How Machine Learning Models Encode Knowledge – and What We Can Learn from Them

Lisa Miou Antoinette Schut

Linacre College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2025

This thesis is dedicated to my family – Han, Heleen, and Donna. Thank you for your love and support, always.

Acknowledgements

Throughout my DPhil and time at Oxford, I've met a lot of wonderful people who have influenced my life and academic journey. I believe that all the conversations and interactions over time have shaped me as a researcher, for which I'm very grateful.

Mentors and Collaborators. I would like to thank my supervisor, Yarin Gal. You took a chance on me years ago and have supported me ever since. Thank you for teaching me how to do research, develop and question ideas, and grow into an independent researcher.

Been Kim – thank you for your mentorship during my internship and beyond. You taught me so much about interpretability, and the field more generally.

Throughout my DPhil, I was lucky to collaborate and learn from many incredible researchers. I learned a lot while working with you, and this has only become more clear with time. Thank you for helping me develop my thoughts and ideas, and for helping me grow as a researcher.

In no particular order, I'd like to thank: Tom McGrath, Demis Hassabis, Tom Zahavy, Vivek Veeriah, Shaobo Hou, Kevin Waugh, Matthew Lai, Edouard Leurent, Nenad Tomašev, Satinder Singh, Clare Lyle, Robin Ru, Sebastian Farquhar, Pascal Notin, Angus Nicolson, Alison J. Noble, Jan M. Brauner, Albert Q. Jiang, Benedikt Höltgen, Miroslav Fil, Jiatong Han, Jannik Kossen, Muhammed Razzak, Shreshth A. Malik, Mark van der Wilk, Edward Hu, Greg Yang, Angus Nicolson, Alison Noble, Evan M. Russek, Ionatan Kuperwajs, Yotam Sagiv, Veniamin Veselovsky, Benedikt Stroebl, Gianluca Bencomo, Dilip Arumugam, Arvind Narayanan, Marcelo G. Mattar, Wei Ji Ma, and Thomas L. Griffiths, Oscar Key, Rory McGrath, Luca Costabello, Bogdan Sacaleanu, Medb Corcoran.

Institutional. I am grateful to be supported by the Engineering and Physical Sciences Research Council, and DeepMind.

Thank you to my examiners, Prof. Max van Kleek and Prof. Bau, for your time and constructive feedback.

Research Labs, Teams and Departments. I was fortunate to visit several labs and teams during my DPhil.

At DeepMind, thank you to Ulrich Paquet and the DEBL team for hosting me. I met a lot wonderful researchers and friends while there – and in particular, I'd like to thank John Hewitt, Bobby He, Kayo Yin, Yash Chandak, and Jordan Hoffmann.

At Google Brain, thank you to Been Kim and the Deep Understanding team. In particular, thank you to Emily Reif, Robert Geirhos, Zi Wang, Greg Yauney, and Blair Bilodeau. I loved coming to the office and chatting with everyone about research.

At Princeton, thank you to the CoCoSci lab and Tom Griffiths for hosting me. I loved spending time at your lab and learning more about Cognitive Science. I am grateful to the friends I've made – Ionatan Kuperwajs, Sally Xie, Ratchit Dubey and Cameron Turner – you are (will be) wonderful professors.

At Oxford, I owe a special thanks to the entire OATML lab. I loved working and learning alongside you all. In particular, I am grateful to Lewis Smith, Clare Lyle, Seb Farquhar, and Pascal Notin for your mentorship and guidance during my DPhil.

I would also like to thank my AIMS cohort, especially Freddie Bickford Smith, Charig Yang, and Baskaran Sripathmanathan, for your friendship over the years. Thank you to Wendy Poole for your continuous support and help throughout my time here.

Finally, I would like to thank my friends in the Department of Statistics – my first academic home at Oxford – Emilien Dupont, Stacy Pu, Fan Wu, and JF Ton.

Personal. To all my friends – thank you for the happiness and support you've brought during these years. I feel so lucky to have you in my life.

Max, I'm so grateful for our friendship and for simply having you in my life. Thank you for coworking sessions, the fun memories and your support over the years. Austeja, thank you for welcoming me in your home, your care and all the delicious food.

Luisa, thank you for all your mentorship and guidance, but even more for the friendship and adventures over the years. I'm excited for many more to come – especially all the auntie memories ahead. Joost, bedankt voor alle advies, support en champagne.

Linde, you are the physical embodiment of a hug; thank you for your support and kindness through the DPhil. Thank you and Youp for your friendship, and always welcoming me in your home.

Emily, thank you for all our incredible adventures, in Seattle, Iceland, NY and beyond. I'm so grateful to have you in my life; our shared experiences and ability to talk about anything mean the world to me.

Lewis, thank you for your friendship and the happiness you've brought during my PhD. I'm grateful for our adventures, and I can't wait to see how you'll continue to make the world a better place.

Katrina, thank you for your support, yoga mornings and brunches that brightened my week. Gunshi, I loved our time together in Oxford and San Francisco, and I look forward to many chai nights in London. Mizu, thank you for the nights out, the long conversations, and for always bringing strength and joy into my life. Finally, thank you to Vittorio, Martin and Jennifer for your friendship.

To my grandparents – Antoinette and Bernardus, thank you for your love and always being proud. I wish you were here to celebrate this moment with me.

To my family – Heleen, Han and Donna – you are everything. Thank you for all your love, guidance and support. You've always made me feel like there was no limit to what I could do but also that it didn't matter what I did, as long as I was happy. Love you always.

Epilogue. With this, my chapter at Oxford comes to an end. I'm grateful for everything I've learned – both as a researcher, but also more generally in life. So long, and thank you all for all the fish.

Abstract

As machine learning systems become increasingly capable–often surpassing human performance–they present both a challenge and an opportunity for understanding. These complex systems may operate in ways that are distinct from human reasoning, making them difficult to interpret. Yet, they also hold the potential to reveal new knowledge and support critical decisions in domains such as healthcare, science, and education.

This thesis examines interpretability as a means of understanding and learning from machine learning models. We focus on two central goals: (1) to understand what and how knowledge is encoded in machine learning models and (2) to extract that knowledge in a form that is meaningful and accessible to humans. To this end, we propose a structured pipeline with four stages: identifying explanation desiderata, locating encoded knowledge or concepts, verifying that the concepts influence model behaviour, and translating the concepts into a human-interpretable form. Within this framework, we adapt existing methods where appropriate and develop new ones where necessary–selecting the approach best suited to the task. Our focus is not only on methodological development but also on understanding how these methods behave and the assumptions they make.

Different parts of the thesis contribute to each stage of this pipeline. We begin by investigating how models encode knowledge: first, by analysing the linear representation hypothesis and then examining the universality of concept representations in multilingual language models. We then shift to the user-facing side of interpretability. We study how to make explanations more user-friendly by leveraging uncertainty to generate realistic and unambiguous explanations. Finally, we apply the full pipeline to develop a framework for extracting novel concepts from AlphaZero and teaching them to chess experts. This final study illustrates how interpretability can help bridge the gap between artificial and human understanding. Together, these contributions advance our understanding of and ability to learn from machine learning systems, laying the groundwork for future research at the intersection of artificial intelligence and human insight.

Contents

List of Figures

List of Tables

xxx

1
Introduction

Contents

1.1 Motivation

At the core of human nature is an innate drive to understand and learn. Humans are inherently curious; we seek to make sense of the world around us. From early childhood, we display a persistent curiosity [Gopnik et al., 1999], which leads us to explore, ask questions, and build an understanding of how things work. This curiosity is vital for our development–shaping how we learn, reason, and make decisions [Kidd and Hayden, 2015]. Over time, this drive has led not only to individual growth but also to collective progress, resulting in scientific and technological advances that shape our society. Understanding is more than a source of insight–it is a foundation for growth and discovery.

The field of machine learning presents new opportunities for understanding and learning. We are observing a paradigm shift as models surpass human performance across various domains. However, these systems may operate in ways that are distinct from human cognitive processes and understanding how these models operate

Figure 1.1: A cartoon of a human peeking into an AI system. This image captures the central idea of this thesis: looking into AI systems to gain a deeper understanding of them and discover new knowledge. This image was generated using DALL · E 3.

remains an open question.[1] This has given rise to interpretability research, which aims to understand and learn from the inner workings of these complex systems.

We can think about leveraging these systems in different ways:

1. (AI → Humans): using AI systems to model humans and the world
2. (Humans → AI): humans understanding AI systems
3. (Humans ↔ AI): learning from AI systems about humans and the world

These three directions are closely connected and often build on each other. We elaborate on each in the paragraphs below.

AI → Humans, World With the emergence of extensive datasets across various domains, AI systems are increasingly used to model real-world phenomena. One of the reasons neural networks are effective is their ability to approximate most functions of interest under some mild assumptions [Leshno et al., 1993]. This allows them to capture complex mechanisms and has led to breakthroughs across different domains. For example, AI systems can discover new biological structures [Jumper et al., 2021]. In cognitive science, AI systems are used to model large-scale

[1] Even if these processes were the same, human cognition itself is not yet fully understood.

behavioural data, generating new hypotheses about human cognition [Kuperwajs et al., 2023, Peterson et al., 2021].

Humans → AI AI systems may operate in a manner that is distinct from humans [Kosoy et al., 2023, Geirhos et al., 2022]. To understand *how* these models operate, we can explore questions such as: What knowledge do they encode? How do they encode knowledge? How do they reason over knowledge? Understanding how models operate is important for improving performance, safe deployment, and, as we will later explain, learning from these systems.

AI ↔ Humans AI systems have surpassed human performance across a range of domains [OpenAI, 2025, Phan et al., 2025]. Given this superhuman performance,[2] we may want to extract new knowledge from these systems. By learning the fundamental principles behind their decisions, we can extend upon and complement our existing collective knowledge. For example, in healthcare, a highly predictive model can be used for diagnosis. However, understanding how the model works may give insight into the cause of disease and prognosis – how the disease develops – which are crucial for prevention and treatment.

These three directions – using AI to model humans and the world, understanding AI systems, and learning from AI to generate new knowledge – are deeply interconnected. Understanding how models operate improves our ability to use them as scientific tools, both to study human behaviour and to extract insights about the world. In turn, these insights can be used to refine and improve the models themselves, creating a feedback loop between humans and AI systems. Interpretability is central to this process, helping us to both understand and learn from these systems. However, despite its importance, deploying interpretability is far from simple. In the next section, we introduce the field of interpretability and explain why understanding machine learning systems is challenging.

1.2 Interpretability

Modern machine learning models are complex systems that often learn from vast amounts of data. These models typically rely on high-dimensional, non-linear transformations to extract patterns from this data and make predictions. However,

[2] We use *superhuman performance* to refer to cases where the AI system outperforms humans on a given task. A more detailed discussion of this term can be found in Sections 6.2 and 7. However, models do not need to outperform humans for this to be valuable. Even models that reason differently – using heuristics or strategies unfamiliar to us – may offer valuable insights.

unlike rule-based systems that follow explicit interpretable logic, deep learning models do not provide a clear step-by-step explanation of how decisions are made. As such, they are often difficult for humans to interpret and are commonly described as black boxes.

At its core, interpretability aims to explain machine learning models in a way that is understandable to humans [Doshi-Velez and Kim, 2017]. One way to think about understanding a model is as the ability to mentally simulate its decision-making process [Lipton, 2018] – similar to reading a manual, where you can work through each step in your mind. However, as machine learning models have grown in scale, manually performing their computations becomes infeasible. For example, consider Llama-3-7B [Dubey et al., 2024], which contains 7 billion parameters. If a human were to manually compute a single forward pass, performing one operation (such as multiplication or addition) per second, the process would take hundreds of years. Understanding the model would likely require more than a single forward pass. This complexity highlights the need to find new approaches to understand how these models operate.

Interpretability focuses on developing new methods to improve our understanding of machine learning models. These methods vary in both the aspects of the model they explain – such as the input, internal representations, or output – and the form they take, ranging from prototype-based explanations to text explanations to finding evidence of a concept encoding in the model's representation space. Often, rather than explaining the entire model, the goal is to provide insights that are useful and relevant to the intended application or setting.

Interpretability can be used for various purposes, such as improving performance, ensuring safety, or enabling knowledge discovery. The most appropriate approach depends on the domain, as different settings introduce different considerations and requirements. For example, in the context of identifying biomarkers in healthcare, the primary goal is to uncover explanations that align with causal mechanisms in human physiology. In such cases, we may emphasise *realism* – ensuring that the explanation captures the true underlying data-generation process, and focusing on less whether it captures the behaviour of the model. In contrast, in the context of AI safety, we care more about the *faithfulness* of the explanation – ensuring that it accurately reflects the model behaviour. For example, if we want to detect potentially harmful or deceptive behaviour, we want to find behaviour that is implemented and used by the model. Beyond the domain, factors such as the data modality and model architecture can impose further constraints on the types of explanations that are suitable. While saliency maps can provide useful visual insights for continuous real-valued data, such as images, they are not suited for discrete input data, where alternatives, such as prototypes of counterfactual explanations, may perform better. Given these varying goals and constraints, a natural next question is how we can design interpretability methods that help us both understand these systems and extract meaningful insights from them.

1.3 Thesis Contributions

Interpretability is an active and evolving field. As models become more capable and are applied to increasingly complex domains, there is an ongoing need for interpretability methods that are both effective and well-suited to the task. In this thesis, we focus on designing interpretability methods to better understand and learn from AI systems. Our goal is two-fold:

1. To understand what and how knowledge is encoded in machine learning models.
2. To identify how this knowledge can be extracted in a human-interpretable form.

We consider a range of settings with different data modalities, constraints, and end users. Despite these differences, our work builds towards a common interpretability pipeline for uncovering knowledge encoded in models, and making that knowledge accessible to humans.

This pipeline consists of the following components:

- **Identify Explanation Desiderata** (Chapters 4, 5 and 6). We begin by identifying the requirements that explanations must meet. These vary depending on the context, the data type, the model architecture, and the intended audience.
- **Identifying Knowledge Encoded in Model Representations** (Chapters 3, 4 and 6). We examine the model's internal representations to determine *what* information is encoded and *how* it is organised in the latent space. Throughout this thesis, we often isolate the concept or phenomenon of interest by contrasting the model's behaviour across two carefully constructed datasets. This contrastive approach provides fine-grained control over what is being analysed, while remaining broadly applicable across domains.
- **Verify the knowledge influences the model behaviour** (Chapters 4 and 6). Neural networks are expressive enough to encode many features by chance – features that may not be used in decision making (see Appendix A).

To ensure our findings from the previous step are meaningful, we test whether the knowledge is used by the model.

- **Extract the knowledge in a human interpretable form** (Chapters 5 and 6). Finally, we develop methods to extract the encoded knowledge into a form that is understandable to humans. The goal is to bridge the gap between the model's internal representations into a form that humans can understand, and potentially learn from. In this thesis, we focus primarily on *prototypical examples* – representative instances that capture the class or concept of interest.

This thesis contributes to each stage of the pipeline. We adapt existing methods where appropriate and develop new ones where necessary–choosing the approach best suited to the problem at hand. From a technical perspective, we focus not only on method development but also on understanding how these methods work and the assumptions they rely on.

1.4 Thesis Structure

The thesis consists of four main parts.

Part I: Foundations. The first part provides a general background on interpretability research, providing an overview of the field, including explanation desiderata, methods and evaluation. This section provides the foundational background necessary for the subsequent chapters.

Part II: Understanding Knowledge Encoding in Models. This section explores how machine learning models represent and encode knowledge.

- **Chapter 3: Understanding the Linear Representation Hypothesis.** This chapter investigates a common assumption about how models encode knowledge – the linear representation hypothesis (LRH), which posits that knowledge is encoded linearly in the model's representation space. We examine the assumptions behind this hypothesis, focusing on polysemanticity, spatial dependence, and the influence of interventions across layers. We use the LRH in later chapters (Chapters 4 and 6).

  This chapter builds on A. Nicolson, L. Schut, J. A. Noble, and Y. Gal. Explaining explainability: Recommendations for effective use of concept activation vectors. *Transactions on Machine Learning Research*, 2025. Accepted February 2025

- **Chapter 4: Universal Representations in LLMs** This chapter investigates how knowledge is encoded in large language models – specifically, whether these models learn universal representations or language-specific representations.

  This chapter builds on L. Schut, Y. Gal, and S. Farquhar. Do multilingual llms think in english? *Building Trust Workshop at the Thirteenth International Conference on Learning Representations*, 2025a

Part III: Human-Interpretable explanations This part shifts focus from internal representations to the human side: how to create explanations that are understandable to users.

- **Chapter 5: Generating Explanations** This chapter focuses on creating prototypical explanations that are easy for users to understand. In particular, we use uncertainty estimates to generate explanations that are both realistic and unambiguous.

  This chapter builds on L. Schut, O. Key, R. Mc Grath, L. Costabello, B. Sacaleanu, M. Corcoran, and Y. Gal. Generating interpretable counterfactual explanations by implicit minimisation of epistemic and aleatoric uncertainties. In A. Banerjee and K. Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 1756–1764. PMLR, 13–15 Apr 2021

- **Chapter 6: Knowledge discovery in chess.** This chapter combines both thesis goals. Here, we explore how we can extract novel concepts from AlphaZero, an AI system that plays chess at a superhuman level. The aim is to uncover knowledge that is not known to humans and teach it to human experts. As in Chapter 4, we find concepts using a contrastive approach, but here, we operate in an unsupervised setting, which requires developing a new method. Building on Chapter 5, we extract concepts as prototypical examples, which we then teach to experts through a proof of concept human experiment.

  This chapter builds on L. Schut, N. Tomašev, T. McGrath, D. Hassabis, U. Paquet, and B. Kim. Bridging the Human–AI Knowledge Gap through Concept Discovery and Transfer in AlphaZero. *Proceedings of the National Academy of Sciences*, 122(13):e2406675122, 2025c. doi: 10.1073/pnas.2406675122

Part IV: The final part reflects on the contributions of this thesis. It summarises the main insights and outlines future directions for interpretability as a tool for understanding and learning from machine learning systems.

Part I

Foundations

2
Foundations

Contents

This chapter provides a general overview of the field of interpretability. Where necessary, we include additional background in later chapters.

2.1 Defining Interpretability

In this section, we discuss what *interpretability* means in the context of machine learning. There is an ongoing debate around the definition of *interpretability*. Some commonly cited definitions include:

> *A method is interpretable if a user can correctly and efficiently predict the method's results* [Kim et al., 2016].
>
> *The ability to explain or present [the model] in terms understandable to a human* [Doshi-Velez and Kim, 2017].
>
> *An interpretable ML model obeys a domain-specific set of constraints to allow it to be more easily understood by humans. These constraints can differ dramatically depending on the domain.* [Rudin et al., 2022].

Despite differences in phrasing and emphasis, these definitions share several common themes. Below, we summarise key components that are frequently cited in the interpretability literature:

- **User comprehensibility**: the target user can understand the explanation [Kim et al., 2016, Doshi-Velez and Kim, 2017, Adadi and Berrada, 2018, Gilpin et al., 2018].
- **Model faithfulness**: the explanation should accurately reflect the model's inner workings [Doshi-Velez and Kim, 2017, Gilpin et al., 2018]
- **Data alignment**: the explanation should reflect the (training) data [Murdoch et al., 2019].
- **Relevant to the output**: the explanation should explain or influence the model's output or decision [Kim et al., 2016, Miller, 2019, Molnar, 2020].
- **Domain-specific**: the explanation should incorporate domain-specific aspects [Rudin, 2019, Murdoch et al., 2019, Rudin et al., 2022].

There may be an inherent trade-off between different components of interpretability. One example is the tension between model faithfulness and user comprehensibility. A faithful explanation accurately reflects the model's internal mechanisms but may be too complex or unintuitive for users to understand. Adversarial examples [Szegedy et al., 2013, Goodfellow et al., 2014] illustrate this tension well. An adversarial example is a minimal perturbation to an input, typically measured by the L_2 or L_∞ norm, that changes the model's prediction. In some sense, these examples are explanations: they provide a counterfactual explanation – an alternative input that changes the prediction of the model. However, these perturbations are often imperceptible to humans and therefore may not be as easily understood. On the other hand, a more intuitive explanation may oversimplify the model's reasoning. Therefore, while easier to understand, this explanation may no longer be faithful to the model's inner workings. This dichotomy was highlighted by Kim [2022], who suggested that the reason for this trade-off may lie in that humans and AI do not share a common language.

This trade-off raises a deeper question: what exactly do we want from the *definition* of interpretability in machine learning? A definition is a statement that explains the meaning of a word or phrase [Cambridge Academic Content Dictionary, 2023]. However, to be useful in the context of machine learning, a definition must also possess the quality of being *operationalisable*, meaning that it should be something we can measure or describe within a machine learning framework [Lipton, 2018].

In this thesis, we take a practical approach to interpretability. Rather than aiming for a single, universal definition, we focus on the usefulness of explanations within a given context. We treat interpretability as a tool – it supports a specific goal, and the form should adapt accordingly. For example, if the goal is to understand model behaviour – for instance, in the context of safety or debugging – then the emphasis should be on model faithfulness. The explanation should help us understand the model's internal mechanisms and how they influence the output. Conversely, if the goal is to use interpretability to discover new knowledge, the focus should shift toward accurately capturing patterns in the underlying data-generating process. In this case, some deviation from model internals may be acceptable as long as it captures the true causal mechanisms within the domain. This task-dependent view allows for flexibility. It acknowledges that different objectives may require different trade-offs – and that no single definition of interpretability can serve all purposes equally well.

This framing also reduces the relevance of the distinction often drawn between *interpretability* and *explainability*. Some argue that interpretability refers to whether a model's components can be directly understood – for example, through sparse weights or decision rules – while explainability refers to post hoc methods applied to black-box models [Flora et al., 2022]. In this work, we do not distinguish between the two terms. Following Kim et al. [2016] and Molnar [2020], we treat both as referring to the broader goal of producing explanations that support understanding, regardless of whether they are intrinsic to the model or externally applied.

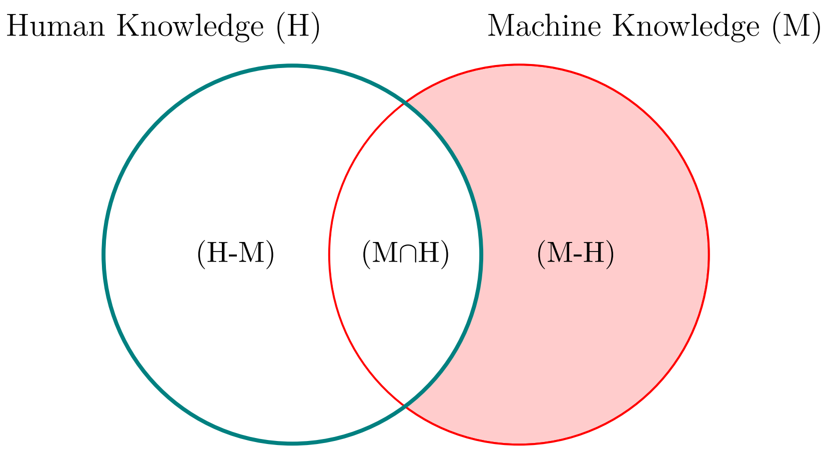
2.2 Why Interpretability?

Following the idea that interpretability should be defined relative to its intended use, the natural next question is: why do we want interpretability? In this section, we summarise several common motivations for interpretability. For a more comprehensive discussion, see Lipton [2018].

Safety and Robustness As AI systems are increasingly deployed in real-world settings, ensuring that they behave reliably and as expected is of critical importance. This applies not only to safety-critical applications, such as healthcare or autonomous driving, but also to high-impact systems like large language models, which affect millions of users. Models can exhibit unexpected behaviours – for instance, sycophancy, where a model insincerely agrees with user assertions [Sharma et al., 2023] or goal misgeneralisation, where a model performs well during training but fails in new settings due to internal objectives that are misaligned with the intended goal [Shah et al., 2022, Denison et al., 2024]. Interpretability can help surface these types of alignment failures by uncovering heuristics or internal rewards the model has learned. In some cases, it can also shed light on the origin of these behaviours – whether they emerge from architectural biases or are learned, and which training examples may have contributed to the behaviour [Koh and Liang, 2017].

Building Trust Interpretability plays an important role in building trust between machine learning systems and their users, particularly in high-stakes settings such as healthcare and finance. By providing explanations, practitioners are better positioned to evaluate whether the model's reasoning aligns with their specialised domain knowledge and ethical standards. This supports epistemic trust [Fonagy and Allison, 2014] – the willingness to treat information as credible and relevant for decision-making. Previous work has shown that domain experts are more likely to adopt machine learning systems when they are able to question and validate the model's reasoning process [Doshi-Velez and Kim, 2017]. As such, interpretability is a necessary component for responsible deployment and sustained practitioner confidence.

Figure 2.1: Learning from machine-unique knowledge, modified from Kim [2022] with permission. Teal represents human knowledge and pink represents knowledge encoded in AI systems. In the context of knowledge discovery, we are interested in extracting machine-unique knowledge from the shaded region ($M - H$).



Knowledge discovery Interpretability is a powerful tool for scientific discovery. As machine learning models are applied to increasingly large and high-dimensional datasets, they offer the potential to reveal new insights and provide domain experts with new hypotheses. In cognitive science, for example, extensive datasets are leveraged to model and better understand human cognitive processes [Kuperwajs et al., 2023, Peterson et al., 2021]. In medicine, models have identified new structures [Jumper et al., 2021], and markers associated with a disease [Lundberg et al., 2018, Kermany et al., 2018].

Another avenue for discovery is analysing AI systems that surpass human performance, such as chess engines. These systems encode knowledge that may not be known to humans and is not captured in human-constructed datasets. The goal of interpretability in this context is to extract new knowledge from these systems. The idea is summarised in Figure 2.1, where we distinguish between the space of human knowledge (H), machine-unique knowledge (M), and the intersection ($M \cap H$). Interpretability may allow us to access ($M - H$), thereby expanding the boundaries of human knowledge (H).

Ethics and Accountability Interpretability plays a key role in evaluating whether machine learning models behave in a fair, transparent, and responsible way. In particular, it can help address ethical concerns across three important areas: fairness, privacy, and accountability.

- **Fairness**: Interpretability methods can help uncover and address discriminatory patterns by showing how a model's predictions vary across demographic groups. For example, saliency maps or counterfactual examples can reveal whether a model relies on protected attributes–such as race or gender–either directly or through correlated proxies [Zemel et al., 2013, Barocas et al., 2019]. These insights allow for targeted audits of disparate impact and can guide interventions to improve fairness.

- **Privacy**: In privacy-sensitive contexts like healthcare, interpretability can help identify when a model inappropriately memorises or relies on sensitive data. For example, analysing feature attributions can expose data leakage or overfitting to individual records. Prior work has shown that models are vulnerable to membership inference [Shokri et al., 2017] and reconstruction attacks [Song et al., 2017]. In this context, interpretability is useful for auditing privacy risks and evaluating privacy-preserving methods.

- **Accountability and algorithmic recourse**: Interpretability is critical for ensuring compliance with AI regulations. Under the EU's GDPR, individuals have a right to an explanation when subject to automated decisions [Sartor and Lagioia, 2020]. Beyond compliance, interpretability supports algorithmic recourse–helping users understand how to change their inputs to receive different outcomes (e.g., loan approvals). This can increase transparency and procedural fairness.

Debugging, Understanding and Improving Model Behaviour Interpretability can be used as a tool to debug and improve models. By understanding how a model operates, we can alter its training and inference behaviour – such as improving reasoning skills [Huang and Chang, 2022, Wei et al., 2023, OpenAI, 2024] Similarly, when models fail unexpectedly, interpretability methods can help pinpoint the underlying causes. These failure modes can span a wide range, from reliance on spurious correlations to limitations in a model's reasoning capabilities. For example, interpretability can be leveraged to find bugs [as in Ribeiro and Lundberg, 2022] or reliance on spurious correlations [Gajcin and Dusparic, 2022]. However, it can also find more general behavioural limitations such as specific problems in inter-agent dynamics [Omidshafiei et al., 2022] or limitations in multi-step reasoning [Yang et al., 2024]. In this way, interpretability allows us to find the cause of problems, which we can, in turn, use to improve models.

2.3 Practical Considerations in Interpretability

While interpretability is often useful, it is not universally necessary. In some settings – particularly low-stakes or well-understood applications – explanations may provide little added value. For example, in sorting or hashing algorithms, the internal logic is already transparent and deterministic, making interpretability tools redundant [Molnar, 2020].

Moreover, interpretability can come at a cost. Inherently interpretable models, such as decision trees or linear models, may struggle to match the predictive performance of more complex alternatives like deep neural networks, particularly on high-dimensional or unstructured data. While post-hoc explanation methods allow us to apply interpretability to black-box models, they often incur computational overhead at inference, particularly when applied at scale.

Interpretability may lower the barrier to security attacks: while many systems can already be reverse-engineered, explanations can make it easier for attackers to identify vulnerabilities or design targeted manipulations. Conversely, explanations themselves can be misleading or unfaithful–especially when generated post hoc [Slack et al., 2020, Lipton, 2018]–creating a false sense of understanding or trust across users. Careful consideration is therefore required not only in how interpretability methods are designed, but also in the safeguards and governance structures that determine who can access them and under what conditions.

When correctly implemented, interpretability rarely poses a drawback on its own, but, in some cases, other tools – such as empirical evaluation or fairness audits – may be more appropriate. The arguments in this section should therefore be read not as objections to interpretability, but as reflections on the factors that inform when and how it provides the greatest benefit.

2.4 Explanation Desiderata

In the previous section, we discussed arguments both for and against the use of interpretability. In this section, we explore what properties we want from an explanation when we use interpretability methods. Unless otherwise stated, these attributes are generally summarised from Lipton [2018] and Molnar [2020].

Informativeness An explanation should be informative – users should be able to understand and learn from explanations [Schut et al., 2025c]. We want explanations to contain information rather than noise. An informative explanation should help users learn something new about the model, the data, or the task, enabling them to form more accurate mental models of how the system operates. This is particularly important in contexts where users need to make decisions based on the explanation, or when explanations are used as a tool for scientific discovery or debugging.

Model Faithfulness *Model faithfulness* refers to the extent to which an explanation captures the inner workings of a model, such as identifying features that are encoded in internal representations or circuits that are used by the model. An explanation that is not faithful fails to describe the true model behaviour – either because it oversimplifies, abstracts away relevant complexity, or incorrectly attributes mechanisms.

A related concept is *fidelity*, which measures how accurately the explanation can predict the model's output. Explanations that are faithful may not have high fidelity. For example, we may identify a feature that is encoded in the model, which, on its own, is not highly predictive of the output. Some methods, such as LIME [Ribeiro et al., 2016] and SHAP [Lundberg and Lee, 2017], offer high fidelity but only locally – i.e., they accurately approximate model behaviour in a neighbourhood around a given instance but do not generalise globally.

Some researchers further introduce the notion of *translucency*, which characterises the degree to which an explanation requires internal access to the model [Selvaraju et al., 2017, Molnar, 2020]. Inherently interpretable models – such as linear regression or decision trees – are translucent. In contrast, post-hoc explanation methods for black-box models that rely on perturbing inputs and observing outputs, are not translucent.

Faithful explanations are particularly important in high-stakes or safety-critical domains, where misleading explanations could result in misplaced trust or oversight.

Human understandable The goal of interpretability is generally to provide explanations to humans. As such, a core requirement is that explanations should be understandable. This can take many forms: highlighting important features, identifying representative prototypes, or describing logic through rules.

However, not all explanations are equally easy to understand. Some may accurately describe model behaviour but remain too complex for human reasoning. For example, complex neural circuits or highly abstract activation patterns may be faithful to the model, but too difficult to follow. Lipton [2018] formalises this idea with *simulatability*–the notion that a person should be able to mentally step through the model's decision process.

In some settings, human understanding is not essential. When the goal is to improve model performance or uncover failure modes, the explanation does not need to be intuitive. Adversarial examples are a clear case: although unintuitive, they can expose vulnerabilities. Here, the explanation is useful to researchers or developers, who do not need to fully understand the explanation to use it to improve the model.

Ultimately, the need for human interpretability depends on the audience and purpose. When explanations are used to inform human decisions, they must be accessible. However, when the goal is technical insight or safety analysis, clarity may be secondary to informativeness.

Expressivity and explanation format Another important consideration is the format of the explanation. Should the explanation be in the form of a vector, written text, or a visual explanation, such as a saliency map? Different formats are better suited to explain different aspects of the model – whether we are trying to explain the representation space, the output, or the important aspects of the input. For example, probing methods are well-suited to explain the latent space, whereas counterfactual explanations are better suited to explain the input.

Each format, however, comes with inherent limitations. Text is often easier to understand as it aligns with natural human language. However, its use of discrete tokens can limit expressivity compared to formats operating in continuous spaces. Even within a given format, there are often additional constraints. For example, many interpretability methods probe models using linear vectors. The linearity assumption simplifies analysis and improves interpretability – but may overlook more complex, non-linear behaviours.

In general, no single format is universally best. The most appropriate choice depends on what aspect of the model is being explained, and who the explanation is for.

Global vs. Local explanations Explanations can be of different granularity. Local explanations focus on explaining the model's behaviour on a specific instance. This property may be useful for algorithmic recourse – when we want to explain the model's behaviour for an individual. Alternatively, sometimes, we may want a global explanation, which describes the model's behaviour across the entire dataset. This allows us to determine how the model performs at scale and whether it is suitable for deployment.

Concise We want an explanation to be as concise as possible [Huysmans et al., 2011, Laugel et al., 2019, Wachter et al., 2017, Lahav et al., 2018, Molnar, 2020, Van Looveren and Klaise, 2021]. By creating a concise explanation, we can avoid overloading an individual with information, ensuring the information can be processed more easily.

Complete We may want an explanation to be *complete*: to explain all of the model's behaviour. Yeh et al. [2020] underscored this requirement in the context of concepts, where they evaluate how *sufficient* concepts are in explaining a model's decision. Alternatively, we may want an explanation to cover all components of the model – the input, model components (e.g., neurons) and output. This is closely related to *decomposability*, where each part of the model is expected to have an interpretable explanation.

Realism Another desideratum is that the explanation should be *realistic*, which is rooted in the idea that it should originate from a 'possible world' [Wachter et al., 2017, Schut et al., 2021]. This means that the explanation should accurately capture the underlying data model. Realism can be important for several reasons. For example, we may want this because it increases the likelihood that the explanation can predict unseen data, which is useful for scientific discovery. Another context in which we may care about realism is algorithmic recourse, where users are provided with actionable feedback. In such cases, realism ensures that the suggested changes are feasible, and more likely to lead to the desired outcome in the real world.

Actionable [Pawelczyk et al., 2020, Rawal and Lakkaraju, 2020, Schut et al., 2021] In some contexts, particularly algorithmic recourse, we may want an explanation to be *actionable*. An actionable explanation is one that users can use to intervene on, and should empower users to make informed decisions or take proactive measures. For example, consider an individual who is rejected for a bank loan. In this case, the explanation should highlight factors the user can realistically modify to improve their eligibility. Some factors – such as the number of outstanding loans – may be more easily addressed than others, like the duration of their banking history. The actionability of the explanation is critical in such settings, as it enables users to understand not just why a decision was made, but also what they can do to achieve a different outcome.

Consistency and Robustness Explanations should be consistent with the model's behaviour. When the model generates a particular prediction, the accompanying explanation should not contradict it. This consistency between the prediction and explanation is crucial for building trust in the model. Similarly, we want an explanation to be robust, meaning that irrelevant information or noise should not lead to a change in the explanation. Zafar et al. [2021] show that model explanations are often not robust. Crabbé and van der Schaar [2023] examine explanation invariance and equivariance.

Computational Complexity Another important aspect is the computational complexity of the methods that generate explanations. In many settings, particularly interactive applications, the explanation must be computed quickly enough to be useful [Schut et al., 2021, Van Looveren and Klaise, 2021]. Space complexity is also a key factor, especially when deploying models at scale. Given the size of current machine learning models, the computational burden of interpreting entire models can be prohibitive. As a result, researchers often focus on analysing selected components, such as individual neurons or layers, rather than attempting to explain the model as a whole.

Model-Agnostic These methods are not restricted to specific models, and can be applied across a wide range of models. This desideratum, sometimes referred to as *portability* [Lipton, 2018], allows explanations to be generated without requiring access to a model's internal structure or parameters. These methods are particularly useful in settings where only black-box access to the model is available or proprietary. However, this generality often comes at the cost of reduced faithfulness to the model's internals.

Target Audience Explanations should be tailored to their intended audience [Doshi-Velez and Kim, 2017]. For example, explanations that highlight influential neurons or weights may be useful for machine learning engineers, while domain experts or general users may require explanations framed in terms of familiar concepts or examples. Consequently, the form, complexity, and terminology of an explanation should be adapted to match the user's background and needs, ensuring that the explanation is both accessible and useful.

Summary. Some desiderata may be at odds with others. We may need to focus on some desiderata, while deprioritising others. For example, explanations that aim to be complete may be less concise, while explanations that prioritise being model-agnostic may be less faithful. Which desiderata are most important depends on the specific use case and user group. In Table 2.1, we provide a high-level summary of which desiderata are likely to be most critical in different contexts. This is not intended to be prescriptive, but rather to highlight common priorities across settings such as safety, trust-building, scientific discovery, ethics, and debugging. Moreover, the entries should be interpreted as general (on average) tendencies – in practice, the desiderata for specific use cases can vary considerably.

It is important to note that certain desiderata – such as computational complexity and model agnosticism – are often more influenced by factors like model access, scale, or deployment environment, rather than the specific application context. For this reason, we omitted computational complexity and model agnosticism from the table. For example, computational complexity is primarily influenced by factors such as model size and deployment type (e.g., large-scale or interactive settings). Similarly, model-agnostic methods may be necessary when model access is unavailable, for example, in proprietary systems.

Different interpretability methods are better suited to different settings. In the next section, we summarise key categories of interpretability methods and outline which desiderata they tend to prioritise.

Table 2.1: Importance of explanation desiderata across different interpretability motivations. Symbols indicate relative importance: ●●●= high importance, ●●= medium importance or depends on context, ●= not critical. Gen = general public, Exp = domain expert, ML = machine learning practitioner.

Explanation desiderata	Safety and robustness	Trust	Knowledge discovery	Ethics and accountability	Model Behaviour
Informativeness	●●	●●	●●●	●●●	●●
Faithfulness	●●●	●●●	●	●●●	●●●
Human understandable	●●	●●●	●●●	●●●	●
Concise	●	●●	●●	●●●	●
Complete	●●	●●●	●	●●	●●
Realism	●	●	●●●	●●●	●
Actionable	●●●	●	●	●●●	●●●
Consistency	●●	●●●	●●	●●	●●
Global vs local	Both	Either	Either	Either	Either
Target audience	Exp, ML	Gen, Exp	Exp	Gen, ML	ML

2.5 Interpretability Methods

In this section, we provide a brief overview of different interpretability methods.

Feature-based explanations These methods explain model decisions by highlighting the importance of individual features or attributes in the input data. Feature-based explanations have been explored in various domains, including language [Karpathy et al., 2015] and computer vision [Olah et al., 2017].

The most well-known techniques are Local Interpretable Model-agnostic Explanations (LIME) [Ribeiro et al., 2016] and Shapley Additive Explanations (SHAP) [Lundberg and Lee, 2017]. LIME explains the model predictions by learning a sparse linear regression that identifies the relative feature importance. SHAP estimates the marginal contribution of each feature towards the model prediction, averaged over all possible predictions. Both methods are post-hoc and generally focus on local explanations, although global variants exist.

However, these methods are known to suffer from limitations, including instability under small input changes and a lack of faithfulness to the model's true decision-making process [Alvarez-Melis and Jaakkola, 2018b, Slack et al., 2020]. Some researchers have suggested modifications of both methods to improve consistency, reliability and computation efficiency [Slack et al., 2021].

Concept-based Methods Concept-based methods explain model predictions using high-level abstractions, or concepts. These are designed to be more interpretable to human users, especially domain experts [Bau et al., 2017, Kim et al., 2018, Alvarez-Melis and Jaakkola, 2018c, Koh et al., 2020, Bai et al., 2022, Achtibat et al., 2022]. Such explanations have been shown to be useful in scientific and biomedical domains, where decisions are often grounded in expert-defined concepts rather than low-level features [Graziani et al., 2018, Sprague et al., 2019, Clough et al., 2019, Bouchacourt and Denoyer, 2019, Yeche et al., 2019, Sreedharan et al., 2020, Mincu et al., 2021, Jia et al., 2022].

Since the introduction of concept activation vectors (CAVs) [Kim et al., 2018], several methods have extended or adapted this approach. For example, Goyal et al. [2019] introduce causal formulations of concepts. Zhang et al. [2020] and Schrouff et al. [2021] propose methods to generate local concept explanations in addition to global ones. Yeh et al. [2020] focus on identifying concepts that sufficiently explain model predictions, while Ramaswamy et al. [2023] aim to improve the faithfulness and controllability of concept representations.

Recent work has also explored discovering concepts with or without supervision. In supervised approaches, models are often probed using a labelled dataset of concept exemplars [Kim et al., 2018]. When going beyond supervised concepts derived from human labels [Yeh et al., 2020, Ghorbani et al., 2019c, Ghandeharioun et al., 2021], concepts may be expressed via example sets of data points [Yeh et al., 2020, Ghorbani et al., 2019c] or by generating new data [Ghandeharioun et al., 2021].

Saliency-based methods Saliency-based methods identify which parts of the input are most influential for the model's prediction. These methods work directly on the input space and produce saliency maps that assign importance scores to individual input components, such as pixels or tokens. Unlike feature-based methods, which often aim to rank or describe the importance of features across many predictions, saliency methods tend to provide local explanations – focusing on the specific factors driving a single prediction.

There are several types of saliency-based methods. Gradient-based saliency maps estimate the gradient of the model's output with respect to the input [Baehrens et al., 2010, Simonyan et al., 2013]. These gradients can be noisy, difficult to interpret, and prone to saturation. To address these issues, several modifications have been proposed, including adding noise and averaging the results [Smilkov et al., 2017], interpolating gradients between a baseline and the input [Sundararajan et al., 2017], modifying the ReLU backpropagation [Dosovitskiy et al., 2015], and applying layer-wise relevance propagation [Bach et al., 2015].

Example-based methods Another family of interpretability methods is example-based explanations. These methods explain model behaviour by presenting specific examples, either from the training data or constructed inputs. One approach is to use *counterfactual explanations*, which show alternative inputs that would have led to a different outcome [Wachter et al., 2017]. These highlight the minimal changes needed to change the model's decision, providing insight into important features. *Adversarial examples* can also be seen as counterfactuals, but their goals differ. They are inputs that are minimally perturbed but cause the model to change its prediction [Szegedy et al., 2013, Goodfellow et al., 2014]. While often imperceptible to humans, they reveal vulnerabilities in the model's behaviour and can be used to probe its sensitivity. *Influential examples* identify training datapoints that had the most influence on a given prediction [Koh and Liang, 2017]. By tracing predictions back to key examples, we can uncover which parts of the training data shaped a particular behaviour. *Prototypical examples* show inputs that are most representative of a concept or class [Kim et al., 2016]. They provide a reference point that can help users understand what the model considers typical for a given category. These methods are often model-agnostic and can be applied to a wide range of models. However, they do not necessarily provide a unique explanation, as different examples may offer different perspectives on the same behaviour. Depending on the context, one approach may be more appropriate than others.

Rule-based methods These methods create explicit rules to explain the model's prediction. Some inherently interpretable models are rule-based; these learn rules on a single feature [Holte, 1993], sequentially learn individual rules [Cohen, 1995], and take Bayesian approaches to learning rules [Letham et al., 2015, Yang et al., 2017]. These methods are often straightforward to interpret, and space and computationally efficient. However, they are less suited to capturing linear relationships or complex interactions between features [Molnar, 2020].

In addition to inherently interpretable models, rule-based explanations can be applied post-hoc. For example, Ribeiro et al. [2018] propose Anchors, a method that identifies a set of conditions under which the model's prediction remains stable. These rules aim to provide high-precision explanations that are easy for users to understand.

Surrogate models Surrogate models are interpretable models that are trained to approximate the behaviour of a more complex, black-box model [see, e.g., Bastani et al., 2017]. They are flexible and model-agnostic, as they can be applied to any predictive model by approximating its outputs on a given dataset. However, this flexibility comes at the cost of faithfulness. The surrogate may not perfectly capture the original model's decision boundaries, especially in regions where the model behaviour is highly non-linear or complex.

Mechanistic Interpretability Mechanistic interpretability is a subfield of interpretability, that contains many different types of methods. The goal is to reverse-engineer neural networks by mapping their learned weights and structures into human-interpretable features, circuits, or graphs [Cammarata et al., 2020, Elhage et al., 2021]. The focus is on identifying how specific neurons, attention heads, or subnetworks contribute to the model's behaviour. This approach has been applied across a range of models and domains, with methods such as feature visualisation [Olah et al., 2018], circuit analysis [Olah et al., 2020], activation patching [Meng et al., 2022] and sparse autoencoders (SAEs) [Cunningham et al., 2023].

Mechanistic interpretability often takes a bottom-up perspective, analysing the behaviour of individual units or small-scale circuits. However, as Anthropic [2023] note, there is also value in complementing this with top-down approaches– starting from observable model behaviours and tracing them back to the responsible components.

Recent work has also highlighted the challenge of superposition – where multiple unrelated features are entangled within the same neuron or direction [Elhage et al., 2022b]. This adds complexity to mechanistic analysis, as components may not map cleanly onto distinct or human-understandable concepts. For a more in-depth review of mechanistic interpretability methods and their limitations, see Bereska and Gavves [2024], Sharkey et al. [2025].

Intrinsically interpretable models Intrinsically interpretable models are designed to be understandable by construction. Common examples include linear regression [Gauss, 1809, Legendre, 1805], decision trees [Breiman et al., 1984, Quinlan, 1986], k-nearest neighbours (KNN) [Cover and Hart, 1967], and rule-based methods [Cohen, 1995]. For example, in the context of regression, we can understand the model's behaviour by using the variable coefficients. The coefficient is the model's sensitivity to that feature – specifically, it measures the change in the output for a unit change in the input feature, holding all other variables constant.

There is a large discussion on using inherently interpretable machine learning models instead of post-hoc interpretability methods. Rudin [2019] argues that inherently interpretable models are essential in safety-critical or high-stakes applications, as they provide more guarantees around the model behaviour than post-hoc methods, which are less faithful and interpretable.

A commonly held belief is that there is an inherent trade-off between interpretability and performance [Dziugaite et al., 2020]. Neural networks, which are typically considered black-box models, can approximate any continuous function given sufficient capacity, as formalised by the universal approximation theorem [Cybenko, 1989, Funahashi, 1989, Hornik et al., 1989]. In contrast, interpretable models can potentially reveal the underlying data-generating process but may require stronger assumptions, domain expertise, or careful design to achieve comparable performance [Rudin, 2019]. Moreover, black-box models may detect patterns that are not easily discoverable by interpretable models, including behaviours or structures previously unknown to domain experts.

Table 2.2: How well different interpretability methods capture key explanation desiderata. The table summarises general tendencies–how likely each method is to support a given property. These should be read as broad characterisations rather than strict classifications, as individual methods within each category may differ. Symbols: ●●●= strong support, ●●= partial, ●= not typical.

Method	Informativeness	Faithfulness	Human understandable	Concise	Complete	Realism	Actionable	Consistency and robustness	Computational complexity	Model agnostic	Global vs local
Feature-based	●●	●●	●●●	●●●	●	●●	●●	●●	●●	●●	Both
Concept-based	●●	●●	●●●	●●●	●	●●	●●	●●	●●	●	Both
Saliency-based	●●	●●	●●	●	●	●	●	●	●●	●	Local
Example-based	●●	●●	●●●	●●	●	●●●	●●●	●●	●●	●●	Local
Rule-based	●●●	●●	●●●	●●●	●●	●●	●●	●●●	●●●	●	Both
Surrogate models	●●	●●	●●	●●	●	●●	●●	●●	●●	●●●	Global
Intrinsic interpretability	●●●	●●●	●●●	●●●	●●●	●●	●●	●●	●●●	●	Both

Summary. Table 2.2 summarises how likely different interpretability methods are to capture the different explanation desiderata. These summaries should be understood as general tendencies – how likely each method is to support a given desideratum varies in practice. However, within each category, methods can vary considerably, and specific implementations may target these properties more or less explicitly.

Moreover, some aspects are also context-dependent. For example, the realism of an explanation depends not only on the method but also on how accurately the underlying model reflects the true data-generating process. Similarly, the actionability of an explanation will vary depending on the type of output it produces and whether this output affords meaningful user interventions.

Finally, we excluded mechanistic interpretability from the table, as it is a group of methods, each with different properties. Including it in the table would oversimplify the diversity of the methods, and lead to misrepresentation.

2.6 Evaluation of Interpretability Methods

We have established that interpretability methods vary in type, the context they are used in, and the desiderata they aim to address. Consequently, no single evaluation approach can capture all aspects of interpretability across all settings. Instead, the evaluation should be aligned with the specific claims made by the method [Doshi-Velez and Kim, 2017].

Evaluation approaches can be broadly categorised into the following categories.

Correctness of the Explanation The correctness of the explanation measures whether it captures the underlying (real-world) mechanisms. For example, whether a feature-based method can accurately identify biomarkers. In some cases, the correctness of an explanation can be directly evaluated against known ground truth [Ribeiro et al., 2016, Arras et al., 2019, Kim et al., 2018]. More generally, this is often possible when constructing datasets so that we can control the ground-truth generation process. To this end, Yang and Kim [2019] introduce a benchmark and metrics to estimate feature importance.

Where direct evaluation is not possible, proxies are often used. A common approach is to measure the impact on model accuracy after removing or perturbing features identified as important by the method [Samek et al., 2016, Fong and Vedaldi, 2017, Ancona et al., 2017, Hooker et al., 2019]. This approach, known as feature removal or ablation, evaluates the selectivity of the explanation, i.e., the method's ability to correctly identify the most influential features [Montavon et al., 2018]. More rigorous variants include protocols such as Remove and Retrain (ROAR) [Hooker et al., 2019], where features are removed across the dataset and the model is retrained, offering a more robust evaluation of feature importance estimates.

Sensitivity of Method to Changes in the Model For post-hoc interpretability methods, it is difficult to evaluate the faithfulness of the explanation to the model [Sundararajan et al., 2017]. However, several proxies have been proposed. These include measuring the sensitivity of explanations to model changes [Adebayo et al., 2018, Heo et al., 2019] or to input perturbations [Ghorbani et al., 2019a, Alvarez-Melis and Jaakkola, 2018a]. For instance, the model parameter randomization test proposed by Adebayo et al. [2018] involves randomising model weights and observing if the explanation degrades accordingly. If explanations remain unchanged despite the model being randomised, they may not reflect the model's *learned* knowledge.

Another approach is to evaluate implementation invariance, ensuring explanations remain stable across functionally equivalent models [Sundararajan et al., 2017]. Practically, this involves evaluating explanations across differently parametrised or even initialised models that produce identical outputs, and assessing the consistency of the explanations.

Consistency This assesses whether explanations remain stable for similar or near-identical inputs [Montavon et al., 2018]. High consistency is generally desirable, as inconsistent explanations can erode user trust. Evaluation approaches often involve measuring the similarity (e.g., using Jaccard similarity or rank correlation) of explanations for perturbed versions of the same input or for inputs within a local neighbourhood [Alvarez-Melis and Jaakkola, 2018a]. In some domains, such as vision, consistency is also measured via metrics such as Intersection over Union (IoU) for saliency maps generated for similar images [Hooker et al., 2019].

Completeness This refers to whether the explanation fully captures the factors contributing to the model's prediction [Samek et al., 2016, Sundararajan et al., 2017, Yeh et al., 2020]. For instance, for concept-based methods, completeness can be assessed by training a model to predict the original model's output using only the concept activations and reporting the fidelity of this approximation [Yeh et al., 2020]. For feature attribution methods, completeness can be tested by verifying that the sum of feature attributions matches the change in the model output relative to a baseline [Sundararajan et al., 2017].

Parsimonious Explanations should be as concise as possible, relating to the desideratum of conciseness. Parsimony can be estimated by the model size [Guidotti et al., 2018, Molnar et al., 2020], explanation complexity (e.g., number of features and operations) [Slack et al., 2019, Molnar et al., 2020] or sparsity metrics. In some settings, explanations can be evaluated along a complexity-fidelity trade-off curve, where one measures fidelity at different levels of explanation complexity [Lakkaraju et al., 2016].

Human Evaluation In some cases, human evaluation may be appropriate [Lakkaraju et al., 2016, Doshi-Velez and Kim, 2017, Ribeiro et al., 2016, Lage et al., 2018, Narayanan et al., 2018, Poursabzi-Sangdeh et al., 2021]. Doshi-Velez and Kim [2017] introduce the following evaluation taxonomy for when to include human evaluation:

1. Application level: the end user should test the explanation.
2. Human level: the explanation should be evaluated by humans – these humans do not need to be experts, but may be laymen.
3. Function level: the purpose of the explanation is to understand the model behaviour. Therefore, this level can be done without human evaluation.

Summary and Practical Considerations In practice, robust evaluation of interpretability methods benefits from combining multiple protocols to assess both the functional fidelity and user-facing effectiveness of explanations. This could be a combination of synthetic benchmarks, perturbation-based tests, and axiomatic evaluations to verify correctness and robustness. Human-centred evaluations, ranging from controlled user studies to real-world deployment trials, are important for uses that exceed the functional-level analysis.

2.7 Problems in Interpretability

There are several open problems with interpretability methods. For example, as highlighted by Leavitt and Morcos [2020]:

> *While these methods may be useful for building intuition, they can also encourage three potentially misleading assumptions: that the visualization is representative of the neuron's behaviour; that the neuron is responsible*

> *for a clearly delineated portion of the task or the network's behaviour; and that the neuron's behaviour is representative of the network's behaviour.*

This highlights core concerns around the faithfulness, sufficiency, and relevance of explanations. Specifically, neuron-based explanations may not accurately reflect the model's internal mechanisms, nor can they ensure that the identified neurons contribute directly or exclusively to the model's predictions.

Below, we summarise some of the known problems in the field of interpretability.

Lack of Faithfulness Explanations may not be faithful to the underlying model. Adebayo et al. [2018] show that saliency maps can produce similar visualisations even when applied to untrained models, suggesting that these methods may not reflect the true model logic or data-generating process. Similarly, Bilodeau et al. [2024], Geirhos et al. [2023] show that feature visualisations are processed very differently from natural images and do not explain how natural input is processed. Moreover, they show that feature visualisation can only be guaranteed to be reliable if we already know how the model internals work.

Explanation is not used by the Model Interpretability methods may highlight patterns present in the model, but this does not ensure that the model uses these patterns when making predictions. This is highlighted in the following excerpt:

> *Units similar to those [hand-picked units] may be the exception rather than the rule, and it is unclear whether they are essential to the functionality of the network. For example, meaningful selectivities could reside in linear combinations of units rather than in single units, with weak distributed activities encoding essential information.* [Kriegeskorte, 2015]

This has been shown in practice. For example, Wiegreffe and Pinter [2019] show that attention mechanisms present in a model may be unused. Furthermore, Karimi et al. [2022] use causal inference to estimate the treatment effect of explanations (E) on predictions (Y). They show that the relationship between the explanation and predictions is often weak, and the gap is worse in high-performing models.

Limited Ability to Detect Model Failures Interpretability methods cannot identify all model problems. Adebayo et al. [2020] investigate whether post-hoc methods can be used for debugging and find that they are ineffective for detecting mislabelled training examples. Moreover, Adebayo et al. [2021] show that post-hoc methods may fail to identify spurious correlations.

Lack of Stability and Susceptibility to Manipulation Interpretability methods can be unstable. Kindermans et al. [2019] show that attribution methods can produce different explanations when the input is shifted, even if the function and gradients remain unchanged. Ramaswamy et al. [2022a] showed that using different probe datasets to interpret the same model can lead to different explanations for the same concept. Similarly, Soni et al. [2020] found that models can be sensitive to the seed used for probe datasets. In addition, interpretability methods can be deceived. This includes fairwashing, where explanations are used to mask model biases [Aïvodji et al., 2019, Anders et al., 2020]; data poisoning, where the training data is manipulated to influence explanations [Goldblum et al., 2022, Baniecki et al., 2022]; and adversarial attacks targeting the model [Sabour et al., 2015, Heo et al., 2019] or the explanation itself [Ghorbani et al., 2019a, Slack et al., 2020].

Limited Scope Most interpretability methods explain only a part of the model. For example, prototype methods focus on explaining the input, whereas concept-based methods tend to focus on explaining the representation space. Moreover, even when explaining one part of the model, it can still differ in the type of explanation. For example, saliency maps only show 'where' the model focused in the image, and not 'what' it focused on [Achtibat et al., 2022, Colin et al., 2022].

Incorrect Assumptions Most interpretability methods will make some assumptions – the form of the explanation, the part of the model that is explained or the model behaviour more generally. One example is that features or concepts are sometimes assumed to have a linear representation. However, this assumption has limitations [Chen et al., 2020b, Soni et al., 2020]. Mahinpei et al. [2021] challenge the assumption that concept vectors align with human mental models, highlighting a potential misalignment between explanation representations and user expectations.

Interpretability is often defined in task- or domain-specific ways, which introduces several problems. First, different desiderata may conflict, leading to trade-offs between evaluation metrics [Tsipras et al., 2018, Tomsett et al., 2020]. Second, methods may overfit to the chosen metrics rather than capturing meaningful explanatory properties. Doshi-Velez and Kim [2017] warn that by explicitly defining what we want from an explanation, we risk encouraging metric hacking, where methods optimise for specific metrics without genuinely improving interpretability.

Human Understandable Explanations generated by interpretability methods may not be understandable or useful to humans. In some cases, they may be misleading if they appear plausible while failing to reflect the true model behaviour [Ribeiro et al., 2016, Adebayo et al., 2018]. Users may overgeneralise from local explanations to global model behaviour [Ribeiro et al., 2016], or overtrust models regardless of explanation quality [Poursabzi-Sangdeh et al., 2021]. Moreover, the format and presentation of explanations play a key role in their effectiveness. Lage et al. [2018] and Lakkaraju et al. [2016] show that different explanation formats, such as rules or prototypes, vary in their usefulness depending on the task, user expertise, and presentation, and that explanations may confuse or hinder users if not carefully designed. These findings suggest that explanations need to be evaluated not only for technical correctness, but also for their ability to support human reasoning and decision-making.

Part II

Knowledge Encoding

3 Examining Assumptions about Concept Vectors

One of the core questions of this thesis is *to understand what knowledge is encoded in machine learning models and how it is represented*. As neural networks are highly expressive, the way in which features are encoded could take many different forms. This raises the question: *how* are feature represented in a neural network?

In this thesis, we assume that features are encoded linearly in the latent space. This chapter examines this assumption, exploring its underlying premises and the implications for interpreting model representations. This analysis lays the foundation for subsequent chapters, where we leverage these insights to extract and evaluate the knowledge that models have learned.

3.1 Introduction

Neural networks are highly expressive; they can approximate any function given enough capacity [Cybenko, 1989]. This raises an important question: how are features represented in a model? From an interpretability perspective, it would be simple if each neuron encoded a feature. Then, we could simply read off the meaning of each neuron to understand the features in a network. However, this is likely not the case. Recent work suggests that models often represent more features than they have neurons by combining features into the same dimensions–a phenomenon known as superposition [Elhage et al., 2022b].

A common assumption that we adopt in this thesis is the linear representation hypothesis, which assumes that features are encoded linearly in the latent space. In this view, a concept corresponds to a direction: a vector in the model's representation space. While this assumption is simple, it has proven effective in practice [Kim et al., 2018, Conneau et al., 2018, Tenney et al., 2019, McGrath et al., 2022a, Nanda, 2023].

One way to think about the linear representation hypothesis is to assume that it is possible to find a *concept vector* – a direction in the model's representation space that corresponds to a concept c. While we formally introduce concept vectors in Section 3.2, we refer to them informally here to explain the key properties we analyse. We use `this font` to denote a concept.

Since the linear representation hypothesis is central to several methods used in this thesis, it is important to understand how these assumptions can influence the conclusions drawn about whether a model encodes and uses a concept. We leverage the insights from this analysis in later chapters of the thesis.

The three properties we focus on are:

Polysemanticity If we find a concept vector v_c that represents a concept c, we can ask: to what extent does v_c only capture a concept c? For example, we can train a concept vector to represent the concepts `striped` or `red`. In doing so, we implicitly assume that the label `striped` accurately describes the information encoded by the $v_{striped}$. However, in practice, the concept vector may capture several concepts. Continuing the previous example, $v_{striped}$ might encode both concepts `striped` and `red`. We refer to this phenomenon as *concept entanglement* – a specific version of polysemanticity, where a single concept vector represents multiple related concepts. This issue is explored further in Chapter 4, where we analyse the entanglement of language and concepts.

Local vs Global Concepts If we find a concept vector v_c to represent a concept c, does it encode a local or a global version of that concept? For example, consider the colour blue in an image. Does the concept vector respond to blue if it appears anywhere in the image? Everywhere in the image? Or only when it is present in a specific part of the image – such as the top or bottom? We refer to this property as *spatial dependence*. The property of spatial dependence is leveraged in Chapter 6.2.

While the first two properties focus on the meaning encoded in the concept vector, our final property examines the influence of a concept on the model's behaviour.

Influence of a Concept in Different Layers We can study the effect of a concept by intervening directly on a model's hidden state. For example, we might amplify a concept by perturbing the latent representation $z_l(x)$ of an input x in the direction of a concept vector $v_{c,l}$

$$z_l(x) \leftarrow z_l(x) + \gamma v_{c,l}, \tag{3.1}$$

where $\gamma \in \mathbb{R}^+$ is an non-negative scalar. This intervention is used in interpretability research for various purposes, such as verifying that a concept influences the model's output or steering the model's behaviour at inference time [Subramani et al., 2022, Turner et al., 2023].

However, a concept can often be detected in several different layers of a model. Does it matter which layer we intervene on? More concretely, does amplifying the concept in different layers lead to the same effect on the output? If not, can this lead to different conclusions? We refer to this property as layer consistency. This property is explored further in Chapter 6.2.

Characterising these properties is therefore a crucial first step towards a more nuanced and reliable application of concept-based interpretability methods. We focus on the first concept-based interpretability method: concept activation vectors [Kim et al., 2018]. For each property discussed above, we provide a formal definition and introduce tools to analyse it. We also show how these properties can influence the conclusions drawn from significance testing.

3.2 Background: Concept Activation Vectors

A concept activation vector (CAV) $v_{c,l}$ is a vector that represents a concept c in the latent space of a layer l of a neural network (NN) [Kim et al., 2018]. For now, we assume binary concepts and denote the presence of a concept c in an input x by $c(x) = 1$ and its absence by $c(x) = 0$. Given a concept c, we can divide a general set of inputs $\mathbb{X}$ into two subsets: $\mathbb{X}^+$, where the concept is present, and $\mathbb{X}^-$, where it is absent:

$$\mathbb{X}^+ = \{x \in \mathbb{X} : c(x) = 1\}$$
$$\mathbb{X}^- = \{x \in \mathbb{X} : c(x) = 0\}.$$

The negative samples are often randomly sampled from in-distribution inputs. For the sets $\mathbb{X}^+$ and $\mathbb{X}^-$, we create two corresponding sets of activations in layer l:

$$\mathbb{Z}^+_{c,l} = \{f_l(\mathbf{x}_i)\ \forall \mathbf{x}_i \in \mathbb{X}^+\}, \text{ and } \mathbb{Z}^-_{c,l} = \{f_l(\mathbf{x}_i)\ \forall \mathbf{x}_i \in \mathbb{X}^-\}, \tag{3.2}$$

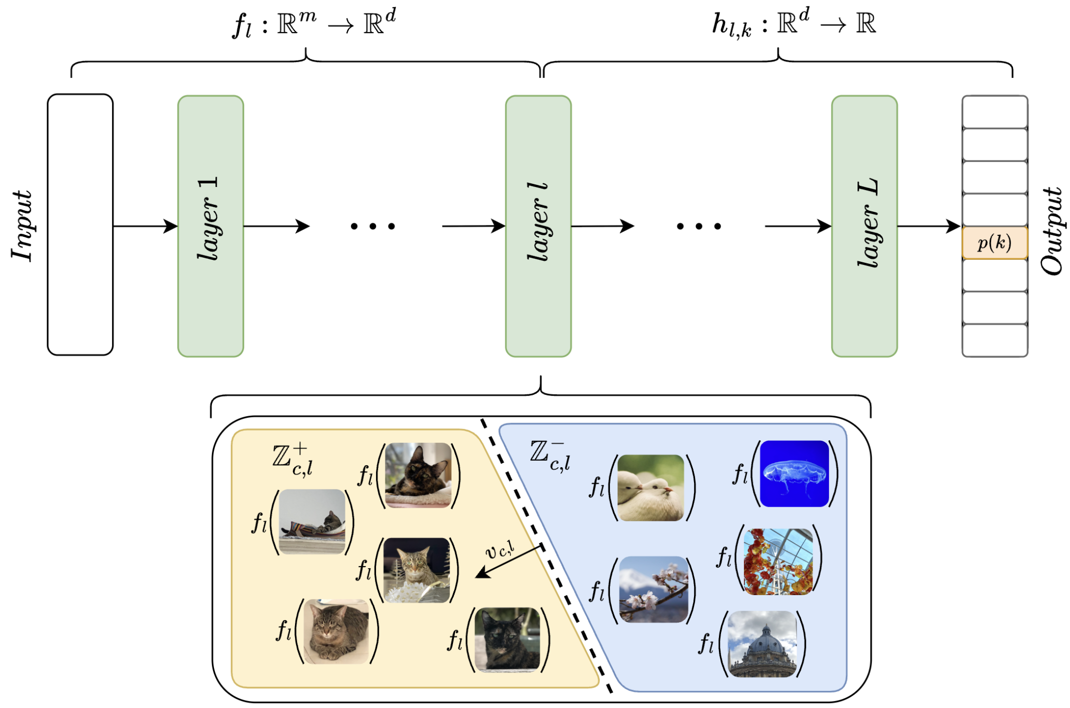


Figure 3.1: Illustration of a Concept Activation Vector (CAV), modified from Kim et al. [2018]. The neural network produces latent representations at each layer $f_l(x)$ for a given input x. For a given concept c (e.g., cat), a CAV $v_{c,l}$ is learned in the latent space of layer l by training a linear classifier to separate the activations of inputs containing the concept ($\mathbb{Z}^+_{c,l}$) from those that do not ($\mathbb{Z}^-_{c,l}$).

where the function $f_l(x)$ generates an activation for layer l given an input x.

We find the CAV $v_{c,l}$ by training a binary linear classifier to distinguish between the two sets $\mathbb{Z}^+_{c,l}$ and $\mathbb{Z}^-_{c,l}$:

$$z^+_{i,l} \cdot v_{c,l} + b_{c,l} > 0 \quad \forall z^+_{i,l} \in \mathbb{Z}^+_{c,l}, \text{ and } z^-_{i,l} \cdot v_{c,l} + b_{c,l} \leq 0 \quad \forall z^-_{i,l} \in \mathbb{Z}^-_{c,l}, \tag{3.3}$$

where $v_{c,l}$ is the normal vector of the hyperplane separating the activations $\mathbb{Z}^+_{c,l}$ and $\mathbb{Z}^-_{c,l}$, and $b_{c,l}$ is the intercept.[1]

To analyse the model's sensitivity to $v_{c,l}$, Kim et al. [2018] introduce sensitivity testing – testing with CAVs (TCAV) – which measures the model's sensitivity to a concept when predicting a class k:

$$\mathrm{TCAV}_{c,k,l} = \frac{|\{x \in \mathbb{X}_k : S_{c,k,l}(x) > 0\}|}{|\mathbb{X}_k|}, \tag{3.4}$$

where $\mathbb{X}_k$ is a set of inputs belonging to class k. The directional derivative of

[1]In this formulation, we make the simplifying assumption that the linear classifier in Eq. 3.3 has hard boundaries.

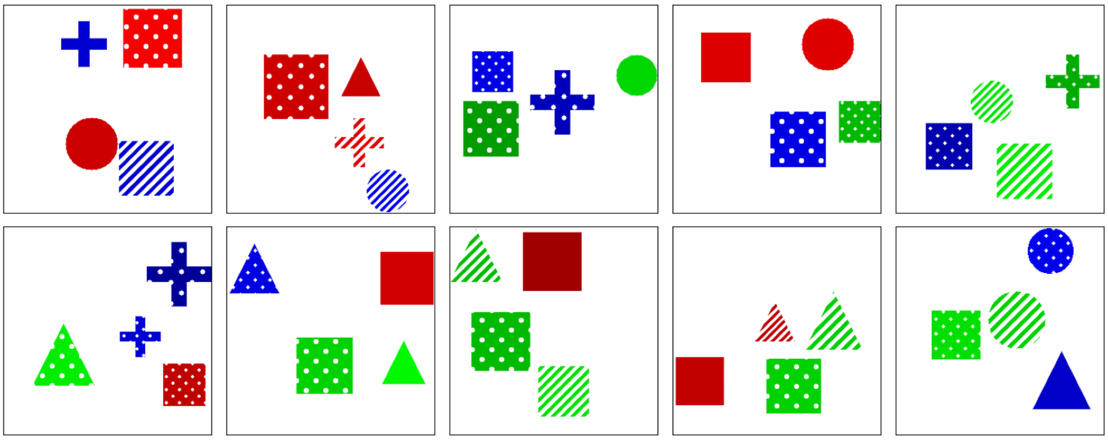


Figure 3.2: Example images from the Elements dataset. Each square shows a different input. Each input contains multiple elements – where an element is a shape with a specific colour, brightness, size, shape, pattern and location. Classes in the dataset are mapped to a specific combinations of shape, colour, and texture.

the concept, $S_{c,k,l}$, is defined as

$$S_{c,k,l}(x) = \lim_{\epsilon \to 0} \frac{h_{l,k}(f_l(x) + \epsilon v_{c,l}) - h_{l,k}(f_l(x))}{\epsilon} \tag{3.5}$$

$$= \nabla h_{l,k}\left(f_l(x)\right) \cdot v_{c,l} \tag{3.6}$$

where $h_{l,k}$ pushforward operation from the activations in layer l to the output class k, and $\nabla h_{l,k}$ is the partial derivative of the model output for class k with respect to the latent representation $f_l(x)$. The TCAV score measures the fraction of class k inputs whose intermediate representation at layer l is positively influenced by the concept vector $v_{c,l}$. To test whether the score is statistically significant, it is compared to the scores of random vectors (see Appendix B.3.1).

3.3 Elements: Configurable Synthetic Dataset

To explore these properties, we introduce a new synthetic dataset: *Elements*. Using a synthetic dataset gives us full control over both the training data and the class definitions. This allows us to directly influence model properties, such as the correlation between concepts in the latent space, and to design probe datasets to test specific concept vector properties, such as spatial dependence.

Inputs In the Elements dataset, each input is an image that contains one or more elements, as shown in 3.2. Each element within an image is defined by its colour, brightness, size, shape, pattern, and location (given by coordinates in the image).

The dataset can be configured by specifying which combinations of these properties are permitted for each element. Detailed descriptions of the property ranges and configurations used in this chapter are provided in Appendix B.1.

Labels Each class in the dataset is mapped to a specific combination of shape, colour, and texture. For example, class 1 may correspond to images with green striped circles. This design gives us access to the ground-truth relationship between each concept and class. To ensure the model must learn these ground-truth relationships, we include a class for every possible combination of concepts. As a result, achieving high accuracy requires the model to develop explicit internal representations of the concepts. This setup allows us to directly analyse the faithfulness of concept-based explanation methods by comparing their outputs to the model's actual use of concepts in making predictions.

Properties The configurable design of the Elements dataset allows us to systematically investigate various aspects of model behaviour and concept activation vectors (CAVs). To study entanglement, for example, we can introduce controlled correlations between concepts by requiring that all red elements in the training set are also striped, thereby linking the `red` and `striped`. To explore spatial dependence, we can construct probe datasets in which all red elements appear only on the right side of the image. This flexibility lets us design targeted experiments to isolate and analyse specific properties of concept representations. An example of such a probe dataset is shown in Figure 3.5.

Other Datasets While several datasets have been introduced for evaluating interpretability methods, they differ from ours in a few key ways. There are three questions we need our dataset to help answer:

1. Is the concept represented in the network?
2. Is the concept used for the network's prediction?
3. How does the network represent correlated concepts?

Existing datasets either do not allow insight into all three or are unsuitable due to other practical reasons (such as runtime).

3.4 Polysemanticity: Concept Entanglement

It is often assumed that a concept vector captures a single, distinct concept–that is, the label in the concept dataset fully describes what the concept vector represents. In practice, however, this assumption may not hold. If a concept is not linearly separable from others in the model's internal representation space, the learned concept vector may capture multiple, overlapping concepts rather than isolating the intended one. In some cases, this may even be desirable, as certain concepts are naturally associated. For example, the concepts of `blue` and `sky` are inherently linked, as a defining characteristic of the sky is that it is often blue.

However, when concepts are entangled, isolating the model's sensitivity to a single concept becomes difficult. As a result, when we compute a TCAV score for concept c_1, we may inadvertently measure the model's sensitivity to another, entangled concept c_2 as well – we elaborate on this later.

In this section, we describe how concept entanglement can be discovered using CAVs and discuss its implications for TCAV analysis. We begin by formalising entanglement:

Definition 3.4.1 (Concept Entanglement)**.** A concept vector $v_{c_1,l}$ for concept c_1 is entangled with concept c_2 if it 'activates' for inputs containing the concept c_2, $\mathbb{Z}_l^+$ and does not activate for inputs corresponding to $\mathbb{Z}_l^-$.

Continuing the example above, this means that a concept vector trained to detect the sky may also activate for images that contain the colour blue but do not include a sky. Importantly, this definition is not bidirectional: if $v_{c_1,l}$ is entangled with c_2, it does not imply that a concept vector for c_2 is entangled with c_1.

Next, we derive this means for different implementations of concept vectors.

Mathematical Formulation and Constraints Concept vectors are typically trained using a linear classifier. While the original implementation used support vector machines (SVMs), the follow-up work often uses a logistic regression [Kim et al., 2018, McGrath et al., 2022a]. In SVMs, the objective is to find a hyperplane that separates the concept exemplars $\mathbb{X}_{c_1}^+$ from a set of negative samples $\mathbb{X}_{c_1}^-$:

$$g_{c_1} = \begin{cases} v_{c_1,l} z_{i,l}^+ - b_1 = +1, & z_{i,l}^+ \in \mathbb{Z}_{c_1}^+ \\ v_{c_1,l} z_{i,l}^- - b_1 = -1, & z_{i,l}^- \in \mathbb{Z}_{c_1}^- \end{cases}$$

for the points on the margin. For all datapoints, it holds that:

$$g_{c_1} = \begin{cases} v_{c_1,l} z^+_{i,l} - b_1 \geq 1, & z^+_{i,l} \in \mathbb{Z}^+_{c_1} \\ v_{c_1,l} z^-_{i,l} - b_1 \leq -1, & z^-_{i,l} \in \mathbb{Z}^-_{c_1} \end{cases}$$

If a concept is entangled, then the same vector $v_{c_1,l}$ also separates the concept exemplars for a different concept:

$$v_{c_1,l} z^+_{i,l} - b_2 \geq +1, \quad z^+_{i,l} \in \mathbb{Z}^+_{c_2} \tag{3.7}$$

$$v_{c_1,l} z^-_{i,l} - b_2 \leq -1, \quad z^-_{i,l} \in \mathbb{Z}^-_{c_2} \tag{3.8}$$

Another common implementation for finding concept vectors is a logistic regression:

$$g_{c_1} = \sigma(v_{c_1,l} z_{i,l} + b_1), \quad z_{i,l} \in (\mathbb{Z}^+_{c_1} \cup \mathbb{Z}^-_{c_1}), \tag{3.9}$$

where σ is the sigmoid function. Assuming a classification threshold of 0.5, entanglement with concept c_2 implies:

$$v_{c_1,l} z^+_{i,l} - b_2 > \ln(0.5), \quad z^+_{i,l} \in \mathbb{Z}^+_{c_2} \tag{3.10}$$

$$v_{c_1,l} z^-_{i,l} - b_2 \leq \ln(0.5), \quad z^-_{i,l} \in \mathbb{Z}^-_{c_2} \tag{3.11}$$

In both cases, the threshold b_2 is not part of the concept vector, and the constants (e.g., $\ln(0.5)$ or ± 1) are arbitrary. Therefore, we can simplify the entanglement condition by removing the bias and focusing on the dot product:

$$z^+_{c_2,l} \cdot v_{c_1,l} > z^-_{c_2,l} \cdot v_{c_1,l} \tag{3.12}$$

We refer to this condition as soft entanglement:

Definition 3.4.2 (Soft Concept Entanglement)**.** A concept activation vector $v_{c_1,l}$ found for concept c_1 is entangled with concept c_2 iff

$$z^+_{c_2,l} \cdot v_{c_1,l} > z^-_{c_2,l} \cdot v_{c_1,l} \qquad \forall z^+_{c_2,l} \in \mathbb{Z}^+_{c_2,l}, z^-_{c_2,l} \in \mathbb{Z}^-_{c_2,l} \tag{3.13}$$

If the above equation holds, then the concept vector is entangled with concept c_2. Having derived the constraints for SVMs and logistic regression, we now distinguish between two main sources of entanglement.

Sources of Entanglement There are two main causes of concept entanglement. One source of entanglement is that the two concepts are not linearly separable in the model's representation space. The second arises from an underspecified objective for learning concept vectors.

Even when the latent space has dimensionality d, and we have more than d datapoints, this may not be sufficient to uniquely determine a concept vector.[2] One reason is that the effective dimensionality of the data – i.e., the span s of the latent space estimated from the training data – is often much smaller than d: i.e., $s \ll d$. As a result, there are extra degrees of freedom when fitting a concept vector. This means that if a concept is encoded in the model, multiple distinct vectors may satisfy the objective.

Consequently, we may learn a concept vector $v_{c_2,l}$ that is entangled with another vector $v_{c_1,l}$. In such cases, we can attempt to disentangle the two by refining the objective by augmenting the negative probe set for c_2 with examples of c_1. If disentangling is possible, we obtain a new concept vector that performs equally well while excluding c_1. If disentangling fails, we can conclude that the two concepts are not linearly separable in that layer.

At a high level, the design of the negative set acts as a tool to refine a concept vector by ruling out specific confounding variables. Next, we examine the implications of entanglement.

Implications for Prototypes If we assume that the magnitudes of the latent representations are approximately equal:

$$||z^+_{c_2,l}|| \approx ||z^-_{c_2,l}|| \tag{3.14}$$

then soft entanglement implies that the positive examples of c_2 have higher cosine similarity with $v_{c_1,l}$ than the negative examples do.

In Kim et al. [2018], cosine similarity is used to rank inputs by how strongly they activate a concept vector. Under the soft entanglement condition, if all representations have similar norms, then inputs containing c_2 will rank higher than those without c_2, even though the vector $v_{c_1,l}$ was not explicitly trained to detect c_2.

[2]In practice, gathering high-quality concept data is often challenging.

Implications for TCAV Concept entanglement has implications for how we interpret TCAV scores. Let $v_{c_2,l}$ be a concept vector for concept c_2. We can decompose it into components that are dependent and independent of $v_{c_1,l}$:

$$v_{c_2,l} = v_{c_2,l}^{||c_1} + v_{c_2,l}^{\perp c_2}, \tag{3.15}$$

where $v_{c_2,l}^{||c_1}$ is the component that is linearly dependent on c_1:

$$v_{c_2,l}^{||c_1} = \frac{v_{c_2,l} \cdot v_{c_1,l}}{||v_{c_2,l}||} v_{c_2,l}. \tag{3.16}$$

We can rewrite this using the cosine similarity:

$$v_{c_2,l}^{||c_1} = \cos\mathrm{sim}(v_{c_1,l}, v_{c_2,l}) \frac{||v_{c_1,l}||}{||v_{c_2,l}||} v_{c_2,l}. \tag{3.17}$$

Assuming unit-length concept vectors, this simplifies to:

$$v_{c_2,l}^{||c_1} = \cos\mathrm{sim}(v_{c_1,l}, v_{c_2,l}) v_{c_2,l}. \tag{3.18}$$

The component that is independent of c_1 is defined as

$$v_{c_2,l}^{\perp c_1} = v_{c_2,l} - v_{c_2,l}^{||c_1} \tag{3.19}$$

This decomposition lets us analyse how entanglement may affect TCAV scores. Let the TCAV score of $v_{c_1,l}$ be α, i.e.:

$$\alpha = \mathrm{TCAV}_{c_1,l} = \frac{|\{x \in \mathbb{X}_k : \nabla h_{l,k}(f_l(x)) \cdot v_{c,l} > 0\}|}{|\mathbb{X}_k|} \tag{3.20}$$

Then the TCAV score of $v_{c_2,l}$ can be written as:

$$\begin{aligned}
\mathrm{TCAV}_{c_2,l} &= \frac{|\{x \in \mathbb{X}_k : \nabla h_{l,k}(f_l(x)) \cdot v_{c,l} > 0\}|}{|\mathbb{X}_k|} && (3.21)\\
&= \frac{\left|\left\{x \in \mathbb{X}_k : \nabla h_{l,k}(f_l(x)) \cdot (v_{c_2,l}^{||c_1} + v_{c_2,l}^{\perp c_2}) > 0\right\}\right|}{|\mathbb{X}_k|} && (3.22)\\
&= \cos\mathrm{sim}(v_{c_1,l}, v_{c_2,l})\alpha + \frac{\left|\left\{x \in \mathbb{X}_k : \nabla h_{l,k}(f_l(x)) \cdot v_{c_2,l}^{\perp c_2} > 0\right\}\right|}{|\mathbb{X}_k|} && (3.23)
\end{aligned}$$

where the first term captures the contribution of the entangled concept vector $v_{c_1,l}$ to the TCAV score of $v_{c_2,l}$.

Borrowing Pearl [2009]'s framework, the TCAV score measures the total effect of $v_{c_2,l}$ when predicting class k. The component that routes through $v_{c_1,l}$ measures the indirect effect, and the component caused by $v_{c_2,l}^{\perp c_2}$ measures the direct effect. In practice, these effects may be further confounded by other variables, but this framing provides an intuition for the impact of entanglement on significance testing.

Toy Experiments

In this section, we empirically investigate the impact of concept entanglement on TCAV scores by designing a controlled toy experiment using the Elements dataset. We analyse two aspects:

- How entanglement between concepts affects their relative orientation in representation space, as measured by cosine similarity between CAVs.
- How entanglement influences the resulting TCAV scores.

We consider three synthetic settings, each varying in the degree of entanglement between the concepts `red` and `triangle` in the training data:

E_1: All combinations of colour, shape, and texture are equally likely.

E_2: Red elements are always triangles, but triangles can appear in other colours.

E_3: Red and triangle always co-occur – every red element is a triangle, and vice versa.

We train a classification model for each setting (see Appendix B.3.1). For each concept, we compute multiple CAVs and report their average. As a baseline, we also calculate the average cosine similarity of each CAV with itself across repeated runs, which serves as a reference for evaluating cross-concept similarity. As noted earlier, concept vectors are often underspecified, and as a result, vectors obtained for the same concept may not have a cosine similarity of 1.

The results are shown in Figure 3.3. In E_1 the concepts are independent by design and we do not observe a significant correlation between the concepts `red` and `triangle` CAVs. In E_2, only triangles are red, but triangles may be other colours. In this setting, we find a mild positive cosine similarity between the `red` and `triangle` CAVs. In E_3, where all triangles are red (and vice-versa), the cosine similarity between the `red` and `triangle` CAVs is similar to the self- cosine similarity of the concepts. These results are consistent with the idea that concept co-occurrence in the training data may induce entanglement between the learned representations. We extend this analysis to ImageNet and observe similar patterns in Appendix B.3.1.

Interestingly, we observe a negative cosine similarity between mutually exclusive concepts. This suggests that the model has encoded the fact that certain concepts cannot co-occur – for example, an object can only have one colour at a time. As a result, the directions of the colour concept vectors are negatively aligned. The presence of `red` reduces the likelihood of `blue` or `green` being present. Consequently, the `red` CAV does not solely represent `red`, but also implicitly encodes `not blue` and `not green`.

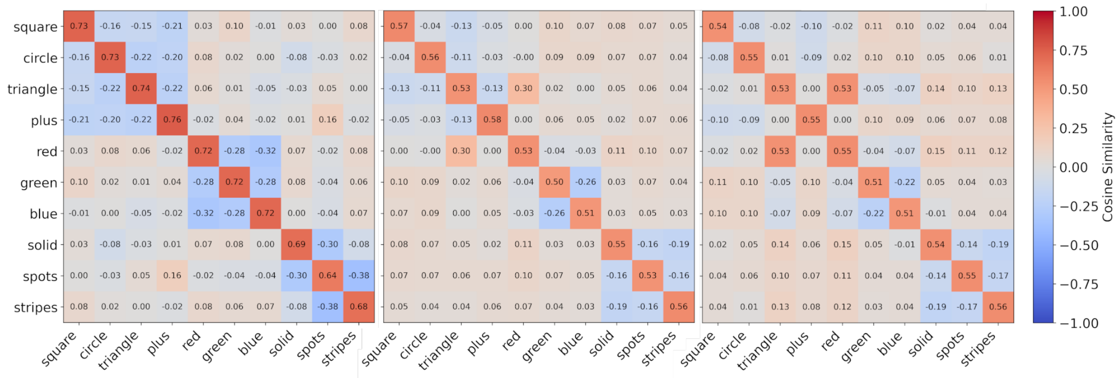


Figure 3.3: Cosine similarities demonstrating entangled concepts. Mean pairwise cosine similarities for all concepts from different versions of the simple Elements dataset, with an increasing association between `red` and `triangle` from left to right: E_1, E_2 and E_3.

Next, we examine how entangled concept vectors can affect TCAV scores. Specifically, we analyse the TCAV scores for the class `striped triangles` in settings E_1 and E_2. Recall that in E_1, all concepts are independent in the training dataset, while in E_2, only triangles are red–though triangles can also appear in other colours. This allows us to use E_2 to assess the influence of concept entanglement on significance testing, with E_1 serving as a control. In both settings, the class label depends solely on the presence of the `stripes` and `triangle` concepts. Accordingly, we expect all other concepts to yield low or insignificant TCAV scores–either because they reduce the likelihood of the class (i.e., negative sensitivity) or are uninformative.[3]

The results are shown in 3.4, with E_1 on the left and E_2 on the right. For E_1, where concepts are not correlated in the dataset, we find that only the `stripes` and `triangle` vectors have a high TCAV score across multiple layers. In contrast, in E_2, the model appears to be sensitive to `red`, `triangle` and `stripes`, with high TCAV scores for each. This occurs even though `red` is not part of the ground-truth definition of the class.

The model obtains a high test accuracy – it classifies 2,374 striped triangle images in the test set of the 5,000 correctly – i.e., around 95%. This makes it unlikely that the model relies on the `red` concept for its prediction. Instead, this suggests that the entanglement of the concepts `red` and `triangle` influences the TCAV scores. In particular, the TCAV score captures the *total* effect, instead of the *direct* effect. This example illustrates that entangled CAVs can lead to explanations that appear meaningful but reflect underlying associations rather than true model behaviour.

[3] Assuming the model uses each concept appropriately.

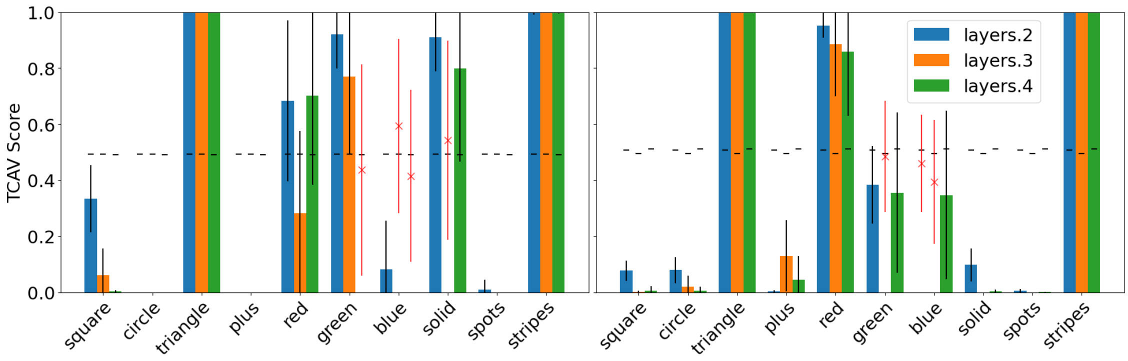


Figure 3.4: Entangled CAVs increase `red` TCAV scores for E_2. TCAV scores for all concepts in E_1 (left) and E_2 (right) for the class of striped triangles. The standard deviation is black or red for significant and insignificant results, respectively. The null for each layer is shown as a horizontal black line.

3.5 Local Concepts: Spatial Dependence

In this section, we focus on the property of spatial dependence. Spatial dependence can arise for various reasons, including architectural choices and biases in the training data. In convolutional neural networks (CNNs), for example, the intermediate representations may retain spatial information, particularly when there is insufficient pooling or data augmentation [Biscione and Bowers, 2021b]. Due to the localised receptive fields of convolutional filters, neuron activations often correspond to specific regions of the input image. As a result, the representation of high-level concepts in CNNs may depend not only on *whether* a feature is present, but also on *where* it appears in the input.

For example, a feature detector sensitive to *wheels* may respond strongly when the pattern appears in the lower part of an image, where wheels are commonly found on vehicles, but not when the same pattern appears at the top. We refer to this phenomenon as *spatial dependence*. Despite this, concept vectors typically assume spatial invariance, implicitly treating concepts as position-independent, ignoring where the features appears. As we will show later in this section, this assumption can be misleading, as it may cause us to overlook concepts that are encoded in a location-specific manner.

In this section, we formalise spatial dependence and explore its implications for CAVs and TCAV scores. We define two types of spatial dependence: in the activations and in the concept vectors. Let $\mathbb{X}_{c,\mu_1}$ and $\mathbb{X}_{c,\mu_2}$ be two probe datasets that contain the same concept c, but in different locations $\mu_1 \neq \mu_2$. For example, $\mathbb{X}_{c,\mu_1}$ contains `striped` objects on the left of the image, while $\mathbb{X}_{c,\mu_2}$ contains `striped` objects on the right (see Figure 3.5 for an example).

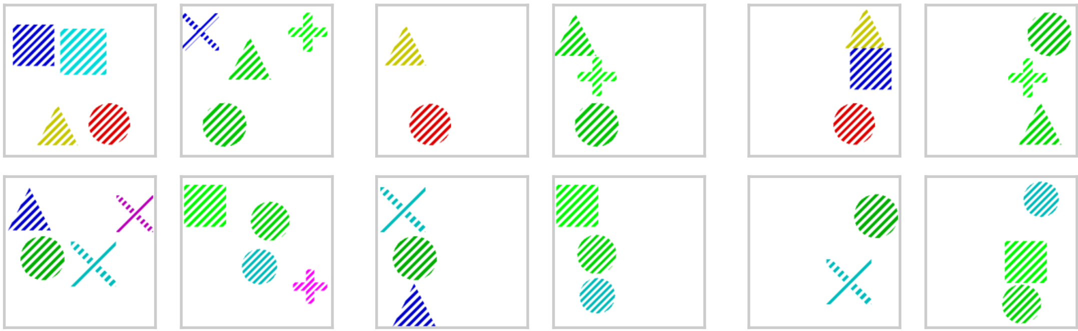


Figure 3.5: Example of the probe dataset for the concept striped. The left show the probe dataset for $\mathbb{X}_{stripes}$, the middles shows the probe dataset for $\mathbb{X}_{stripes\ left}$ and the right shows the probe dataset for $\mathbb{X}_{stripes\ right}$.

Definition 3.5.1 (activation spatial dependence)**.** Let $z_{i,l}$ be the activations corresponding to input x_i in layer l, and let $\mu_{i,c}$ be the location of concept c in x_i. A layer has a spatially dependent representation of a concept iff there exists some function ϕ which maps $z_{i,l}$ to $\mu_{i,c}$ for all inputs x_i:

$$\exists\phi : \forall x_i \in \mathbb{X}_c^+, \phi(h_{l,i}) = \mu_{c,i} \tag{3.24}$$

Activation spatial dependence can arise from the design of the architecture (e.g., convolutional filters), the dataset, or training procedures. If activations are spatially dependent and the probe dataset is location-specific, the resulting CAV may also exhibit spatial dependence.

Definition 3.5.2 (concept vector spatial dependence)**.** A concept vector $v_{c,l}$ is spatially dependent with respect to the locations μ_1 and μ_2 iff

$$z^+_{c,l,\mu_1} \cdot v_{c,l} > z^+_{c,l,\mu_2} \cdot v_{c,l} \quad \forall z^+_{l,\mu_1} \in \mathbb{Z}^+_{l,\mu_1}, z^+_{l,\mu_2} \in \mathbb{Z}^+_{l,\mu_2}, \tag{3.25}$$

where $\mathbb{Z}_{c,l,\mu_1}$ and $\mathbb{Z}_{c,l,\mu_2}$ are latent representations in layer l corresponding to the probe datasets $\mathbb{X}_{c,\mu_1}$ and $\mathbb{X}_{c,\mu_2}$. This means the concept vector is more similar to representations of the concept in one location than another. In such cases, the CAV encodes not just the concept, but the concept at a specific location (e.g., "striped on the right" instead of just "striped"). As we will show later, we can use concept vector spatial dependence to detect activation spatial dependence.

Spatial Dependence Metric

We propose a simple metric to determine whether a concept vector encodes a spatial dependence. In the case of a convolutional neural network, where the activations are of shape $H \times W \times D$, we can reshape the CAVs back into the original shape of the activations, and compute the channel-wise norm as follows:

$$\mathbf{S}_{c,l} = \|\text{reshape}(v_{c,l}, (H, W, D))\|_2, \tag{3.26}$$

where $\mathbf{S}_{c,l} \in \mathbb{R}^{H \times W}$, and $\| \cdot \|_2$ is the L_2 norm across the channel dimension. We refer to this array as the spatial norms of the CAV.

If a CAV's spatial norm varies substantially across the (H, W) dimensions, it indicates that the CAV is spatially dependent (see Appendix B.6.2 for an explanation). Visualising the spatial norm shows us which regions contribute most to the directional derivative and, consequently, to the TCAV score.

Toy Experiments

We empirically investigate spatial dependence in concept vectors by examining two key aspects:

- whether spatial dependence in the probe dataset leads to spatial dependence in the corresponding concept vectors; and
- how spatial dependence affects significance testing.

To induce spatial dependence in concept activation vectors (CAVs), we constructed spatially dependent probe datasets for both Elements and ImageNet. In these datasets, we restricted the concepts to specific regions of the image or by masked out parts of the input image. An example is shown in Figure 3.2. In the example, the concept of stripes is presented under three conditions:

- unrestricted across the entire image (left),
- restricted to the left side of the image (middle), and
- restricted to the right side of the image (right).

Further details on dataset can be found in the Appendix B.3.1.

To address our first research question–whether spatial dependence in the probe dataset leads to spatial dependence in the corresponding concept vectors. Specifically, we measured the spatial norms of the resulting CAVs and present the results in Figure 3.6. When using spatially independent probe datasets (top row of Figure 3.6), the spatial norms appear uniform, indicating that the CAVs themselves do not exhibit spatial dependence.[4] In contrast, the second and third rows, corresponding to spatially dependent probe datasets, reveal clear spatial dependence in the CAVs. Bright regions highlight areas where the gradients contribute significantly to the

[4]While individual CAVs may still display some degree of spatial dependence, this effect tends to cancel out when aggregated across training runs. See Appendix B.6.3 for details.

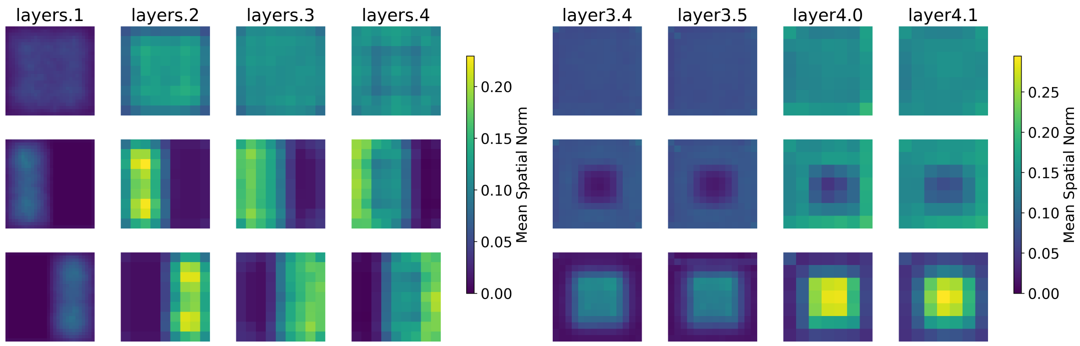


Figure 3.6: Spatial norms reflect the spatial dependence of the probe dataset. Left: Mean spatial norms for `red` (top), `red left` (middle) and `red right` (bottom) for Elements. Right: Mean spatial norms across for `striped` (top), `striped edges` (middle) and `striped middle` (bottom) for ImageNet.

TCAV score, while areas with near-zero spatial norms indicate regions of the image that do not contribute meaningfully to the directional derivative or the resulting TCAV score.

We next turn to our second research question: *Does the concept's location in the input image influence conclusions draw about the model's conceptual sensitivity?* Taking a step back, we can show that a model is not translation invariant if:

1. The model has activation spatial dependence, i.e. concepts in different input locations leads to (predictably) different activation patterns in a given layer.

2. Each depth-wise slice of the activations, of shape $(1, 1, D)$, affects the logit output differently.

Both of these components influence the TCAV score with (1) influencing the concept vector $v_{c,l}$ and (2) influencing the partial derivative $\nabla h_{l,k}\left(f_l(x)\right)$. The results in Figure 3.6 address part (1) – these results show that the model has activation spatial dependence as the locations with the highest spatial norms approximately correspond to the location of the concept in the image space.

To address (2), we computed TCAV scores using different sets of spatially dependent CAVs to assess whether the model's sensitivity to a concept varies with its spatial location in the input. To investigate this, we created spatially dependent classes in the Elements dataset, where the class depends on what concepts are present *and* where they are in the image. 3.7 shows the results for the classes 'striped triangles on the left' (left) and 'striped triangles on the right' classes.

The TCAV scores for `stripes`, `triangle`, `stripes left` and `triangle left` are significant, indicating a positive influence of these concepts on the class. The

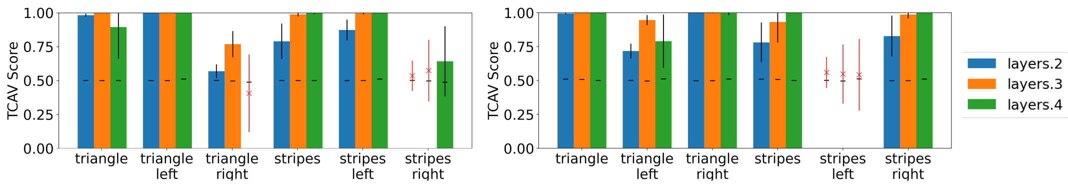


Figure 3.7: TCAV scores are spatially dependent for the 'striped triangles on the left' (left) and 'striped triangles on the right' (right) classes in Elements. The standard deviation is black or red for significant and insignificant results, respectively. The null for each layer is shown as a horizontal black line.

difference between the left- and right-biased TCAV scores highlights that the model's conceptual sensitivity is location-dependent.

Overall, these findings demonstrate that CAVs can be effectively used to detect spatial dependence in models. Further examples for ImageNet are provided in the Appendix B.3.1. Additionally, we present preliminary evidence suggesting that spatially dependent CAVs can also be found in transformer-based architectures (see Appendix B.3.1).

3.6 Consistency of Concept Interventions across Layers

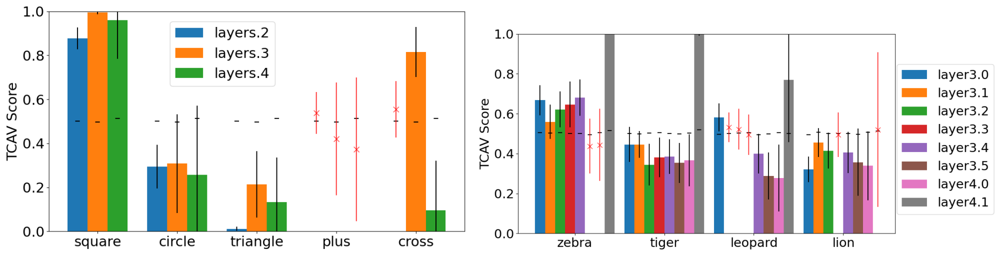
There has been a recent thread of work on intervening on concepts or features in a model's latent space [Turner et al., 2023]. The idea is simple: we amplify a concept by perturbing the hidden state in the direction of a concept vector. This is used in interpretability for two main purposes: (1) to test whether a concept influences the model's output, and (2) to steer model behaviour at inference time.

Often, we can detect a concept in several different layers of the model. Suppose we find a representation of concept c in both layers l_1 and l_2. The question we ask is: does it matter which layer we intervene on? More concretely, does amplifying the concept in different layers lead to the same effect on the model's output?

We formalise this as follows:

Definition 3.6.1 (Layer Consistency)**.** $g_{l_1,l_2}(\cdot)$ maps the latent representations from layer l_1 into activations in layer l_2, where $l_1 < l_2$. Two concept vectors, v_{c,l_1} and v_{c,l_2} are consistent across layers iff for every input x with corresponding latent representations z_{l_1} and z_{l_2}, $g(z_{l_1} + v_{c,l_1}) = z_{l_2} + v_{c,l_2}$.

It follows from the definition that if two CAVs are consistent across layers, then they have the same downstream effect on the model when activations are perturbed in their direction. Our first question is:



(a) Inconsistent TCAV scores across layers. Left: TCAV scores for shape concepts for the ‘solid red squares’ class in Elements. Right: TCAV scores for `striped` for a subset of ImageNet classes.

Figure 3.8: Consistency can affect TCAV scores. The standard deviation is black or red for significant and insignificant results, respectively. The null for each layer is shown as a horizontal black line.

Are concept vector representations consistent across layers?

We analyse this question from a theoretical perspective and find the conditions $g(\cdot)$ must meet for CAVs to be consistent. Then, we explore this question in a toy setting using the Elements and ImageNet [Deng et al., 2009] datasets.

Theory

We begin by analysing layer consistency for different activations – trying to understand under which conditions consistency can hold. Let $z_{l,i}$ be the activation vector in layer l for the input $x_i \in \mathbb{X}$. g_{l_1,l_2} maps the activations in layer l_1 to layer l_2, where $l_1 < l_2$, i.e. $g_{l_1,l_2}(z_{l_1,i}) = z_{l_2,i}$. We assume that g_{l_1,l_2} is continuous and differentiable. Let $\hat{z}_{l_1}$ and $\hat{z}_{l_2}$ be perturbed latent representations in layers l_1 and l_2, defined as:

$$\hat{z}_{l_1} = z_{l_1} + v_{c,l_1} \tag{3.27}$$

$$\hat{z}_{l_2} = z_{l_2} + v_{c,l_2} = g_{l_1,l_2}(z_{l_1}) + v_{c,l_2} \tag{3.28}$$

We want to investigate if v_{c,l_1} and v_{c,l_2} have the same effect on the activations (and hence the model), i.e. if $g_{l_1,l_2}(\hat{z}_{l_1}) = \hat{z}_{l_2}$. Assuming g is continuous and differentiable, in the next section, we will prove the result that this can hold if and only if g takes the following form:

$$g_{l_1,l_2}(z_{l_1}) = w(z_{l_1}) + Mz_{l_1} + b, \tag{3.29}$$

where $w(\cdot)$ is a periodic function with period v_{c,l_1} and $M \in^{m_{l_2} \times m_{l_1}}$ is a non-zero matrix and $b \in \mathbb{R}^{m_{l_2}}$. In theory, g could approximate a function of this form, as neural networks are universal approximators [Sonoda and Murata, 2017]. However, it is unlikely that, even if the model captured a periodic function, its period would align exactly with the direction of a CAV. The derivations for common activations can be found in Appendix B.3.1.

Experiments

Our goal is to investigate the question *are the concept vectors consistent across layers?* We measure the consistency of two perturbations using the consistency error:

$$\epsilon_{consistency} = ||g_{l_1,l_2}(\hat{z}_{l_1}) - \hat{z}_{l_2}|| = ||g_{l_1,l_2}(z_{l_1} + v_{c,l_1}) - (z_{l_2} + v_{c,l_2})|| \tag{3.30}$$

In our experiments, we use a scaling term to reduce the size of v_{c,l_1} and v_{c,l_2} to ensure the perturbed activation remains in distribution – see Appendix B.3.1 for details. If two perturbations have a consistency error of 0, then they have the same downstream effect on the model. We include the following benchmarks:

- *Optimised CAV* (lower bound): when using an SVM, we may not find a v_{c,l_2} that has a consistency error of 0 with v_{c,l_1}. Therefore, we use gradient descent on v_{c,l_2} to minimise the consistency error, which acts as a lower bound.
- *Projected CAV*: the error between $g_{l_1,l_2}(v_{c,l_1})$ and v_{c,l_2}, which measures how consistent the vectors are when projected into the next layer. If $g_{l_1,l_2}(\cdot)$ conserves vector addition, the projected CAVs would have 0 error.
- *Random* (upper bound): included as benchmarks. We include two variants: Random CAVs are found using probe datasets containing random images, and a Random Direction vector: $v_{c,l_2} \sim \text{Uniform}(-1, 1)$. If the consistency error is similar to random, it suggests that the CAVs between layers are as similar to each other as random directions.

Figure 3.9 shows the $\epsilon_{consistency}$ for different v_{c,l_2} across different training runs (see Appendix B.3.1 for details). The concept CAV obtains a nonzero consistency error, suggesting that CAVs across different layers are not consistent. When we compare it with the benchmarks, we find:

- The consistency error for the optimised CAVs is lower, implying that the standard approach for finding CAVs does not find optimally layer consistent CAVs. However, the nonzero error for optimised CAVs suggests it is not possible to find consistent vectors across these layers.
- As expected, the projected CAVs have a nonzero error, indicating that vector addition is not preserved.

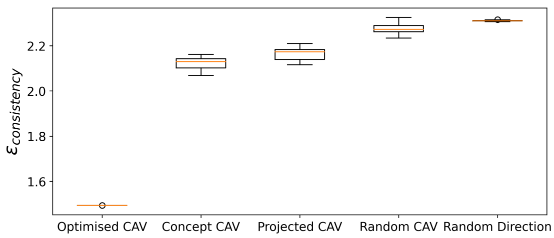


Figure 3.9: Empirical evidence for inconsistent CAVs across layers. The consistency error for different v_{c,l_2} for `striped` in the penultimate convolutional layer of a ResNet-50 trained on ImageNet. The optimised CAV acts as a lower bound, whereas the random CAV and direction act as baselines that provide upper bounds. Concept CAV: `striped` CAVs, trained as normal. Projected CAV: `striped` CAVs from layer l_1 projected into layer l_2, $g_{l_1,l_2}(v_{c,l_1})$.

- The random CAVs have a higher error, suggesting the concept CAVs are more similar than random vectors.

The inability to (easily) find consistent concept vectors across layers suggests that the directions encoded by CAVs in different layers are not equivalent; instead, we speculate that they represent different components of the same concept. This result is unsurprising given that previous works have demonstrated that model representations are more complex later in the NN [Mordvintsev et al., 2015, Olah et al., 2017, Bau et al., 2017], therefore it is unlikely that the same aspects of a concept are represented in different layers (discussed further in Appendix B.4.3). Consequentially, TCAV scores across layers can vary as they perform different tests – they measure the class sensitivity to a different version of the concept.

Figure 3.8a shows that concept vectors found in different layers of a model can give contradictory TCAV scores (further examples available in Appendix B.4.4). In the Elements dataset, shape concepts are encoded in each layer as the test accuracy for each layer is above 93%. Therefore, we expect to be able to use TCAV on each of these layers. However, the TCAV scores for `cross` in the Elements dataset contradict each other across 'layers.3' and 'layers.4', suggesting a positive and negative influence, respectively. This contradiction makes it difficult to draw a conclusion about the model's class sensitivity to `cross`.

On the right of 3.8a, we show the TCAV scores for `striped` for various classes in a ResNet-50 model trained on ImageNet. The accuracy for the `striped` vectors in ImageNet is above 96% for all layers tested, suggesting that the concept is encoded by the model in each of the layers. As in Elements, we do not observe consistent TCAV scores across layers. Instead, we observe a large change in the TCAV scores for `striped` in the penultimate layer, compared to earlier layers. 'layer4.1' suggests `striped` positively influences the likelihood of the classes tiger and leopard. However, earlier layers suggest that the class is not sensitive to the concept. This shows how, depending on the layers that are tested, different conclusions can be drawn.

3.7 Related Work

Concept Correlation and Entanglement Chen et al. [2020a] discuss how concept vectors can be correlated, making it challenging to create a vector that solely represents one concept. While their work focuses on de-correlating concepts *during training*, we focus on analysing the impact of correlated concepts *after training* and show how they can lead to misleading explanations (§3.4). Fong and Vedaldi [2018] use cosine similarity to demonstrate that the similarity between concepts varies based on the vector creation method. In our work, we also use cosine similarity to compare concept vectors. The distinction lies in our focus on CAVs and the insights they provide into the dataset and model. There have been several works analysing correlated concepts in interpretable-by-design networks [Heidemann et al., 2023, Zarlenga et al., 2023]. Our work complements these works by studying standard neural network architectures and the post-hoc explanations of TCAV. Raman et al. [2024] examine the effect of inter-concept relationships in CAVs and provide several useful metrics for measuring how well these relationships are represented, but they do not explore its effect on TCAV scores.

Spatial Dependence Biscione and Bowers [2021a] describe how CNNs are not inherently translation invariant but can learn to be (under certain conditions on the dataset). This finding challenges the common assumption that CNNs possess inherent translation invariance. Through *spatially dependent CAVs*, we demonstrate translation invariance with respect to a specific concept and class rather than, in general, providing more detailed information about a model. Raman et al. [2023] examine the locality of concept bottleneck models (CBMs) and find that, in some cases, CBMs make concept predictions using information far from the object of interest. In this work, we examine post-hoc concept-based explanations rather than predictions of an interpretable-by-design model. Additionally, our definition of spatial dependence is more related to whether we can learn CAVs, which mean `stripes on the right of the image`, rather than whether a CAV is using information near/far from where the concept is located in the image.

What concept representations does our analysis apply to? Most concept-based interpretability methods represent concepts as *vectors* in the activation space of a trained neural network [Kim et al., 2018, Fong and Vedaldi, 2018, Zhou et al., 2018, Ghorbani et al., 2019b, Zhang et al., 2020, Ramaswamy et al., 2022b, Fel et al., 2023]. However, some concept-based methods use different representations: individual neurons [Bau et al., 2017], regions of activation space [Crabbé and van der Schaar, 2022] or non-linear concepts [Bai et al., 2022, Li et al., 2023]. Our work focuses on the properties of concept *vectors*.

How is our work relevant in practice? To give insight into when the various properties may be relevant, we performed a review of computer vision papers which use CAVs in (1) the high-stakes applications of medical imaging (including skin cancer, skin lesions, breast cancer, and histology [Yan et al., 2023, Fürböck et al., 2022, Pfau et al., 2020]), and (2) computer vision research on models trained with well-known datasets [Krizhevsky, 2009, Lin et al., 2014, Wah et al., 2011, Zhou et al., 2017, Sagawa et al., 2020, Deng et al., 2009]. A summary table can be found in Nicolson et al. [2025]. We found that the following papers could have benefited from evaluating: consistency [Yan et al., 2023, Ramaswamy et al., 2022a, Fürböck et al., 2022, Yuksekgonul et al., 2023, Ghosh et al., 2023, Lucieri et al., 2020], entanglement [Yan et al., 2023, Ramaswamy et al., 2022a, Fürböck et al., 2022, Yuksekgonul et al., 2023, Ghosh et al., 2023, Graziani et al., 2020, McGrath et al., 2022b, Lucieri et al., 2020, Pfau et al., 2020], and spatial dependence [Yan et al., 2023, Ramaswamy et al., 2022a, Fürböck et al., 2022, Yuksekgonul et al., 2023, Ghosh et al., 2023, McGrath et al., 2022b, Lucieri et al., 2020, Pfau et al., 2020]. We provide a detailed example, using the application of skin cancer diagnosis [Yan et al., 2023], in Nicolson et al. [2025].

3.8 Conclusion and Future Work

In this chapter, we examined three key properties underlying the linear representation hypothesis that shape how we interpret and apply Concept Activation Vectors (CAVs): *concept entanglement*, *spatial dependence*, and the *consistency of intervention effects across layers*.

We began by analysing concept entanglement. We derived the conditions under which entanglement can arise and showed how it can affect the conclusions drawn about whether a concept is present and used by the model. Next, we formalised two forms of spatial dependence: *activation spatial dependence* and *concept spatial* *dependence*. We demonstrated how concept vectors can be used to detect spatial dependence in a model's internal representations. Finally, we investigated the consistency of concept interventions across layers. We showed that, unless certain structural constraints hold, intervening on a concept in different layers may lead to different effects on the model's output. This sheds light on why concept vector explanation methods can give different conclusions across layers.

These findings have practical implications for how CAVs are used in interpretability research. Building on this analysis, Nicolson et al. [2025] provide concrete recommendations for incorporating these insights into real-world use cases, including a detailed application to medical imaging. We further explore these properties in Chapters 4 and 6. By establishing this groundwork, we can proceed with a clearer understanding of the assumptions and limitations of our methods in those subsequent analyses.

4
Universal Representations in LLMs

In this chapter, we continue to work toward the broader goal of understanding *what* knowledge is encoded in machine learning models and *how* it is represented. Here, we focus on the latter – examining *how* knowledge is encoded in large language models (LLMs). We choose to study LLMs because they are widely deployed and influence many aspects of society [Weidinger et al., 2023], making it increasingly important to understand how they operate.

4.1 Introduction

While LLMs are predominantly trained on English data, they are deployed in several languages, including some that are rarely seen during training. This raises an important question: how do LLMs operate across different languages?

LLMs are hypothesised to operate in an abstract concept space [Chris Olah, 2023, Nanda et al., 2023a, Wendler et al., 2024, Dumas et al., 2024]. From the multilingual perspective, one important question is whether the concept space is language-specific or language-agnostic. We consider three different hypotheses:

1. LLMs operate in a language-agnostic space (i.e., universal).

2. LLMs operate in a language-specific space, which is determined by the input language.

3. LLMs operate in a space that is English-centric (or centred on the dominant pretraining language).

This first view aligns with an *objectivist* stance, in which concepts are assumed to exist independently of culture or context. In the second and third case, the views are aligned with a setting where concepts are shaped by the cultural traditions and usage of the language. Existing work follows two main threads: the concept space is universal, and the concept space is biased towards the training-dominant language.

A universal representation space One stream of research argues that the concept space is *universal*. When analysing the latent space with the logit lens, Wendler et al. [2024] find that the concept space is language-agnostic, but more closely aligned with the English output space. In their follow-up work, Dumas et al. [2024] used tracing to find further evidence of language-agnostic representations of concepts.[1]. Concurrent to this work, Brinkmann et al. [2025] showed that models share morpho-syntactic concept representations across languages.

A language-specific representation space Other researchers found that the concept space is biased towards the training-dominant language. Using the logit lens, Zhong et al. [2024] focused on Japanese, showing that Swallow [Fujii et al., 2024], which is fine-tuned on Japanese, and LLM-jap Aizawa et al. [2024], pre-trained on Japanese and English, are Japanese-centric. Fierro et al. [2025] found that subject enrichment – retrieving attributes about the subject – is language-agnostic whereas object extraction is language-dependent, when studying EuroLLM [Martins et al., 2024], XLGM [Lin et al., 2022] and mT5 [Xue et al., 2021].

In this chapter, we show that the underlying mechanism is more nuanced, and depends on the part of speech modelled. While prior work focuses predominantly on single-token nouns, we consider the open multi-token generation setting. As a result, this allows us to focus on different parts of speech, whereas prior work focuses predominantly on nouns.

To provide an intuition, Figure 4.1 shows the logit lens to Llama-3.1-70B as it generates the French text "Le bateau naviguait en douceur sur l'**eau au calme du lac. Le soleil ...**", with the bold text representing Llama-3.1-70B's output.[2] The English translation is "The boat sailed smoothly on the **calm water in of the lake. The sun ...**". Lexical words like "water", "lake," and "sun" are selected in English, whereas grammatical elements such as "du" and "le" are not.

We analyse this question on three levels:

[1]They used tracing to mitigate potential shortcomings of cosine-similarity [Steck et al., 2024]

[2]In general, we use parentheses for (generated) text and words. We use `this font` to refer to a concept.

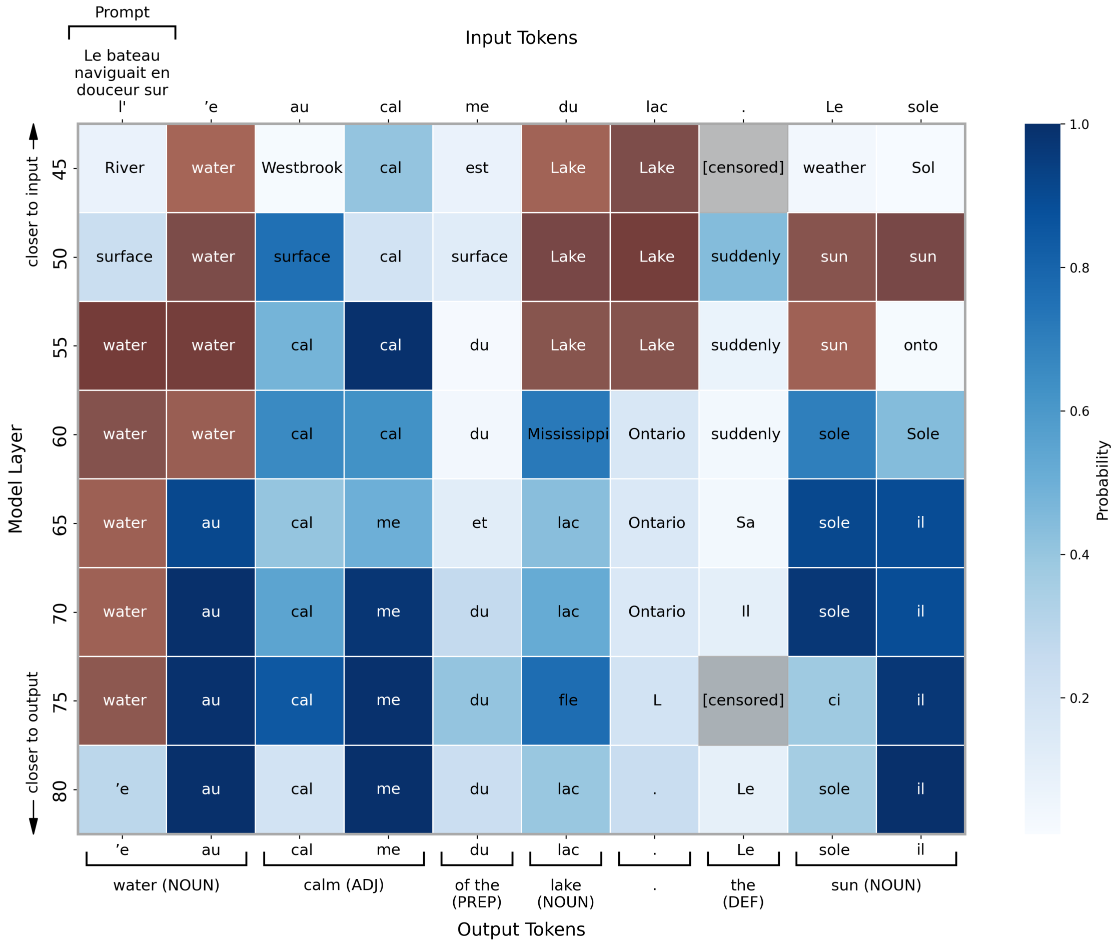


Figure 4.1: Logit lens applied to Llama-3.1-70B's latent space, when prompted with *Le bateau naviguait en douceur sur l'*. Each row depicts the decoded latent representations for one layer and each column corresponds to the generated token. Dark red boxes highlight words selected in English. The nouns "eau", "lac", and "soleil" are selected in English, whereas other parts of speech are not.

1. Latent space representation: we study how representations evolve within the model, showing that for lexical words, English-focused representations often appear first before being transformed into the target language.
2. Intervening on the behaviour: we show that steering the representations is more effective when using vectors constructed in English than in the target language.
3. Understanding the behaviour: we show that the latent representation structure is consistent with the language and semantic context being represented separately.

Recall Chapter 2, where we discussed model desiderata. When the goal is to understand model behaviour, a key consideration is faithfulness – that is, whether the explanation reflects what the model actually does. This is why we include all three steps in our approach – identifying behavioural patterns, verifying the behavioural patterns and understanding the behavioural patterns. Appendix A.1 provides a more formal intuition for why the verification step matters for model faithfulness.

We analyse four open-source models – Llama-3.1-70B, Gemma-2-27b, Aya-23-35B, and Mixtral-8x22B– which vary in architecture and language coverage. Overall, we find that LLMs tend to make semantically-loaded decisions in regions of the representation space closely aligned with English, regardless of the input or output language. In contrast, other aspects of language–such as grammar–appear to be processed in a more language-specific way. These patterns may help explain observed performance gaps in multilingual settings.

The English-centric bias in semantic representation can negatively affect downstream performance in other languages [Shafayat et al., 2024, Huang et al., 2023, Bang et al., 2023, Shi et al., 2022]. This also raises fairness concerns, as models exhibit cultural and linguistic biases [Shafayat et al., 2024].

Language-specific processing also introduces additional challenges – in particular for lower-resource languages. First, these models tend to be less fluent in non-English languages [Guo et al., 2024]. Second, this may also affect the robustness and reliability in multilingual settings [Marchisio et al., 2024, Deng et al., 2024]. This observation is consistent with our finding that safety-tuning does not appear to generalise across languages, suggesting that safety interventions may be language-specific.

We further find that the English-centric behaviour occurs the most in models that are trained with less diverse training datasets. This suggests that diverse training datasets can mitigate this behaviour.

4.2 Background

4.2.1 Large Language Models

Language models are trained to operate across different languages. Table 4.1 summarises the four LLMs we study, which differ in the number of languages they were trained on. Aya-23-35B supports the widest range of languages, while Gemma-2-27b covers the fewest.

We evaluate these models across five languages, selected based on their varying levels of representation during training. English, the predominant training language, serves as a baseline. French and German represent high-resource, non-English languages, while Dutch and Chinese are lower-resource languages. Dutch is only

Table 4.1: Summary of pre-training dataset languages of the open-source large language models studied in this work.

Model	Nr. Languages Trained On	Trained on Language? English	French	German	Dutch	Mandarin
Aya-23-35B [Aryabumi et al., 2024]	23	√	√	√	√	√
Llama-3.1-70B [Dubey et al., 2024]	8	√	√	√	×	×
Mixtral-8x22B-v0.1 [Jiang et al., 2024]	5	√	√	√	×	×
Gemma-2-27b [Google, 2023]	1	√	×	×	×	×

Table 4.2: LLM-Insight dataset examples: sentences and prompts for the word animal.

Language	Sentence Example	Prompt Example
English	The zoo has a wide variety of animal species.	They adopted a
Dutch	De boerderij had elk type huisdier.	In de dierentuin zag ik een bijzonder
French	Le lion est un animal sauvage qui vit dans la savane.	Il a vu un
Mandarin	森林中生活着许多野生动物	每年都会有新的
German	Der Zoo beherbergt viele faszinierende Tiere.	Sie liebt es Zeit mit ihrem

a high-resource language in Aya-23-35B and therefore provides an interesting comparison to German due to their linguistic similarity. This analysis allows us to understand the performance disparities across languages with varying levels of representation in training.

4.2.2 Methods

Our goal is to understand whether LLMs have a universal representation space. To address this question, we use three mechanistic interpretability methods. The logit lens allows us to examine the internal representations, while causal tracing provides insight into where facts are encoded in the model across different languages. Finally, steering vectors let us intervene on the models' internal representations, which allows us to verify that the representations influence the output.

Logit Lens

The logit lens [nostalgebraist, 2020] decodes the internal representations of an LLM into tokens. LLMs take an input x and output a probability distribution over the next token. The logit lens decodes the intermediate representation $z_l(x)$ at layer l into an output token by applying the unembedding layer:

$$\text{argmax}_t \; \text{softmax}(W_u z_l(\text{norm}(x))) \tag{4.1}$$

where x is the input, W_u is the unembedding matrix of the model and the subscript t corresponds to the token. Figure 4.1 shows the logit lens applied to Llama-3.1-70B when generating:

> "Le bateau naviguiait en douceur sur l'**eau au calme du lac. Le soleil ...**".

For each layer (y-axis) and token position in the generation (x-axis), a token is decoded from the internal representation. The decoded tokens from the middle layers onward are more interpretable, whereas earlier layers are less interpretable.

The logit lens is a common approach in mechanistic interpretability for analysing the latent space of large language models (LLMs) [Wendler et al., 2024, Wu et al., 2024, Zhong et al., 2024]. While it has known limitations [Belrose et al., 2023, Ghandeharioun et al., 2024], we favour it for its simplicity and minimal assumptions – it avoids introducing additional inductive biases that more complex interpretability methods may impose.

Causal Tracing

Causal tracing [Meng et al., 2022, Vig et al., 2020] uses causal mediation analysis to identify where facts are stored within a network. For example, imagine that we want to find where the fact "The capital of Canada is Ottawa" is represented in an LLM. We could prompt the model with "The capital of Canada is" to find where "Ottawa" is stored in the network. There are two main steps in causal tracing:

1. **Corrupt the signal**: destroy the information so that the model no longer outputs the fact.
2. **Restore the signal**: determine where in the network the representation needs to be restored so that the LLM can recover the correct output.

Let $e^{\text{clean}} \in \mathbb{R}^{m,d}$ be the embedding of the prompt "The capital of Canada is", where m is the number of tokens and d is the embedding dimension. In the first step, the information is destroyed by adding noise to the embedding of the subject token:[3]

$$e_j^{\text{corrupted}} = \begin{cases} e_j^{\text{clean}} + \varepsilon & \text{if token } j \text{ is a subject token} \\ e_j^{\text{clean}} & \text{otherwise} \end{cases} \tag{4.2}$$

where ε is noise sampled from an isotropic Gaussian distribution, and $e^{\text{corrupted}}$ is the corrupted embedding. We apply the pushforward operator to the corrupted embeddings $e^{\text{corrupted}}$ through the network to obtain the probability that the model outputs Ottawa, $f[\text{Ottawa}|e^{\text{corrupted}}]$.

Next, we want to determine which part of the hidden states encodes the relevant information to restore the correct output. At a given layer l in the network, we restore part of the corrupt hidden state by copying back part of the clean hidden state z at position p:

$$z_{j,l}^{\text{restored}} = \begin{cases} z_{j,l}^{\text{clean}} & \text{if } j = p \\ z_{j,l}^{\text{corrupted}} & \text{otherwise,} \end{cases} \tag{4.3}$$

where the hidden state $z_l^{\text{clean}} = [z_{1,l}^{\text{clean}}, \ldots, z_{m,l}^{\text{clean}}]$ is obtained by pushing the original embeddings, e^{clean}, through the network.

Finally, we propagate z_l^{restored} through the remaining layers to produce the output probability $f[\text{Ottawa}|z_{l,p}^{\text{restored}}]$. The difference $f[\text{Ottawa}|z_{l,p}^{\text{restored}}] - f[\text{Ottawa}|z_l^{\text{corrupted}}]$, measures the importance of layer l and token position p in encoding a fact. The difference in the output probabilities of the target token in the two forward passes is the average indirect effect (AIE) (see Meng et al. [2022] for further details). Through this approach, causal tracing helps identify which parts of the representation are sufficient to retrieve the correct output.

[3]The subject token is the token(s) corresponding to the subject of the sentence.

Steering Vectors

Steering vectors [Subramani et al., 2022, Turner et al., 2023, Panickssery et al., 2024] are used to nudge the behaviour of the LLM in the desired direction. The main idea is to add activation vectors during the forward pass of a model to modify its behaviour as follows:

$$z_l(x) \leftarrow z_l(x) + \gamma v_l, \tag{4.4}$$

where v_l is the steering vector, and $\gamma \in \mathbb{R}^+$ is a scalar hyperparameter. One example of how steering vector could be used is to generate an output that contains more `love`. We can compute a steering vector as follows:

$$v_l = z_l(\text{love}) - z_l(\text{hate}), \tag{4.5}$$

and modify the model behaviour as in Equation 4.4. Further details can be found in Subramani et al. [2022] and Turner et al. [2023].

4.3 Datasets

LLM-Insight We created a dataset to analyse the behaviour of LLMs, specifically designed to study steering in LLMs. It includes 72 target words, each paired with 10 prompts and 10 sentences in English, Dutch, French, German and Mandarin. Table 4.2 shows a sample of the data.

A prompt is a partial sentence that the LLM needs to complete. Each prompt is designed so the target word could appear as the next token, but the prompt sufficiently open-ended so that other (semantically unrelated) words can be used to complete the sentence. For example “They adopted a” can be completed with the word “animal” as well as “daughter” (see Appendix C.2 for examples).

The sentences provided in the dataset can be used to find steering vectors. Some words in the dataset naturally form pairs (as in Equation 4.2.2) that can be used to create steering vectors, such as the words “good” and “bad”. For words without a natural pairing, such as “thermodynamics”, we provide a general set of sentences as the counter set to create the steering vector. Further details on the dataset can be found in Appendix C.2.

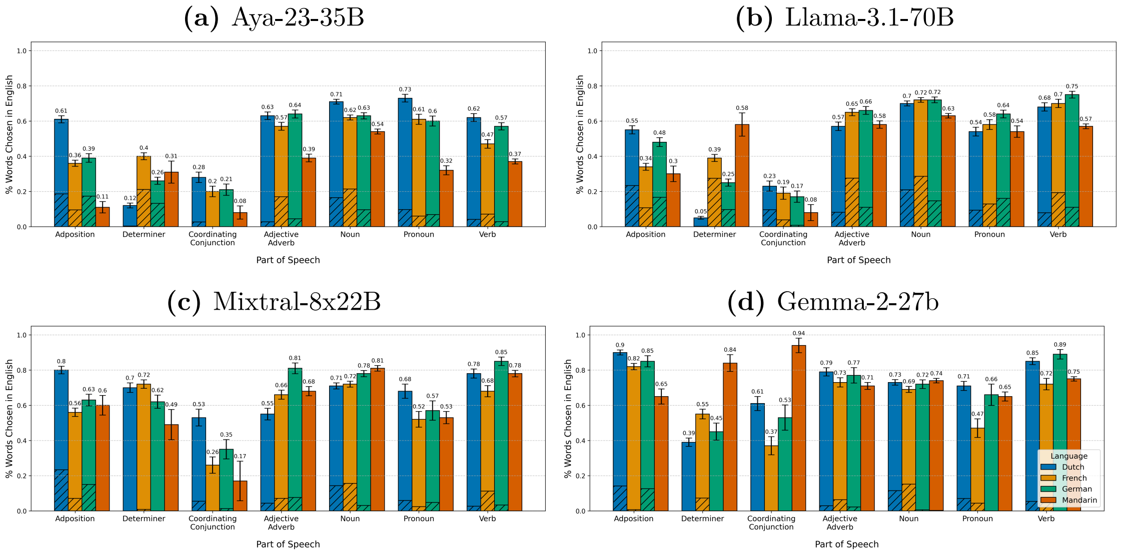


Figure 4.2: Logit lens analysis of LLMs routing through English. Each plot shows the proportion of words routed through the English representation space for each model. The shaded bars indicate the portion explained by homographs – words that are spelled the same in English and the specified language. Overall, the degree of English routing depends on the model: less diverse pretraining leads to more English routing. Similarly, most routing occurs for lexical words.

City facts Ghandeharioun et al. [2024] We use this dataset to investigate how facts in different languages are encoded in LLMs. The task is to provide the capital city of a given country. For example, when prompted with “The capital of Canada is”, the model should output “Ottawa”. This allows us to identify where the concept of Ottawa is encoded in the network. We augment the dataset by translating these facts into German, Dutch, French and Mandarin to analyse cross-lingual representations.

4.4 Experiments

We want to understand whether LLMs process prompts differently depending on the input/output language. First, we analyse the latent space to find that LLMs make semantic decisions that are more closely aligned with the English representation space (Section 4.4.1). Next, we show that we can steer activations better when using English steering vectors (Section 4.4.2). Lastly, in Section 4.4.3, we show that the representations of facts are shared across languages, but have an English-centric bias when decoded.

4.4.1 Inspecting the latent space of LLMs using the Logit Lens

Qualitative Examples To build an intuition on how LLMs operate when prompted in different languages, we analyse their latent space using the logit lens, which decodes the internal representations. In Figure 4.1, nouns and pronouns are routed through English, whereas the coordinating conjunction is not. Similarly, Figure 4.3 shows the logit lens applied to Llama-3.1-70B for the Dutch prompt "Ze telen hun eigen". The noun "fruit", verb "kweken", and pronoun "they" are all routed through the English words, whereas the coordinating conjunction "en" is not routed through the English word "and". Interestingly, the word "growing" appears in the latent space several tokens before "kweken" is generated, suggesting that the LLM may plan words in advance in English, which builds on Pal et al. [2023]'s finding that LLMs encode future tokens in the latent space.

Quantitative Evaluation The qualitative examples in Figures 4.1 and 4.3 suggest that the part of speech determines whether LLMs employ English routing. To investigate this, we prompt each LLM to generate 720 sentences. For each generated word, we evaluate whether the English equivalent of a word appears in the latent space. For example, in Figure 4.3, for the word "groenten", we check whether the English equivalent, "vegetables", appears in the decoded latent space. We then aggregate the results across different parts of speech. Further implementation details are provided in Appendix C.3.

Figure 4.2 shows the results for Aya-23-35B, Llama-3.1-70B, Mixtral-8x22B and Gemma-2-27b. Each bar shows the percentage of words that route through the English representation space. The shaded part shows the proportion explained by cross-lingual homographs – words that are the same in English and the specified language (e.g., water in English and Dutch). For homographs, it is not possible to disambiguate whether the word routes through English.

Lexical words – nouns and verbs – are often chosen in English. These parts of speech influence the semantic meaning of the sentence. Other parts of speech, such as adpositions, determiners and compositional conjugates are infrequently routed through English in Aya-23-35B and Llama-3.1-70B.

The degree of English routing is model-dependent, as shown in Table 4.3. One explanation is the degree of multilingualism in the pre-training data – with more multilingual models, such as Aya-23-35B, routing less through English, in contrast to the least multilingual model, Gemma-2-27b, which routes the most through English.

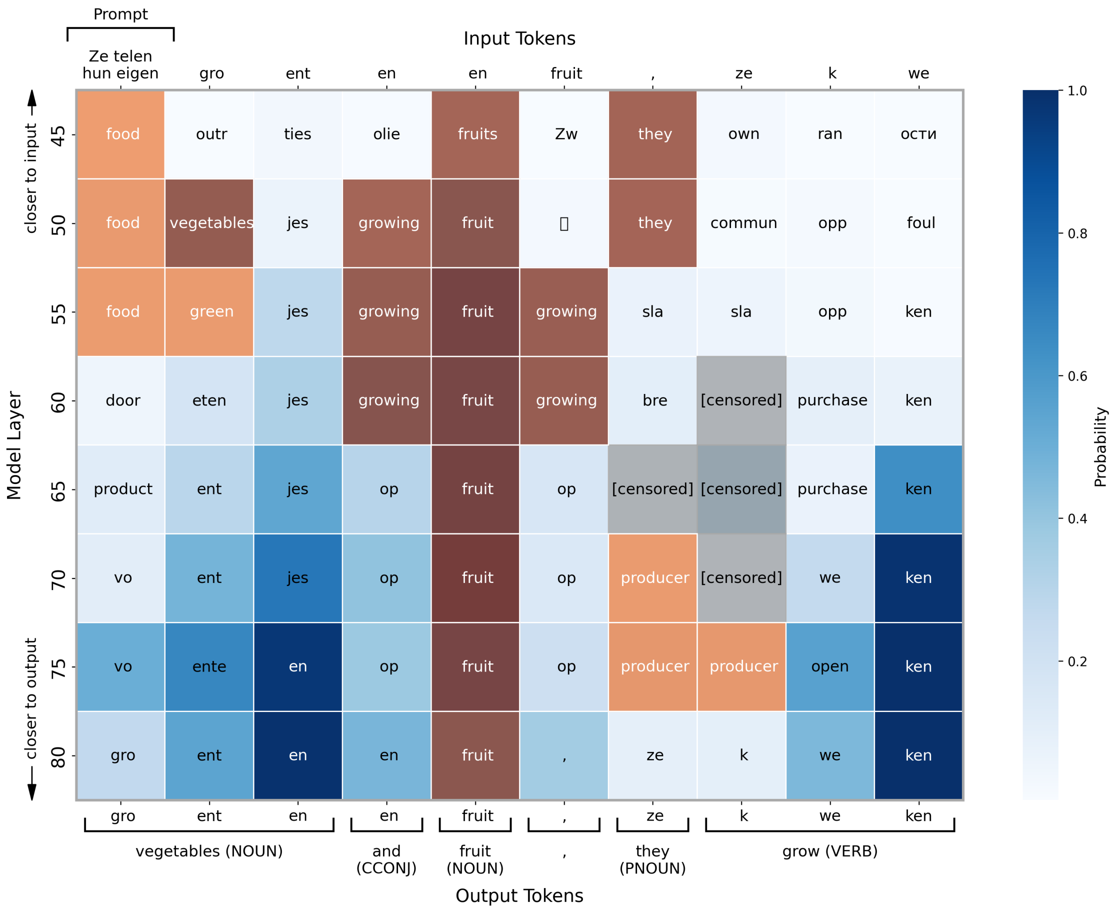


Figure 4.3: Logit lens applied to the latent space of Llama-3.1-70B, prompted in Dutch with "Ze telen hun eigen". Each row depicts decoded latent representations across layers, and each column corresponds to the token generated at a specific time step. Orange boxes highlight words selected in English, darker red boxes highlight related words, while grey boxes indicate explicit terms omitted from the figure (see Appendix C.3.1). The noun "fruit" and the pronoun "they" are selected in English.

However, this does not account for the differences observed between Mixtral-8x22B and Llama-3.1-70B, for which French and German are both high-resource languages. Another possible explanation is model size. Smaller models, such as Mixtral-8x22B and Gemma-2-27b, route through English more frequently than larger models, potentially due to their more limited representation space.

4.4.2 Cross-Lingual Steering

Our experiments in Section 4.4.1 suggest that large language models (LLMs) may make conceptual choices in an English latent representation space, before translating this content into the target output language in later layers. To further investigate this hypothesis, we test whether non-English outputs can be influenced using

Table 4.3: English routing in LLMs: percentage of generated words routed through English. Aya-23-35B shows the least routing behaviour, whereas Gemma-2-27b shows the most routing behaviour.

Model	Dutch	French	German	Mandarin	Average
Gemma-2-27b	0.72 ± 0.01	0.67 ± 0.01	0.72 ± 0.01	0.71 ± 0.01	0.70 ± 0.00
Mixtral-8x22B	0.69 ± 0.01	0.63 ± 0.01	0.71 ± 0.01	0.69 ± 0.01	0.68 ± 0.01
Llama-3.1-70B	0.51 ± 0.01	0.57 ± 0.01	0.58 ± 0.01	0.55 ± 0.01	0.55 ± 0.00
Aya-23-35B	0.58 ± 0.01	0.49 ± 0.01	0.50 ± 0.01	0.41 ± 0.01	0.50 ± 0.00

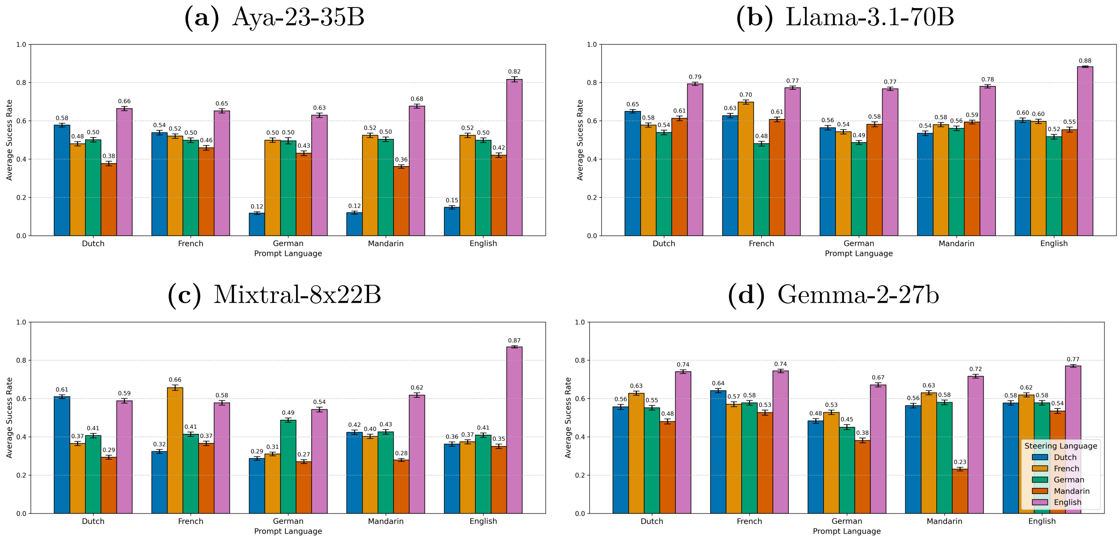


Figure 4.4: Cross-Lingual Steering LLMs: The language on the x-axis is the prompt and the desired output language, while the colour of each bar indicates the language used to generate the topic steering vectors.

steering vectors derived from English inputs.

More concretely, we test whether we can steer models to generate a sentence in a specified output language using two types of steering vectors:

- **Topic steering vector** – encourages the LLM to generate a sentence with the given topic, such as animals.
- **Language steering vector** – encourages the model to generate text in the desired output language.

We evaluate the effectiveness of steering across various topics and prompts, using the LLM-Insight dataset (see Section 4.3). As an example, consider the word "slecht", which is "bad" in Dutch. Here, Dutch is the input and output language. Imagine we select English as the steering language. For the word "slecht", we have 10 prompts in LLM-Insight. An example of a Dutch prompt is "Ze had een", which translates to "She had a". Our goal is to steer the LLM to complete this sentence with the word "slecht", which is `bad` in Dutch. We do this as follows:

1. **Compute the Topic Steering Vector.** For the word "bad", there are 10 English sentences in the dataset, $\mathbb{X}^{\text{topic},+}$ and 10 sentences containing the counter-word "good", $\mathbb{X}^{\text{topic},-}$. For each sentence, we compute its representation at layer l_1, averaged across token positions, to create the sets $\mathbb{Z}_{l_1}^{\text{topic},+}$ and $\mathbb{Z}_{l_1}^{\text{topic},-}$ for each sentence in $\mathbb{X}^{\text{topic},+}$ and $\mathbb{X}^{\text{topic},-}$, respectively. We compute the topic steering vector as

$$v_{c,l_1,topic} = \sum_i z_{i,l_1} - \sum_j z_{j,l_1} \quad \forall z_{i,l_1} \in \mathbb{Z}_{l_1}^{\text{topic},+}, z_{j,l_1} \in \mathbb{Z}_{l_1}^{\text{topic},-}. \tag{4.6}$$

2. **Compute a Language Steering Vector.** To encourage the LLM to complete the sentence in the target language (Dutch), we compute a language vector. We encode 10 Dutch sentences, $\mathbb{X}^{\text{language},+}$, and 10 English sentences, $\mathbb{X}^{\text{language},-}$. As in the previous step, we construct $\mathbb{Z}_{l_2}^{\text{language},+}$ and $\mathbb{Z}_{l_2}^{\text{language},-}$ for $\mathbb{X}^{\text{language},+}$ and $\mathbb{X}^{\text{language},-}$, respectively. We compute the language steering vector as

$$v_{c,l}^{language} = \sum_i z_{i,l_2} - \sum_j z_{j,l_2} \quad \forall z_{i,l_2} \in \mathbb{Z}_{l_2}^{\text{language},+}, z_{j,l_2} \in \mathbb{Z}_{l_2}^{\text{language},-} \tag{4.7}$$

3. **Apply Steering Vectors to the Hidden State**. We encode the Dutch prompt "Ze had een" to obtain its hidden state z_{l_1}. We then steer the model by modifying the hidden states as follows:

$$z_{l_1} \leftarrow z_{l_1} + \alpha v_{c,l_1}^{topic} \tag{4.8}$$

$$z_{l_2} \leftarrow z_{l_2} + \alpha v_{c,l_2}^{language} \tag{4.9}$$

The layers l_1 and l_2 may differ; different parts of the model may perform concept selection and language selection. We evaluate the performance at every fifth layer, starting with the post-embedding layer, and report results for the layer combination that yields the best performance.

We evaluate steering as successful if the generated sentence meets the following three criteria:

- Topic: Sentences that contain or is about the topic word obtain the the maximum score of 1.
- Avoids stuttering: Outputs containing repetitions at the end of the sentence (e.g., "they went to the store store store") are assigned the lowest score, 0.

- Fluency: We use GPT-4o to assess fluency. Sentences that are grammatically correct but exhibit collapsed or unnatural structure (e.g., "the doctor treated the animal and the animal was treated by the doctor") receive a score of 0. To ensure the reliability of the GPT-4o fluency evaluations, we manually verified 100 examples per language.

We aggregate the scores across each criteria by multiplying the three scores. We perform this experiment for the entire LLM-Insight dataset – 72 words, 10 prompts per word, for each of the combinations of the five languages. Additional implementation details are provided in Appendix C.

Figure 4.4 shows results when steering different LLMs. In general, we observe that English steering vectors perform the best – outperforming steering vectors generated using the desired output language. This suggests that the representation space is not universal – if it were, we would expect the cross-lingual performance to be roughly equal across languages. Instead, this supports the hypothesis that these models make semantic decisions in English.

4.4.3 Investigating the Representation Space

The previous sections showed that LLMs have an English-centric conceptual bias, but what does this imply for the structure of the latent space? We begin by using causal tracing to identify which layers encode factual information. Next, we interpolate between concept representations to explore the underlying geometry of fact encoding.

Causal Tracing In this section, we study *how* cross-lingual facts are encoded relative to each other using the city facts dataset (see Section 4.3). First, we perform causal tracing to determine whether facts in different languages are encoded in the same part of the model. Figure 4.5 shows the causal traces for Aya-23-35B, Llama-3.1-70B, Mixtral-8x22B, and Gemma-2-27b. We find that facts are generally localised in similar layers, regardless of the language.

Fact Interpolation Next, we want to understand if the representation of a fact is shared across different languages. In particular, if we have the same fact in two different languages, such as English and Dutch, can we decompose the representation as follows:

$$h(\text{capital of Canada}) = z_{Ottawa} + z_{English} \tag{4.10}$$

$$h(\text{hoofdstad van Canada}) = z_{Ottawa} + z_{Dutch}, \tag{4.11}$$

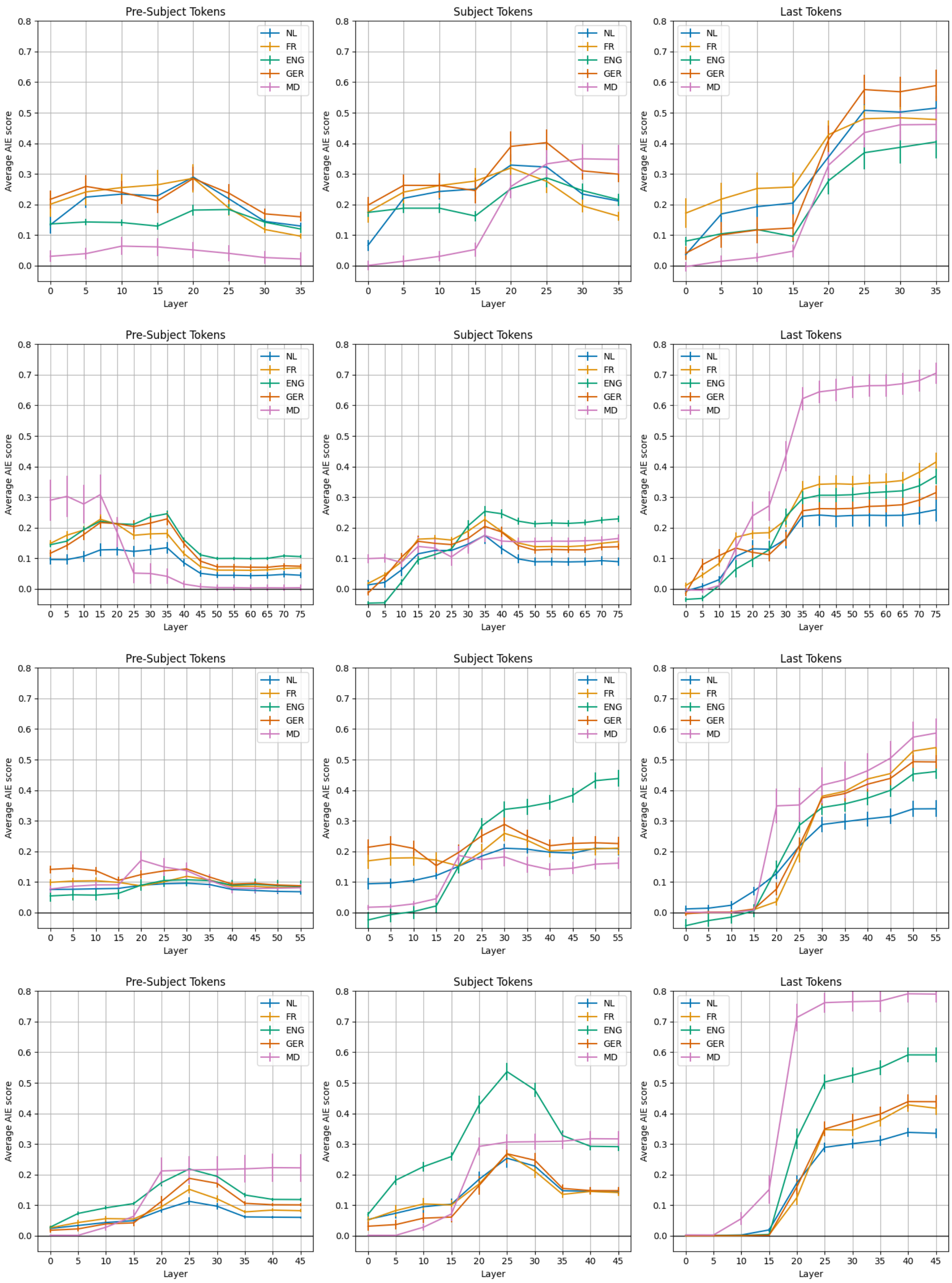


Figure 4.5: Causal traces of the City Facts dataset for (top-to-bottom): Aya-23-35B, Llama-3.1-70B, Mixtral-8x22B, and Gemma-2-27b. Each sentence varies in length, so we divide it into three segments: pre-subject tokens, subject tokens (i.e., tokens corresponding to the subject of of the sentence), and final tokens. Corrupting either the subject or final tokens typically has the greatest impact on the model's output, as these regions carry critical information needed to recover the correct response. The AIE scores are similar across different languages, suggesting that facts are localised in the same area of the model.

where h represents a vector in the latent space. If the above equations hold, we may be able to interpolate between the facts:

$$\alpha h(\text{hoofdstad van Canada}) + (1-\alpha)h(\text{capital of Canada}),$$
$$= z_{Ottawa} \ + \alpha z_{Dutch} + (1-\alpha)z_{English}$$

If we pushforward the interpolated hidden state, and the output is correct, then this suggests that we may be able to disentangle the language from the semantic meaning.

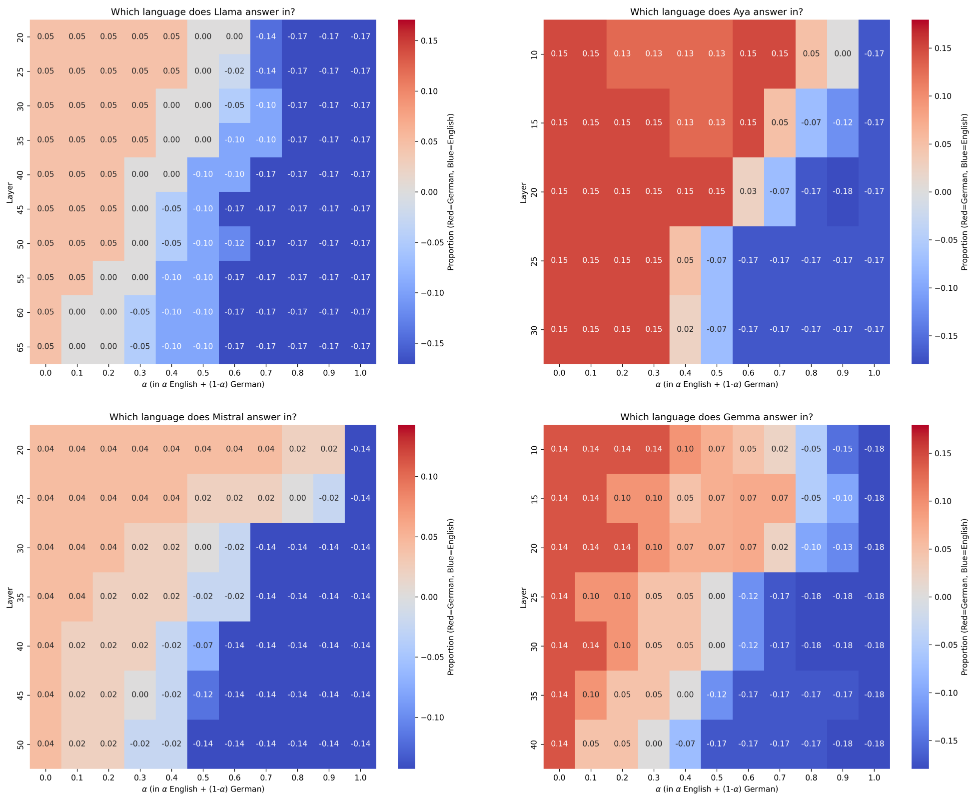


Figure 4.6: Relative propensity of (starting top left, clockwise) Llama-3.1-70B, Aya-23-35B, Mixtral-8x22B, and Gemma-2-27bto answer in German (red) vs English (blue). Llama-3.1-70B is most likely to answer in English.

We find that interpolating between hidden states corresponding to different languages does not result in significant accuracy degradation; in most cases, performance smoothly interpolates between the accuracies of the two languages (see Appendix C.6). Additionally, models exhibit a strong bias toward generating English responses, as shown in Figure 4.6, where we quantify this bias as the difference in probability assigned to the English versus German output when the interpolated hidden state is decoded.

These findings support the conclusion that LLMs represent factual content in a language-agnostic latent space, with language information encoded along a distinct and separable direction. This is consistent with prior work by Dumas et al. [2024], who also observed a disentangling of conceptual and linguistic representations.

Our contribution extends this insight: we show that factual knowledge can be smoothly transferred between languages via latent space interpolation, without any loss in task performance – only the output language shifts. This suggests that the direction between the hidden states primarily encodes language, while the underlying factual content remains stable. Notably, English appears to occupy a disproportionately large region of this latent space, as evidenced by the model's tendency to default to English during interpolation.

4.5 Limitations

Our work provides evidence suggesting that LLMs primarily operate in English. Below, we outline potential limitations and directions for future research.

Tokenization Sentences in different languages often vary in tokenization length [Rust et al., 2021, Muller et al., 2021, Petrov et al., 2024], which complicates cross-lingual comparisons. In this work, we provide heuristics (e.g., for causal tracing, which operates on a per-token level) to compare the results when tokenization lengths vary. However, tokenization remains an important consideration for developing interpretability methods that are used in a cross-lingual setting.

Language confidence and confusion Models often assign higher probabilities to outputs in certain languages, which can affect analyses such as causal tracing by requiring higher noise levels. Similarly, models often exhibit language confusion [Marchisio et al., 2024], continuing to respond in English even when prompted in other languages. Both factors influence our analysis. We can mitigate some issues associated with the first problem – e.g., in causal tracing, we ensure the probabilities all fall below a specified threshold when a prompt is noised. However, we do not actively address language confusion, as doing so could alter the natural behaviour of the LLMs, which we aim to understand.

Factors affecting interpretability methods Interpretability methods are influenced by various factors. Steering performance, for example, depends on the intrinsic steerability of a prompt [Turner et al., 2023, Tan et al., 2024]. To address this, we designed a custom dataset that, to the best of our knowledge, is equally steerable across all languages. Another challenge is that steering could push activations outside the expected data distribution, leading to unintended outputs. To mitigate this, we checked for stuttering in the generated outputs. However, further work is needed to deepen our understanding of steering mechanisms and to develop more robust evaluation procedures.

Other Methods Exploring alternative methods could provide valuable insights. For example, sparse autoencoders (SAEs) [Olshausen and Field, 1997, Hinton and Salakhutdinov, 2006, Templeton et al., 2024] are a popular interpretability tool. However, training SAEs for each layer is computationally expensive and beyond our computational budget. While some pre-trained SAEs are available, they are predominantly trained on English data, which introduces biases we aim to avoid [Lieberum et al., 2024].

4.6 Related Work

We can think about understanding a model from two perspectives:

- An **internal perspective**, focused on analysing the model through the latent space and operations performed inside of the model. Examples of questions include: How do models represent knowledge across different languages? How does a model retrieve facts?
- An **external perspective**, focused on analysing the model output. For example, how well do models perform in different languages?

Having a unifying theory that combines both perspectives is important – the internal perspective helps us understand the mechanisms underlying behaviour, while the external perspective examines the real-world impact of that behaviour. Below, we summarise the research on multilingual language models from both perspectives.

4.6.1 How do LLMs Operate Internally?

The current main theory in mechanistic interpretability suggests that there are three general phases in the forward pass of an LLM Chris Olah [2023], Nanda et al. [2023a], Wendler et al. [2024], Dumas et al. [2024], Fierro et al. [2025]:

1. **Detokenization**: In this phase, individual tokens are combined into abstract units that the model uses for analysis. These units can be referents – for example, Nanda et al. [2023a] found evidence that the tokens [Michael] and [Jordan] are combined into a unit representing the basketball player Michael Jordan. Similarly, these units can encode instructions, as shown by Dumas

et al. [2024], where the model extracts the target language during translation tasks in these layers.

2. **Processing**: In this phase, the model processes or reasons over abstract units. For instance, this stage may involve tasks like fact recall Geva et al. [2023], Nanda et al. [2023a].

3. **Selecting the output**: In this phase, the model selects the output. This may involve selecting the correct attribute Nanda et al. [2023a], mapping an abstract concept to the corresponding word in the target language Wendler et al. [2024] and/or selecting the correct token for the intended word.

In the context of multilingual models, an important question is whether the concept space (in phase 2) is universal. Here, *universal* means the representation is *shared* across languages, i.e., the representation of "cat" (`cat` in English) and "kat" (`cat` in Dutch) is the same.

One stream of research argues that the concept space is *universal*. When analysing the latent space with the logit lens, Wendler et al. [2024] find that the concept space is language-agnostic, but more closely aligned with the English output space. In their follow-up work, Dumas et al. [2024] used tracing to find further evidence of language-agnostic representations of concepts.[4]. Concurrent to this work, Brinkmann et al. [2025] showed that models share morpho-syntactic concept representations across languages.

Other researchers found that the concept space is biased towards the training-dominant language. Using the logit lens, Zhong et al. [2024] focused on Japanese, showing that Swallow [Fujii et al., 2024], which is fine-tuned on Japanese, and LLM-jap Aizawa et al. [2024], pre-trained on Japanese and English, are Japanese-centric. Fierro et al. [2025] found that subject enrichment – retrieving attributes about the subject – is language-agnostic whereas object extraction is language-dependent, when studying EuroLLM [Martins et al., 2024], XLGM [Lin et al., 2022] and mT5 [Xue et al., 2021].

Overall, our findings suggest the presence of a language-specific latent space, in line with Zhao et al. [2024]. Specifically, our results indicate that the concept space is largely English-centric, consistent with prior work Zhong et al. [2024], Wu et al. [2024], where English is the dominant training language. However, unlike previous studies, we uncover additional nuance: non-lexical words are not necessarily routed through the English representation space. Moreover, we find that the behaviour varies across models. One potential explanation is that we focus on open language

[4]They used tracing to mitigate potential shortcomings of cosine-similarity [Steck et al., 2024]

generation – which differs from prior work that predominantly focuses on single-token and translation tasks. We provide a more in-depth discussion in Section 4.8.

4.6.2 Multilingual LLM Behaviour

The internal mechanisms of LLMs affect their performance in several different ways, which we summarise below.

Performance The performance of multilingual language models varies across languages Shafayat et al. [2024], Huang et al. [2023], Bang et al. [2023], Shi et al. [2022], Ahuja et al. [2023], often performing best in English. This can even be leveraged to improve performance in other languages, for example, through cross-lingual chain-of-thought reasoning Chai et al. [2024], or by modifying prompts, such as using multilingual instructions or asking the LLM to translate the task into English before completing it Zhu et al. [2023], Etxaniz et al. [2023].

Fluency and language confusion Marchisio et al. [2024] showed that English-centric models are prone to language confusion, i.e., providing the answer in the incorrect language. Moreover, even when LLMs output text in the correct language, they can produce unnatural sentences in other languages, akin to an accent Guo et al. [2024].

Bias and culture Moreover, LLMs tend to be biased toward certain cultures, with content performing better when dealing with facts originating from Western contexts Naous et al. [2024], Shafayat et al. [2024], falling short when answering questions on other cultures Chiu et al. [2024]. Liu et al. [2024] investigate the cultural diversity of LLMs using proverbs and find that these models often struggle to reason with or effectively use proverbs in conversation. Their understanding appears limited to memorisation rather than true comprehension, creating a culture gap when translating or reasoning with culturally specific content across languages.

4.7 Conclusion

Our results provide evidence that semantic decisions in LLMs are predominantly made in a representation space close to English, while non-lexical words are processed in the prompt language. However, we find that this behaviour varies across models, likely due to differences in multilingual proficiency and model size. The English-centric behaviour is further validated by our findings that steering non-English prompts using vectors derived from English sentences is more effective than those from the prompt language.

Exploring the structure of the latent space, we find that factual knowledge across languages is stored in roughly the same regions of the model. Interpolating between the latent representations of these facts in different languages preserves predictive accuracy, with the only change being the output language. This suggests that facts encoded in different languages likely share a common representation. However, when interpolating, we find that the model output is most frequently in English, further underlining the English-centric bias of the latent space.

The English-centricity of the latent space is consistent with prior observations about LLM behaviour. In particular, Etxaniz et al. [2023] found that instructing LLMs to first translate a non-English prompt into English improves model performance. However, this bias can be detrimental. If the latent space is English-centric, this may lead the LLMs to exhibit Western-centric biases [Naous et al., 2024, Shafayat et al., 2024].

Future work could investigate ways to mitigate the English-centricity. We hypothesise that training with a more diverse corpus leads to less English-centric behaviour, as observed in Aya-23-35B. However, retraining was beyond our computational capacity.

4.8 Discussion

There are currently two perspectives in interpretability research on concept representations in multilingual models: (1) concept representations are universal; and (2) concepts have language-centric representations, where the language is the training-dominant language. Our work aligns more closely with the second perspective – namely, that LLMs encode language-specific representations. However, we argue that there are two sources: the training-dominant language and input/output language. Below, we discuss how the different theories may be reconciled.

Wendler et al. [2024] and Dumas et al. [2024] argue that the concept space is universal, but likely more aligned with the English output space. However, our findings contest this conclusion, as we find that interventions in the latent space are more effective when using English text, even when the target language is not English. If the concept space were truly universal, we would expect interventions using all languages to perform equally well. Our findings are consistent with concurrent work by Wu et al. [2024], who similarly find that steering using English performs comparably to, or slightly better than, the target language.

One possible way to reconcile the two theories is via the difference between concepts that are encoded and concepts that are used [as discussed in Brinkmann et al., 2025]. There may be multiple representations of any given concept [Hase et al., 2023, McGrath et al., 2023], or a concept may be represented in an LLM but not used during generation. Wendler et al. [2024] focus more on the encoding of concepts, whereas our work focuses more on the generation of text.

An alternative explanation is that different behaviour is captured in the each task. In Wendler et al. [2024], Dumas et al. [2024], the tasks are designed to generate a single token. In this setting, the task is to select the correct token, and we expect a high probability mass on a single token. In contrast, we focus on a more open-ended setting where there are several different possible continuations. These two settings are inherently different, leading to different conclusions about the behaviour of LLMs. Even within the same task of fact retrieval, prior work found that different components of the forward pass are language-specific and language-agnostic [Fierro et al., 2025].

More generally, the open-generation setting allows us to analyse different parts of speech. This leads to the second main difference in conclusions, which is that LLMs encode language-specific representations. For semantically loaded words, we find evidence that the latent space is English-centric (in LLMs where English is the dominant training language). This is consistent with one line of prior work, which generally focuses on nouns [Wu et al., 2024, Zhong et al., 2024]. However, we find that the same pattern does not hold for non-lexical words.

This is in contrast to concurrent work by Brinkmann et al. [2025], who showed that models share morpho-syntactic concept representations across languages in Llama-3-7b and Aya-8B. In line with their previous work [Wendler et al., 2024], they argue that the representations are universal. While our high-level conclusions differ, our findings also support the hypothesis that smaller models emit more shared representations than larger models, which permit more language-specific representations.

Lastly, there is an inherent challenge in disentangling whether a model thinks in English or whether its encoder and decoder perform better in English than in other languages. One way to probe this distinction is to ask whether the concepts represented by the model encoded English versions of those concepts. Zhang et al. [2025] explore this idea – for example, if we ask an LLM which colour a pumpkin is does it answer green or orange? In Japan, pumpkins are typically green whereas in English-speaking countries, they tend to be orange. Their findings provide further evidence that LLMs exhibit English-centric concept biases.

In conclusion, our findings suggest that the sharing of representations across languages is more nuanced than previously assumed. Representations appear both to reflect universal structure–yet often centred on English, the dominant training language–and to be shaped by the specific input or output language. Conceptually, this is significant: universal representations capture abstractions that extend beyond individual languages. We explore this idea of generality in Chapter 6.

Part III

Human-Understandable Explanations

5 Leveraging Uncertainty for Realistic Prototypes

In the previous part, we focused on understanding *what* knowledge is encoded in machine learning models and *how* it is represented. In this part, we turn to the second goal: how this knowledge can be extracted in a form that is interpretable to humans.

This chapter explores how to generate explanations that are both useful and understandable to end users – whether the general public, in the case of algorithmic recourse or experts in the medical domain, In particular, we focus on prototypical explanations and propose a method for generating explanations that are both realistic and unambiguous by leveraging model uncertainty.

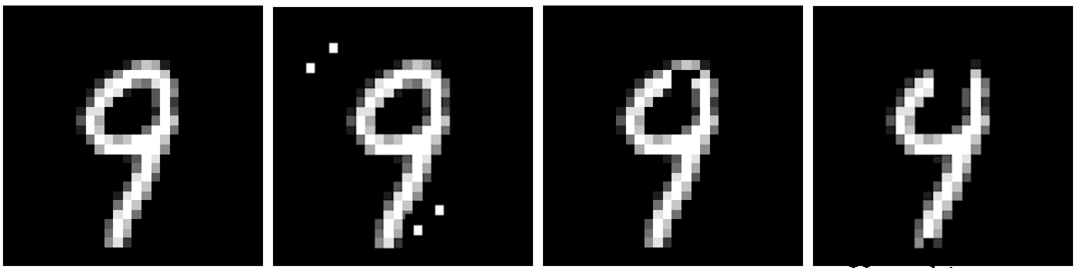
5.1 Introduction

Machine learning systems are increasingly used to make decisions with real-world consequences. As these systems are deployed in high-stakes settings, there is a growing need to explain *why* a particular prediction was made [Sartor and Lagioia, 2020]. Explanations help users understand *which factors* influenced a decision, and *what changes* might lead to a different outcome.

One setting where this is particularly important is algorithmic recourse, where the goal is to provide individuals–such as someone denied a loan–with actionable guidance: what they could realistically change to receive a different result [Spangher et al., 2018]. Another critical domain is healthcare, where explanations not only help build trust in predictions but can also offer insights into disease pathology.

In this chapter, we focus on prototypical explanations. In particular, counterfactual explanation (CE) of the form "*If X had not occurred, then Y would not have occurred*" [Wachter et al., 2017]. For example, consider the following binary classification problem: "*Given the current specifications of my house (e.g., location, number of bedrooms, etc.), am I likely or unlikely to sell it for $300,000?*". On inputting the details of their apartment, the user might receive the classification "unlikely". In this example, a CE could be the same house with upgraded furnishings to increase the desirability, resulting in the classification "likely".

Methods for generating CEs focus on finding an alternate input that is *close* to the original input, but with the desired classification [Molnar, 2020]. However, this highlights a fundamental difficulty in designing CEs, namely their similarity to adversarial examples [Szegedy et al., 2013, Goodfellow et al., 2014]. Both CEs and adversarial examples search for a minimal perturbation to add to the original input that changes the classification. The distinguishing conceptual feature is interpretability: while CEs should be interpretable, adversarial examples need not be. [1] However, interpretability is an ambiguous term, with varying definitions in existing literature [Lipton, 2018].



Original | Unrealistic CE | Ambiguous CE | Unambiguous, Realistic CE

Figure 5.1: Examples of possible CEs for an input image of the digit 9 when changing the classification to 4. From left to right: the original image, an unrealistic CE, an ambiguous CE (it can be interpreted as either a 4 or 9), a realistic and unambiguous CE (output from our algorithm).

We propose defining an interpretable CE as one that is *realistic*, i.e., a likely scenario for the user in question, and *unambiguous*, i.e., not a pathological 'borderline' case. Figure 5.1 provides an illustration of these two properties for an MNIST image [LeCun et al., 2010]. Here, we want to find a minimal change to alter the original image of a 9 so that it is 4. Second from the left is an example of a CE that is

[1]Although there is a common conception that adversarial attacks generate *imperceptible changes*, the term 'perceptible' is ill-defined, and many adversarial perturbations are visible to the human eye [see e.g. Papernot et al., 2016, Sharif et al., 2016, Brown et al., 2017].

not realistic – it doesn't resemble a "normal" 4 found in MNIST. Third from the left is an example of an ambiguous counterfactual; it is unclear whether it depicts a 4 or 9. In the final image is a CE that is both realistic and unambiguous, which is clearly preferable. We give a more extensive definition of realism and unambiguity in 5.2.

Existing work largely focuses on generating realistic CEs, and does not consider ambiguity [Wachter et al., 2017, Dhurandhar et al., 2018, Joshi et al., 2019]. Additionally, many of these approaches rely on using an auxiliary generative model, in addition to the classifier, to either generate realistic CEs or evaluate the realism of CEs in order to guide a search process. This may impose a bottleneck, as generative models may be poorly suited for some datasets and often require additional engineering and maintenance.

In this chapter, we propose capturing realism and ambiguity using the predictive uncertainty of the classifier. We leverage two types of uncertainty: epistemic and aleatoric uncertainty [Kendall and Gal, 2017]. Epistemic uncertainty is uncertainty due to a lack of knowledge, stemming from observing only a subset of all possible data points. We argue that CEs for which the classifier has low epistemic uncertainty are more realistic, as they are more likely under the data distribution. Aleatoric uncertainty captures inherent stochasticity in the dataset, such as proximity to the decision boundary between two classes. Therefore, CEs with lower aleatoric uncertainty will have lower ambiguity. We discuss both types of uncertainty in more detail in Section 5.3.1.

Based on these insights, we propose a new method for generating interpretable CEs by using a classifier that provides estimates of both epistemic and aleatoric uncertainty. Our approach avoids the need for a separate generative model and requires less hyperparameter tuning than existing methods. Existing neural network classifiers can often modified to capture uncertainty without incurring additional overhead, for example, by using Monte Carlo dropout [Gal and Ghahramani, 2016]. Moreover, in many real-world applications where explanations are required, uncertainty estimates are also important for downstream decisions. In such cases, uncertainty-aware classifiers may already be in use, and their outputs can be repurposed to generate counterfactuals with minimal additional cost.

In summary, our contributions are that we:

- link the concepts of aleatoric and epistemic uncertainty to the concepts of unambiguous and realistic CEs (5.3.1),
- introduce a new method for generating interpretable CEs based on implicit minimisation of both epistemic and aleatoric uncertainty (5.3.1),

- demonstrate empirically, from both a qualitative and quantitative perspective, that our method generates more interpretable CEs than existing methods, despite not requiring an auxiliary model (5.3.5).

5.2 Explanation Desiderata

We begin by clarifying the terminology used for counterfactual explanations. Consider an input x, the goal is to find an alternative input x' such that the model's prediction changes. We can write this as $x' = x + \Delta$, where Δ is the minimal change required to alter the prediction. Throughout this chapter, we will use *explanation* to refer to x', and counterfactual *perturbation* (CP) to refer to Δ.

Next, we define the desiderata of CEs. Explanation desiderata are subjective, and some are not mentioned below. Our goal is not to define an exhaustive list of all possible desiderata, but to make explicit the framework and targets we consider in this chapter.

To illustrate each desideratum, we use the example introduced in the introduction: a landlord has a two bedroom, one bathroom, one garage house in Boston with a small garden. A classifier answers the question "Is this property likely to sell for $300,000?" with `False`. The goal is to generate counterfactual explanations of the form "*If the property had X, then the classifier would return* ***`True`***".

Minimal The CE should be as similar as possible to the original instance, i.e. there should be as few changes as possible between x and x' [Huysmans et al., 2011, Wachter et al., 2017, Molnar, 2020, Laugel et al., 2019, Van Looveren and Klaise, 2019]. Minimal changes result in more concise and interpretable explanations, reducing cognitive load for users [Lahav et al., 2018]. For example, consider the following two CPs that both change the classification to `True`:

Δ_1 repainting the kitchen

Δ_2 repainting both the kitchen and the bathroom

As both obtain the desired outcome, Δ_1 is more desirable as it is more concise.

Realistic Explanation The suggested explanation must reflect a "possible world" [Wachter et al., 2017]. Realistic explanations are important for several reasons. They are often easier for users to understand, as they resemble familiar scenarios. More importantly, they are more likely to reflect true real-world dynamics.

For example, the explanation "*if the garage were rebuilt into 100 small rooms, then the house would likely sell for* $300,000$" is clearly unrealistic and unlikely to affect the outcome in practice. In contrast, "*if the garage were rebuilt into an ensuite bedroom*" is a plausible and informative alternative. In addition, the feature values should be realistic when considered *together* [Joshi et al., 2019]. For example, a one-bedroom house with 6 bathrooms would not be a realistic explanation as most houses have a higher bedroom-to-bathroom ratio.

Unambiguous Explanation CEs should be unambiguous to be *informative.* In this context, we take informative to mean explanations that humans can understand and learn from. For example, doctors may be interested in informative explanations from a breast cancer detection model.

Ambiguous inputs may be classified with a low confidence score, resulting in 'borderline' cases or inputs that resemble multiple classifications. For example, an 'ambiguous' house specification is one that one buyer might value over $300,000, but another buyer might value under $300,000. For a visual example, see 5.1, where the input resembles both a 4 and 9.

Realistic or Actionable Perturbation It must be possible for the user to apply the suggested CP in practice. While the 'Realistic Explanation' property ensures that the *explanation* is a possible instance, it will only provide the user with recourse if it is possible for them to apply the suggested *perturbation* to transition from their original input to the explanation. For example, while having an identical house to the original but in New York City would be a realistic counterfactual, it is not an actionable perturbation because the user cannot move their house to a different city.

Run Time of the Algorithm The algorithm must generate counterfactual examples (CEs) efficiently enough to be usable in practice [Van Looveren and Klaise, 2019]. While other computational properties of the algorithm, such as memory usage, are also important, we highlight run time because recourse is often delivered through an interactive, user-facing system, where latency directly affects user experience. In these settings, the algorithm must produce CEs fast enough to support real-time interaction. However, many CE generation methods rely on non-convex optimisation and repeated evaluations of a potentially expensive model. As a result, run time becomes a primary bottleneck.

Our approach is designed to satisfy all of the desiderata outlined above. We explicitly target the *unambiguous* and *realistic* desiderata through the design of our loss function. These properties are particularly important, as they distinguish counterfactual examples from adversarial examples. The remaining desiderata are addressed implicitly through the optimisation procedure that underlies our CE generation algorithm. In the next section, we introduce our method and explain how each desideratum is incorporated.

5.3 Leveraging Uncertainty for Explanations

In this section, we introduce our method for generating interpretable CEs. Our approach is motivated by the idea that realism and unambiguity can be captured through epistemic and aleatoric uncertainty. We show that minimising both forms of uncertainty can be achieved efficiently by optimising a cross-entropy objective of specific model classes. Based on these insights, we present a fast, greedy algorithm that generates minimal perturbations for a given input that minimises both types of uncertainty, resulting in interpretable explanations. Our method is post hoc: it operates on a trained classifier to generate CEs.

5.3.1 Uncertainty as a Proxy for Realism and Unambiguity

We begin by following Wachter et al. [2017] in framing the task of generating CEs as an optimisation problem. Given an input x, we can generate an explanation x' with target class y' by solving

$$x' = \arg\min_{\bar{x}} \lambda \ell(f, \bar{x}, y') + h(\bar{x}), \tag{5.1}$$

where f is the classifier, $\ell(\cdot)$ is a loss function, $\lambda \in \mathbb{R}^+$ is a hyperparameter[2], and $h(\cdot)$ is a metric of interpretability (for which lower is better). Intuitively, we want to generate an explanation in class y', which is encouraged by $\ell(\cdot)$, and is interpretable, as encouraged by $h(\cdot)$. The main difficulty is the definition of $h(\cdot)$, the measure of interpretability. As previously introduced, we define $h(\cdot)$ by considering two key aspects of interpretability: realism and unambiguity.

Prior work has shown that realism is one of the most difficult desiderata to satisfy (see Section 5.5). As a result, we place particular emphasis on targeting this property. To do so, we generate counterfactuals x' for a target class y' by maximising $\mathcal{D}(x' \mid y')$, the distribution of training examples in that class. This idea builds on insights from previous work [Dhurandhar et al., 2018, Joshi et al., 2019, Van Looveren and Klaise, 2019], which we review in Section 5.5.1.

[2] λ can be selected by iteratively solving the optimisation problem and increasing λ until x' is classified as y' with a loss below a user selected threshold.

The core intuition is simple: explanations that are likely under the training distribution are more likely to be accurately captured by the model. In other words, the model is more likely to behave reliably on examples that resemble the data it was trained on, making these counterfactuals more faithful to the true data-generating process. Such examples also tend to appear more familiar to users, and are therefore more likely to be perceived as realistic.

When generating CEs, we focus on the distribution for the target class, i.e. $\mathcal{D}(x' \mid y')$, in order to generate examples which look realistic for the particular target class y'. For example, it would not be realistic for a house classified as expensive to be very small and in a cheap area.

Given this definition of *realistic*, Bayes' rule gives us the following expression for the un-normalized density function,

$$p_{\mathcal{D}}(x'|y') = \frac{p_{\mathcal{D}}(y'|x')p_{\mathcal{D}}(x')}{p_{\mathcal{D}}(y')} \tag{5.2}$$

$$\propto p_{\mathcal{D}}(y'|x')p_{\mathcal{D}}(x'). \tag{5.3}$$

If we use a standard classification model with a softmax output, then $p_{\mathcal{D}}(y'|x')$ is estimated by the output of the model. To compute $p_{\mathcal{D}}(x')$, the likelihood of x' under the training data distribution, we have several choices. One option would be to use a separate generative model to estimate $p_{\mathcal{D}}(x')$. This would lead us to a similar objective to that introduced by Dhurandhar et al. [2018] and Van Looveren and Klaise [2019].

Instead of training an additional model to estimate $p_{\mathcal{D}}(x')$, we can approximate it directly using a classifier that provides uncertainty estimates over its predictions [Smith and Gal, 2018, Grathwohl et al., 2020]. In particular, we focus on epistemic uncertainty – which captures uncertainty about which function best explains the data, due to the fact that many functions may fit the finite training set. In the input space, a Bayesian classifier will have lower epistemic uncertainty on points which are close to the training data, and the uncertainty will increase as we move away from the training data. As a result, low epistemic uncertainty is correlated with a higher $p_{\mathcal{D}}(x')$. Gal and Smith [2018] show empirically for Bayesian neural networks implemented using deep ensembles. Therefore, given a classifier that estimates epistemic uncertainty, we can use it to approximate an unnormalised value of $p_{\mathcal{D}}(x' \mid y')$.

Lastly, we consider how to generate unambiguous counterfactual examples. To capture ambiguity, we rely on aleatoric uncertainty – which arises from inherent noise or stochasticity in the data distribution [Smith and Gal, 2018]. To ensure unambiguous explanations, we generate counterfactuals in regions of the input space where the classifier exhibits low aleatoric uncertainty.

5.3.2 Uncertainty in Practice

There are several different approaches for obtaining classifiers that offer estimates of epistemic and aleatoric uncertainty. For the experiments in this section we choose an ensemble of deep neural networks, as this is a simple method for computing high-quality uncertainty estimates [Lakshminarayanan et al., 2017]. Contrary to other methods for estimating uncertainty in deep learning, deep ensembles do not impose constraints on the architecture class of the classifier. However, our approach will work with any model that offers uncertainty estimates.

We define $h(x')$ as the predictive entropy of the classifier when evaluated on input x'. Predictive entropy captures both aleatoric and epistemic uncertainty, and both are low when the predictive entropy is low. The predictive entropy estimated using ensembles is

$$h(x') = -\sum_{y \in \mathcal{Y}} \bar{p}(y|x') \log \bar{p}(y|x') \tag{5.4}$$

$$\bar{p}(y|x') = \frac{1}{M} \sum_{m} p_m(y|x'), \tag{5.5}$$

where we have M models in the ensemble [Smith and Gal, 2018]. Here, $p_m(y|x')$ is the softmax output of the mth model in the ensemble.

Having defined $h(x')$ as the predictive entropy, we note that the term $h(x')$ in 5.1 is redundant. This is because a counterfactual that minimizes the cross-entropy (i.e., that maximizes the probability to be assigned to a class) must also minimize the predictive entropy (i.e., be likely under our approximation of the data distribution). Intuitively, the cross-entropy is minimized (equal to 0) when the target class has a probability 1 and all other classes have a probability 0. In this scenario, predictive entropy will also be minimized at 0. We provide a formal derivation in D.1.

As a result of this intuition, we can drop the terms λ and $h(\cdot)$ from 5.1 and objective becomes

$$x' = \arg\min_{\bar{x}} \ell(f, \bar{x}, y'). \tag{5.6}$$

This simplification of the objective makes it cheaper and easier to generate CEs. We avoid the iterative optimization of the hyperparameter λ, which might otherwise increase the computational cost of the optimization. As we discuss in detail in 5.5.1, this is an improvement over existing approaches such as Wachter et al. [2017], which requires tuning of λ, and Van Looveren and Klaise [2019], which uses an objective with several hyperparameters.

5.3.3 A Greedy Algorithm for Generating Minimal CEs

In this section, we present the algorithm used to generate counterfactual examples, targeting the objective defined in Equation 5.6. Optimising this loss function directly yields a sample in the target class y', but does not account for the minimality or realistic perturbation desiderata. To address these, we constrain the optimisation process through the design of the algorithm itself. Specifically, we adapt the Jacobian-based Saliency Map Attack (JSMA), originally proposed by Papernot et al. [2016] for generating adversarial examples.

JSMA is an iterative algorithm that updates the most salient feature, i.e. the feature that has the largest influence on the classification, at each step. To generate realistic CEs rather than adversarial examples, we replace the original definition of saliency by defining the most salient feature as that which has the largest gradient with respect to the objective in 5.6:

$$\text{most\_salient\_feature}(x') = \arg\max_{j} \nabla_{x'_j} \left[\ell(f, x', y') \right], \tag{5.7}$$

where ∇ denotes the partial derivative and x'_j denotes the jth feature of x. Updating each feature iteratively by δ acts as a heuristic for minimising the L_0 distance between the original input and the CE [Papernot et al., 2016]. The algorithm terminates when the input is classified as the target class with *high confidence*, or after reaching the maximum number of iterations. Alternatively, the algorithm can be configured to fail if the explanation does not reach the predefined confidence level. This enforces that generated explanations are those on which the classifier has low uncertainty, which may be important for certain applications. We give pseudocode in 1.

This is a fast algorithm for generating realistic and unambiguous explanations using minimal perturbations. We also want to ensure that the perturbation is actionable. In many cases, we can manually identify the features a user cannot change, and lock these features to prevent the algorithm from perturbing them. For example, we might prevent the algorithm from changing the location of a house. This simple approach assumes that we are explicitly aware of factors that can be changed, which is often but not always the case. We leave a detailed investigation into other approaches for generating realistic perturbations for future work.

Algorithm 1 Generating Counterfactuals
(For more detail, see D.2)

Input original observation x; target class y'; ensemble of models $\{f_m\}_{m=1}^{M}$; maximum number of iterations N; minimum confidence of target class γ; perturbation size δ; maximum number of times each feature is changed n; optional: a function that clips the values to a permitted pre-defined range clip.
Output counterfactual x'
$x' \leftarrow x$
$c \leftarrow 0$
$P = 0_{n_p}$, where n_p is the number of input features
while $\frac{1}{M}\sum_m p_m(y'|x') \leq \gamma$ and $c \leq N$ **do**
  Compute forward derivative $S(x', y') = \nabla_{x'} \frac{1}{M} \sum_m^M \ell(f_m, x', y')$
  $i = \text{argmax}_{i:i \in P, P[i] < n} S(x', y')[i]$
  $x'[i] = x'[i] + \text{sign}(S(x', y')[i]) \cdot \delta$
  $x' = \text{clip}(x')$
  $P[i] \leftarrow P[i] + 1$
  $c \leftarrow c + 1$
end while
return x'

5.3.4 Adversarial Training

Our proposed method works with any classifier that both offers uncertainty estimates and for which we have access to the gradients (of cross-entropy loss with respect to some input). However, if it is possible to retrain the classifier then the realism of the generated explanations can be improved by applying adversarial training, as we demonstrate empirically in 5.3.5. Specifically, we augment the dataset during training using adversarial examples generated by FGSM [Goodfellow et al., 2014], see D.3 for details.

We suggest that adversarial training might improve the realism of the generated CEs for two reasons. First, Lakshminarayanan et al. [2017] demonstrate that adversarial training improves uncertainty estimation, both on in-distribution and out-of-distribution inputs. This should improve the performance of our method, as we generate CEs in areas of input space where the classifier has low uncertainty.

Second, adversarial training can lead to learning more robust features [Tsipras et al., 2018, Ilyas et al., 2019]. Augmenting the training set with adversarial examples during training ensures that the model does not focus on noise when learning features for classification. As such, the model is more likely to learn features that are not noise, and therefore are more interpretable [Tsipras et al., 2018]. An example of the effect of adversarial training is shown in 5.2. The saliency of an

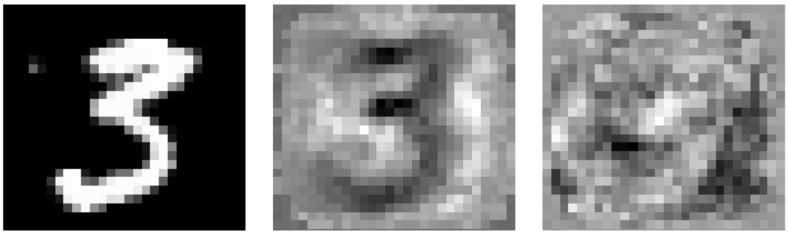


Figure 5.2: Gradients of classifiers trained with adversarial training (middle image), versus without (right image) for an input image (left image). We observe that adversarial training results in gradients (which can used to identify a salient features) that are more aligned with human perception. This example is inspired by Tsipras et al. [2018, Figure 2].

adversarially trained model, as shown by the middle image, is more aligned with human interpretation than the saliency of a regular model (shown by the right image). We discuss these two effects further in D.3.

5.3.5 Evaluation Metrics

To evaluate counterfactual explanation (CE) generation algorithms, we assess two key properties: realism and minimality. To measure minimality, we report the L_1 distance between the original input and the counterfactual. To measure realism, we adopt the metric proposed by Van Looveren and Klaise [2019], based on reconstruction losses from autoencoders. Two variants are defined as follows:

$$\mathrm{IM1}(x', y, y') = \frac{\|x' - \mathrm{AE}y'(x')\|_2^2}{\|x' - \mathrm{AE}y(x')\|_2^2 + \epsilon} \quad \mathrm{IM2}(x', y') = \frac{\|\mathrm{AE}_{y'}(x') - \mathrm{AE}(x')\|_2^2}{\|x'\|_1 + \epsilon} \tag{5.8}$$

Here, AE_y denotes an autoencoder trained on instances from class y, and AE is trained on all classes.

IM1 computes the ratio between the reconstruction loss under the counterfactual class and the original class. A lower value indicates that the counterfactual resembles the target class more closely than the original. IM2 measures the normalized difference between reconstructions from the class-specific and general autoencoders.

We use IM1 to evaluate the realism of generated counterfactuals. We omit IM2, as it fails a basic sanity check: it does not reliably distinguish between in-distribution data and out-of-distribution "junk" inputs. See Appendix D.4 for further discussion.

5.4 Empirical Analysis

We train a classifiers for each of the datasets – D.5 gives full details of the models we use in each experiment.

5.4.1 Datasets

We perform our analysis on three datasets: MNIST [LeCun et al., 2010], the Breast Cancer Wisconsin Diagnostic dataset [Dua and Graff, 2017], and the Boston Housing dataset [Dua and Graff, 2017]. We chose MNIST as it is easy to visualize, which allows non-experts to evaluate the interpretability of the generated CEs. We consider the two tabular datasets because this type of data is frequently used in the interpretability literature. We briefly describe each dataset below.

MNIST dataset A dataset containing gray-scale images of handwritten digits ranging between 0 and 9. The goal of the CE explanation is to find a perturbation that changes the image classification from, e.g., a 1 to a 7. We consider MNIST, as image-based data allows us to visually inspect the quality of the CEs. Our classifiers obtain an accuracy of 98.5% on the test set.

Breast Cancer Wisconsin Diagnostic dataset A tabular dataset where each row contains various measurements of a cell sample from a tumour, alongside a binary diagnosis of whether the tumour is benign or malignant. A CE for a particular input changes the classification from benign to malignant, or vice-versa. Our classifiers obtain an accuracy of 96.9% on the test set.

Boston Housing dataset A tabular dataset where each row contains statistics about a suburb of Boston, alongside the median house value. To construct a classification problem, we divide the dataset into suburbs where the price is below the median, and those where it is above. A CE for a particular input changes the classification from below the median to above, or vice-versa. Our classifiers obtain an accuracy of 86.3% on the test set.

5.4.2 Benchmarks

We benchmark the performance of our method against Van Looveren and Klaise [2019], a state-of-the-art approach for generating CEs. We also compare against JSMA, the adversarial attack from which we draw inspiration for our algorithm. We include JSMA for two reasons: [1] to determine that we are able to generate more interpretable counterfactuals, and [2] to validate that JSMA can efficiently create minimal perturbations.

Table 5.1: Realism (IM1) and minimality (L_1) of generated explanations. IM1 serves as a proxy for the distance to the class data manifold, measured via reconstruction error (see Section 5.3.5); lower values indicate more realistic. The L_1 distance is measured between the original input and the counterfactual example; lower values are better. Results are reported as the mean over 100 randomly selected test points, averaged across 10 random seeds, and the standard deviations across seeds are shown in parentheses. VLK refers to the method introduced by Van Looveren and Klaise [2019]. Note that the reported values for VLK are based on our own implementation and differ from those in the original paper. While we improve upon their reported results on MNIST, we obtain a worse performance on the Breast Cancer dataset. We discuss the steps we took to reproduce their results in Appendix D.5.2.

Method	**Realism** (IM1 $\downarrow$)		**Minimality** ($L_1 \downarrow$)	
	MEAN	STD	MEAN	STD
		MNIST		
JSMA	1.11	(0.01)	14.4	(2.3)
VLK	1.12	(0.06)	47.7	(4.9)
OURS	0.98	(0.02)	38.3	(3.5)
		BREAST CANCER DIAGNOSIS		
JSMA	1.06	(0.01)	1.23	(0.04)
VLK	2.81	(0.27)	1.27	(0.09)
OURS	0.89	(0.04)	2.57	(0.13)
		BOSTON HOUSING		
JSMA	1.50	(0.00)	0.72	(0.01)
VLK	2.55	(0.08)	0.73	(0.20)
OURS	0.85	(0.00)	1.47	(0.02)

5.4.3 Results

Table 5.1 shows our results. Overall, we find that our approach generates more realistic CEs than Van Looveren and Klaise [2019] despite not requiring an auxiliary generative model, as can be seen from the lower IM1 scores. Comparing our method to JSMA, we note that our method generates larger perturbations but with better IM1 scores. This demonstrates that our adapted loss function successfully trades off perturbation size for realism, as desired. We emphasise that we only report JSMA to show that it can efficiently create minimal perturbations, although it does not generate realistic explanations. This can be seen in 5.3, which shows qualitative examples of the explanations generated by the three methods.

5.3 shows qualitative examples of counterfactual examples. Here, we observe a failure mode of our proposed algorithm: the strokes in the counterfactual examples are less smooth than the strokes in real images. This is due to the algorithm design,

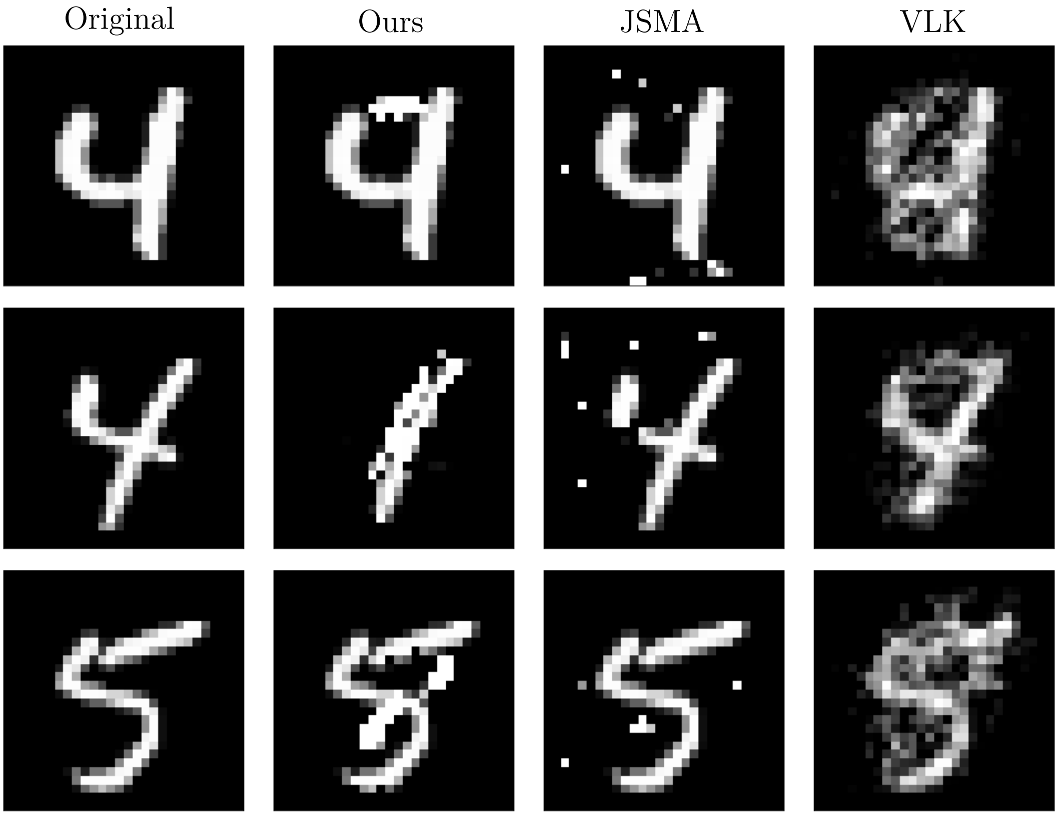


Figure 5.3: Qualitative Examples of Counterfactual Explanations. Each row shows a different example. From top to bottom, the goal is to change the classification from a $4 \rightarrow 9$, $4 \rightarrow 1$, and $5 \rightarrow 8$. The left column shows original images. The following three columns show examples of generated CEs by the algorithms. From left to right: our algorithm, JSMA, VLK. Our algorithm is able to create more realistic counterfactuals – i.e., images that are likely under the data distribution. We note that the CEs above generated by VLK appear less realistic than those shown in the original paper. This is likely because we consider *targeted* CEs, i.e., we randomly specify a target class for the explanation, whereas in Van Looveren and Klaise [2019] the explanation can be in any class, which is an easier task.

which changes single pixels iteratively. As a result, our method does not capture the stylistic properties of the dataset, which would be important if we want to employ our method as a generative model. However, our method is able to grasp high-level changes required, such as adding a white stroke to 4 to turn it into 9, which is more important for explanatory purposes. In D.6, we show more examples of CEs generated by our method on both MNIST and tabular data, and provide further insight into which features are altered.

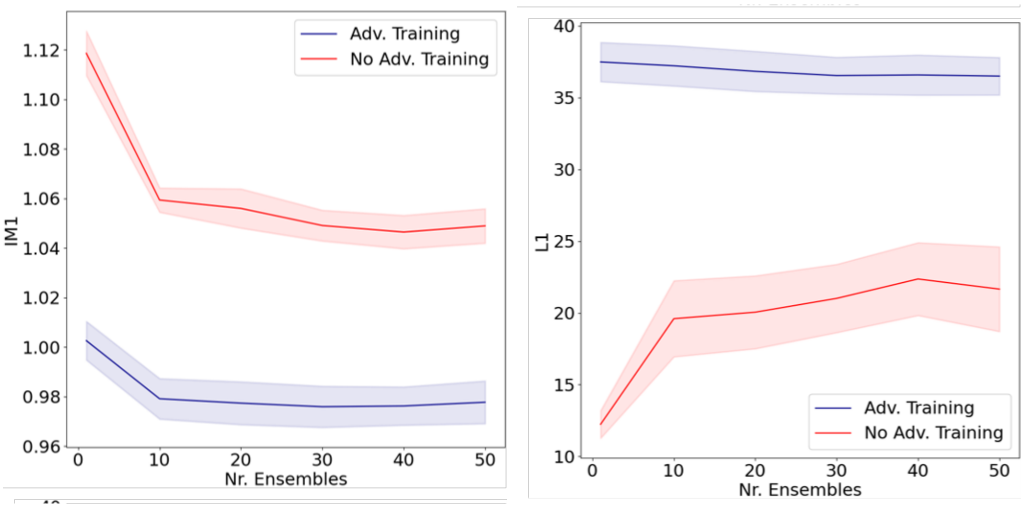


Figure 5.4: Ablation Study on the MNIST: effect of adversarial training and number of models in the ensemble on interpretability (top), as measured by the IM1 score, and minimality (bottom), as measure by the mean L_1 distance. The shaded areas show 95% CIs, as estimated over 10 seeds.

5.4.4 Ablation Study

We perform an ablation study to investigate the effects of adversarial training, and the number of models in the ensemble, on the quality of generated CEs for MNIST images. The results are shown in 5.4. Initially, the interpretability of CEs tends to improve as the number of ensemble components increases – this can be seen from the initial downward slopes of IM1 in the top graph of 5.4. However, after 10 ensembles, the improvement in performance saturates; likely because uncertainty estimation does not improve further. Adversarial training improves the interpretability scores, however leads to less sparse explanations.

5.5 Related Work

5.5.1 Generating CEs

Below we summarize the different methods used to generate CEs. We begin with Wachter et al. [2017], who frame the task of finding a CE x' in target class y' for initial input x as the optimization problem

$$x' = \arg\min_{\bar{x}} \lambda \ell(f, \bar{x}, y') + d(x, \bar{x}), \tag{5.9}$$

where f is the classifier, $\ell(\cdot)$ is a loss function (the authors use MSE loss), d is some measure of distance (the authors use a weighted L_1 distance), and λ is a hyperparameter which is selected by iteratively increasing it until the target class is predicted with a desired level of confidence. This is equivalent to the objective function we use in our approach, as given in 5.1, if $h(x')$ is defined as distance to the original input. This definition of $h(x')$ does not give any consideration to ensuring that x' is realistic, and Wachter et al. [2017] note that it risks generating adversarial examples.

Various approaches adapt 5.9 in an attempt to generate realistic CEs:

- Dhurandhar et al. [2018] include an additional penalty in the objective to encourage CEs to lie on the training data manifold. The authors fit an auxiliary autoencoder model to the training data. In the objective, they then include the reconstruction loss of applying this autoencoder to the CE. The assumption is that reconstruction loss will be higher for CEs which are not likely under the training distribution, which will encourage the approach to generate realistic CEs.

- Van Looveren and Klaise [2019] note that the approach introduced by Dhurandhar et al. [2018] does not take into account the data distribution of each class, for example a very large house is unlikely to also be very cheap. Thus, the authors include an additional loss term which guides the search process towards a 'prototype' instance of the relevant class. The prototype for each class is defined as the average location in a latent space of all the training points in that class. Again, the authors use an autoencoder to map inputs into the latent space. One disadvantage of this approach is that the objective function contains several terms, and a hyperparameter must be specified for each in order to scale them appropriately.

- The methods above generate CEs by searching in input space. In contrast, Joshi et al. [2019] search in a latent space, and use a generative model to map instances from this latent space into the input space in order to evaluate them. The objective is

$$x' = \underset{z \sim p(z)}{\arg\min}\, \lambda \ell \left(f(\mathcal{G}(z)), y'\right) + c\left(x, \mathcal{G}(z)\right), \tag{5.10}$$

  where $p(z)$ is a distribution over the latent space, $\mathcal{G}(\cdot)$ is the generative model mapping the latent space to input space, $c(\cdot)$ is a cost function (which we can view the same as the distance function), and λ is a hyperparameter. One limitation of this approach is that the CEs are produced by the generative model, thus suffer from the pathologies of that model. For example, a VAE is likely to generate blurry explanations.

We claim that our approach has several advantages over these methods. First, we avoid the engineering, and potential computational, overheads of implementing, training, and maintaining an auxiliary generative model. Second, we have a simple objective function which does not involve minimax optimization or specifying hyperparameters, both of which incur additional computational cost.

One weakness of our method is that it requires a classifier which offers reliable uncertainty estimates. This is likely to have higher computational cost, and in particular the ensemble of classifiers that we use in our experimental work is more expensive to train and evaluate than a single model. However, our method can be used with any classifier that offers both epistemic and aleatoric uncertainty, and several fast approaches are available for deep learning models [Gal and Ghahramani, 2016, Liu et al., 2020, Van Amersfoort et al., 2020]. Additionally, we argue that many applications where the machine learning system must offer the user recourse, estimates of the uncertainty in the classification will also be required, and so will already be available from the classifier. For example, when using a machine learning tool to make a decision, estimates of the uncertainty in the predictions are very important to be able to act cautiously, or defer to a human expert, when the model is unsure.

5.5.2 Adversarial examples

Counterfactual examples are closely related to adversarial examples [Pawelczyk et al., 2021, Freiesleben, 2021]. Adversarial examples are crafted by finding the minimal perturbations required to change the classification of an image Szegedy et al. [2013]. Mathematically, this can be formulated as

$$x' = \arg\min_{\bar{x}} \quad d(x, \bar{x}) \tag{5.11}$$

$$\text{s.t.} \quad f(\bar{x}) = y', \tag{5.12}$$

where x is the original input, x' is the adversarial example, y' is the target class, $d(\cdot)$ is a distance metric and $f(\cdot)$ is the classifier.

This is very similar to the mathematical formulation used to generate CEs in 5.9. In literature, the distinguishing feature between the two fields is *interpretability*. While we want counterfactual examples to be interpretable, adversarial examples need not be. Our approach focuses on this distinguishing feature; we design an algorithm that leverages uncertainty to generate interpretable CEs.

5.6 Conclusion

We have introduced a fast method for generating realistic, unambiguous, and minimal CEs. In the process, we collect, define, and discuss the properties which CEs should have. In comparison to existing methods, our algorithm does not rely on an auxiliary generative model, reducing the engineering overhead. Nevertheless, we demonstrate empirically that our approach is able to match or exceed the performance of existing methods, with respect to the realism of the CEs generated.

One could argue against the chosen desiderata – for example, that minimality is unnecessary, or at least secondary to realism. Yet without constraining optimisation for minimality, explanations risk becoming superfluous, including features that either (1) do not influence the outcome or (2) form a superset of those truly required. Moreover, our approach incorporates realism by enforcing a threshold that must be satisfied before optimisation terminates.

In future work, methodological developments could be explored by adapting the proposed method to work for black-box models [Afrabandpey et al., 2020]. Similarly, it would be interesting to extend these ideas to more complex models, such as LLMs. While using ensembles would be computationally infeasible, researchers could explore generating CEs that leverage the uncertainty in the latent space [Höltgen et al., 2021, Kossen et al., 2024].

It would also be valuable to compare a model's epistemic uncertainty with that of humans. From the perspective of learning and knowledge discovery, counterfactual explanations may be especially informative in cases where humans are uncertain but the model is confident. Conversely, inherently ambiguous examples can be pedagogically useful: as complex cases, they help humans refine their internal decision boundaries. More broadly, focusing on cases of maximal user uncertainty may reveal insights that are particularly informative for improving human understanding.

6
Knowledge Discovery in Chess

This chapter brings together the two central goals of this thesis: (1) understanding *what* knowledge is encoded in machine learning models, and (2) extracting that knowledge in a form that is interpretable to humans. In particular, we explore whether we can extract *new* knowledge from AI systems– knowledge that was not previously known to humans. We consider the setting of chess, focusing on AlphaZero, a highly capable system that plays at a superhuman level.

The chapter follows the interpretability pipeline introduced earlier in this thesis: identifying explanation desiderata, locating knowledge encoded in the model, verifying that it influences behaviour, and translating it into a human-interpretable form. Moreover, we demonstrate – through a small-scale study – that the extracted knowledge can be communicated to and learned by domain experts.

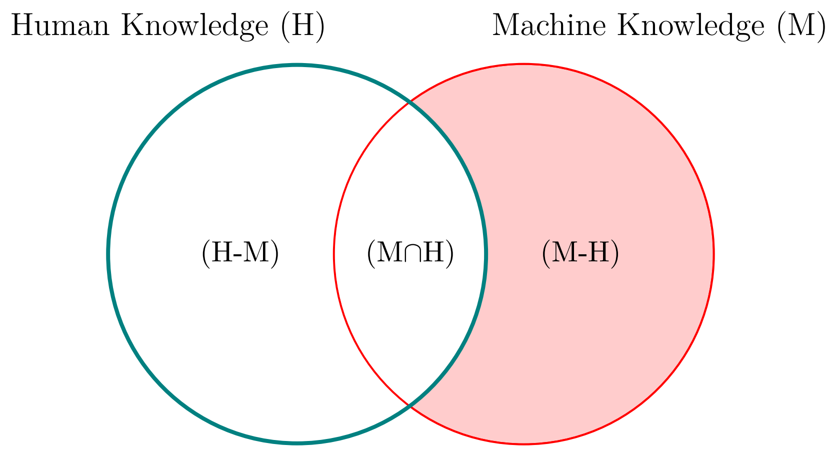
6.1 Introduction

Artificial Intelligence (AI) systems are often treated as problem-solving machines; they can carry out the jobs humans are already capable of but more efficiently or with less effort, which brings benefits in several domains. In this chapter, we pursue a different goal: treat AI systems as learning machines and *demand* that they teach us the fundamental principles behind their decisions to extend upon and complement our knowledge. We can imagine many benefits of learning from machines. For example, while a system that provides accurate cancer diagnosis or effective personalised treatment than human experts is useful, transferring the rationale behind their decisions to doctors could not only bring advances in medicine but also leverage human doctors' strength and generalisation ability to enable new breakthroughs. There is a tremendous untapped opportunity across various domains where the capabilities of AI systems are reaching or exceeding those of human experts (super-human AI systems).

The super-human capabilities of AI systems may arise in a few different ways: pure computational power of machines, new ways of reasoning over existing knowledge, or super-human knowledge we do not possess. In the next two chapters, we focus on the last two cases and for simplicity, refer to both as super-human knowledge from now on.

What does this mean from a research standpoint? The human representational space (H) has some overlap with the machine representational space (M) (see Figure 6.1 [Kim, 2022]). A *representational space* forms the basis of and gives rise to knowledge and abilities, which we are ultimately interested in. Thus, we use representational space and knowledge interchangeably – roughly speaking, H to represent what humans know and M to represent what a machine knows.

Figure 6.1: Learning from machine-unique knowledge, modified from Kim [2022] with permission. Teal represents human knowledge and pink represents knowledge encoded in AI systems. In the context of knowledge discovery, we are interested in extracting machine-unique knowledge from the shaded region ($M - H$).



We believe that the knowledge gap represented by ($M - H$) is crucial to empowering humans by identifying new concepts and new connections between existing concepts within highly performant AI systems.

One prominent example of (M-H) in the history of AI is move 37 – an exceptionally strong, novel idea – that AlphaGo made in the second game of its match against Lee Sedol. This move surprised both commentators and Lee Sedol, and is still discussed to this day as an example of machine-unique knowledge. If there are more such examples, exemplifying new conceptual knowledge that enables new capabilities, we would like to discover them using a structured approach.

In this chapter, we take a first step towards discovering super-human knowledge in $(M - H)$. We focus on a domain that has inspired AI practitioners for decades, and captivated human imagination for centuries: the game of chess. Chess is a well-suited domain to explore the existence and usefulness of set (M − H) for several reasons. First, chess knowledge has been developed over a long period of time, and the ground truth is easier to validate compared to the frontiers of other fields, such as science or medicine. Moreover, we have a quantitative measure of the quality of play, both for human experts as well as machines, known as the Elo rating [Wikipedia contributors, 2023c].

Chess engines have performed at a super-human level for a long time, ever since DeepBlue's match against Garry Kasparov. While early engines were based on human knowledge, AlphaZero [Silver et al., 2018] (AZ) was the first self-taught deep learning model that surpassed human capabilities without relying on any prior human knowledge. However, as humans, we have not yet been able to fully tap into their knowledge. Through analysis of AZ's games, humans manually distilled patterns, such as its proclivity to play on the flank of the chess board with moves such as `a4` or `h4` [Sadler and Regan, 2019]. However, this still analyses M through the lens of H, a bias that limits what we can find from $M \cap H$.

In this chapter, we aim to take the first step to change that by facilitating learning from the super-human knowledge in the $(M - H)$ set of AZ. We hypothesise that $(M - H)$ exists, and can be taught to humans. We validate our hypothesis by showing that we can teach new chess concepts to four top human grandmasters, the best chess players in the world. Also, due to their undeniable strength, and talent, $(M - H)$ may fall into their 'proximal zone of development' in Vygotsky's education theory: "the space between what a learner can do without assistance and what a learner can do with adult guidance or in collaboration with more capable peers". While communicating $(M - H)$ may require new language [Kim, 2022], we bypass this need in this work by leveraging chess champions' ability to connect the dots and generalise from patterns that arise in chess positions.

We develop a new framework to search for concepts in $(M - H)$, i.e., unearth AZ's super-human knowledge. In our framework, we:

- develop a **new method for finding unsupervised concepts** in the latent space. By using the full AZ machinery, both the policy value network and MCTS tree, our method discovers *dynamic* concepts that motivate a sequence of actions in chess. We show that our method can find vector representations of concepts in a data-efficient manner.

- ensure concepts are novel. Through spectral analysis, our framework only select concepts that contain information unique to the vector space of AZ's games compared to that of human games.[1]
- ensure that concepts are teachable. We develop a new metric that evaluates whether concepts are **teachable** to another AI agent with no prior knowledge of the concept. Through this metric, we select concepts based on their informativeness (i.e., useful for the AI agent in a downstream task).
- provide insight into the meaning of the new concepts via graph analysis to reveal new concepts' relations to human-labelled concepts.

We validate the new methodology in the supervised setting in Section 6.4.2. After finding these concepts in $(M - H)$, we analyse whether we can expand the human representational space (H) to include these new concepts. We collaborate with four world top chess grandmasters and former World Champions to test whether they could learn and apply the concepts, by learning from prototypical examples.

6.2 AlphaZero

AlphaZero (AZ) is the first chess AI system that mastered the game without any prior human knowledge [Schrittwieser et al., 2019]. Here, we provide a brief description; for a complete description, we refer the reader to Schrittwieser et al. [2019]. AZ leverages two main components: a policy value network and Monte Carlo Tree Search (MCTS).

Policy Value Network For a given input state, the policy-value network f_θ outputs a policy p and value estimate v:

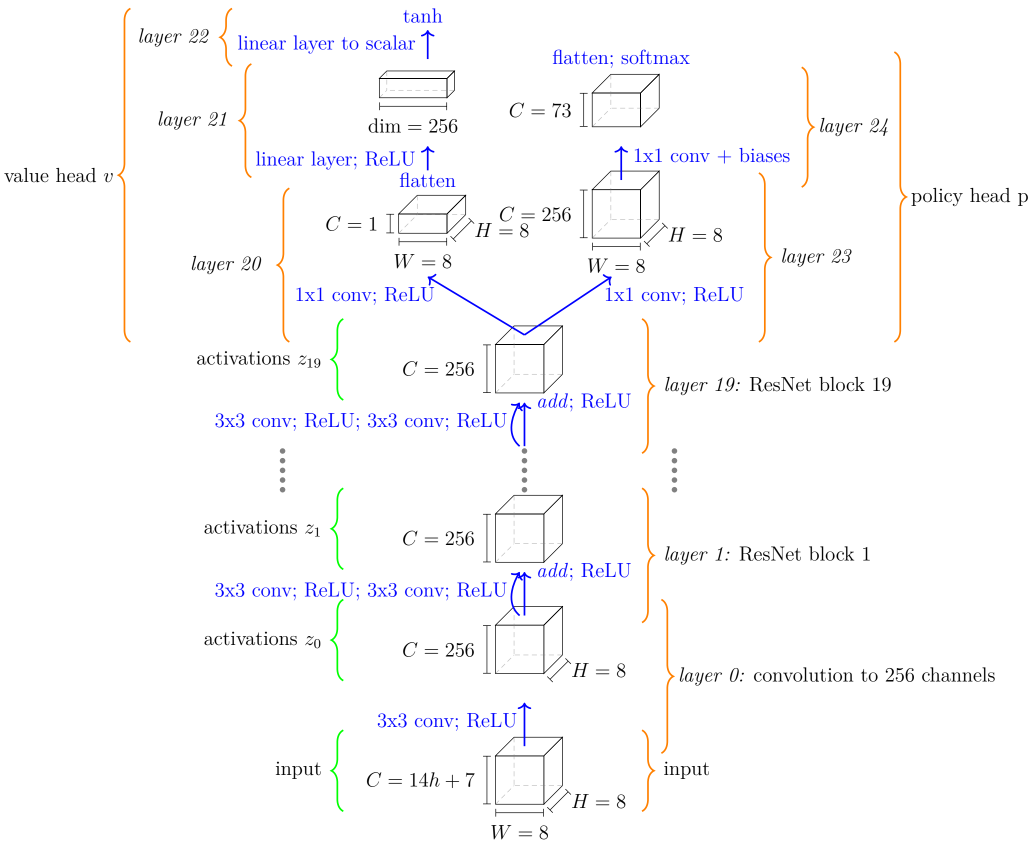
$$(p, v) = f_\theta(s), \tag{6.1}$$

where s is the input state. The input state contains the chess position, history and metadata required for legal moves, such as castling rights. The policy is the probability distribution over the next actions. The value estimate whether the player to move in the state will win (1), or whether the opponent is likely to win (-1).

The main body of the network is a ResNet backbone [McGrath et al., 2022a]. The body consists of 19 residual blocks. Each block contains two convolutional

[1]This difference is not due to variance, as the human games have a larger span in input space.

Figure 6.2: Policy Value Network in AlphaZero. Adapted with permission from ref. McGrath et al. [2022a], which is licensed under CC BY 4.0.



layers, followed by a skip connection. The post-activation latent representation z_l the corresponding to block l is given by:

$$z_l = \mathrm{ReLU}(z_{l-1} + g_l(z_{l-1})), \tag{6.2}$$

where $g_l(\cdot)$ is the composition of two convolution layers. The intermediate layers in PV network extract features before outputting a policy and value estimate.

Figure 6.2 shows an overview of the network. We refer to layers using integers. For the main body of the network, these integers denote the ResNet blocks. In the value head, we have three layers, which we will refer to as layers 20, 21 and 22 (from the body to value output). In the policy head, we have two layers, which we refer to as layers 23 and 24 (from body to policy output).

Search The search algorithm used in AlphaZero is Monte Carlo Tree Search (MCTS). For a state-action (s, a), the visit count $N(s, a)$, the total action value $W(s, a)$, mean action value $Q(s, a)$ and prior probability $P(s, a)$ are stored. $W(s, a)$, $Q(s, a)$ and $N(s, a)$ are initialised to zero. $P(s, a)$ is the probability assigned to action

a in the policy p of the policy-value network $f_\theta(s)$. During training, noise sampled from a Dirichlet distribution is added to the prior probabilities of the root node.

The search algorithm starts at the initial state s_0 and finishes at maximum depth T. At each time step $t < t$, the next action is chosen as follows:

$$\text{argmax}_a Q(s_t, a) + U(s_t, a) \tag{6.3}$$

The first term, $Q(s_t, a)$ is for exploitation whereas $U(s_t, a)$ ensures exploration. $U(s_t, a)$ is defined as:

$$U(s, a) = C(s)P(s, a)\frac{\sqrt{(N(s)}}{1 + N(s, a)}, \tag{6.4}$$

where $N(s)$ is the parent visit count and $C(s)$ is the exploration rate, define as:

$$\log \frac{1 + N(s) + c_{\text{base}}}{c_{\text{base}}} + c_{\text{init}} \tag{6.5}$$

The value v of the leaf node s_T is estimated using the policy value network.

The visit counts and values are updated in the backward pass:

$$N(s_t, a_t) \leftarrow N(s_t, a_t) + 1 \tag{6.6}$$

$$W(s_t, a_t) \leftarrow W(s_t, a_t) + v \tag{6.7}$$

$$Q(s_t, a_t) \leftarrow \frac{W(s_t, a_t)}{N(s_t, a_t)} \tag{6.8}$$

Training The PV network is trained using RL from self-play. Each game starts at the initial position. At every time step, a move is sampled proportionally to the root node visit count. Games are simulated until a terminal state is reached. A score of -1 is assigned for a loss, 0 for a draw and 1 for a win. The model parameters are trained using gradient descent on the loss:

$$\ell = (z - v)^2 - \pi^T \log p + c||\theta||^2, \tag{6.9}$$

where c is a hyperparameter controlling the L_2 regularisation.

Evaluation During evaluation, the moves are greedily sampled with respect to the root node visit count.

6.3 What are Concepts?

There are several possible definitions of a concept – varying from a human-understandable high-level feature to an abstract idea. In this work, we use the following definition:

Definition 6.3.1 (Concept)**.** A concept is a unit of knowledge

There are two properties we focus on. The first is that a concept contains *knowledge*: information that is useful. In the context of machine learning, we take this to mean that it can be used to solve a task.[2] For example, consider the concept of a beak. We can *teach* an algorithm or a person what a beak is. If the person grasps the beak concept, they can use it to identify birds. The second property is *unit*, which implies minimality; a concept is concise and irrelevant information has been removed.

There are many ways to operationalise this definition and properties, and we choose one of them: showing a concept can be *transferred* to another agent to help them solve a task (e.g., follow the strategy represented in a concept). Being able to do so implies that the concept is self-contained and useful for the task.

Concepts in Reinforcement Learning What types of concepts do we aim to discover in the RL setting? We aim to discover concepts that give rise to a plan, where a plan is a deliberate sequence of actions optimizing for one or more relevant concepts. We take deliberate to mean that there is an underlying reason. More specifically, we assume a plan is motivated by one or more concepts. Although the *terminal* goal of a plan is the same across states – maximizing the outcome (win or draw) – plans in a specific state will have more context-specific *instrumental* goals along the way, for instance, capturing a particular piece in an advantageous position, or maximising one's board control. We assume that plans in similar contexts will share similar instrumental goals, and thus give rise to similar concepts.

How do we represent concepts? We leverage rich literature that assumes concepts are linearly encoded in the latent space of a neural network [McGrath et al., 2022a, Kim et al., 2018, Gurnee et al., 2023, Conneau et al., 2018, Tenney et al., 2019, Nanda, 2023]. The latent space refers to the space spanned by post-activation features of a neural network. Although our assumption of linearity is a strong assumption, it has a surprising amount of empirical support: linear probing and

[2]The definition of a task can be broadly defined. Relating this to previous – in the context of LLMs, this task may simply be communicating a fact.

related techniques have successfully extracted a wide range of complex concepts from neural networks across multiple domains [Kim et al., 2018, Gurnee et al., 2023, Conneau et al., 2018, Tenney et al., 2019, McGrath et al., 2022a, Nanda, 2023]. Although we may miss concepts with non-linear representations, we nevertheless show that we can find useful concepts for our goal using purely linear representations.

6.4 Finding Concepts

Recall, Figure 6.1. Our goal is to find concepts in $(M - H)$. Our approach is as follows:

1. Find concepts in (M), using an unsupervised approach.
2. Ensure concepts are useful, using a teachability metric.
3. Ensure concepts are in $(M - H)$, using a novelty metric

We elaborate on each step in the following subsections.

6.4.1 Finding concepts in (M)

To find concepts in (M) without bias from (H), we need to use an unsupervised approach. We propose a new approach as existing methods are not well-suited for the type of data. Firstly, the model input is a mix of binary and real-valued inputs, and existing methods often require continuous values inputs (e.g., saliency maps). Second, we want to analyse both parts of AZ machinery – the policy-value network and MCTS. Leveraging both the network and MCTS is crucial, since each component plays a different yet important role in deciding the optimal action (see Section E.1 for more detail).

We formulate concept discovery as a convex optimisation problem. Using a convex optimisation framework is not new; many existing methods for finding concept vectors, such as non-negative matrix formulation, can often be approximated as a convex optimisation problem [Ding et al., 2008].

For each concept vector we want to find, we formulate a separate convex optimisation problem. As mentioned in Section 6.3, we define a concept as *a unit of knowledge*. Minimality is achieved by encouraging sparsity [Tibshirani, 1996] through the L_1 norm

$$\begin{aligned} &\min \|v_{c,l}\|_1 \\ &\text{such that concept constraints hold,} \end{aligned} \tag{6.10}$$

where $v_{c,l} \in \mathbb{R}^{d_l}$ is a vector that lives in the latent space of layer l to represent concept c, and d_l is the dimension of layer l.

We outline the concept constraints used for two different types of concepts: static and dynamic concepts.

Definition 6.4.1. Static concept. Static concepts are found in a single state

An example of a static concept in autonomous driving is that the car is *located on the highway*.

Definition 6.4.2. Dynamic concept. Dynamic concepts are found in a sequence of states.

A dynamic concept is that the car is *accelerating*. While our framework only aims to discover dynamic concepts, we use both static and dynamic concepts to validate our method.

Static concepts

We use *supervised data*, where the label indicates whether a state contains a concept c, to learn static concept vectors. These concepts encode human knowledge, and therefore, can be used to validate our approach. One example of a static concept in chess is the concept of 'space', which we can be inferred from the pawn structure in a single state.

For now, assume we have binary concepts[3] and denote the presence of concept c (concept score) in chess position x by $c(x) = 1$, and $c(x) = 0$ otherwise. For each concept c, we can split a general set of chess positions $\mathbb{X}$ into positive examples $\mathbb{X}^+$, where the concept is present, and $\mathbb{X}^-$, where it is absent

$$\mathbb{X}^+ = \{x \in \mathbb{X} : c(x) = 1\}$$
$$\mathbb{X}^- = \{x \in \mathbb{X} : c(x) = 0\}.$$

These positive and negative examples allow us to generate corresponding positive and negative examples of latent representations (intermediate post-activation representations in the network). The function $f_l(x)$ generates an activation for layer l given an input x:

$$\mathbb{Z}_l^+ = \{f_l(x) : x \in \mathbb{X}^+\}$$
$$\mathbb{Z}_l^- = \{f_l(x) : x \in \mathbb{X}^-\},$$

[3]We show how to handle non-binary concepts in Section E.2.3.

where $z_l = f_l(x)$ denotes the latent representation obtained at layer l by passing input x through the network.

The convex optimisation goal is to learn a sparse vector $v_{c,l}$ that represents a concept c. We hypothesise that the inner product $v_{c,l} \cdot z_l$ is higher[4] for activations from $\mathbb{Z}_l^+$ (the set where the concept is present) than for activations from $\mathbb{Z}_l^-$ (the set where the concept is absent). Thus, the formulation becomes

$$\min \|v_{c,l}\|_1 \quad \text{such that} \quad v_{c,l} \cdot z_l^+ \geq v_{c,l} \cdot z_l^- \quad \forall\, z_l^+ \in \mathbb{Z}_l^+,\, z_l^- \in \mathbb{Z}_l^-. \tag{6.11}$$

We can evaluate how well a concept is represented by $v_{c,l}$ in the supervised setting by splitting $\mathbb{X}$ into two sets: $\mathbb{X}_{\text{train}}$ and $\mathbb{X}_{\text{test}}$ and then $v_{c,l}$ only using $\mathbb{X}_{\text{train}}$. We then measure the fraction of elements in $\mathbb{X}_{\text{test}}$ on which the concept constraints hold. If $v_{c,l}$ represents the concept c well, we expect the concept constraint to hold on previously unseen activations derived from $\mathbb{X}_{\text{test}}$.

Dynamic concepts

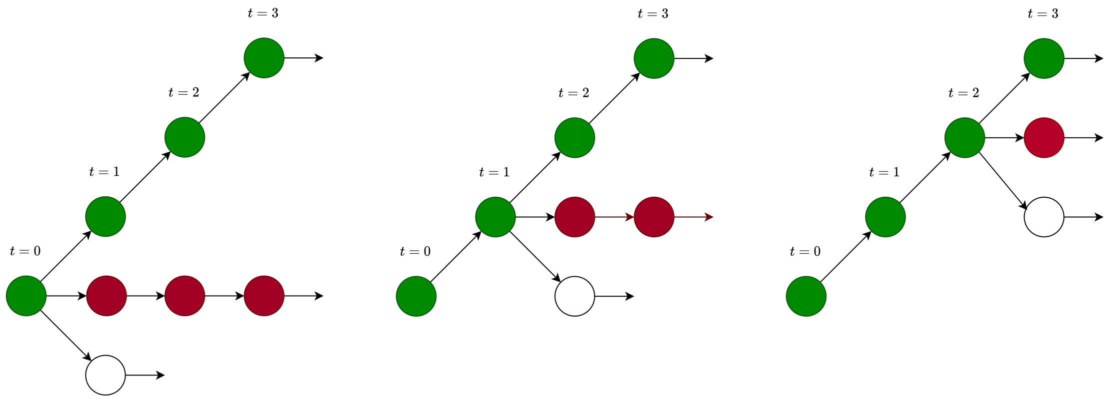
Dynamic concepts are found in a sequence of states. We use the MCTS statistics to find candidates for meaningful sequences of states – sequences where we may expect to find dynamic concepts. MCTS generates a tree of possible moves for a given start state s_0, including a chess position x_0.[5] While the exact details are not essential to understand for our procedure, what matters is that AZ chooses a rollout (i.e., a sequence of potential moves and corresponding states) $\mathbb{X}^+_{\leq T} = (x_1^+, x_2^+, x_3^+, \ldots, x_T^+)$, where T is the maximum depth of the rollout, which terminates in the most favourable state according to AZ. We contrast this optimal rollout $\mathbb{X}^+_{\leq T}$ with a sub-par rollout $\mathbb{X}^-_{\leq T}$, which is defined as a path in the MCTS search tree that is suboptimal, according to the value estimate or visit count in MCTS. This intuition is visualised in Figure 6.3.

The mathematical reason for AZ choosing $\mathbb{X}^+_{\leq T}$ over $\mathbb{X}^-_{\leq T}$ is that it is a continuation where at each state the player maximises their action-value estimate. There may be many conceptual explanations why AZ chooses $\mathbb{X}^+_{\leq T}$ over $\mathbb{X}^-_{\leq T}$. However, we can broadly classify these explanations into the following three scenarios:

1. **Active planning** $\mathbb{X}^+_{\leq T}$ increases the presence of a concept c. For example, the rollout $\mathbb{X}^+_{\leq T}$ may increase the concept of piece activity.
2. **Prophylactic planning** $\mathbb{X}^+_{\leq T}$ *avoids* increasing the presence of a concept c. For example, the plan in $\mathbb{X}^+_{\leq T}$ avoids losing a piece.

[4] A larger inner product corresponds to a higher cosine similarly.
[5] In addition to x_0, s_0 contains the game history, and metadata.

Figure 6.3: Contrasting the optimal rollout with subpar MCTS rollouts at different time steps. The green rollout shows the optimal rollout, and the red rollouts depict subpar trajectories. At each time step, MCTS finds subpar trajectories. We include each of these pairs in the concept constraints.



3. **Random** $\mathbb{X}^+_{\leq T}$ is arbitrarily chosen above the $\mathbb{X}^-_{\leq T}$, as all concepts are equally present in both rollouts and the value estimates of the final states are approximately equal.

We are interested in scenarios 1 and 2, but not scenario 3. Scenario 3 can be filtered out by ignoring rollout pairs with similar value estimates and visit counts in the MCTS statistics.

Using a similar approach to static concepts, we derive our concept constraints on the vector $v_{c,l}$ by contrasting the positive and negative examples, except that this time our contrasting examples are pairs from the chosen rollout $\mathbb{X}^+_{\leq T}$ and the subpar rollout $\mathbb{X}^-_{\leq T}$. We denote the activations at layer l at depth t by $z^+_{t,l}$ and $z^-_{t,l}$ for positive and negative examples, respectively. A single pair of positive and negative rollouts gives rise to the following convex optimisation problem

$$\min \quad \|v_{c,l}\|_1 \tag{6.12}$$

$$\text{such that} \quad v_{c,l} \cdot z^+_{t,l} \geq v_{c,l} \cdot z^-_{t,l} \quad \forall t \leq T, \tag{6.13}$$

for scenario 1, with the inequality reversed for scenario 2.

We extend this idea by contrasting the optimal trajectory with multiple subpar trajectories across different MCTS depths, as illustrated in Figure 6.3. On the left side of Figure 6.3, we find the optimal and subpar trajectory at the initial chess position, $t = 0$. However, we can also use the MCTS statistics (value estimate and visit count) to find subpar trajectories at $t = 1$ (shown in the middle) and $t = 2$ (shown on the right). The idea behind using multiple subpar trajectories is to further reduce the solution space with the goal of reducing noise (thereby

Table 6.1: Datasets Summaries: from concepts more known to humans (top rows) to AZ (bottom rows). Complexity refers to the concept complexity, where low is beginner level and high exceeds the level of the strongest human chess players.

Name	Concept Type	Data type	Complexity
Piece	Static	Human	Low
Stockfish	Static	Human	Varies
Strategic Test Suite (STS)	Dynamic	Human	Medium
Opening	Dynamic	Both	Varies
AZ games	Dynamic	AI	High

increasing the likelihood that we find a meaningful concept) and decreasing the likelihood of learning a polysemantic concept vector.

Let $\mathbb{Z}^{+}_{\leq T,l}$ denote the latent representations in layer l corresponding to the optimal rollout $\mathbb{X}^{+}_{\leq T}$, and $\mathbb{Z}^{-}_{\leq T,l,j}$ denote the latent representations in layer l corresponding to the subpar rollout $\mathbb{X}^{-}_{\leq T,j}$ selected at time step j. We find the dynamic concept as follows:

$$\begin{aligned} \min \quad & ||v_{c,l}||_1 \\ \text{such that} \quad & v_{c,l} \cdot z^{+}_{t,l} \geq v_{c,l} \cdot z^{-}_{t,l,j} \quad \forall t \leq T, j \leq \tilde{T}, \end{aligned} \tag{6.14}$$

where $\tilde{T}$ denotes the maximum depth at which we find suboptimal rollouts. $\tilde{T} = 3$ in Figure 6.3. In general, we set $\tilde{T} = T - 5$ to ensure the rollout is sufficiently deep. Details on how T is set can be found in Section E.2.1.

6.4.2 Method Evaluation

Having introduced our new method for identifying concept vectors, we first evaluate its performance in a supervised setting. We focus on the supervised setting, as we can leverage existing datasets that allow us to evaluate our method in a controlled setting on known concepts. Table 6.1 summarises the datasets used (further details can be found in Section E.2.2). For each datasets we find concept vectors using the approaches described in Section 6.4.1 and Section 6.4.1. Implementation details for each individual dataset can be found in Section E.2.3.

We want to verify whether the concept vector represents the intended concept, however, evaluating this directly is difficult. We hypothesise that if the concept vector represents a concept, it will be able to identify unseen[6] positions that contain

[6]Unseen means that the positions were not used to in the concept constraints.

the concept. We implement this by testing whether the concept constraints hold for positions in a held-out dataset.

However, this does not ensure that AlphaZero *uses* the concept. A concept may be encoded in the latent space, but not used by AlphaZero. To address this, we further validate our approach by showing that amplifying the concept vector in the latent representation influences AZ to select moves that leverage the concept (Section 6.4.2).

Unfortunately, it is computationally expensive to evaluate all layers. Therefore, we focus on evaluating our method on layers 19, 20, 21 and 23 (See Section 6.2). These layers are selected due to their proximity to the network outputs – the policy and value estimate.

Do the concept constraints hold for a test dataset?

For each concept vector, we measure the percentage of times the concept constraints are satisfied on a held-out set. We use an $80-20$ train-test split. The aggregate results for each dataset are shown in Table 6.2. Overall, the concept vectors achieve a high accuracy in detecting unseen concept prototypes, indicating that the concept vectors capture the intended concept. A concept-level breakdown of the accuracy for each dataset can be found in Section E.3.1.

Table 6.2: Evaluation: % the concept constraints hold on test data. The standard error is shown in parentheses.

Concept	Layer 19	Layer 20	Layer 21	Layer 23
Pieces	0.99 (0.00)	0.98 (0.00)	0.95 (0.00)	0.99 (0.00)
Stockfish	0.76 (0.03)	0.75 (0.03)	0.73 (0.02)	0.77 (0.02)
STS	0.92 (0.09)	0.92 (0.09)	0.92 (0.06)	0.90 (0.06)
Opening (general)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
Openings (per line)	0.99 (0.13)	0.99 (0.14)	0.99 (0.14)	0.99 (0.13)

Does amplifying concept increase concept-related behaviour?

We use concept amplification to test whether the concept influences model behaviour. Let z_l denote the latent representation in layer l of a chess position x. To amplify the presence of a concept, we nudge the latent representation in the direction of the concept vector $v_{c,l}$

$$\tilde{z}_l = (1-\alpha)z_l + \alpha\beta\frac{\|z_l\|}{\|v_{c,l}\|}v_{c,l}, \tag{6.15}$$

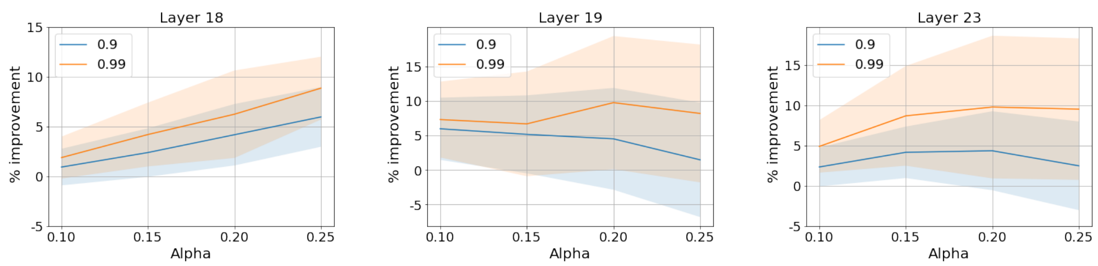
where $\alpha \in [0, 1]$ and β are hyperparameters for the size of the perturbation; and $||\cdot||$ is the L_2 norm. We use cross-validation to determine β, and find that $\beta = 0.01$ is the best overall (see Section E.2.4).

For our evaluation, we use the STS dataset (see Section E.2.2), which includes different types of puzzles grouped according to a strategic theme, such as 'square vacancy'. For each concept, there are 100 chess positions ($\mathbb{X}$) and a solution set $\mathbb{S}_i$ for each chess position $x_i \in \mathbb{X}$. The solution moves require applying the concept given a chess position. As a baseline, we first evaluate AZ's performance by recording the percentage of times the move selected by AZ under the policy is in the solution set:

$$A = \sum_i \mathbb{1}[\text{argmax } \pi_l(z_{i,l}) \in \mathbb{S}_i], \tag{6.16}$$

where $\pi_l(z_{i,l})$ is AZ's policy on the latent representation $z_{i,l}$, and $\mathbb{1}[\cdot]$ is an indicator function that is equal to 1 if the move selected by AZ is in the solution set S_i. We compare the performance difference between with and without concept amplification and report the normalised values $(\tilde{A} - A)/A$ in Figure 6.4.

Figure 6.4: Performance improvement in solving puzzles using STS dataset across different α values (x-axis). Layers 18 (left) and 19 (centre) are before value/policy head split, and the policy head (layer 23) (right). Each line indicates a set of concepts of different quality (measured by test accuracy as in Section 6.4.2). Higher quality (high threshold, orange line) achieves the highest improvement.

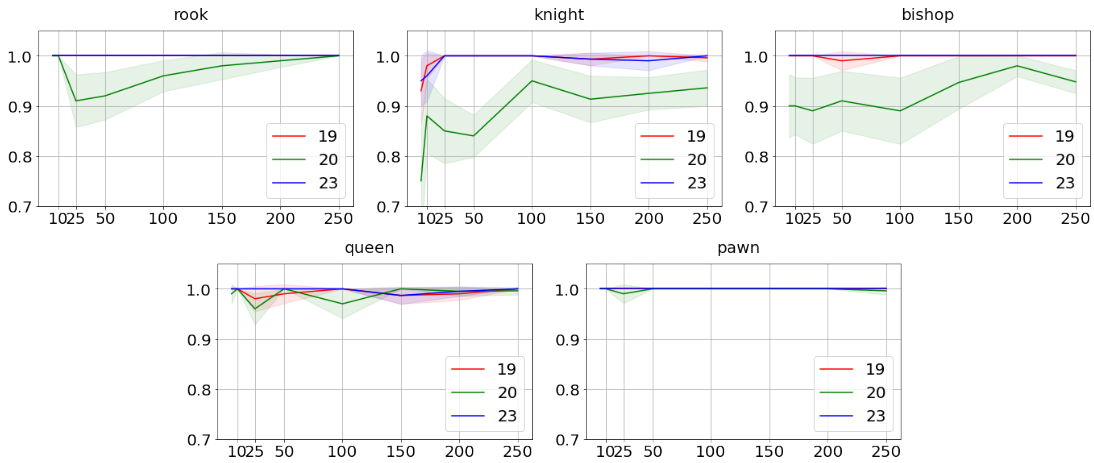


As this experiment focuses on the impact of the concept on the predicted move (only the policy output, not the value), we analyse the performance of concepts found in layers 18 and 19 (layers before policy and value head split, see Section 6.2), and layer 23 (policy head). Each line in Figure 6.4 represents the results for a different concept quality, where concept quality is measured by test accuracy as in Section 6.4.2. We observe that amplifying the concept can improve AZ's performance on the puzzles of the concept. Naturally, the quality of concepts influences this; concepts with higher test accuracy (shown by the line in orange) lead to a larger performance improvement, suggesting a higher test score is a good proxy for how well the concept vector captures the semantic meaning of the concept.

How many data points do we need to learn a concept?

So far, we have focused on labelled datasets to evaluate the concept vector quality (as in the previous section). However, these can be hard to come by in practice. This section shows that our formulation can find concept vectors efficiently using a few examples (for the concept constraints). We measure the test set accuracy (as in Section 6.4.2) while varying the size of the training set for two datasets: pieces and the strategic test suite (STS). Here, we discuss the result for the piece dataset. The results for the STS dataset are similar and can be found in Section E.3.2.

Figure 6.5: Sample efficiency of the convex optimisation framework, averaged across 10 seeds, for the bottleneck layer (19), value head (20) and policy head (23).



As shown in Figure 6.5, we find that the method reaches close to full-set accuracy with only a few samples – often as little as 10 data points on the pieces dataset.

Interestingly, we observe relatively lower performance in the value head (layer 20) than the policy head (layer 23). One potential explanation is that the concept of a specific piece no longer has to exist when estimating the value, which is a scalar – a highly compressed representation of the state of the game. We speculate that it is possible that simple concepts are combined with other concepts. For example, the network may encode the presence of light pieces, i.e., bishops and knights, rather than the presence of bishops. This may explain the relatively low, but not significantly lower performance in layer 20.

6.5 Finding New Concepts

Previously, we have proposed a method to find concept vectors, and showed that it is a data-efficient approach in the supervised setting. In this section, we show how the method can be adopted in an unsupervised setting to find new concepts.

6.5.1 Curating Position Sets

There are three main steps to finding concepts:

1. Curate a set of positions, $\mathbb{X}$, in which we may find concepts.
2. For each position, run MCTS.
3. For each position, formulate a convex optimisation formulation problem using the MCTS statistics.

Steps (2) and (3) were described earlier. In the previous setting, the supervised setting, step (1) is provided; in unsupervised case, we need to construct the position set $\mathbb{X}$. This section explains how we generate $\mathbb{X}$.

Simulate Games As our goal is to find concepts unique to AlphaZero, we search for concepts in AlphaZero's games. To ensure diversity, we generate games from different starting positions.[7] We use the Encyclopedia of Chess Openings (ECO) to find a diverse set of starting chess positions (see the ECO) and simulate $1,308$ games from these initial chess positions.

Find candidate positions in each game From these games, our aim is to identify positions that may contain concepts. To do so, we focus on positions where two agents make meaningfully different choices.

We compare two versions of AlphaZero that differ by 75 Elo points.[8] Both models are from the final stages of AZ's training, when it has surpassed the strongest human chess player's level. This increases the likelihood that we find concepts in $(M - H)$.

From each self-play game, we select positions where the two versions disagree on the best move according to their policies, and where this disagreement leads to a different evaluation by the stronger version. Using this approach, we constructed a dataset with $3,974$ chess positions. In each of these positions, we search for a concept.

6.5.2 Layer Selection

In the field of interpretability, deciding which layers to analyse is an open discussion. We want to determine which layers are most likely to contain new concepts. Taking a step back, the most information-rich latent spaces are likely those where the information contained in AZ games exceeds the information contained in human games. To this end, we contrast AZ's games with human games, through the lens

[7] Conditional on hardware, AZ's play is deterministic (after training).

[8] For reference, a player rated 75 Elo points higher has an expected score of 0.68, with 1 for a win, 0.5 for a draw, and 0 for a loss.

Table 6.3: Rank of latent representation of Human Games and AZ's Games. Max denotes the maximum theoretical rank.

	Input	Layer 19	Layer 20	Layer 21	Layer 23
Human	730	7857	64	86	6544
AZ	728	8269	64	88	6771
Max.	7616	16384	64	256	16384

of the encoding in AZ's PV network, to determine which activation spaces may contain new concepts in AZ's games compared to human games.

Let Z_l^{a} denote a matrix where we stack the latent representations in layer l of chess positions sampled from AZ's games. Each row represents a chess position, and each column represents a dimension in the latent space in layer l. Similarly, we define Z_l^{h} as a matrix of chess positions sampled from grandmaster games. The number of the basis vectors (i.e., the ranks of Z_l^{h} and Z_l^{a}) estimates the size of the span of the latent representations of the games – informally, we can think of it as a proxy for the number of concepts.

Table 6.3 shows the ranks for Z_l^{h} and Z_l^{a} for different layers. We focus on the final layers in the architecture as these tend to be more complex concepts [Kim et al., 2018, Ghandeharioun et al., 2021] and are more likely to influence the policy.

We include the rank of the input as a reference. Differences in input rank may not be meaningful, e.g. AZ has longer games than humans, which may not contain informative concepts. However, the higher variance of position types may lead to a larger input rank, and therefore a larger rank in other layers. However, we observe that this is not the case. The rank of AZ's games is smaller, and therefore we can assume that the differences in rank are due to other factors driving the observed difference.

In Table 6.3, Max. shows the maximum possible rank for each layer. In our analysis, we used $17,184$ positions to ensure that the number of samples does not impact the rank estimation, as $17,184$ is larger than the number of dimensions in each layer. Therefore, the maximum is equal to the number of neurons in each layer.

AZ games' rank is higher than the human games' rank at layers 19 (final layer in main bottleneck) and 23 (policy layer), suggesting that there may be concepts present in AZ games that are not in human games. Therefore, we focus on finding new concepts in these layers.

6.5.3 Finding Meaningful Concepts

Our approach provides many concept vectors, some or many of which represent known concepts or non-generalisable concepts (i.e., only applicable to a single chess position). In this section, we describe how we further filter concepts to ensure that they are useful (transferable) and novel.

Recall that we defined a concept as *a unit of knowledge* – a key aspect is that it is teachable to another AI agent or person, who can apply the concept to solve an unseen task. To ensure our concepts are teachable, we use *teachability* as a selection criterion for the final concepts. The idea is simple:

1. **Phase 0: Find prototypes of a concept vector**. Given a concept $v_{c,l}$, find chess positions that epitomise the concept.
2. **Phase 1: Baseline** find an agent that does not *know* the concept. We can estimate the agent's knowledge by evaluating its performance on a concept-related task.
3. **Phase 2: Teach** the agent the concept using concept prototypes.
4. **Phase 3: Evaluate** the agent's performance on an unseen concept-related task

If a concept is teachable, we expect the agent's performance to improve between the first and third steps. We use a similar process to evaluate when we evaluate our approach with humans (Section 6.7).

Selecting prototypes. In Phase 2, we use AZ as a teacher to supervise a student network on a set of 'prototypes' – chess positions where the concept motivates the plan in the optimal rollout. For each candidate concept c, we have a concept vector $v_{c,l}$. Our goal is to find chess positions x from a dataset $\mathbb{X}$ that epitomise the concept c. We create $\mathbb{X}$ by sampling $30,000$ chess positions from AZ games. We find these prototypes by selecting chess positions for which the convex optimisation constraints for $v_{c,l}$ hold. For static concepts, we do this by computing a concept score $c(x) = v_{c,l} \cdot f_l(x)$ for every $x \in \mathbb{X}$ and then selecting the top 2.5% of $\mathbb{X}$ according to the concept score $c(x)$. We use 2.5%, as we found the concept score $c(x)$ to be comparable to $v_{c,l} \cdot z_l^+$, where $z_l^+ \in \mathbb{Z}_l^+$ is the training set used to find $v_{c,l}$. This procedure gives us a prototype set $\mathbb{X}^{\text{proto}} = \{x \in \mathbb{X} : c(x) \text{ in 2.5th percentile of } c(x)\}$. For dynamic concepts, we find chess positions using the MCTS statistics. For each chess position $x \in \mathbb{X}$, we ran MCTS. Next, we found the chosen rollout $\mathbb{X}^+$ (and the corresponding latent representations $\mathbb{Z}^+$) and subpar rollout $\mathbb{X}^-$ (and the corresponding latent representations $\mathbb{Z}^-$) as in Section 6.4.1. For a prototype $x_i \in \mathbb{X}^{\text{proto}}$, we require that $v_{c,l} \cdot z^+_{i,t} \geq v_{c,l} \cdot z^-_{i,t} \forall t \leq T$.

Student and teacher. We use AZ as the teacher network. For the student network, we want to find an agent that does not know the concept but does understand chess. As chess is a complex game, we cannot train an agent from scratch (using only the curriculum). Instead, we select a training checkpoint of AZ (i.e., the model parameters at a specific point during training) and estimate the model's knowledge of the concept using the empirical overlap of the neural network policies:

$$\text{overlap}(\pi^{\text{s}}, \pi^{\text{t}}) = \frac{1}{|\mathbb{X}_{\text{test}}|} \sum_{\text{x}_i \in \mathbb{X}_{\text{test}}} \mathbb{1}\left[\text{argmax}(\pi^{\text{s}}(x_i)) = \text{argmax}(\pi^{\text{t}}(x_i))\right]. \tag{6.17}$$

Here, π^{s} and π^{t} are the student and teacher neural network policy heads, which output a probability distribution over the possible actions. The overlap measures how often the teacher and student agree on the best move. We select the student as the latest checkpoint for which the top-1 policy overlap is less than 0.2.

Teaching and measuring learning. Intuitively, we can interpret the set of prototypes as a curriculum. We can split $\mathbb{X}_{proto}$ into a training set $\mathbb{X}_{train}$ and a test set $\mathbb{X}_{test}$. The teacher (AZ) teaches the student by minimising the Kullback-Leibler (KL) divergence between the policies of the teacher (π_t) and the student (π_s) on the training prototypes: $\sum_{x_i \in \mathbb{X}_{train}} \text{KL}[\pi_t(x_i), \pi_s(x_i)]$. We use the Adam optimiser [Kingma and Ba, 2014] with learning rate of 1×10^{-4}. As we have several concepts, we train each student for 50 epochs, as we find that this is sufficiently indicative of performance if we train for longer.

To determine whether the student has acquired the new knowledge, we evaluate the student's performance on the test set prototypes by estimating how often the student and teacher select the same top-1 move

$$T = \sum_{x_i \in \mathbb{X}_{test}} \mathbb{1}[\text{argmax}(\pi_s(x_i)), \text{argmax}(\pi_t(x_i))]. \tag{6.18}$$

Benchmark. Teaching using any curriculum may improve students' performance on the task due to the difference in strength between the teacher and the student. To differentiate general learning from concept-specific learning, we compare the student's performance when taught using concept prototypes versus random chess positions sampled randomly from AZ's games (but with meaningful plans). We sample the chess positions from AZ's games instead of human games for two reasons:

(1) AZ's games tend to be of a higher quality than human games (as AZ has a higher Elo rating), and (2) the data is closer to AZ's natural training data (avoiding any confounding effects due out-of-distribution data).

Moreover, we further test whether the student performs better on concept prototypes as well as randomly sampled positions. The concept prototypes allow us to test whether the student improves their understanding of the concept. The randomly sample positions allows us to test whether the student improve their play more generally.

Figure 6.6 shows the student's performance in four settings:

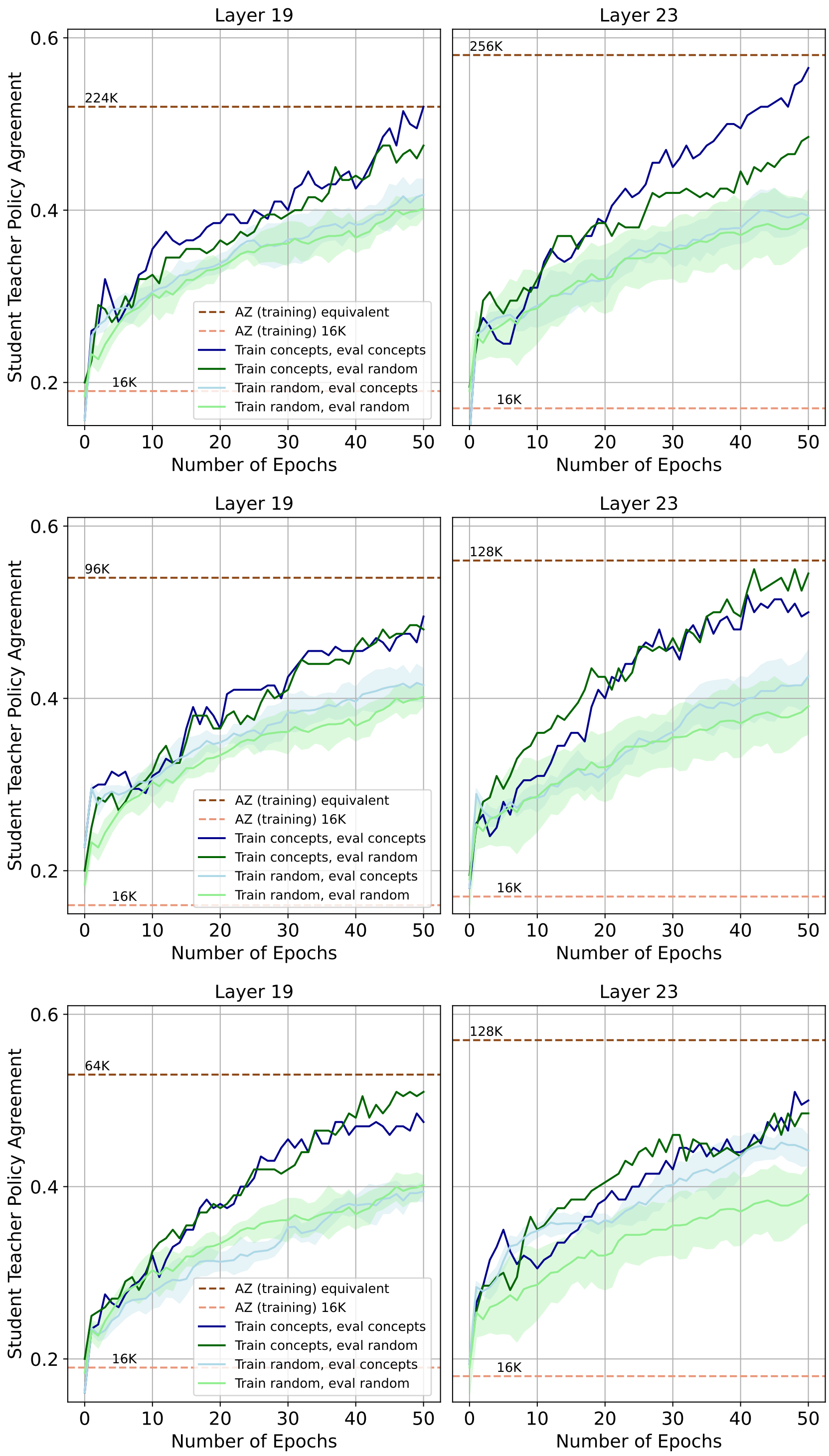
1. Student trained on concept c and evaluated on concept c (dark blue line);
2. Student trained on concept c and evaluated on random data from AZ's games (dark green line);
3. Benchmark: student trained on random data and evaluated on concept c (light blue line); and
4. Benchmark: student trained on random data and evaluated on random data (light green line).

When teaching a student with concept-specific prototypes, the student improves its performance on concept-specific prototypes (dark blue line) and on a random set of prototypes (dark green line). The improvement on concept prototypes (dark blue line) indicates that it learns to apply the concept. The improvement on randomly sampled data (dark green line) indicates that it learns to improve its play more generally. The student taught with randomly sampled chess positions, labelled with optimal play, improves its performance significantly less (light lines) than when taught with concept-specific prototypes (dark lines).

Overall, the student learns quicker when taught with concept-specific prototypes (dark blue line) than random prototypes (light blue line). Further, the concepts can be taught efficiently. The student's performance after training for 50 epochs on a small set of prototypes would have taken $10K - 100K$ epochs using self-play (dark dotted line). Recall that the student is evaluated on a holdout test set, ruling out the possibility that the student network memorised the chess positions.

We select the student to be the latest checkpoint for which the top-1 policy overlap is less than 0.2, resulting in 97.6% of the concepts being filtered out.

Figure 6.6: Teachability: AZ Concepts. The y-axis shows how often the student and teacher select the same move (normalised version of Equation 6.17), and the x-axis shows the training time step. The dark dotted lines show the level of a training checkpoint at which AZ obtains the same level on the concept set as our student. Each plot is a different concept found in layer 19 (top) and layer 23 (bottom).



6.5.4 Novelty

There are many different ways to ensure the novelty of concepts. There is a level of subjectivity involved when evaluating the novelty of chess concepts from a human point of view, as any potentially newly discovered complex concept may be closely related to a number of known, more general chess concepts (e.g. material sacrifice, dynamic compensation, long-term structural weakness). However, even if related in this way, these concepts may still be novel insofar as they are contextualised and instantiated within certain types of piece configurations and patterns on the chessboard, which come with a unique set of considerations that influence how a position is conventionally approached. Moreover, any description of a concept in natural language will inevitably resemble existing concepts, as words are inherently tied to pre-existing ideas; instead, a truly novel concept may require a new corresponding term. Lastly, even if descriptions could correctly encompass a concept, our method does not provide a written description. It is difficult to qualitatively establish novelty in the absence of a detailed language description.

We aim to sidestep the more subjective pitfalls above by focusing on a more quantitative and objective measure, aiming to determine whether the concepts arise in AZ's games but not in human games. Leveraging the fact that concepts are represented in the latent space as a vector, we can compare the vector space of AZ games to that of human games.

Setup to measure novelty. While the data selection process (see Section 6.5.1) ensures that concepts that emerge in the final stages of training in AZ are complex by construction, it does not ensure that the concepts are novel to humans. One way to validate novelty is to determine whether the concepts arise in AZ's games but not in human games. Leveraging the fact that concepts are represented in the latent space as a vector, we can compare the vector space of AZ games to that of human games. Specifically, let Z_l^a denote a matrix where we stack the latent representations in layer l of $17,184$ chess positions sampled from AZ's games. Each row represents a chess position, and each column represents a dimension in the latent space of layer l. Similarly, we define Z_l^h as a matrix of $17,184$ chess positions sampled from human games. Using the latent representations, (1) we first get evidence that AZ's game is likely to contain new concepts using a rank experiment, and then (2) measure the novelty scores by regressing concepts onto AZ's games' vector space and human games vector space, on which we filter concepts based.

Novelty scores for filtering. We define the novelty scores based on how well a concept vector can be reconstructed using a set of basis vectors that derived from AZ's games. Naturally, a lower reconstruction loss means the concept is better represented using a set of given basis vectors. In other words, we look for concepts that are better explained using AZ's language (basis vectors) than humans' language. We define novelty score as the difference between the concept's reconstruction loss (see Equation 6.19) using basis vectors from humans' games and AZ games. A higher score means a closer alignment with the basis vectors arising from AZ's games.

Specifically, for Z_l^h and Z_l^a, we find the singular value decomposition to find the basis of the space spanned by AZ's and human games

$$\begin{aligned} Z_l^h &= U_l^h \Sigma_l^h V_l^{h\top}, \\ Z_l^a &= U_l^a \Sigma_l^a V_l^{a\top}, \end{aligned}$$

where the columns of U_l^h and U_l^a form an orthonormal basis for the rows of Z_l^h and Z_l^a, respectively; Σ_l^h and Σ_l^a are the singular value matrices; and the columns of $V_l^{h\top}$ and $V_l^{a\top}$ form the orthonormal basis for the columns of Z_l^h and Z_l^a, respectively.

The novelty score of concept vector $v_{c,l}$ is defined as

$$\min_{\beta_{i,l}} \left(v_{c,l} - \sum_{i=1}^{k} \beta_{i,l} u_{i,l}^h \right)^2 - \min_{\gamma_{i,l}} \left(v_{c,l} - \sum_{i=1}^{k} \gamma_{i,l} u_{i,l}^a \right)^2, \tag{6.19}$$

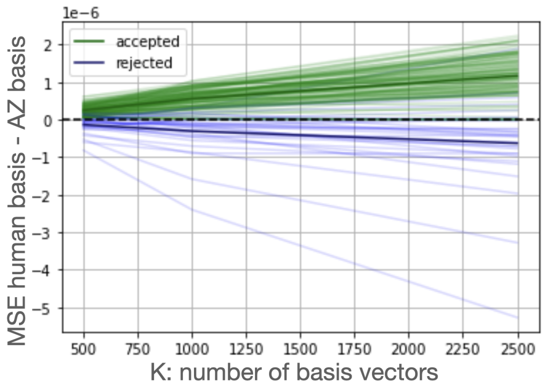
where $\beta_{i,l}$, $\gamma_{i,l}$ are coefficients estimated using linear regression, $u_{i,l}^a$ and $u_{i,l}^h$ are columns of U_l^a and U_l^h, respectively, and k is the number of basis vectors used. We do not set k as the rank of the matrix because they differ for Z_l^a and Z_l^h, and doing so would favour the matrix with the largest rank. Instead, we estimate Equation 6.19 for various values of k.

Figure 6.7 shows the novelty scores for concepts in layer 19 for 120 concepts. We accept concepts for which the reconstruction error using AZ's basis vectors is less than that of the human game's basis vectors. The light green lines denote the novelty scores for the concepts we accept, and the light blue lines denote the novelty scores for the concepts we reject. We filter 27.1% using the novelty metric.

6.6 Learning a Mapping between Human Vocabulary and AI Vocabulary

In the previous section, we introduced a method to discover concept vectors within $(M - H)$. These vectors represent a 'new' vocabulary specific to AlphaZero. One

Figure 6.7: Filtering concepts based on novelty scores. Concepts for which the reconstruction error using AZ's basis vectors is less than the reconstruction error using human game's basis vectors for every k are accepted (not filtered). The darker green and blue lines show the average over the accepted and rejected concepts.



approach to understanding them is to connect these new concepts back to human-labelled concepts.

This idea is analogous to learning lacunae in a foreign language. For example, the Dutch word *gezellig* is a lacuna – there is no translation in English. However, we can gain insight into the meaning of gezellig by relating it to English words we know, such as 'cosinesss', 'fun' and 'warmth'. Following this idea, we construct a directed graph that links AZ's unlabelled concept vectors to human-labelled concepts, providing insight into the potential meaning of AZ's concepts.

How do we fit a graph? We assume that if two concepts are related, they will arise in the same chess positions. Let $s_{c,l}(x)$ denote whether the concept represented by $v_{c,l}$ is present in a chess position x. For each AZ concept c, we learn which human-labelled concepts ($\mathbb{V}$) it shares an edge with by fitting a regression model (similar to Meinshausen and Bühlmann [2006]):

$$s_{c,l}(x) = \sum_{l \in L} \sum_{k \in \mathbb{V}} \beta_{k,l} s_{k,l}(x) + \lambda \|\beta_{k,l}\|^2 \quad \text{for all} \quad x \in \mathbb{X}, \tag{6.20}$$

where $\beta_{k,l}$ are regression coefficients, $\mathbb{X}$ is a set of chess positions, and L is the layers we want to consider. If $\beta_{k,l}$ is significant at a 5% level, we add a directed edge from $v_{k,l}$ to $v_{c,l}$. We use a regression as a concept $v_{c,l}$ may be important for $v_{k,l}$, but not vice-versa. We run the same regression for all human-labelled concepts, using AZ's concepts as the explanatory variables to compute the edges in the opposite direction.

Is the concept present in a position? As before, we assume that product $v_{c,l} \cdot f_l(x)$, where $f_l(x)$ extracts the post-activation latent representation in layer l for the position x, corresponding to positions that contain concept c. For static concepts, we measure the concept presence as

$$s_{c,l}^{\text{static}} = v_{c,l} \cdot f_l(x) \quad x \in \mathbb{X}^s. \tag{6.21}$$

We create $\mathbb{X}^s$ by sampling $2,000$ positions from human games (taken from ChessBase [2021]) as static concepts are found using human-labelled positions (see Section 6.4.1).

For dynamics concepts, we estimate the concept relevance as

$$s_{c,l}^{\text{dynamic}}(x) = \sum_{t<T} \in v_{c,l} \cdot f_l(x_t^+) \tag{6.22}$$

where $x_1^+, \ldots, x_T^+$ corresponds to AZ's chosen MCTS rollout of depth T for a starting chess position x_0. We sample $2,000$ positions from AZ's games to construct $\mathbb{X}^d$, as the dynamic concepts are predominantly found in AZ's games.

In our analysis, we only keep human concepts of high quality, i.e., those with a concept constraint score of higher than 0.90 (see Section 6.4.2). Further, for our analysis to be stable, we removed variables with a correlation coefficient higher than 0.99 [Robinson, 1974, Snee, 1973, Snee and Marquardt, 1984].

Graph Summary. We focused our analysis on layers 19 and 23. We included all concepts found using our unsupervised approach for AlphaZero, and human concepts found using the datasets described in Section 6.5.1. The resulting graph contains 29% $(180,609/625,681)$ of the possible directed edges. As the graph is very dense, we do not provide the entire graph. Instead, we provide parts of the graph in Section E.5 when presenting examples of AZ's concepts.

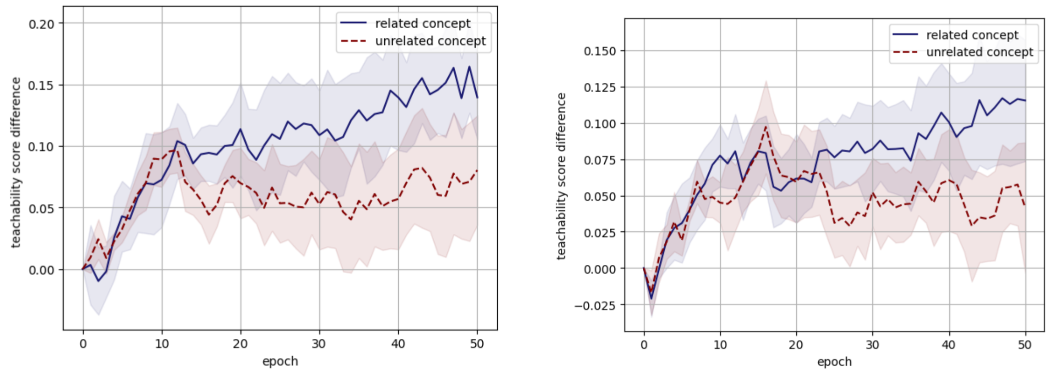
Graph Verification. To verify the graph, we run an experiment to test whether two adjacent concepts – concepts that share an edge – contain more related knowledge than non-adjacent concepts. For a concept c, let $\mathbb{C}_e$ denote the set of adjacent concepts and $\mathbb{C}_n$ denote the set of non-adjacent concepts. Following the teachability procedure in Section 6.5.3, we train a student model using prototypes of concept c. Next, using Equation 6.18, we evaluated students' performance on:

- adjacent concepts $\mathbb{C}_e$; we denote the performance by T_{c,c_e}, and
- non-adjacent concepts $\mathbb{C}_n$; we denote the performance by T_{c,c_n},

where T is defined in Equation 6.18. If the graph correctly captures the relationships between concepts, then we expect that a model trained on c performs better on $\mathbb{C}_e$ than $\mathbb{C}_n$, i.e. $T_{c,c_e} > T_{c,c_n}$.

However, the concepts in $\mathbb{C}_e$ may be inherently easier to learn than those in $\mathbb{C}_n$. To account for this, we train a model on a random set of data (as in Section 6.5.3), which we use as a benchmark. We estimate the performance of this model on $\mathbb{C}_e$ and $\mathbb{C}_n$, which we denote by T_{r,c_e} and T_{r,c_n}, respectively.

Figure 6.8: Teachability score on adjacent concepts (blue line) and non-adjacent concepts (red line).



If our graph accurately captures the underlying relationships, we expect that $T_{c,c_e} - T_{r,c_e} > T_{c,c_n} - T_{r,c_n}$. Figure 6.8 shows the training curves for two concepts, each with 5 adjacent concepts and 5 non-adjacent concepts. We find that the performance on adjacent concepts is significantly better than on non-adjacent concepts at a 5% significance level. This suggests that the graph structure may accurately capture the relationship between concepts. We provide examples of subgraphs in the next section, alongside chess prototypes.

6.7 Human evaluation

We investigate whether top chess grandmasters can learn and subsequently apply the concepts we discovered (in Section 6.4). In particular, we investigate whether they can learn these concepts via exposure to a small set of prototypes of each concept. These prototypes are found using a concept vector as in Section 6.5.3, and then filtered based on a set of criteria aimed at identifying the most relevant high-quality prototypes (see Section E.4 for more information). Learning from prototypes is similar to the traditional learning approaches in chess, where students are often presented with thematic puzzles (opening, chess position type, piece sacrifices, etc.). Solving these puzzles (i.e., finding the correct next moves) allows chess players to learn a new skill, thereby improving their overall strength and ability [Alisa Melekhina, 2014].

In this chapter, we use puzzles to denote prototypes and their 'solutions' (AZ's selected move). The human evaluation follows three phases, similar to how teachability is measured in Section 6.5.3:

- **Phase 1: Establishing a baseline performance.** Each grandmaster is asked to solve a set of puzzles based on concepts. This phase determines the baseline performance: the number of puzzles in which the chess grandmaster identifies the correct continuation *before* the learning phase.
- **Phase 2: Learning from AZ's calculations.** AZ's suggested top line, based on MCTS calculations for each puzzle from Phase 1 is shown to chess grandmasters. This 'explanation' follows the tandem set-up in chess.
- **Phase 3: Assessing the final performance.** Grandmasters are tasked with providing solutions for a test set of unseen puzzles sampled from the same concepts as in Phase 1. We compute the grandmasters' accuracy on the puzzle test set and compare it to their performance on the puzzle training set in Phase 1 to measure whether their performance changes.

The train-test split of puzzles (i.e., allocation to Phase 1 and Phase 3) is random. At no point during any phase are the grandmasters shown labels identifying the underlying concepts behind the prototypes. At each stage, we also asked the grandmasters to summarise their thought process in free form.

We worked with four experts, all of whom hold the grandmaster title;[9] one of our participants is rated 2600-2700, and three are rated 2700-2800. As each puzzle is complex, the study participants would likely spend considerable time on each puzzle to analyse the chess positions in great depth. Therefore, to avoid over-burdening the study participants, they were presented with four puzzles (per concept) for 3-4 discovered concepts at each stage of the study. In total, each grandmaster saw 36 to 48 chess puzzles. While there was some overlap between study participants in terms of puzzles shown, different participants were shown different concepts. Given that the participants had a limited amount of time and the high time investment per puzzle, this randomisation allows us to explore a more extensive concept set (compared to showing every grandmaster the same concepts and puzzles). While some chess concepts may be more teachable than others, it is difficult to establish this beforehand without inserting human bias.

[9]They are ranked in the top 0.004% of internationally ranked players worldwide.

6.7.1 Grandmaster performance

Table 6.4: Improvements in grandmasters' performance. The percentage scores are the % of puzzles that the grandmaster solved correctly (according to AZ's solution). # Puzzles is the number of puzzles shown to the grandmaster in total.

Grandmaster	Percentage Score			# Puzzles
	Phase 1	Phase 3	Improvement	
1	0	42	+42	36
2	33	58	+25	36
3	25	42	+16	36
4	38	44	+6	48

Overall, we find that all study participants improve notably between phases 1 and 3, as shown in Table 6.4, suggesting that the chess grandmasters were able to learn and apply their understanding of the represented AZ chess concepts. The magnitude of improvement does not correlate with the chess player's strength (i.e., Elo rating). Breaking down performance at a concept level, we found an average improvement of solving 0.85 more puzzles correctly between phases 1 and 3, with a standard error of 0.12, which is statistically significant. Of the 13 concepts tested, participants improved their performance on 8 concepts, showed no change on 3, and showed a decrease in performance on 2 concepts. Within the context of the small-scale experiment, this suggests that the majority of concepts were learnable by humans.

6.7.2 Limitations and Confounders

There are limitations and possible confounders in our setup.

Sample Size The number of participants in the sample is small, as we focus on world experts. Therefore, while our results suggest that the grandmasters may acquire the concepts, this should only be seen as a proof of concept.

Confounders Other factors may have influenced performance, such as:

- **Variability in difficulty and quality.** We filter prototypes (as described in Section E.4) to ensure quality and complexity. However, the difficulty and quality of puzzles players received may vary across puzzles.
- **Variability in teachability.** While we filter based on teachability (as described in Section 6.5.3), this metric is based on teaching the concept to another AI agent, which may be inherently different from teaching humans.

- **Overthinking.** We observed that grandmasters often mention AZ's move in Phase 3 in their free-form comments but ultimately did not choose the move (which was not counted as 'correct'). We speculate that this may be because players are more familiar with existing strategies in their decision process, despite having learnt the concept.
- **Baseline** We did not include a baseline to measure the grandmasters' improvement on an unseen concept. Therefore, it is possible their improved performance was caused by a more general learning – they learnt to look for patterns that a chess engine would consider. However, the significance of this effect is debatable. The participants are all top-level players who are accustomed to working extensively with chess engines. Moreover, this type of generalized learning would still be a beneficial outcome, given that chess engines perform at a superhuman level.

6.7.3 Qualitative Analysis of the Concepts and Illustrative Examples

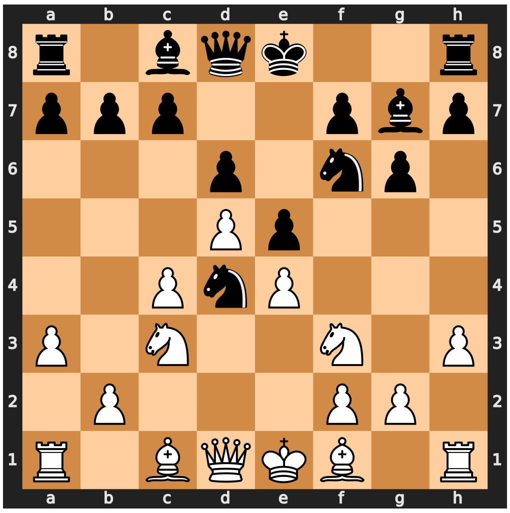
In this section, we provide a qualitative analysis of some concepts with the associated puzzles (plotted using the `chess` python package [Fiekas, 2023]), along with the grandmasters' analysis of the puzzles and (part of) AZ's calculations. Any text or notation referring to the chessboard or moves is written in `this font`. When giving direct quotes, we refrain from providing players' names to preserve anonymity. These examples will cover some cases of successful and some cases of unsuccessful learning. We discuss potential sources of differences between humans and AZ in Section 6.7.4.

Overall, the grandmasters appreciated the concepts, describing them as 'clever' (Figure 6.9), 'very interesting' (Figure E.4), and 'very nice' (Figure E.6). Further, they found that the ideas often contained novel elements, commenting that the moves were 'something new' and even 'not natural' (Figures E.4) and 6.11). Often, the grandmasters found the positions were very complex – making remarks such as that it was "very complicated - not easy to understand what to do". Even when seeing AZ's solutions, they remarked that it was a 'very nice idea which is hard to spot' (Figure E.10).

Concept example: positive knowledge transfer

Here, we explore a concept that possesses both strategic and prophylactic characteristics, involving plans that improve the player's piece placement while restricting the opponent's piece activity. This concept contains an additional element of exploiting tactical motifs and weaknesses, combining strategic and tactical play. We speculate that the concept is learnable to humans, as the grandmasters who trained on the concept puzzles improved their performance between Phases 1 and 3. In Phase 1, they were not able to identify AZ's plan in any concept puzzle (0/4), whereas they successfully identified the correct AZ plan in 2/4 of the concept puzzles in Phase 3.

Figure 6.9: Puzzle of shown in Phase 1. White is to play.



AZ's calculations: `9.Bg5 (9. Be3 Nxf3+ (9...O-O 10.Nxd4 exd4 11.Bxd4 Nxe4 12.Nxe4 Qh4 13.Bxg7 Qxe4+ 14.Qe2 Qxe2+ 15.Bxe2 Kxg7 16.Kd2 Bd7) 10.Qxf3 Nh5 11.g3 O-O 12.Be2 f5 13.exf5 Bxf5 14.Qg2 Bd7 15.O-O-O a6 16.c5) 9...Nxf3+ 10.Qxf3 h6 11.Be3 b6 12.Bd3 Qe7 13.Qe2 a6 14.Qd2 Nd7 15.Bc2 h5 16.O-O-O h4 17.g4 White is slightly better`

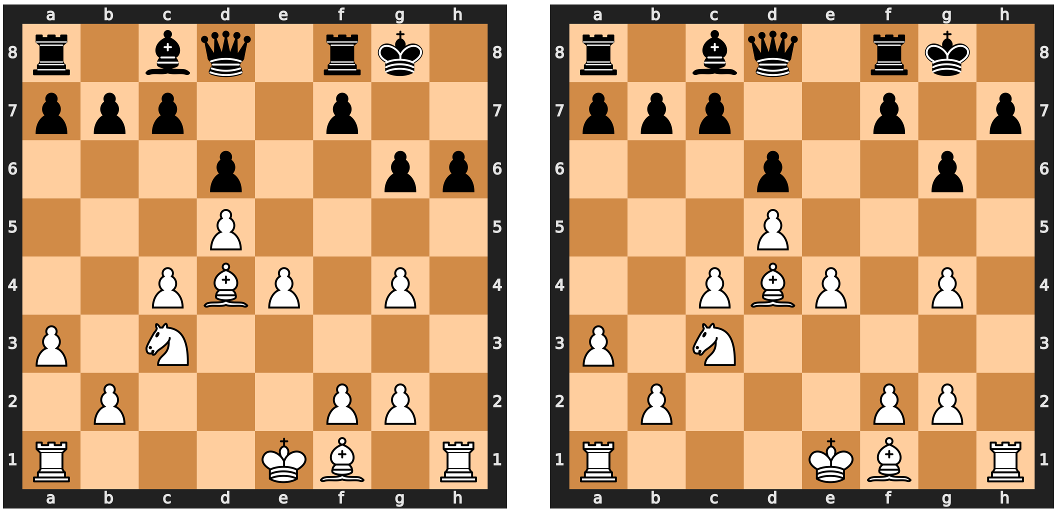
Figures 6.9 and 6.11 show two of the puzzles provided to a grandmaster in Phase 1. In Figure 6.9, AZ plays the move `9.Bg5`; the idea is to provoke `9...h6` before retreating to the square `e3`, thereby inducing a structural weakness. Instead, the grandmaster chose `9.Be3`, a natural move to develop the bishop. After seeing AZ's calculations, the grandmaster acknowledged the strength of provoking this weakness:

> "`9.Be3` allows Black the clever option of playing [`9...Nxf3 10.Qxf3`] `Nh5` [as provided by AZ, followed by] f5 ... but `9.Bg5` is clever as it provokes h6 after which f5 is not great and also [the pawn on] h6 serves as a hook for the pawn advance `g4-g5`. Blacks' plan of stopping `c5` with `b6` and playing `h4-h5` is interesting too but with `g4` anyways White manages to open lines so I would prefer White there."

The idea of playing provocative bishop moves to induce pawn weaknesses is not new and arises in human play. However, the interesting and potentially novel element here lies in the planned strategic queen sacrifice that emerges in one of the critical lines in the MCTS calculations. Consider one such critical continuation:

```
9.Bg5 h6 10.Be3 O-O 11.Nxd4 exd4 12.Qxd4 Ng4 13.hxg4!
```

Figure 6.10: Further analysis of the puzzle in Figure 6.9. In both puzzles, Black is to move.

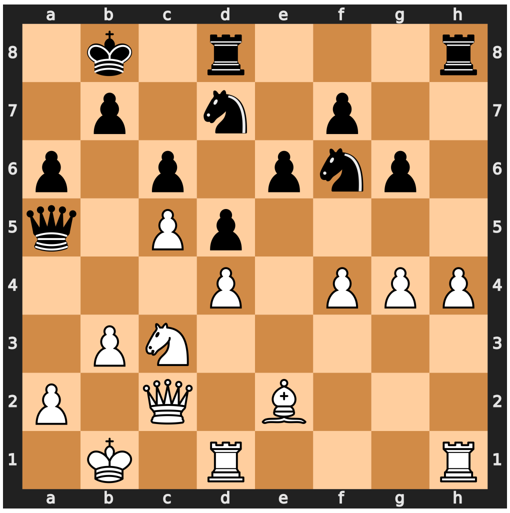


Queen sacrifices are among the most beautiful (and rare) tactical motifs in chess, as they go against the established chess principles – do not trade more valuable pieces (i.e., the queen) for less valuable pieces (a knight and bishop). However, here AZ's queen sacrifice is strategic – after sacrificing the queen, White continues developing their pieces. The line continues `13...Bxd4 14.Bxd4`, as shown on the left of Figure 6.10. Here, due to the pawn on `h6`, Black's king is vulnerable, and White is better. Therefore, it was critical to play `9.Bg5`, rather than `9.Be3`, to make the queen sacrifice feasible by creating this weakness. The puzzle on the right of Figure 6.10 arises in the same line if one opts for immediate `9.Be3` instead, failing to induce `h6` first. Without the pawn on `h6`, White is lost, highlighting that this is the critical positional element.

In general, some moves in the AZ's calculations are more important for the concept than others. In our convex optimisation formulation, we do not require the concept to be equally important for every chess position (see Equation 6.14). This is illustrated in the previous example, where `Bg5` is more important for the concept.

The next puzzle from the same concept, also given in Phase 1, is shown in Figure 6.11. This puzzle is of particular interest due to the unconventional plan of AZ – it increases space on both sides of the board, expanding with the pawn move `b4` while the king is still on `b1`. To this end, AZ initiates this plan with the prophylactic move `21.Qd2`, preventing the immediate `21...Ne4` by Black.

Figure 6.11: Second puzzle shown in Phase 1. White is to move.



AZ's calculations: `21.Qd2 Qc7 (21...Rh7 22.h5 Rdh8 23.Rh3 Qc7 24.b4 Ne4 25.Nxe4 dxe4 26.g5 gxh5 27.a4) 22.Bf3 Rh7 23.Rh3 Rdh8 24.Rdh1 Ne8 25.Bd1 (25.b4 f6 26.Bd1 e5) White is slightly better`

Unlike AZ, the grandmaster's suggestion in this puzzle was `21.g5`, with the intention of following up with `21..Nh5 22.Bxh5 Rxh5 23.Rh3 Rdh8 24.Rdh1`. Upon seeing the suggested AZ line starting with `21.Qd2`, however, the grandmaster study participant remarked

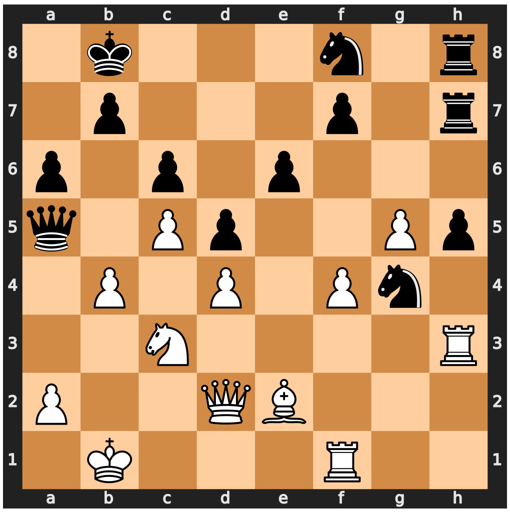
> "`21.Qd2` is a useful move as it stops Ne4 and protects f4 and can be better placed in case of b4 in the future. One curious line [given by AZ] is `21...Rh7 [22.h5 Rdh8] 23.Rh3 gxh5 24.g5 Ng4` White can just play `25.Rf1` and then focus on getting the b4 [pawn] break, which is not natural."

The unconventional plan of pushing the pawn to `b4` with the king potentially exposed on `b1` is particularly strong in this chess position (shown in Figure 6.12) as it allows White to gain space and open up the chess position under unfavourable circumstances for Black, and claim an advantage. Therefore, the more general rules here are discarded based on concrete analysis. For example

```
21.Qd2 Rh7 22.h5 Rdh8 23.Rh3 gxh5 24.g5 Ng4 25.Rf1!? Nf8 26.b4!!
Qxb4+ 27.Ka1 Qa5 28.f5! exf5 29.Qb2 Ne6 30.Nxd5!! cxd5 31.Ra3
Qc7 32.Bxa6 White is better.
```

In this line, we see the dynamic play of AZ: the rooks from `f1` and `h3`, switch over to the b-file to attack Black's king.

Figure 6.12: Expanding on the critical line in the puzzle shown in Figure 6.11. White has just played `b4`.



The grandmaster missed the ideas in both positions. AZ's ideas require unconventional continuations that go against common human chess principles. These observations hint at the existence of super-human knowledge $(M - H)$.

Figure 6.13: In the left puzzle, White is to move. In the right puzzle, Black is to move.

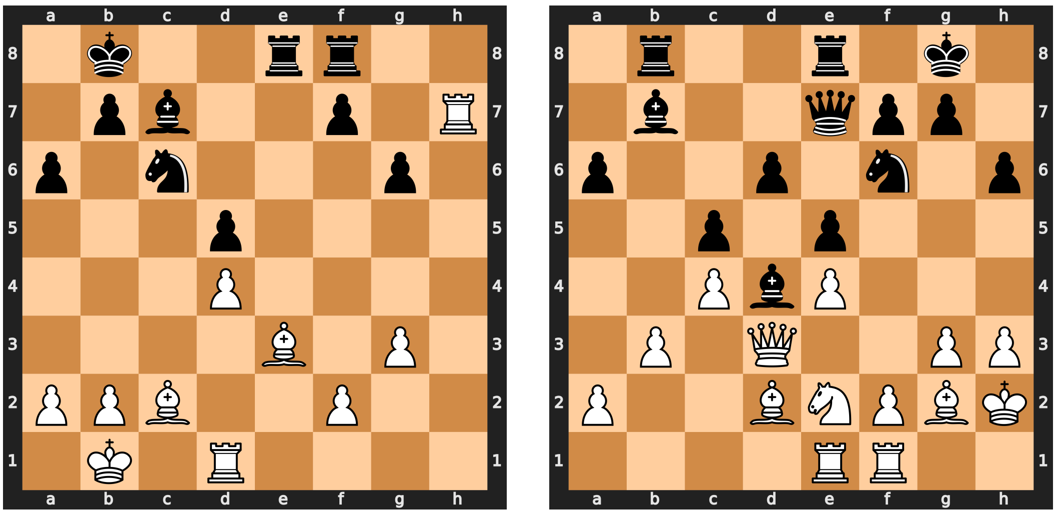


Figure 6.13 shows the puzzle from the same concept provided to the same grandmaster in Phase 3; Phase 3 tested whether the grandmasters had learnt the concept. As before, the puzzles underscore the concept's multifaceted attributes, encompassing its prophylactic characteristics and its integration of tactical and strategic elements.

In the puzzle (from Phase 3) shown on the left of Figure 6.13, the grandmaster correctly found the move suggested by AZ: `24.Bb3`, with the idea of forcing the

Black rook on `e8` into a more passive position (`d8` to defend the pawn) before commencing activity on the other side of the board. The idea can be seen in a possible continuation:

> `24...Rd8 25.Ba4 Na5 26.Rdh1 Nc4? 27.Bh6 and White picks up the pawn on f7.`

As in the previous puzzles, AZ uses both sides of the board to optimise piece activity.

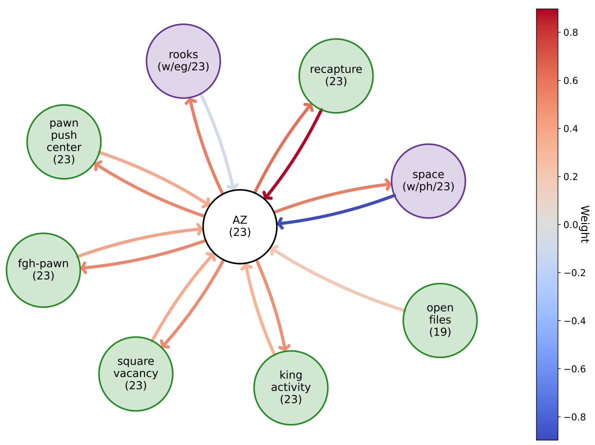
In the puzzle, shown on the right in Figure 6.13, the grandmaster again found the correct idea: `22...Bc6`, with the idea of preventing the critical pawn advance `23.f4` because of:

> `23.f4 Qb7 24.Nc3 exf4 25.Rxf4, resulting in a weakened pawn structure,`

where White cannot recapture with `25.gxf4` as `e4` is hanging. The move `22...Bc6` is both prophylactic and tactical; it prevents White from executing their plan to advance the kingside pawns while improving Black's position by activating the bishop and rook.

The overall improvement of the grandmaster on this concept suggests that they may have learnt AZ's concept, thereby expanding H with $(M - H)$. Other examples of concept learning can be found in Section E.5.

Figure 6.14: Graph of AZ's concept in Figures 6.9, 6.11 6.10 and 6.13 between AZ's (white), strategic (green) and Stockfish concepts (purple). The information in the parentheses means the layer in which the concept is found, and w = white, b=black, eg= endgame, mg = middlegame, ph=phased. The edge colour denotes the edge weight.



It is perhaps not immediately clear what the exact commonality between these examples is, and what specifically makes these applications of advanced (yet more generally known and recognisable) chess principles form a distinct concept, from the perspective of AlphaZero. While we recognise that interpretability remains an open challenge, especially for intricate and complex concepts, here we outline one possible approach that can act as a starting point for improving our understanding of the discovered concepts - graph analysis.

Understanding AZ's concept using graph analysis and human-labelled concepts. Using a simple way to learn a graph (see Section 6.6) between concept vectors, we discover strong relationships between existing and discovered concepts to gain further insight into the concept meaning (shown in Figure 6.14). Edge weight is influenced by (1) the strength of the relation between two concepts and (2) the frequencies at which concepts co-occur. Below, we further discuss the two concepts with the largest edge weights.

Space. AZ's concept has a strong positive weight on the outgoing edge with the (White-side) space concept. In puzzles where White is to move (Figures 6.9, 6.11 and 6.13), an important component of the plan is to increase space. In a similar vein, given that the idea is to *increase* space, which is 'easier'/more likely if the initial value is lower, AZ has a negatively weighted incoming edge with the concept space.

Recapture. We observe positive incoming and outgoing edge weights with the recapture concept. Recall that we have dynamic concepts, which refer to a sequence of states. As such, we postulate that this connection is because the plan may be to recapture/gain material in the subsequent chess positions, as in the puzzles in Figures 6.9 and 6.13 (left side).

Concept example: unsuccessful learning

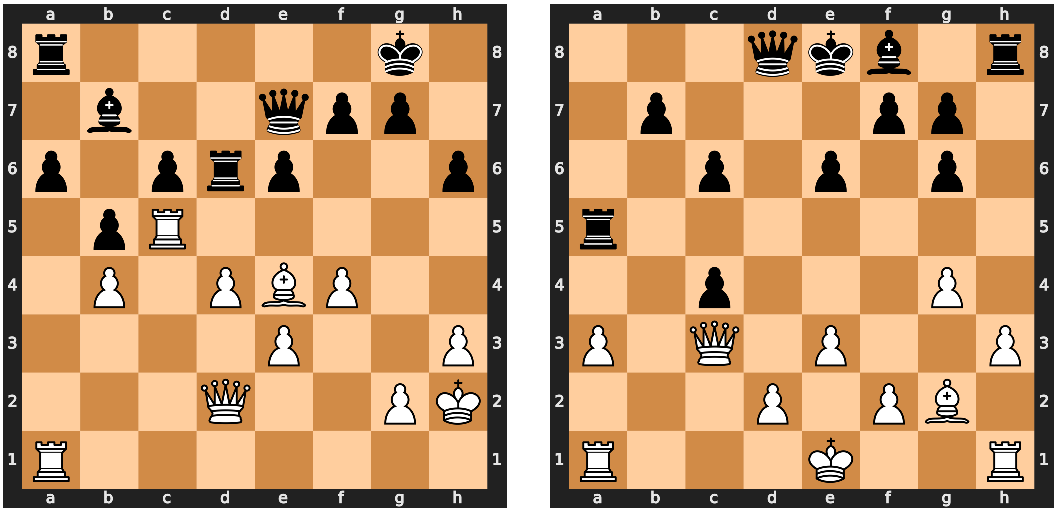
This concept is related to gaining and playing with a space advantage with positional advantage despite less material. In this section, we provide an example of when a grandmaster found the correct move in Phase 1, but provided an incorrect (i.e., not AZ's choice) move in Phase 3. The puzzle on the left side in Figure 6.15 was provided to the grandmaster in Phase 1, and they correctly chose the same move as AZ: `g4`. However, while the grandmaster found the correct move, there were further finesses in AZ's calculations that the grandmaster missed during their time-constrained analysis:

> White is a pawn down, [and AZ plays the move I suggested] `g4` ... The computer plays with `h4-h5` [and] `g4` and the queen on `f2` or `g2`. White is playing on the kingside.[10] Black has no active moves, is playing

[10] For readers - the queenside refers to the left side marked a-d and the kingside refers to the right side of the board marked e-h.

Rd8 or [R]f8, Kh8. Seems convincing to me. On g6 [AZ] goes [Q]g2 which is nice, it didn't occur to me. I was mainly focused on making h4 work for White. White is not in a hurry, will at some point play g5. Compensation, zero counterplay, and AZ is acting on these premises.

Figure 6.15: In both puzzles, White is to move.



AZ's calculations: 33.g4 g6 (33...f6 34.Qg2 Rf8 35.Rf1; 33...Rad8 34.Qf2 f6 35.Rg1) 34.Qf2 Kh8 35.h4 White is better

As remarked by the grandmaster, AZ is a pawn down, however, Black's pieces, particularly the Bishop on b7, are placed passively. Instead of prioritising regaining material, AZ focuses on improving the kingside position. While the grandmaster understood the general plan, they missed the intricate idea Qg2, and slowly advancing the g- and h- pawns.

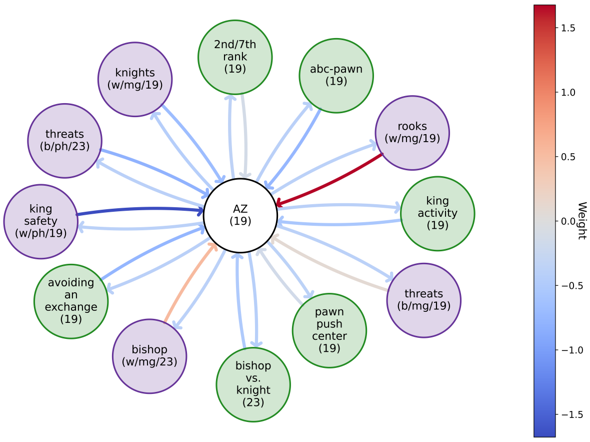
The puzzle on the right in Figure 6.15 was provided to the grandmaster in Phase 3. Similar to the previous example, AZ focuses on space rather than recapturing material. It continues with the move 18.a4, which was rejected by the grandmaster on account of 18.Rc5, where Black tries to maintain the material advantage. However, AZ finds the rook on c5 misplaced and continues

18.a4 Rc5 19.a5 Be7 20.a6 where after Bf6? White has 21.d4

The idea behind the pawn advance is to weaken Black's pawn structure.

Overall, the evidence suggests that the grandmaster did not learn this concept. Here, AZ selects stronger moves due to its prioritisation of concepts (e.g., focusing on space and activity). Humans tend to prioritise these concepts differently (e.g., prioritising bringing the king to a safe location as soon as possible). This concept may be inherently difficult and require further examples to learn.

Figure 6.16: Graph of AZ Concept in Figure 6.15 between AZ's (white), strategic (green) and Stockfish concepts (purple). The information in the parentheses is the layer number in which the concept is found. w = white, b=black, eg= endgame, mg = middlegame, ph=phased. The edge colour denotes the edge weight.



Understanding AZ's concept using graph analysis and human-labelled concepts. Figure 6.16 shows the relationship between AZ's concept and high-quality human-labelled concepts. The graph is dense, and we elaborate on the two concepts with the largest edge weight.

Rook. AZ's concept has an incoming positive edge with the rook (activity) concept. In the puzzles in Figure 6.15, we observe that White has active rooks (the rooks on `a1` and `c5`, in the left chess position) or plans to activate the rook (the rook on `a1`, in the chess position on the right).

King. AZ's concept has an outgoing negative edge with king safety. In the chess positions in Figure 6.15, the king is less safe than usual. In the left chess position in Figure 6.15, AZ pushes forward the pawn to `g4` (and later the pawn to `h4`) around the king – thereby removing some of the king's defenders. In the position on the right, White does not castle to improve the king's safety but instead leaves the king in the centre.

Further examples of concept puzzles can be found in Section E.5.

6.7.4 Differences between Humans and AZ

In this section, we share a few observed differences between the grandmasters and AZ and speculate where they come from. While we do not have definitive answers, the discussion may lead to further research.

The qualitative examples suggest that AZ has different priors over the relevance of concepts in a chess position than humans. Human chess players formulate and adopt heuristic chess principles to inform their analysis, predisposing them to biases that influence which concepts they deem relevant for specific chess positions. An example is the three 'golden rules' of the opening: control the centre, develop your pieces, and bring your king to safety [Hansen, 2021, Brunia and van Wijgerden, 2021, King, 2000]. Consequently, in the opening, humans may focus on moves that align with these guidelines. Instead, AZ is self-taught and does not seem to have the same priors over chess concepts as humans. We believe this lack of prior allows AZ to be more flexible – it can apply concepts to various different chess positions and change plans quickly. In essence, AZ formulates its own priors over the relevance of chess concepts for a given chess position. Examples of this behaviour are that AZ plays over the entire board, as opposed to focusing on a specific side (see, e.g., Figures E.4, E.5, 6.13, and E.7); places less importance on the material value of pieces, and prioritises space and piece activity (see, e.g., Figures 6.10 or 6.15). This may result in the super-human application of concepts, and new concepts.

One may ask where does the differences between AZ's and humans' play come from? We conjecture that they may arise from differences in objectives and capabilities. AZ learnt to play chess against itself. As such, AZ assumes optimal play and information symmetry.[11] On the other hand, humans play chess against other humans and, therefore, may assume information asymmetry and imperfect play. Moreover, psychological tactics may underlay human's move selection. This leads to a difference in behaviour: while AZ focuses on finding the best move, human chess players often make *practical* choices. Humans' choices do not always increase the expected outcome against an optimal opponent (their choices may even slightly decrease the expected outcome), but may increase their odds against another human. For example, in drawn chess positions, humans may try to complicate the chess position or opt for continuations where the best moves are less clear-cut in hopes that their opponent makes a mistake (see Figure E.11). However, AZ will try to find the optimal plan, disregarding aspects such as complexity. Therefore, AZ's play may be fundamentally different and better reflect conceptually relevant plans in a chess position.

Another difference between humans and AZ's play is the role of time. Humans have limited energy[12] and time allocation for a game. In chess positions where humans are better, they often simplify the chess position to try to secure the win as quickly as possible and minimise risk (see, e.g., the grandmasters' chosen moves

[11]With information symmetry, we mean that Black and White have the same general knowledge and perform the same calculations in a given chess position.

[12]Classical time control (see match time controls in FIDE [2019]) chess games take hours.

in Figures E.10 and E.9). However, AZ does not care about how quickly the game finishes. The training loss function does not have a penalty term to encourage winning as quickly as possible. As a result, it has a different treatment of time. This results in sometimes choosing slow strategic wins (as can be seen in the chess positions in Figure E.9). While the lack of time constraint may lead to super-human concepts, it also may result in complex concepts that are difficult for humans to learn.

AZ and humans have different computational capacities. As a result, AZ can opt for more computationally expensive moves and, therefore, defend complicated chess positions where humans might be more hesitant. In terms of playing style, AZ will often opt for what it calculates as the *optimal* move, while humans have a more limited computing budget and may opt for a safer move. In chess, *safer* is used to describe continuation where there is less probability of making an incorrect move. Sometimes, humans may even play a slightly suboptimal move to minimise the risk. We see this phenomenon in Figures E.10 and E.11 i. While computational capacity cannot be transferred, it may still lead to super-human concepts, as AZ can find new ideas that can still be taught to humans.

6.8 Related Work

6.8.1 Generating explanations in Reinforcement Learning

Explanations for Reinforcement Learning (RL) methods [Alharin et al., 2020, Vouros, 2022] have different requirements compared to traditional supervised learning settings. These differences arise due to the temporal dependency between states, actions, and subsequent states, where an agent's past, present and future state-action sequences may relate to a long-term goal [Dazeley et al., 2021]. Explainability methods in RL can help identify issues such as over-fitting to training data, reliance on spurious correlations [Gajcin and Dusparic, 2022], poor out-of-distribution performance [Annasamy and Sycara, 2019] and challenges in inter-agent dynamics [Omidshafiei et al., 2022]. Moreover, these methods in RL may help provide useful counterfactuals [Deshmukh et al., 2023, Olson et al., 2019].

There is a need for post-hoc RL interpretability methods. Currently, input saliency maps are commonly used [Wang et al., 2016, Selvaraju et al., 2019, Greydanus et al., 2018, Mundhenk et al., 2020]. However, saliency maps can suffer from unfalsifiability and cognitive bias [Atrey et al., 2020, Adebayo et al., 2018] and lead to incorrect results [Bilodeau et al., 2024]. Moreover, while saliency maps may help give insights into individual actions in terms of the inputs, they do not allow us to easily identify generalisable higher-level concepts across sets of inputs. Other extensively used methods include tree-based methods [Craven and Shavlik, 1995, Bastani et al., 2018, Roth et al., 2019, Coppens et al., 2019, Liu et al., 2019, Vasić et al., 2022, Madumal et al., 2020], and various techniques for analysing agent behaviour – e.g., visualising the agent memory over trajectories [Jaunet et al., 2020], extracting finite-state models [Koul et al., 2018], and leveraging Markov decision processes [Finkelstein et al., 2022, Zahavy et al., 2016] for detecting sub-goals or emerging structures [Rupprecht et al., 2019].

While most prior work focuses on understanding agent policies without search, our work considers discovering concepts for explaining agent decisions based on MCTS. Given that different explainability methods offer different insights into model predictions, future work should move towards more holistic, composite approaches that bridge and combine existing techniques to give a more comprehensive and reliable set of explanations [Gilpin et al., 2018, 2022].

6.8.2 Chess and AI

Chess has been a test bed for AI ideas for decades. Early engines were based on human knowledge, and their super-human strength came from their computational capacity, which allowed them to evaluate orders of magnitude more variations than human chess players. The introduction of neural networks and RL-based approaches led to a surge of improvements in computer chess engines. These advances were in part inspired by the prominent results of AZ in chess [Silver et al., 2018], and Lc0 [LCZero Development Community, 2018], an open-source re-implementation of the original model.

As interactions with chess engines play a key role in chess players' preparation and training, interpretability can play an important role in helping players improve their abilities. To this end, prior work has looked at piece saliency [Gupta et al., 2020], tree-based explanations [Kerner, 1995] and natural language [Jhamtani et al., 2018, DecodeChess, 2017]. Recently, natural language explanations for chess were explored in ChessGPT [Feng et al., 2023].

Using probing-based techniques, researchers found that AZ and Stockfish encode human-like concepts in its network [McGrath et al., 2022a, Pálsson and Björnsson, 2023]. This prior investigation of concepts in AZ focused on known human concepts in the policy-value network and did not consider search, move sequences, or new concepts. This work is the first to use an unsupervised approach to find new concepts. However, preliminary questions have been raised regarding whether human players have been adopting AZ's ideas [González-Díaz and Palacios-Huerta, 2022], as some prominent motifs had been analysed in detail in the book Game Changer [Sadler and Regan, 2019].

6.9 Conclusion

Our research represents a step towards understanding the potential of human learning from Artificial Intelligence (AI). In this work, we focused on AlphaZero (AZ) – an AI model that attained a super-human chess level through self-play, without prior knowledge or human bias. We provide evidence for the existence of super-human knowledge in AZ using spectral analysis to show that AZ's games encode features that are not present in human games. To extract this knowledge from AZ's representational space, we developed a framework to uncover new chess concepts. To mitigate human bias, we use an unsupervised approach that leverages AZ's training history to curate a set of complex chess positions. We ensured each concept is useful, by verifying that the concept can be taught to another AI agent who can apply the concept to solve a downstream task, and novel, by measuring the alignment with the basis vectors of human and AZ games. Communicating novel concepts requires developing a common language between humans and AI. We bypass the need for this language by leveraging puzzles (i.e., prototypes) for each concept.

We collaborated with four world-top grandmasters to (1) validate the human capacity to understand and apply these concepts on an unseen test set by studying AZ's concept prototypes and (2) improve our understanding of the differences between AZ's and humans' chess representation space. In our small-scale study, all grandmasters improved their performance after learning concepts compared to baseline performance. We speculate that the differences in the representation space between AZ and humans may stem from: (1) prior biases over concepts, including their perceived applicability, importance, and how they can be combined with other concepts. For example, AZ shows a reduced emphasis on factors such as material value and is more agile in switching between playing on different sides of the board. (2) a difference in the motivation and objectives when playing chess; AZ is trained to play optimally rather than competitively. Human players may leverage their opponents' weaknesses, such as limited strength, time or energy.

There are several aspects of the work that could be further explored. In our work, we identified a subset of all possible concepts. From those found, we only tested 10% in experiments with human participants. Moreover, our analysis was subject to several limitations: we considered only two layers (less than 10% of the network's total layers), focused on concepts learned by AZ at the end of training, and represented concepts solely as linear sparse concept vectors. Relaxing these assumptions could potentially reveal a broader spectrum of interesting concepts, such as those with non-linear representations, thereby expanding the concepts we discover in AlphaZero.

Beyond individual concepts, the current work focuses on finding a single concept to explain a plan. However, a plan may inherently contain multiple concepts. As such, an interesting aspect to further explore is how these concepts relate to each other, influence the overall plan, and, critically, how their combined presentation might further aid human learning.

Further work could also expand on the human experiment study. While our work provides a proof of concept, a larger-scale study is crucial to draw statistical conclusions regarding human learning from AI. We limited our subjects to top-ranked grandmasters, as we suspected that acquiring novel knowledge at the frontier of human understanding (in $(M - H)$) would require experts. However, a different group of learners may require alternative pedagogical approaches beyond the prototype-based learning studied here.

Moreover, it would be interesting to further explore the optimal conditions for humans to learn concepts. We allotted a fixed time budget for grandmasters to assimilate the concepts due to the time availability of the world-class players. However, it is plausible that an unlimited time budget would improve learning and performance. In our research, we provided grandmasters with part of AZ's Monte Carlo Tree Search (MCTS), in which the rollouts are motivated by the concept, as an explanation for the concept. We used this approach to keep the explanations as familiar and simple as possible. Nonetheless, it would be interesting to augment this phase with an interactive component: e.g., for each puzzle, humans can actively engage with AZ by playing moves and asking AZ what its response is. This interactive element would allow humans to investigate counterfactual scenarios, allowing for a deeper understanding of why AZ did not select their solutions or approaches. More generally, optimizing individual learning strategies when interacting with advanced AI systems remains a significant open question.

While this chapter explored AlphaZero within the domain of chess, it would be interesting for future work to investigate knowledge discovery in other complex domains. A crucial direction for future research is the quantification of the human-AI knowledge gap. An important question lies in identifying when there exists a meaningful difference in performance or conceptual grasp exists between humans and advanced AI systems. Developing methods to quantify this gap, and subsequently how much of this difference is learnable by humans, is an open challenge.

This work also prompts deeper reflection on the nature of human learning from AI. To what extent can and do humans learn the concepts encoded by the AI system versus learn more generally through interaction with AI's knowledge? For example, humans may acquire new knowledge or skills that, while not identical to the AI systems' internal representations, are nonetheless useful for solving the downstream task. If the primary goal is to advance human understanding, it may not be paramount to understand *exactly* what the AI systems know, as long as learning is advanced more generally.

Learning from and interacting with AI systems may lead to significant shifts in human behavior and strategic thinking. Within the context of chess, for instance, the advent of chess engines has fundamentally altered how humans play [González-Díaz and Palacios-Huerta, 2022] and prepare for matches. At a surface level, players leverage chess engines to better understand the game and find new ideas. However, this already leads to a shift in required human skills – moving towards finding and memorizing engine-generated ideas. This, in turn, can lead to a ripple effect that follows game-theoretic notions of information (a)symmetry. On the next level, players prepare knowing that their opponent also has access to an engine. This can lead to optimizing for trajectories where the engine might perform suboptimally or where it's easy for engines to play but difficult for humans without specific preparation. Given this, it becomes strategically important for players to infer whether their opponent is 'in-preparation'. This then leads to a further layer: players may use subtle cues, such as body language and time usage, to signal (or more importantly, to influence their opponent's belief) about whether they have prepared ideas with a chess engine in particular positions.

While these dynamics are particularly evident in chess, they raise critical questions about how AI influence will reshape human interactions and skill sets across other domains. There may be a broader shift in human skills as AI automates some tasks while emphasizing others, alongside a need for more proactive thought on how human intelligence can best interact with, and perhaps even strategically circumvent, these advanced systems.

Part IV

Discussion

7 Conclusion

The goal of this thesis is to understand how knowledge is encoded in machine learning models and how that knowledge can be extracted in a meaningful way.

First, in Part II, we examine how knowledge is represented within neural networks. In Chapter 3, we start by examining the linear representation hypothesis, which states that concepts are encoded as directions within a model's representation space. We analyse the implications of this assumption, focusing on three key properties: spatial dependence, polysemanticity, and the effects of interventions across layers. We formalise each property and show how it can influence the conclusions drawn from concept-based analyses.

In Chapter 4, we turn to the question of how knowledge is encoded in large language models (LLMs). Specifically, we investigate whether LLMs learn universal representations or language-specific representations. Our findings suggest that models exhibit language-specific behaviour. Conceptual knowledge tends to be tied to the training language, while the input language primarily determines other linguistic features, such as grammar.

Next, in Part III, we shift to the second goal of this thesis: extracting knowledge that humans can easily understand. We begin by focusing on generating human-understandable prototypical explanations in Chapter 5. By leveraging uncertainty estimates, we produce realistic and unambiguous explanations.

Finally, we combine both goals of the thesis. In Chapter 6, we explore how interpretability can be used to discover new knowledge. In the domain of chess, we introduce an unsupervised framework for concept discovery. We ensure that the identified concepts are both novel – not observed in human games – and teachable –

able to be transferred to another AI agent. In a small-scale experiment, we further demonstrate that these concepts can be taught to human experts, who then apply them to solve previously unseen puzzles.

Taking a step back, the work presented in this thesis can be contextualised within the broader perspective on interpretability introduced in Chapter 2. We adopt a pragmatic stance, moving away from a single, rigid definition of interpretability. Instead, we argue for a more practical approach where the methods and goals are tailored to the task at hand. This perspective motivates a structured process, and the research in this thesis builds towards a general interpretability framework composed of four key steps:

1. **Identify Explanation Desiderata** Different explanations require different properties. These can vary depending on the context, data type, model architecture, and intended audience. Therefore, it is important to identify these objectives upfront.

2. **Find a concept or evidence of a behavioural phenomenon.** The main technique employed in this thesis is a contrastive approach, comparing model behaviour across two settings that differ along a single axis, which helps isolate and analyse the behaviour of interest.

3. **Verify that the model uses the concept.** A concept may be present in a model's representation space without being used by the model. Therefore, it is important to verify that the model uses the concept. In this thesis, we do this by intervening on a concept – modifying the latent representation to amplify a concept – to estimate its influence on the model's output.

4. **Extract the explanation in a human-interpretable form.** The goal is often to present explanations in a form that humans can understand and/or learn from. In this thesis, we focus on prototypical examples, as humans are highly effective at learning from examples.

In practice, each of these steps presents challenges. Consider the first step: identifying explanation desiderata. The goals we care about often vary as interpretability is applied in different settings. Translating high-level aims, such as fairness or usability, into concrete mathematical objectives is not always straightforward. Some properties are difficult to define precisely without losing important nuance. Even when goals can be formalised, they may be defined in different ways by different researchers, making it difficult to compare or evaluate methods in a consistent way. The lack of unified evaluation protocols can lead to a chicken-and-egg problem in research papers: a researcher defines a property of interest and then creates a metric to measure it. However, if a method performs well on that metric, it may be because the metric is tailored to the property, not because the method is truly effective. In essence, we risk metric hacking, resulting in a strong performance that may not generalise beyond the setup.

Next, finding meaningful concepts in a model is like looking for a needle in a haystack. As machine learning models have grown in size and complexity, pinpointing where knowledge is stored has become increasingly difficult. Even when we find evidence of a concept or phenomenon, it is still difficult to verify whether we truly identified a part of the model behaviour or whether our finding is a false positive – an artefact of the interpretability method. This is why the next step – verification–is so important.

However, verifying concept usage is also non-trivial. The correct way to intervene on a concept is often unclear. For example, if we choose to intervene on a neuron, what value should we assign to the neuron? The mean? The minimum? Moreover, does this value make sense given the joint distribution over the other neurons? Poorly designed interventions can lead to out-of-distribution behaviour, making results hard to interpret. Furthermore, simple interventions may be misleading altogether. A single feature might not be sufficient to influence behaviour, due to reasons such as the hydra effect [McGrath et al., 2023], where the model encodes backup mechanisms for important model behaviour.

However, verifying concept usage is not always necessary – particularly when the goal is to extract novel insights for human use. In such cases, the utility of a concept to humans can outweigh the need for strict faithfulness to the model's internal representations. However, human evaluation is costly. To minimise reliance on manual assessment, it is important to reduce the number of candidate concepts through filtering techniques, such as novelty and teachability scores (see Chapter 6).

Lastly, extracting concepts in a human-interpretable form is inherently challenging. In translating internal representations into forms accessible to humans – such as natural language, examples, or rules – we may lose important nuance or fidelity. The model's internal concepts may not map cleanly to human concepts, and some representations may encode abstractions for which we lack a corresponding representation or vocabulary. For example, in language models, we are limited to human language. However, the model may encode a useful concept that we do not yet have a word for. This step remains one of the key challenges in making interpretability truly valuable.

In conclusion, all of the steps are difficult. However, the broader motivation behind this framework is to move away from thinking about interpretability as a single-step process. Instead, we can think about it as hypothesis testing. Each step provides evidence for a hypothesis about model behaviour. While each individual has a false positive rate, combining multiple, partially independent steps lowers the risk of drawing the wrong conclusion. This makes it easier to build confidence in our conclusions, particularly in the absence of a ground truth that can be easily verified.

As stated in Chapter 2, this entire process is guided by intended use. While this thesis explores several applications, its ultimate trajectory points towards one of the most ambitious goals for the field: knowledge discovery. Two main takeaways emerge. First, interpretability is most effective when we focus on the usefulness of an explanation for a specific purpose rather than aiming for a perfect or complete one. Second, by structuring interpretability as an iterative, evidence-building process, we gain a more robust and practical way to understand model behaviour. It is this structured approach that can allow us to explain what AI models have learned, and to leverage them as tools for discovering novel, human-understandable knowledge about the world.

8
Discussion and Future Work

In this section, we discuss open challenges in the field.

8.1 Definitions are Difficult

One of the central challenges in interpretability research is specifying definitions [Lipton, 2018]. Interpretability, at its core, aims to provide explanations for model behaviour–but what is an explanation? Even if we could agree on a definition, explanations are often not unique. Multiple valid explanations may account for the same behaviour, depending on the type and the criteria used for evaluation.

Consider concepts and features – both terms are frequently used in interpretability research but rarely formally defined. A definition we use in this thesis is that a concept is *a unit of knowledge*. However, this definition makes assumptions – for example, that a concept is concise or minimal. While this assumption is common in practice, it imposes constraints on what counts as a valid explanation, and these constraints may not always be justified. Consider a bike.[1] What are the features of a bike? Should a feature be a partial curve, a tire, or a complete wheel? Perhaps, more importantly, what level of feature granularity does the model use?

A similar problem is found in the field of object-oriented learning, where objects are difficult to define. It has been argued that objects only gain meaning relative to the task or context [Smith, 2020, Bienenstock et al., 1996]. Similarly, in this thesis, we argued that concepts are defined relative to their usefulness for a task.

[1] This idea arose when working on Smith et al. [2021]. See also Smith [2020].

This perspective aligns with the idea that concepts are cognitive abilities [Margolis and Laurence, 2023] and frames them functionally.

While this approach may not be perfect, it offers a practical way forward. It allows us to move from abstract debates to empirical investigation – trading off some philosophical precision for tractability and enabling progress in understanding and explaining model behaviour.

8.2 Mutable Concepts

The inherent ambiguity surrounding the definition of concepts underscores a deeper philosophical challenge. Within the field of philosophy, there is extensive research on trying to define concepts and their role in human cognition. Concepts can refer to mental representations, cognitive abilities, or abstract categories [Margolis and Laurence, 2023]. Two different people may hold different internal definitions of the same concept – even one as familiar as a `cat`. Moreover, these definitions can evolve over time with experience, culture, and context.

Building on the idea that the definition of a concept is not fixed, perhaps we should reconsider how we think about concepts and features. Often, we extract features or concepts from a network operating under the assumption that these entities exist independently of specific inputs and can be interpreted in isolation. This perspective assumes that concepts or features are immutable and context-independent.

However, this viewpoint may oversimplify the intricate behaviour of concepts. In reality, the meaning and function of a concept can vary significantly depending on the context or input.[2] Extracting a singular, absolute representation of a concept may not capture its multifaceted nature. It would be interesting to understand the progression of a concept as a model trains. Similarly, we could more explicitly capture the multifaceted nature of a concept. For example, we could treat it as a function that maps an input to a vector. This approach may allow us to improve our understanding of the concept's role in a model – and how it changes across inputs.

8.3 Human Learning

When learning from AI systems, we can think about the three core components:

1. Training models that meaningfully outperform humans.

[2] We, as researchers, take a similar approach to word embeddings [Peters et al., 2018].

2. Finding knowledge that is unique to AI systems.

3. Extracting the knowledge in a human-interpretable form.

Each of these components is non-trivial and requires careful consideration.

Consider the first step. By *meaningfully*, we mean that the model outperforms humans due to better reasoning or novel ideas – rather than a computational advantage. While AI systems may outperform humans by calculating more or faster, this kind of performance gap is often not what we care about from a learning perspective. However, evaluating whether a performance gap is meaningful is inherently difficult. In practice, we evaluate performance on a specific task under specific conditions. This often amounts to asking whether a human could solve the task within a limited time. However, a more relevant question is often: could a human solve the task given unlimited time or more information?

Another important factor is which human we consider. Are we comparing against a randomly sampled individual or an expert? Finally, the evaluation metric itself is often a proxy – measuring only part of the behaviour we care about. As a result, it is often difficult to definitively conclude whether an AI system performs meaningfully better than a human in a given domain.

Now, moving on to the second part. Let us suppose we have a model we want to learn from. A natural question follows: *to what extent can we understand it?* The representation space of a model is high-dimensional and complex. Some of this knowledge may be interpretable if we can extract it. However, some of it may be fundamentally inaccessible to humans. Here, we refer to 'humans' in the collective sense–but it is important to note that people differ in their backgrounds, experiences, and cognitive abilities. Understanding across humans is not uniform, and this diversity is a strength, as it may increase the likelihood of learning from AI systems.

One component of interest is quantifying the human-AI knowledge gap. In Chapter 6, we take a first step by comparing the span of human and AI representation spaces. However, more refined approaches are possible, such as estimating distributional differences between human and model behaviours. A second step would be to estimate how much of the model's knowledge is learnable by humans. In Chapter 6, we approached this using estimating *teachability*. However, this idea could be improved upon. Inspired by cognitive science, we could build models that simulate human learners to estimate more accurately what type of knowledge is learnable.

Now, moving on to the last step – extracting knowledge in human-interpretable form is difficult and unlikely to be lossless. In Chapter 6, we approached this in three ways: identifying prototypical examples, showing optimal actions, and linking new concepts to known concepts. These methods rely on formats humans are already familiar with – concepts, inputs, and actions. This constraint can lead to important limitations. First, what happens when a concept cannot be expressed in any of these forms? Second, does our insistence on mapping new ideas to familiar ones distort our understanding of the new concept? It may lead us to interpret every concept as a reapplication or recombination of existing ideas. This suggests the need for richer representational tools–or entirely new conceptual languages [Hewitt et al., 2025].

Finally, suppose we can extract knowledge from a model. How can we best teach it to humans? While this depends on the individual, there are useful design principles we can draw on. In Chapter 6, we allowed humans to learn by observing the AZ's calculations. A more powerful approach would involve interaction, for example, by allowing the experts to play against the model to explore the reasoning behind its suggestions. One important signal in this process is understanding *why* the human's solution was incorrect. More broadly, the fields of cognitive science and human-computer interaction are critical for designing learning environments that support human understanding. One concrete direction is identifying which settings are most conducive to human learning [Schut et al., 2025b].

This challenge of teaching is further compounded by the mutable nature of concepts, a central theme of our earlier discussion. As we have established, different people will inevitably hold different conceptualisations of the same concept, each shaped by unique backgrounds and prior knowledge. This inherent variability in human cognition implies that a one-size-fits-all approach to knowledge transfer is unlikely to succeed. Consequently, effective learning environments must be adaptive, capable of acknowledging and bridging the gap between an AI's representation and the individual learner's unique conceptual framework.

8.4 Multi-step Explanations

Most model explanations are 'one-step' – for a given model prediction, we provide a single explanation – typically without feedback, iteration, or clarification. While this can be informative, it may not be optimal for how humans best understand or learn from information.

As previously discussed, humans often learn better through interaction. This may help explain the success of systems like ChatGPT: the explanation is not fixed but part of an ongoing exchange. A user can ask for clarification, probe edge cases, or challenge assumptions. The conversation continues until the human either understands or decides to disagree.

This type of interaction improves performance and understanding on multiple levels. For example, in a debate between agents, an explanation is refined and tested through challenge. Interaction is not only useful for improving human understanding – it can also improve model behaviour.

Now, consider the context of clinical decision-making. If a model fails to use a feature that clinicians view as essential, trust in the system is likely to be low. An ideal setup would not only allow the model to explain its reasoning but also allow the clinician to provide a counter-explanation, highlighting the missing feature or reasoning path. The model can then incorporate this feedback by updating its representations, circuits and/or adjusting its predictions.

This idea is simple: explanations should not be static, one-shot outputs but part of an interactive loop. A multi-step explanation allows both the human and the model to update, resulting in improved understanding, trust, and performance.

Bibliography

R. Achtibat, M. Dreyer, I. Eisenbraun, S. Bosse, T. Wiegand, W. Samek, and S. Lapuschkin. From "where" to "what": Towards human-understandable explanations through concept relevance propagation, 2022.

A. Adadi and M. Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160, 2018.

J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, pages 9525–9536, 2018.

J. Adebayo, M. Muelly, I. Liccardi, and B. Kim. Debugging tests for model explanations. *arXiv preprint arXiv:2011.05429*, 2020.

J. Adebayo, M. Muelly, H. Abelson, and B. Kim. Post hoc explanations may be ineffective for detecting unknown spurious correlation. In *International conference on learning representations*, 2021.

H. Afrabandpey, T. Peltola, J. Piironen, A. Vehtari, and S. Kaski. A decision-theoretic approach for model interpretability in bayesian framework. *Machine Learning*, pages 1–22, 2020.

A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

K. Ahuja, H. Diddee, R. Hada, M. Ochieng, K. Ramesh, P. Jain, A. Nambi, T. Ganu, S. Segal, M. Axmed, K. Bali, and S. Sitaram. Mega: Multilingual evaluation of generative ai, 2023. URL `https://arxiv.org/abs/2303.12528`.

U. Aïvodji, H. Arai, O. Fortineau, S. Gambs, S. Hara, and A. Tapp. Fairwashing: the risk of rationalization. In *International Conference on Machine Learning*, pages 161–170. PMLR, 2019.

A. Aizawa, E. Aramaki, B. Chen, F. Cheng, H. Deguchi, R. Enomoto, K. Fujii, K. Fukumoto, T. Fukushima, N. Han, et al. Llm-jp: A cross-organizational project for the research and development of fully open japanese llms. *arXiv preprint arXiv:2407.03963*, 2024.

A. Alharin, T.-N. Doan, and M. Sartipi. Reinforcement learning interpretation methods: A survey. *IEEE Access*, 8:171058–171077, 2020.

Alisa Melekhina. Interview with gm chuchelov - caruana's coach, 2014. URL `https://en.chessbase.com/post/interview-with-gm-chuchelov-caruana-s-coach`. [Online; accessed 1-August-2023].

D. Alvarez-Melis and T. S. Jaakkola. On the robustness of interpretability methods. *arXiv preprint arXiv:1806.08049*, 2018a.

D. Alvarez-Melis and T. S. Jaakkola. On the robustness of interpretability methods. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 149–158. PMLR, 2018b.

D. Alvarez-Melis and T. S. Jaakkola. Towards robust interpretability with self-explaining neural networks. In *Advances in Neural Information Processing Systems*, pages 7786–7795, 2018c.

M. Ancona, E. Ceolini, C. Öztireli, and M. Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv preprint arXiv:1711.06104*, 2017.

C. Anders, P. Pasliev, A.-K. Dombrowski, K.-R. Müller, and P. Kessel. Fairwashing explanations with off-manifold detergent. In *International Conference on Machine Learning*, pages 314–323. PMLR, 2020.

R. M. Annasamy and K. Sycara. Towards better interpretability in deep q-networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4561–4569, 2019.

Anthropic. Tracing model outputs to the training data. `https://www.anthropic.com/index/influence-functions`, 2023. Accessed: 4 November.

L. Arras, A. Osman, K.-R. Müller, and W. Samek. Evaluating recurrent neural network explanations. *arXiv preprint arXiv:1904.11829*, 2019.

V. Aryabumi, J. Dang, D. Talupuru, S. Dash, D. Cairuz, H. Lin, B. Venkitesh, M. Smith, K. Marchisio, S. Ruder, A. Locatelli, J. Kreutzer, N. Frosst, P. Blunsom, M. Fadaee, A. Üstün, and S. Hooker. Aya 23: Open weight releases to further multilingual progress, 2024.

A. Atrey, K. Clary, and D. Jensen. Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning, 2020.

S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, and K.-R. MÃžller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11(Jun):1803–1831, 2010.

A. Bai, C.-K. Yeh, P. Ravikumar, N. Y. C. Lin, and C.-J. Hsieh. Concept gradient: Concept-based interpretation without linear assumption, 2022.

Y. Bang, S. Cahyawijaya, N. Lee, W. Dai, D. Su, B. Wilie, H. Lovenia, Z. Ji, T. Yu, W. Chung, Q. V. Do, Y. Xu, and P. Fung. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity, 2023. URL `https://arxiv.org/abs/2302.04023`.

H. Baniecki, W. Kretowicz, and P. Biecek. Fooling partial dependence via data poisoning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 121–136. Springer, 2022.

S. Barocas, M. Hardt, and A. Narayanan. Fairness and machine learning. 2019. Book draft. `https://fairmlbook.org/`.

O. Bastani, C. Kim, and H. Bastani. Interpreting blackbox models via model extraction. *arXiv preprint arXiv:1705.08504*, 2017.

O. Bastani, Y. Pu, and A. Solar-Lezama. Verifiable reinforcement learning via policy extraction. *Advances in neural information processing systems*, 31, 2018.

D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6541–6549, 2017.

N. Belrose, Z. Furman, L. Smith, D. Halawi, I. Ostrovsky, L. McKinney, S. Biderman, and J. Steinhardt. Eliciting latent predictions from transformers with the tuned lens, 2023. URL `https://arxiv.org/abs/2303.08112`.

L. Bereska and E. Gavves. Mechanistic interpretability for ai safety–a review. *arXiv preprint arXiv:2404.14082*, 2024.

E. Bienenstock, S. Geman, and D. Potter. Compositionality, mdl priors, and object recognition. In *Advances in Neural Information Processing Systems 9*, pages 838–844, 1996.

B. Bilodeau, N. Jaques, P. W. Koh, and B. Kim. Impossibility theorems for feature attribution. *Proceedings of the National Academy of Sciences*, 121(2):e2304406120, 2024. doi: 10.1073/pnas.2304406120. URL `https://www.pnas.org/doi/abs/10.1073/pnas.2304406120`.

V. Biscione and J. S. Bowers. Convolutional neural networks are not invariant to translation, but they can learn to be. *Journal of Machine Learning Research*, 22 (229):1–28, 2021a.

V. Biscione and J. S. Bowers. Convolutional neural networks are not invariant to translation, but they can learn to be, 2021b. URL `https://arxiv.org/abs/2110.05861`.

Bojun Guo. Syzygy tablebase, 2023. URL `https://syzygy-tables.info/`. [Online; accessed 23-July-2023].

D. Bouchacourt and L. Denoyer. EDUCE: Explaining model Decisions through Unsupervised Concepts Extraction. *arXiv preprint arXiv:1905.11852*, 2019.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984. ISBN 9780412048418. URL `https://www.taylorfrancis.com/books/mono/10.1201/9781315139470/classification-regression-trees-leo-breiman-jerome-friedman-olshen-charles-sto`

J. Brinkmann, C. Wendler, C. Bartelt, and A. Mueller. Large language models share representations of latent grammatical concepts across typologically diverse languages. *arXiv preprint arXiv:2501.06346*, 2025.

T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.

R. Brunia and C. van Wijgerden. *Steps Method*. Self-published, 2021.

Cambridge Academic Content Dictionary. *Definition: Explanation*. Cambridge University Press, 2023.

N. Cammarata, S. Carter, G. Goh, C. Olah, M. Petrov, L. Schubert, C. Voss, B. Egan, and S. K. Lim. Thread: Circuits. *Distill*, 2020. doi: 10.23915/distill.00024. https://distill.pub/2020/circuits.

L. Chai, J. Yang, T. Sun, H. Guo, J. Liu, B. Wang, X. Liang, J. Bai, T. Li, Q. Peng, and Z. Li. xcot: Cross-lingual instruction tuning for cross-lingual chain-of-thought reasoning, 2024. URL `https://arxiv.org/abs/2401.07037`.

P. Chalasani, J. Chen, A. R. Chowdhury, S. Jha, and X. Wu. Concise explanations of neural networks using adversarial training. *International Conference on Machine Learning 2020*, pages arXiv–1810.06583v9, 2020.

Z. Chen, Y. Bei, and C. Rudin. Concept whitening for interpretable image recognition. *Nature Machine Intelligence*, 2:772–782, 2020a.

Z. Chen, Y. Bei, and C. Rudin. Concept whitening for interpretable image recognition. *Nature Machine Intelligence*, 2(12):772–782, 2020b.

ChessBase. Chessbase Mega Database. database.chessbase.com, 2021.

Y. Y. Chiu, L. Jiang, B. Y. Lin, C. Y. Park, S. S. Li, S. Ravi, M. Bhatia, M. Antoniak, Y. Tsvetkov, V. Shwartz, and Y. Choi. Culturalbench: a robust, diverse and challenging benchmark on measuring the (lack of) cultural knowledge of llms, 2024. URL `https://arxiv.org/abs/2410.02677`.

A. J. Chris Olah. What would be the most safety-relevant features in language models? *Transformer Circuits Thread*, 2023. URL `https://transformer-circuits.pub/2023/july-update/index.html`.

J. R. Clough, I. Oksuz, E. Puyol-Antón, B. Ruijsink, A. P. King, and J. A. Schnabel. Global and local interpretability for cardiac MRI classification. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 656–664. Springer, 2019.

W. W. Cohen. Fast effective rule induction. In *Machine learning proceedings 1995*, pages 115–123. Elsevier, 1995.

J. Colin, T. FEL, R. Cadene, and T. Serre. What i cannot predict, i do not understand: A human-centered evaluation framework for explainability methods. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

A. Conneau, G. Kruszewski, G. Lample, L. Barrault, and M. Baroni. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *arXiv preprint arXiv:1805.01070*, 2018.

Y. Coppens, K. Efthymiadis, T. Lenaerts, A. Nowé, T. Miller, R. Weber, and D. Magazzeni. Distilling deep reinforcement learning policies in soft decision trees. In *Proceedings of the IJCAI 2019 workshop on explainable artificial intelligence*, pages 1–6, 2019.

D. Corbit, S. Natarajan, and F. Mosca. Strategic test suite, 2014. URL `https://sites.google.com/site/strategictestsuite/`.

T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

J. Crabbé and M. van der Schaar. Concept activation regions: A generalized framework for concept-based explanations. In *Advances in Neural Information Processing Systems*, volume 35, pages 2590–2607, 2022.

J. Crabbé and M. van der Schaar. Evaluating the robustness of interpretability methods through explanation invariance and equivariance. *arXiv preprint arXiv:2304.06715*, 2023.

M. Craven and J. Shavlik. Extracting tree-structured representations of trained networks. *Advances in neural information processing systems*, 8, 1995.

T. Cunningham, N. Nanda, T. Li, C. Chughtai, A. Chen, and D. Ganguli. Sparse autoencoders find interpretable directions in gpt, 2023.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

R. Dazeley, P. Vamplew, and F. Cruz. Explainable reinforcement learning for broad-xai: A conceptual framework and survey, 2021.

DecodeChess. Understanding Chess with Explainable AI. `https://decodechess.com/about/`,, 2017.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Y. Deng, W. Zhang, S. J. Pan, and L. Bing. Multilingual jailbreak challenges in large language models, 2024. URL `https://arxiv.org/abs/2310.06474`.

C. Denison, M. MacDiarmid, F. Barez, D. Duvenaud, S. Kravec, S. Marks, N. Schiefer, R. Soklaski, A. Tamkin, J. Kaplan, B. Shlegeris, S. R. Bowman, E. Perez, and E. Hubinger. Sycophancy to subterfuge: Investigating reward-tampering in large language models, 2024. URL `https://arxiv.org/abs/2406.10162`.

S. V. Deshmukh, S. R, S. Vijay, J. Subramanian, and C. Agarwal. Counterfactual explanation policies in rl, 2023.

A. Dhurandhar, P.-Y. Chen, R. Luss, C.-C. Tu, P. Ting, K. Shanmugam, and P. Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Advances in Neural Information Processing Systems*, pages 592–603, 2018.

S. Diamond and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

C. H. Ding, T. Li, and M. I. Jordan. Convex and semi-nonnegative matrix factorizations. *IEEE transactions on pattern analysis and machine intelligence*, 32(1):45–55, 2008.

F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

A. Dosovitskiy, J. Tobias Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1538–1546, 2015.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

D. Dua and C. Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

A. Dubey et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

C. Dumas, V. Veselovsky, G. Monea, R. West, and C. Wendler. How do llamas process multilingual text? a latent exploration through activation patching. In *ICML 2024 Workshop on Mechanistic Interpretability*, 2024.

G. K. Dziugaite, S. Ben-David, and D. M. Roy. Enforcing interpretability and its statistical impacts: Trade-offs between accuracy and interpretability. *arXiv preprint arXiv:2010.13764*, 2020.

N. Elhage, N. Nanda, C. Olsson, T. Henighan, N. Joseph, B. Mann, A. Askell, Y. Bai, A. Chen, T. Conerly, N. DasSarma, D. Drain, D. Ganguli, Z. Hatfield-Dodds, D. Hernandez, A. Jones, J. Kernion, L. Lovitt, K. Ndousse, D. Amodei, T. Brown, J. Clark, J. Kaplan, S. McCandlish, and C. Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

N. Elhage, T. Hume, C. Olsson, N. Schiefer, T. Henighan, S. Kravec, Z. Hatfield-Dodds, R. Lasenby, D. Drain, C. Chen, R. Grosse, S. McCandlish, J. Kaplan, D. Amodei, M. Wattenberg, and C. Olah. Toy models of superposition, 2022a.

N. Elhage, T. Hume, C. Olsson, N. Schiefer, T. Henighan, S. Kravec, Z. Hatfield-Dodds, R. Lasenby, D. Drain, C. Chen, R. Grosse, S. McCandlish, J. Kaplan, D. Amodei, M. Wattenberg, and C. Olah. Toy models of superposition, 2022b. URL https://arxiv.org/abs/2209.10652.

J. Etxaniz, G. Azkune, A. Soroa, O. L. de Lacalle, and M. Artetxe. Do multilingual language models think better in english?, 2023. URL https://arxiv.org/abs/2308.01223.

T. Fel, A. Picard, L. Bethune, T. Boissin, D. Vigouroux, J. Colin, R. Cadène, and T. Serre. Craft: Concept recursive activation factorization for explainability. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2711–2721, 2023.

X. Feng, Y. Luo, Z. Wang, H. Tang, M. Yang, K. Shao, D. Mguni, Y. Du, and J. Wang. Chessgpt: Bridging policy learning and language modeling, 2023.

FIDE. Fide handbook c.02, 2019. URL https://handbook.fide.com/. [Online; accessed 13-October-2023].

N. Fiekas. python-chess. https://github.com/niklasf/python-chess, 10 2023.

C. Fierro, N. Foroutan, D. Elliott, and A. Søgaard. How do multilingual language models remember facts?, 2025. URL https://arxiv.org/abs/2410.14387.

M. Finkelstein, L. Liu, N. L. Schlot, Y. Kolumbus, D. C. Parkes, J. S. Rosenshein, and S. Keren. Explainable reinforcement learning via model transforms, 2022.

M. Flora, C. Potvin, A. McGovern, and S. Handler. Comparing explanation methods for traditional machine learning models part 1: An overview of current methods and quantifying their disagreement. *arXiv preprint arXiv:2211.08943*, 2022.

P. Fonagy and E. Allison. The role of epistemic trust in child development: implications for intervention. *Mind & Brain: The Journal of Psychiatry*, 5 (1):1–12, 2014.

R. Fong and A. Vedaldi. Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. *CoRR*, abs/1801.03454, 2018.

R. C. Fong and A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE international conference on computer vision*, pages 3429–3437, 2017.

J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019. URL `https://arxiv.org/abs/1803.03635`.

T. Freiesleben. The intriguing relation between counterfactual explanations and adversarial examples. *Minds and Machines*, 32(1):77–109, Oct. 2021. ISSN 1572-8641. doi: 10.1007/s11023-021-09580-9. URL `http://dx.doi.org/10.1007/s11023-021-09580-9`.

K. Fujii, T. Nakamura, M. Loem, H. Iida, M. Ohi, K. Hattori, H. Shota, S. Mizuki, R. Yokota, and N. Okazaki. Continual pre-training for cross-lingual llm adaptation: Enhancing japanese language capabilities. *arXiv preprint arXiv:2404.17790*, 2024.

K.-I. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural networks*, 2(3):183–192, 1989.

C. Fürböck, M. Perkonigg, T. Helbich, K. Pinker, V. Romeo, and G. Langs. Identifying Phenotypic Concepts Discriminating Molecular Breast Cancer Sub-Types. In *MICCAI*, 2022.

J. Gajcin and I. Dusparic. Reccover: Detecting causal confusion for explainable reinforcement learning, 2022.

Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.

Y. Gal and L. Smith. Sufficient conditions for idealised models to have no adversarial examples: a theoretical and empirical study with bayesian neural networks. *arXiv preprint arXiv:1806.00667*, 2018.

C. F. Gauss. *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*. Perthes and Besser, 1809.

R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness, 2022. URL `https://arxiv.org/abs/1811.12231`.

R. Geirhos, R. S. Zimmermann, B. Bilodeau, W. Brendel, and B. Kim. Don't trust your eyes: on the (un) reliability of feature visualizations. *arXiv preprint arXiv:2306.04719*, 2023.

M. Geva, J. Bastings, K. Filippova, and A. Globerson. Dissecting recall of factual associations in auto-regressive language models. In H. Bouamor, J. Pino, and K. Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12216–12235, Singapore, Dec. 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.751. URL `https://aclanthology.org/2023.emnlp-main.751/`.

A. Ghandeharioun, B. Kim, C.-L. Li, B. Jou, B. Eoff, and R. W. Picard. Dissect: Disentangled simultaneous explanations via concept traversals. *arXiv preprint arXiv:2105.15164*, 2021.

A. Ghandeharioun, A. Caciularu, A. Pearce, L. Dixon, and M. Geva. Patchscope: A unifying framework for inspecting hidden representations of language models. *arXiv preprint arXiv:2401.06102*, 2024.

A. Ghorbani, A. Abid, and J. Zou. Interpretation of neural networks is fragile. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3681–3688, 2019a.

A. Ghorbani, J. Wexler, J. Zou, and B. Kim. Towards automatic concept-based explanations. In *Advances in Neural Information Processing Systems*, 2019b.

A. Ghorbani, J. Wexler, J. Y. Zou, and B. Kim. Towards automatic concept-based explanations. In *Advances in Neural Information Processing Systems*, pages 9273–9282, 2019c.

S. Ghosh, K. Yu, F. Arabshahi, and K. Batmanghelich. Dividing and conquering a blackbox to a mixture of interpretable models: Route, interpret, repeat. In *ICML*, 2023.

L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE, 2018.

L. H. Gilpin, A. R. Paley, M. A. Alam, S. Spurlock, and K. J. Hammond. "explanation" is not a technical term: The problem of ambiguity in xai, 2022. URL `https://arxiv.org/abs/2207.00007`.

M. Goldblum, D. Tsipras, C. Xie, X. Chen, A. Schwarzschild, D. Song, A. Mądry, B. Li, and T. Goldstein. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1563–1580, 2022.

J. González-Díaz and I. Palacios-Huerta. Alphazero ideas. *Available at SSRN 4140916*, 2022.

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. URL `http://arxiv.org/abs/1412.6572`.

Google. gemma-2-27b. Hugging Face repository, 2023. URL `https://huggingface.co/google/gemma-2-27b`. Accessed: November 1st 2024.

A. Gopnik, A. N. Meltzoff, and P. K. Kuhl. *The Scientist in the Crib: Minds, Brains, and How Children Learn*. HarperCollins, 1999. ISBN 9780060933845.

Y. Goyal, A. Feder, U. Shalit, and B. Kim. Explaining classifiers with causal concept effect (cace). *arXiv preprint arXiv:1907.07165*, 2019.

W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, M. Norouzi, and K. Swersky. Your classifier is secretly an energy based model and you should treat it like one. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=Hkxzx0NtDB`.

M. Graziani, V. Andrearczyk, and H. Müller. Regression concept vectors for bidirectional explanations in histopathology. In *Understanding and Interpreting Machine Learning in Medical Image Computing Applications*, pages 124–132. Springer, 2018.

M. Graziani, V. Andrearczyk, M. M. S., and H. Müller. Concept attribution: Explaining CNN decisions to physicians. *Computers in Biology and Medicine*, 123, Aug. 2020.

S. Greydanus, A. Koul, J. Dodge, and A. Fern. Visualizing and understanding atari agents. In *International Conference on Machine Learning*, pages 1792–1801. PMLR, 2018.

R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.

Y. Guo, G. Shang, and C. Clavel. Benchmarking linguistic diversity of large language models. *arXiv preprint arXiv:2412.10271*, 2024.

P. Gupta, N. Puri, S. Verma, D. Kayastha, S. Deshmukh, B. Krishnamurthy, and S. Singh. Explain your move: understanding agent actions using specific and relevant feature attribution. In *International Conference on Learning Representations (ICLR)*, 2020.

W. Gurnee, N. Nanda, M. Pauly, K. Harvey, D. Troitskii, and D. Bertsimas. Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610*, 2023.

C. Hansen. *Back to Basics: Chess Openings*. Russel Enterprises, 2021.

P. Hase, M. Bansal, B. Kim, and A. Ghandeharioun. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 17643–17668, 2023.

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034. IEEE Computer Society, 2015. doi: 10.1109/ICCV.2015.123. URL `https://doi.org/10.1109/ICCV.2015.123`.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

T. Heap, T. Lawson, L. Farnik, and L. Aitchison. Sparse autoencoders can interpret randomly initialized transformers, 2025. URL `https://arxiv.org/abs/2501.17727`.

L. Heidemann, M. Monnet, and K. Roscher. Concept Correlation and Its Effects on Concept-Based Models. In *WACV*, pages 4769–4777, 2023. doi: 10.1109/WACV56688.2023.00476.

J. Heo, S. Joo, and T. Moon. Fooling neural network interpretations via adversarial model manipulation. *Advances in neural information processing systems*, 32, 2019.

J. Hewitt, R. Geirhos, and B. Kim. We can't understand ai using our existing vocabulary, 2025. URL `https://arxiv.org/abs/2502.07586`.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

E. Hoffer, T. Ben-Nun, I. Hubara, N. Giladi, T. Hoefler, and D. Soudry. Augment Your Batch: Improving Generalization Through Instance Repetition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8126–8135, 2020.

R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine learning*, 11:63–90, 1993.

M. Honnibal and I. Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

S. Hooker, D. Erhan, P.-J. Kindermans, and B. Kim. A benchmark for interpretability methods in deep neural networks. *Advances in neural information processing systems*, 32, 2019.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

H. Huang, T. Tang, D. Zhang, W. X. Zhao, T. Song, Y. Xia, and F. Wei. Not all languages are created equal in llms: Improving multilingual capability by cross-lingual-thought prompting, 2023. URL `https://arxiv.org/abs/2305.07004`.

J. Huang and K. C.-C. Chang. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*, 2022.

J. Huysmans, K. Dejaeger, C. Mues, J. Vanthienen, and B. Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51(1):141–154, 2011.

B. Höltgen, L. Schut, J. M. Brauner, and Y. Gal. Deduce: Generating counterfactual explanations efficiently, 2021. URL `https://arxiv.org/abs/2111.15639`.

A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, pages 125–136, 2019.

T. Jaunet, R. Vuillemot, and C. Wolf. Drlviz: Understanding decisions and memory in deep reinforcement learning. In *Computer Graphics Forum*, volume 39, pages 49–61. Wiley Online Library, 2020.

H. Jhamtani, V. Gangal, E. Hovy, G. Neubig, and T. Berg-Kirkpatrick. Learning to generate move-by-move commentary for chess games from large-scale social forum data. In *The 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, Melbourne, Australia, July 2018.

Y. Jia, J. McDermid, T. Lawton, and I. Habli. The role of explainability in assuring safety of machine learning in healthcare. *IEEE Transactions on Emerging Topics in Computing*, 10(4):1746–1760, 2022.

A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. de las Casas, E. B. Hanna, F. Bressand, G. Lengyel, G. Bour, G. Lample, L. R. Lavaud, L. Saulnier, M.-A. Lachaux, P. Stock, S. Subramanian, S. Yang, S. Antoniak, T. L. Scao, T. Gervet, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed. Mixtral of experts, 2024. URL `https://arxiv.org/abs/2401.04088`.

S. Joshi, O. Koyejo, W. Vijitbenjaronk, B. Kim, and J. Ghosh. Towards realistic individual recourse and actionable explanations in black-box decision making systems. *arXiv preprint arXiv:1907.09615*, 2019.

J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

A.-H. Karimi, K. Muandet, S. Kornblith, B. Schölkopf, and B. Kim. On the relationship between explanation and prediction: A causal view. *arXiv preprint arXiv:2212.06925*, 2022.

A. Karpathy, J. Johnson, and L. Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, pages 5574–5584, 2017.

D. S. Kermany, M. Goldbaum, W. Cai, C. C. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan, and et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131.e9, 2018. doi: 10.1016/j.cell.2018.02.010.

Y. Kerner. Case-based evaluation in computer chess. In J.-P. Haton, M. Keane, and M. Manago, editors, *Advances in Case-Based Reasoning*, pages 240–254, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg. ISBN 978-3-540-45052-8.

C. Kidd and B. Y. Hayden. The psychology and neuroscience of curiosity. *Neuron*, 88(3):449–460, 2015. doi: 10.1016/j.neuron.2015.09.010.

B. Kim. Beyond interpretability: developing a language to shape our relationships with ai, 05 2022. URL `https://medium.com/@beenkim/beyond-interpretability-4bf03bbd9394`.

B. Kim, R. Khanna, and O. O. Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. *Advances in neural information processing systems*, 29, 2016.

B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR, 2018.

P.-J. Kindermans, S. Hooker, J. Adebayo, M. Alber, K. T. Schütt, S. Dähne, D. Erhan, and B. Kim. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer, 2019.

D. King. *How to Win at Chess*. Everyman Chess, 2000.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. URL `http://arxiv.org/abs/1412.6980`.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

J. Klaise, A. Van Looveren, G. Vacanti, and A. Coca. Alibi: Algorithms for monitoring and explaining machine learning models. *GitHub*, 2020. URL `https://github.com/SeldonIO/alibi`.

P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.

P. W. Koh, T. Nguyen, Y. S. Tang, S. Mussmann, E. Pierson, B. Kim, and P. Liang. Concept bottleneck models. *arXiv preprint arXiv:2007.04612*, 2020.

E. Kosoy, E. R. Reagan, L. Lai, A. Gopnik, and D. K. Cobb. Comparing machines and children: Using developmental psychology experiments to assess the strengths and weaknesses of lamda responses. *arXiv preprint arXiv:2305.11243*, 2023.

J. Kossen, J. Han, M. Razzak, L. Schut, S. Malik, and Y. Gal. Semantic entropy probes: Robust and cheap hallucination detection in llms, 2024. URL `https://arxiv.org/abs/2406.15927`.

A. Koul, S. Greydanus, and A. Fern. Learning finite state representations of recurrent policy networks, 2018.

N. Kriegeskorte. Deep neural networks: a new framework for modeling biological vision and brain information processing. *Annual review of vision science*, 1: 417–446, 2015.

A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

I. Kuperwajs, H. H. Schütt, and W. J. Ma. Using deep neural networks as a guide for modeling human planning. *Scientific reports*, 13(1):20269, 2023.

I. Lage, A. Ross, S. J. Gershman, B. Kim, and F. Doshi-Velez. Human-in-the-loop interpretability prior. *Advances in neural information processing systems*, 31, 2018.

O. Lahav, N. Mastronarde, and M. van der Schaar. What is interpretable? using machine learning to design interpretable decision-support systems. *arXiv preprint arXiv:1811.10799*, 2018.

H. Lakkaraju, S. H. Bach, and J. Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1675–1684, 2016.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413, 2017.

T. Laugel, M.-J. Lesot, C. Marsala, X. Renard, and M. Detyniecki. The dangers of post-hoc interpretability: Unjustified counterfactual explanations. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 2801–2807. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/388. URL `https://doi.org/10.24963/ijcai.2019/388`.

LCZero Development Community. Leela Chess Zero. `lczero.org`, 2018. Accessed: 20 November 2019.

M. L. Leavitt and A. Morcos. Towards falsifiable interpretability research. *arXiv preprint arXiv:2010.12016*, 2020.

Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

A.-M. Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes*. Paris: F. Didot, 1805.

M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

B. Letham, C. Rudin, T. H. McCormick, and D. Madigan. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, 2015.

K. Li, A. K. Hopkins, D. Bau, F. Viégas, H. Pfister, and M. Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task, 2023.

LiChess. Encyclopedia of chess openings, 2023. URL `https://github.com/lichess-org/chess-openings`. [Online; accessed 1-May-2023].

T. Lieberum, S. Rajamanoharan, A. Conmy, L. Smith, N. Sonnerat, V. Varma, J. Kramár, A. Dragan, R. Shah, and N. Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2, 2024. URL `https://arxiv.org/abs/2408.05147`.

T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

X. V. Lin, T. Mihaylov, M. Artetxe, T. Wang, S. Chen, D. Simig, M. Ott, N. Goyal, S. Bhosale, J. Du, R. Pasunuru, S. Shleifer, P. S. Koura, V. Chaudhary, B. O'Horo, J. Wang, L. Zettlemoyer, Z. Kozareva, M. Diab, V. Stoyanov, and X. Li. Few-shot learning with multilingual language models, 2022. URL `https://arxiv.org/abs/2112.10668`.

Z. C. Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

C. C. Liu, F. Koto, T. Baldwin, and I. Gurevych. Are multilingual llms culturally-diverse reasoners? an investigation into multicultural proverbs and sayings, 2024. URL `https://arxiv.org/abs/2309.08591`.

G. Liu, O. Schulte, W. Zhu, and Q. Li. Toward interpretable deep reinforcement learning with linear model u-trees. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part II 18*, pages 414–429. Springer, 2019.

J. Liu, Z. Lin, S. Padhy, D. Tran, T. Bedrax Weiss, and B. Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7498–7512. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/543e83748234f7cbab21aa0ade66565f-Paper.pdf`.

A. Lucieri, M. N. Bajwa, S. A. Braun, M. I. Malik, A. Dengel, and S. Ahmed. On interpretability of deep learning based skin lesion classifiers using concept activation vectors. In *IJCNN*, 2020.

S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

S. M. Lundberg, B. Nair, M. Vavilala, M. Horibe, M. J. Eisses, T. Adams, D. E. Liston, D. K. Low, S.-F. Newman, J. Kim, and et al. Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. *Nature Biomedical Engineering*, 2(10):749–760, 2018. doi: 10.1038/s41551-018-0304-0.

P. Madumal, T. Miller, L. Sonenberg, and F. Vetere. Distal explanations for model-free explainable reinforcement learning, 2020.

A. Mahinpei, J. Clark, I. Lage, F. Doshi-Velez, and W. Pan. Promises and pitfalls of black-box concept learning models. *CoRR*, abs/2106.13314, 2021. URL `https://arxiv.org/abs/2106.13314`.

E. Malach, G. Yehudai, S. Shalev-Shwartz, and O. Shamir. Proving the lottery ticket hypothesis: Pruning is all you need, 2020. URL `https://arxiv.org/abs/2002.00585`.

K. Marchisio, W.-Y. Ko, A. Bérard, T. Dehaze, and S. Ruder. Understanding and mitigating language confusion in llms, 2024. URL `https://arxiv.org/abs/2406.20052`.

E. Margolis and S. Laurence. Concepts. In E. N. Zalta and U. Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2023 edition, 2023.

P. H. Martins, P. Fernandes, J. Alves, N. M. Guerreiro, R. Rei, D. M. Alves, J. Pombal, A. Farajian, M. Faysse, M. Klimaszewski, P. Colombo, B. Haddow, J. G. C. de Souza, A. Birch, and A. F. T. Martins. Eurollm: Multilingual language models for europe, 2024. URL `https://arxiv.org/abs/2409.16235`.

T. McGrath, A. Kapishnikov, N. Tomašev, A. Pearce, M. Wattenberg, D. Hassabis, B. Kim, U. Paquet, and V. Kramnik. Acquisition of chess knowledge in alphazero. *Proceedings of the National Academy of Sciences*, 119(47):e2206625119, 2022a.

T. McGrath, A. Kapishnikov, N. Tomašev, A. Pearce, M. Wattenberg, D. Hassabis, B. Kim, U. Paquet, and V. Kramnik. Acquisition of chess knowledge in alphazero. *Proceedings of the National Academy of Sciences*, 119(47), Nov. 2022b. ISSN 1091-6490. doi: 10.1073/pnas.2206625119.

T. McGrath, M. Rahtz, J. Kramar, V. Mikulik, and S. Legg. The hydra effect: Emergent self-repair in language model computations, 2023. URL `https://arxiv.org/abs/2307.15771`.

N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, 34(3), jun 2006. doi: 10.1214/009053606000000281. URL `https://doi.org/10.1214%2F009053606000000281`.

K. Meng, D. Bau, A. Andonian, and Y. Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35: 17359–17372, 2022.

G. A. Miller. WordNet: A lexical database for English. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994. URL `https://aclanthology.org/H94-1111/`.

T. Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

D. Mincu, E. Loreaux, S. Hou, S. Baur, I. Protsyuk, M. Seneviratne, A. Mottram, N. Tomasev, A. Karthikesalingam, and J. Schrouff. Concept-based model explanations for electronic health records. In *Proceedings of the Conference on Health, Inference, and Learning*, pages 36–46, 2021.

C. Molnar. *Interpretable Machine Learning.* 3 edition, 2020. ISBN 978-3-911578-03-5. URL `https://christophm.github.io/interpretable-ml-book`.

C. Molnar, G. Casalicchio, and B. Bischl. Quantifying model complexity via functional decomposition for better post-hoc interpretability. In *Machine Learning and Knowledge Discovery in Databases: International Workshops of ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I*, pages 193–204. Springer, 2020.

G. Montavon, W. Samek, and K.-R. Müller. Methods for interpreting and understanding deep neural networks. *Digital signal processing*, 73:1–15, 2018.

A. Mordvintsev, C. Olah, and M. Tyka. Inceptionism: Going deeper into neural networks. *Google Research Blog*, 2015. Retrieved November 2017.

B. Muller, A. Anastasopoulos, B. Sagot, and D. Seddah. When being unseen from mBERT is just the beginning: Handling new languages with multilingual language models. In K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 448–462, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.38. URL `https://aclanthology.org/2021.naacl-main.38/`.

T. N. Mundhenk, B. Y. Chen, and G. Friedland. Efficient saliency maps for explainable ai, 2020.

W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, 2019.

S. G. Müller and F. Hutter. Trivialaugment: Tuning-free yet state-of-the-art data augmentation. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 754–762. ICCV, 2021.

N. Nanda. Actually, othello-gpt has a linear emergent world model, Mar 2023. URL `<https://neelnanda.io/mechanistic-interpretability/othello>`.

N. Nanda, L. Chan, T. Lieberum, J. Smith, and J. Steinhardt. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023a.

N. Nanda, S. Rajamanoharan, J. Kramar, and R. Shah. Fact finding: Attempting to reverse-engineer factual recall on the neuron level, Dec 2023b. URL `https://www.alignmentforum.org/s/hpWHhjvjn67LJ4xXX`.

T. Naous, M. J. Ryan, A. Ritter, and W. Xu. Having beer after prayer? measuring cultural bias in large language models, 2024. URL `https://arxiv.org/abs/2305.14456`.

M. Narayanan, E. Chen, J. He, B. Kim, S. Gershman, and F. Doshi-Velez. How do humans understand explanations from machine learning systems? an evaluation of the human-interpretability of explanation. *arXiv preprint arXiv:1802.00682*, 2018.

A. Nicolson, L. Schut, J. A. Noble, and Y. Gal. Explaining explainability: Recommendations for effective use of concept activation vectors. *Transactions on Machine Learning Research*, 2025. Accepted February 2025.

nostalgebraist. interpreting gpt: the logit lens, Aug 2020. URL https://www.alignmentforum.org/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

C. Olah, A. Mordvintsev, and L. Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.

C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev. Building blocks of interpretability. *Distill*, 2018. URL https://distill.pub/2018/building-blocks/.

C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, and S. Carter. Zoom in: An introduction to circuits. *Distill*, 2020. URL https://distill.pub/2020/circuits/zoom-in/.

B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):3311–3325, 1997.

M. L. Olson, L. Neal, F. Li, and W.-K. Wong. Counterfactual states for atari agents via generative deep learning, 2019.

S. Omidshafiei, A. Kapishnikov, Y. Assogba, L. Dixon, and B. Kim. Beyond rewards: a hierarchical perspective on offline multiagent behavioral analysis. *Advances in Neural Information Processing Systems*, 35:3444–3460, 2022.

OpenAI. Learning to Reason with LLMs. https://openai.com/index/learning-to-reason-with-llms/, 2024. [Online; accessed 17-March-2025].

OpenAI. Introducing Deep Research. https://openai.com/index/introducing-deep-research/, 2025. [Online; accessed 17-March-2025].

K. Pal, J. Sun, A. Yuan, B. Wallace, and D. Bau. Future lens: Anticipating subsequent tokens from a single hidden state. In *Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL)*, page 548–560. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.conll-1.37. URL http://dx.doi.org/10.18653/v1/2023.conll-1.37.

A. Pálsson and Y. Björnsson. Unveiling concepts learned by a world-class chess-playing agent. In *International Joint Conference on Artificial Intelligence*, 2023.

N. Panickssery, N. Gabrieli, J. Schulz, M. Tong, E. Hubinger, and A. M. Turner. Steering llama 2 via contrastive activation addition, 2024. URL https://arxiv.org/abs/2312.06681.

N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

M. Pawelczyk, K. Broelemann, and G. Kasneci. Learning model-agnostic counterfactual explanations for tabular data. In *Proceedings of the web conference 2020*, pages 3126–3132, 2020.

M. Pawelczyk, C. Agarwal, S. Joshi, S. Upadhyay, and H. Lakkaraju. Exploring counterfactual explanations through the lens of adversarial examples: A theoretical and empirical analysis, 2021. URL https://arxiv.org/abs/2106.09992.

J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, Cambridge, UK, 2nd edition, 2009. ISBN 9780521895606.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237, 2018.

J. C. Peterson, D. D. Bourgin, M. Agrawal, D. Reichman, and T. L. Griffiths. Using large-scale experiments and machine learning to discover theories of human decision-making. *Science*, 372(6547):1209–1214, 2021.

A. Petrov, E. La Malfa, P. Torr, and A. Bibi. Language model tokenizers introduce unfairness between languages. *Advances in Neural Information Processing Systems*, 36, 2024.

J. Pfau, A. T. Young, J. Wei, M. L. Wei, and M. J. Keiser. Robust Semantic Interpretability: Revisiting Concept Activation Vectors. In *ICML WHI*, 2020.

L. Phan, A. Gatti, Z. Han, N. Li, J. Hu, H. Zhang, S. Shi, M. Choi, A. Agrawal, A. Chopra, et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025.

F. Poursabzi-Sangdeh, D. G. Goldstein, J. M. Hofman, J. W. Wortman Vaughan, and H. Wallach. Manipulating and measuring model interpretability. In *Proceedings of the 2021 CHI conference on human factors in computing systems*, pages 1–52, 2021.

J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

N. Raman, M. E. Zarlenga, J. Heo, and M. Jamnik. Do concept bottleneck models obey locality? In *NeurIPS Workshop XAI in Action: Past, Present, and Future Applications*, 2023.

N. J. Raman, M. E. Zarlenga, and M. Jamnik. Understanding inter-concept relationships in concept-based models. In *ICML*, 2024.

V. Ramanujan, M. Wortsman, A. Kembhavi, A. Farhadi, and M. Rastegari. What's hidden in a randomly weighted neural network?, 2020. URL `https://arxiv.org/abs/1911.13299`.

V. V. Ramaswamy, S. S. Y. Kim, R. C. Fong, and O. Russakovsky. Overlooked factors in concept-based explanations: Dataset choice, concept learnability, and human capability. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10932–10941, 2022a.

V. V. Ramaswamy, S. S. Y. Kim, N. Meister, R. Fong, and O. Russakovsky. Elude: Generating interpretable explanations via a decomposition into labelled and unlabelled features, 2022b.

V. V. Ramaswamy, S. S. Kim, R. Fong, and O. Russakovsky. Ufo: A unified method for controlling understandability and faithfulness objectives in concept-based explanations for cnns. *arXiv preprint arXiv:2303.15632*, 2023.

K. Rawal and H. Lakkaraju. Beyond individualized recourse: Interpretable and interactive summaries of actionable recourses. *Advances in Neural Information Processing Systems*, 33:12187–12198, 2020.

M. T. Ribeiro and S. Lundberg. Adaptive testing and debugging of nlp models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3253–3267, 2022.

M. T. Ribeiro, S. Singh, and C. Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

P. J. Robinson. Fitting equations to data: Computer analysis of multifactor data for scientists and engineers. *JMR, Journal of Marketing Research (pre-1986)*, 11 (000003):346, 1974.

A. M. Roth, N. Topin, P. Jamshidi, and M. Veloso. Conservative q-improvement: Reinforcement learning for an interpretable decision-tree policy, 2019.

C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1 (5):206–215, 2019.

C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, and C. Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys*, 16:1–85, 2022.

C. Rupprecht, C. Ibrahim, and C. J. Pal. Finding and visualizing weaknesses of deep reinforcement learning agents, 2019.

P. Rust, J. Pfeiffer, I. Vulić, S. Ruder, and I. Gurevych. How good is your tokenizer? on the monolingual performance of multilingual language models. In C. Zong, F. Xia, W. Li, and R. Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3118–3135, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.243. URL `https://aclanthology.org/2021.acl-long.243/`.

S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet. Adversarial manipulation of deep representations. *arXiv preprint arXiv:1511.05122*, 2015.

M. Sadler and N. Regan. *Game Changer: AlphaZero's Groundbreaking Chess Strategies and the Promise of AI.* New In Chess, 2019.

S. Sagawa, P. W. Koh, T. B. Hashimoto, and P. Liang. Distributionally Robust Neural Networks for Group Shifts: On the Importance of Regularization for Worst-Case Generalization. 2020.

W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K.-R. Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673, 2016.

G. Sartor and F. Lagioia. The impact of the General Data Protection Regulation (GDPR) on artificial intelligence. *European Parliamentary Research Service*, pages 1–100, 2020.

J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver. Mastering Atari, Go, chess and shogi by planning with a learned model, 2019.

J. Schrouff, S. Baur, S. Hou, D. Mincu, E. Loreaux, R. Blanes, J. Wexler, A. Karthikesalingam, and B. Kim. Best of both worlds: local and global explanations with human-understandable concepts. *ArXiv*, abs/2106.08641, 2021.

L. Schut, O. Key, R. Mc Grath, L. Costabello, B. Sacaleanu, M. Corcoran, and Y. Gal. Generating interpretable counterfactual explanations by implicit minimisation of epistemic and aleatoric uncertainties. In A. Banerjee and K. Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 1756–1764. PMLR, 13–15 Apr 2021.

L. Schut, Y. Gal, and S. Farquhar. Do multilingual llms think in english? *Building Trust Workshop at the Thirteenth International Conference on Learning Representations*, 2025a.

L. Schut, E. M. Russek, I. Kuperwajs, M. G. Mattar, W. J. Ma, and T. L. Griffiths. Learning in online chess increases with time spent thinking and diversity of experience. Ongoing work, 2025b.

L. Schut, N. Tomašev, T. McGrath, D. Hassabis, U. Paquet, and B. Kim. Bridging the Human–AI Knowledge Gap through Concept Discovery and Transfer in AlphaZero. *Proceedings of the National Academy of Sciences*, 122(13):e2406675122, 2025c. doi: 10.1073/pnas.2406675122.

R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, Oct 2019. ISSN 1573-1405. doi: 10.1007/s11263-019-01228-7. URL `http://dx.doi.org/10.1007/s11263-019-01228-7`.

S. Shafayat, E. Kim, J. Oh, and A. Oh. Multi-fact: Assessing factuality of multilingual llms using factscore, 2024. URL `https://arxiv.org/abs/2402.18045`.

R. Shah, V. Varma, R. Kumar, M. Phuong, V. Krakovna, J. Uesato, and Z. Kenton. Goal misgeneralization: Why correct specifications aren't enough for correct goals, 2022. URL `https://arxiv.org/abs/2210.01790`.

M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1528–1540. ACM, 2016.

L. Sharkey, B. Chughtai, J. Batson, J. Lindsey, J. Wu, L. Bushnaq, N. Goldowsky-Dill, S. Heimersheim, A. Ortega, J. Bloom, et al. Open problems in mechanistic interpretability. *arXiv preprint arXiv:2501.16496*, 2025.

M. Sharma, M. Tong, T. Korbak, D. Duvenaud, A. Askell, S. R. Bowman, N. Cheng, E. Durmus, Z. Hatfield-Dodds, S. R. Johnston, S. Kravec, T. Maxwell, S. McCandlish, K. Ndousse, O. Rausch, N. Schiefer, D. Yan, M. Zhang, and

E. Perez. Towards understanding sycophancy in language models, 2023. URL https://arxiv.org/abs/2310.13548.

F. Shi, M. Suzgun, M. Freitag, X. Wang, S. Srivats, S. Vosoughi, H. W. Chung, Y. Tay, S. Ruder, D. Zhou, et al. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*, 2022.

R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *IEEE Symposium on Security and Privacy*, pages 3–18, 2017.

D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

D. Slack, S. A. Friedler, C. Scheidegger, and C. D. Roy. Assessing the local interpretability of machine learning models. *arXiv preprint arXiv:1902.03501*, 2019.

D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186, 2020.

D. Slack, A. Hilgard, S. Singh, and H. Lakkaraju. Reliable post hoc explanations: Modeling uncertainty in explainability. *Advances in neural information processing systems*, 34:9391–9404, 2021.

D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

L. Smith. Are capsules a good idea? a generative perspective. https://oatml.cs.ox.ac.uk/blog/2020/07/10/are-capsules-a-good-idea-a-generative-perspective.html, 2020. Accessed: 4 November.

L. Smith and Y. Gal. Understanding measures of uncertainty for adversarial example detection. In A. Globerson and R. Silva, editors, *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pages 560–569. AUAI Press, 2018. URL http://auai.org/uai2018/proceedings/papers/207.pdf.

L. Smith, L. Schut, Y. Gal, and M. van der Wilk. Capsule networks – a probabilistic perspective, 2021. URL https://arxiv.org/abs/2004.03553.

R. D. Snee. Some aspects of nonorthogonal data analysis: Part i. developing prediction equations. *Journal of Quality Technology*, 5(2):67–79, 1973.

R. D. Snee and D. W. Marquardt. Comment: Collinearity diagnostics depend on the domain of prediction, the model, and the data. *The American Statistician*, 38(2):83–87, 1984.

C. Song, T. Ristenpart, and V. Shmatikov. Machine learning models that remember too much. In *ACM Conference on Computer and Communications Security (CCS)*, pages 587–601, 2017.

R. Soni, N. Shah, C. T. Seng, and J. D. Moore. Adversarial tcav–robust and effective interpretation of intermediate layers in neural networks. *arXiv preprint arXiv:2002.03549*, 2020.

S. Sonoda and N. Murata. Neural Network with Unbounded Activation Functions is Universal Approximator. *Applied and Computational Harmonic Analysis*, 43 (2):233–268, 2017. ISSN 10635203. doi: 10.1016/j.acha.2015.12.005.

A. Spangher, B. Ustun, and Y. Liu. Actionable recourse in linear classification. In *Proceedings of the 5th Workshop on Fairness, Accountability and Transparency in Machine Learning*, 2018.

C. Sprague, E. B. Wendoloski, and I. Guch. Interpretable AI for deep learning-based meteorological applications. In *American Meteorological Society Annual Meeting*, 2019.

S. Sreedharan, U. Soni, M. Verma, S. Srivastava, and S. Kambhampati. Bridging the gap: Providing post-hoc symbolic explanations for sequential decision-making problems with black box simulators. *arXiv preprint arXiv:2002.01080*, 2020.

H. Steck, C. Ekanadham, and N. Kallus. Is cosine-similarity of embeddings really about similarity? In *Companion Proceedings of the ACM on Web Conference 2024*, pages 887–890, 2024.

Stockfish Community. Stockfish Chess. `https://stockfishchess.org/`, 2018. Accessed: 23 July 2023.

N. Subramani, N. Suresh, and M. E. Peters. Extracting latent steering vectors from pretrained language models. *arXiv preprint arXiv:2205.05124*, 2022.

M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, 2017.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

D. Tan, D. Chanin, A. Lynch, D. Kanoulas, B. Paige, A. Garriga-Alonso, and R. Kirk. Analyzing the generalization and reliability of steering vectors, 2024. URL `https://arxiv.org/abs/2407.12404`.

G. Team. Gemma. 2024. doi: 10.34740/KAGGLE/M/3301. URL `https://www.kaggle.com/m/3301`.

A. Templeton, T. Conerly, J. Marcus, J. Lindsey, T. Bricken, B. Chen, A. Pearce, C. Citro, E. Ameisen, A. Jones, H. Cunningham, N. L. Turner, C. McDougall, M. MacDiarmid, C. D. Freeman, T. R. Sumers, E. Rees, J. Batson, A. Jermyn, S. Carter, C. Olah, and T. Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL `https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html`.

I. Tenney, D. Das, and E. Pavlick. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*, 2019.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

R. Tomsett, D. Harborne, S. Chakraborty, P. Gurram, and A. Preece. Sanity checks for saliency metrics. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 6021–6029, 2020.

D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018. URL https://openreview.net/forum?id=SyxAb30cY7.

A. M. Turner, L. Thiergart, G. Leech, D. Udell, J. J. Vazquez, U. Mini, and M. MacDiarmid. Activation addition: Steering language models without optimization. *arXiv preprint arXiv:2308.10248*, 2023.

J. Van Amersfoort, L. Smith, Y. W. Teh, and Y. Gal. Uncertainty estimation using a single deep deterministic neural network. In *International Conference on Machine Learning*, pages 9690–9700. PMLR, 2020.

A. Van Looveren and J. Klaise. Interpretable counterfactual explanations guided by prototypes. *arXiv preprint arXiv:1907.02584*, 2019.

A. Van Looveren and J. Klaise. Interpretable counterfactual explanations guided by prototypes. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 650–665. Springer, 2021.

M. Vasić, A. Petrović, K. Wang, M. Nikolić, R. Singh, and S. Khurshid. MoËT: Mixture of expert trees and its application to verifiable reinforcement learning. *Neural Networks*, 151:34–47, jul 2022. doi: 10.1016/j.neunet.2022.03.022. URL https://doi.org/10.1016%2Fj.neunet.2022.03.022.

J. Vig, S. Gehrmann, Y. Belinkov, S. Qian, D. Nevo, Y. Singer, and S. Shieber. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33:12388–12401, 2020.

G. A. Vouros. Explainable deep reinforcement learning: state of the art and challenges. *ACM Computing Surveys*, 55(5):1–39, 2022.

S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31: 841, 2017.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. *Computation Neural Systems Technical Report*, 2011.

Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

L. Weidinger, M. Rauh, N. Marchal, A. Manzini, L. A. Hendricks, J. Mateos-Garcia, S. Bergman, J. Kay, C. Griffin, B. Bariach, I. Gabriel, V. Rieser, and W. Isaac. Sociotechnical safety evaluation of generative ai systems, 2023. URL https://arxiv.org/abs/2310.11986.

C. Wendler, V. Veselovsky, G. Monea, and R. West. Do llamas work in english? on the latent language of multilingual transformers, 2024. URL https://arxiv.org/abs/2402.10588.

S. Wiegreffe and Y. Pinter. Attention is not not explanation. *arXiv preprint arXiv:1908.04626*, 2019.

Wikipedia contributors. Chess strategy, 2023a. URL https://en.wikipedia.org/wiki/Chess_strategy. [Online; accessed 13-October-2023].

Wikipedia contributors. Chess tactic, 2023b. URL https://en.wikipedia.org/wiki/Chess_tactic. [Online; accessed 13-October-2023].

Wikipedia contributors. Chess rating system, 2023c. URL https://en.wikipedia.org/wiki/Chess_rating_system. [Online; accessed 23-July-2023].

Z. Wu, X. V. Yu, D. Yogatama, J. Lu, and Y. Kim. The semantic hub hypothesis: Language models share semantic representations across languages and modalities. *arXiv preprint arXiv:2411.04986*, 2024.

Y. Xu, S. Zhao, J. Song, R. Stewart, and S. Ermon. A theory of usable information under computational constraints, 2020.

L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel. mT5: A massively multilingual pre-trained text-to-text transformer. In K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.41. URL https://aclanthology.org/2021.naacl-main.41/.

S. Yan, Z. Yu, X. Zhang, D. Mahapatra, S. S. Chandra, M. Janda, P. Soyer, and Z. Ge. Towards trustable skin cancer diagnosis via rewriting model's decision. In *CVPR*, 2023.

H. Yang, C. Rudin, and M. Seltzer. Scalable bayesian rule lists. In *International conference on machine learning*, pages 3921–3930. PMLR, 2017.

M. Yang and B. Kim. Benchmarking attribution methods with relative feature importance. *arXiv preprint arXiv:1907.09701*, 2019.

S. Yang, E. Gribovskaya, N. Kassner, M. Geva, and S. Riedel. Do large language models latently perform multi-hop reasoning?, 2024. URL https://arxiv.org/abs/2402.16837.

H. Yeche, J. Harrison, and T. Berthier. Ubs: A dimension-agnostic metric for concept vector interpretability applied to radiomics. In *Interpretability of Machine Intelligence in Medical Image Computing and Multimodal Learning for Clinical Decision Support*, pages 12–20. Springer, 2019.

C.-K. Yeh, B. Kim, S. Arik, C.-L. Li, T. Pfister, and P. Ravikumar. On completeness-aware concept-based explanations in deep neural networks. *Advances in Neural Information Processing Systems*, 33:20554–20565, 2020.

M. Yuksekgonul, M. Wang, and J. Zou. Post-hoc concept bottleneck models. In *ICLR*, 2023.

S. Yun, D. Han, S. Chun, S. Oh, Y. Yoo, and J. Choe. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6022–6031, 2019.

M. B. Zafar, M. Donini, D. Slack, C. Archambeau, S. Das, and K. Kenthapadi. On the lack of robust interpretability of neural text classifiers. *arXiv preprint arXiv:2106.04631*, 2021.

T. Zahavy, N. Ben-Zrihem, and S. Mannor. Graying the black box: Understanding dqns. In *International conference on machine learning*, pages 1899–1908. PMLR, 2016.

M. E. Zarlenga, P. Barbiero, Z. Shams, D. Kazhdan, U. Bhatt, A. Weller, and M. Jamnik. Towards Robust Metrics for Concept Representation Evaluation. *AAAI*, 2023. doi: 10.48550/arXiv.2301.10367.

R. Zemel, Y. Wu, K. Swersky, T. Pitassi, and C. Dwork. Learning fair representations. In *International Conference on Machine Learning*, pages 325–333, 2013.

H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

H. Zhang, H. Zhang, and C.-J. Hsieh. Universal approximation in dropout neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 11034–11043. PMLR, 2019. URL `https://proceedings.mlr.press/v97/zhang19d.html`.

L. Zhang, V. Veselovsky, R. T. McCoy, and T. L. Griffiths. Identifying and mitigating the influence of the prior distribution in large language models, 2025. URL `https://arxiv.org/abs/2504.12585`.

R. Zhang, P. Madumal, T. Miller, K. A. Ehinger, and B. I. P. Rubinstein. Invertible concept-based explanations for cnn models with non-negative concept activation vectors. In *AAAI Conference on Artificial Intelligence*, 2020.

Y. Zhao, W. Zhang, G. Chen, K. Kawaguchi, and L. Bing. How do large language models handle multilingualism?, 2024. URL `https://arxiv.org/abs/2402.18815`.

C. Zhong, F. Cheng, Q. Liu, J. Jiang, Z. Wan, C. Chu, Y. Murawaki, and S. Kurohashi. Beyond english-centric llms: What language do multilingual language models think in? *arXiv preprint arXiv:2408.10811*, 2024.

B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million Image Database for Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

B. Zhou, Y. Sun, D. Bau, and A. Torralba. Interpretable basis decomposition for visual explanation. In *Computer Vision – ECCV 2018*, pages 122–138. Springer International Publishing, 2018.

W. Zhu, Y. Lv, Q. Dong, F. Yuan, J. Xu, S. Huang, L. Kong, J. Chen, and L. Li. Extrapolating large language models to non-english by aligning languages, 2023. URL `https://arxiv.org/abs/2308.04948`.

Appendices

A
Further Material

A.1 Concept Encoding is a Weak Requirement

A large part of interpretability research focuses on identifying *what* is encoded inside neural networks. However, we argue that this is often not sufficient.[1] In many settings, we are not only interested in whether a concept is encoded, but whether it is used by the model. In this section, we show that concepts can be encoded by chance.

This result is not intended as a novel contribution. To the best of our knowledge, it has not been explicitly stated in the interpretability literature, but it follows directly from established results.

A.1.1 Theory

Assume we have an untrained neural network f with parameters θ_{init} sampled from a standard initialisation scheme, such as Kaiming initialisation [He et al., 2015]. Let $f_l(x)$ denote the intermediate representation at layer l for input x.

Further, let $c(x)$ be a function that indicates whether a concept c is present in x,

$$c(x) = \begin{cases} \geq k & \text{if the concept is present in } x \\ \leq k & \text{otherwise,} \end{cases} \tag{A.1}$$

where k is an arbitrary value.

Our goal is to show that, under some mild assumptions, for any concept c, there exists a trainable decoder ϕ such that $|\phi(f_l(x)) - c(x)| \leq \varepsilon$.

The universal approximation theorem states that any continuous function on a compact domain can be approximated arbitrarily well by a neural network with at least two layers, under some conditions [Cybenko, 1989, Hornik et al., 1989]. Zhang et al. [2019] extended this result for the finite-data regime:

Theorem A.1.1. There exists a two-layer neural network with ReLU activations and $2n + d$ weights that can represent any function on a sample of size n in d dimensions.

Therefore, if we assume $c(x)$ is a continuous function, there must exist a two-layer neural network with ReLU activations ϕ that can approximate this function. An implication for interpretability is that a model $f(x)$ can learn *any* $c(x)$ – i.e., it can extract any concept from an input x, under the assumptions given above.

We can make this claim even stronger by invoking the lottery ticket hypothesis (LTH) [Frankle and Carbin, 2019, Ramanujan et al., 2020, Malach et al., 2020]. The LTH states that:

[1]There are exceptions – particularly in applications where we care about whether a feature is present, regardless of whether it is used.

For any such target network h^*, there exists a large enough randomly initialized network f, and a binary mask $m \in \{0,1\}^{|\theta|}$, such that:

$$f_m(x) := f_{m \odot \theta}(x) \approx h^*(x)$$

Combining this with the universal approximation theorem, it follows that:

- An **untrained** network f contains a subnetwork that extract any concept function $c(x)$.
- there exists a decoder ϕ that can learn to decode this masked subnetwork's activations.

More formally, there exists a neural network decoder ϕ such that:

$$|\phi(f_\ell(x)) - c(x)| < \varepsilon, \quad \forall x \in \mathbb{X}$$

This implies that, given a sufficiently expressive decoder, any concept can, in principle, be decoded from an untrained network. Empirical work has also found evidence that concepts can indeed be decoded from untrained models [Adebayo et al., 2018, Heap et al., 2025].

A.1.2 Implications

These results suggest that models may encode concepts by chance at initialisation, which can be detected by expressive decoders. The ability to decode a concept from a representation does not imply that the model has learned or uses the concept meaningfully.

From a research perspective, this implies that we should do more than identify that concepts are encoded in a network. For example, it places importance on showing that the model uses the concept during inference. Alternatively, perhaps rather than focusing on finding concepts, we should think about how easy it is to extract or use concepts. For example, we could place restrictions on the expressivity of the decoder ϕ.

More generally, an important implication of this result is that we need to think about whether behaviour is learned or unlearned during training. Are we actively encoding features or removing subnetworks for irrelevant ones? This distinction is important as it would imply that mechanisms –including potentially unsafe behaviour– may already be present at initialisation rather than being learned during training. This suggests that alignment failures or shortcut behaviours could, in some cases, be rooted in the model's inductive biases.

B

Appendix: Examining Assumptions of Concept Vectors

B.1 Derivations for Activation Functions

Let $\hat{z}_{l_1}$ and $\hat{z}_{l_2}$ be linearly perturbed activations in each of these layers:

$$\hat{z}_{l_1} = z_{l_1,i} + v_{c,l_1} \tag{B.1}$$

$$\hat{z}_{l_2} = z_{l_2,i} + v_{c,l_2} = g_{l_1,l_2}(z_{l_1,i}) + v_{c,l_2} \tag{B.2}$$

where $u \in \mathbb{R}^{m_{l_1}}$ and $v \in \mathbb{R}^{m_{l_2}}$ are vectors of non-zero norm. The non-zero norm follows from that CAVs are directions in activation space and m_{l_1} and m_{l_2} are the dimensions of layer l_1 and l_2, respectively.

For the two perturbations to have the same effect on the activations (and hence the model) it must hold that:

$$\begin{aligned} g_{l_1,l_2}(\hat{z}_{l_1}) &= \hat{z}_{l_2} \\ g_{l_1,l_2}(z_{l_1,i} + v_{c,l_1}) &= g_{l_1,l_2}(z_{l_1,i}) + v \end{aligned} \tag{B.3}$$

For u and v to be consistent for all activations, they must be constant with respect to $z_{l_1,i}$, i.e. we can assume that u and v are not functions of $z_{l_1,i}$. If we rearrange Eq. B.3 to obtain an equation for v, we obtain

$$v = g_{l_1,l_2}(z + v_{c,l_1}) - g_{l_1,l_2}(z). \tag{B.4}$$

where we have simplified the notation by writing $z_{l_1,i}$ as z. Let us differentiate Eq. B.4

$$\frac{d}{dz}v = \frac{d}{dz}(g_{l_1,l_2}(z + v_{c,l_1}) - g_{l_1,l_2}(z)) \tag{B.5}$$

$$\mathbf{0} = \frac{d}{dz}g_{l_1,l_2}(z + v_{c,l_1}) - \frac{d}{dz}g_{l_1,l_2}(z) \tag{B.6}$$

$$\mathbf{0} = g'_{l_1,l_2}(z + v_{c,l_1}) - g'_{l_1,l_2}(z) \tag{B.7}$$

$$g'_{l_1,l_2}(z + v_{c,l_1}) = g'_{l_1,l_2}(z) \tag{B.8}$$

which implies that the derivative of g_{l_1,l_2}, g'_{l_1,l_2}, is periodic with period u. This is a strong restriction on the form that g'_{l_1,l_2} can take. Let's integrate to find out what implications it has on the form of f. First, let's split the periodic function g'_{l_1,l_2} into its mean value and its oscillatory part:

$$g'_{l_1,l_2}(z) = w'(z) + M \tag{B.9}$$

where $M \in \mathbb{R}^{m_{l_2} \times m_{l_1}}$ is the mean value across one interval for each component of $g'_{l_1,l_2}(z)$ and $w'(z)$ is a periodic function with zero integral across a single period, i.e.

$$\int_{\mathbf{0}}^{u} w'(z)dz = \mathbf{0}. \tag{B.10}$$

If we integrate $g'(z)$ from z to $z + v_{c,l_1}$ (using a change of variables with $t \in \mathbb{R}^{m_{l_1}}$) we find

$$\int_{z}^{z+v_{c,l_1}} g'(t)dt = g(z + v_{c,l_1}) - g(z) \tag{B.11}$$

and by using Eq. B.10 we find

$$g(z + v_{c,l_1}) - g(z) = 0. \tag{B.12}$$

Therefore g is periodic with period u. If we now take the integral of $g'_{l_1,l_2}(z)$, we find

$$g_{l_1,l_2}(z) = \int g'_{l_1,l_2}(z)dz \tag{B.13}$$

$$g_{l_1,l_2}(z) = \int w'(z)dz + \int Mdz \tag{B.14}$$

$$g_{l_1,l_2}(z) = w(z) + Mz + b. \tag{B.15}$$

Hence, g'_{l_1,l_2} satisfies Eq. B.5 (and therefore Eq. B.4) if and only if it is composed of a periodic function with period u and a linear term. However, there are further restrictions upon f. Let $M = \mathbf{0}$ so that f is simply a periodic function. In this case

$$v = g_{l_1,l_2}(z + v_{c,l_1}) - g_{l_1,l_2}(z) \tag{B.16}$$

$$v = g_{l_1,l_2}(z) - g_{l_1,l_2}(z) \tag{B.17}$$

$$v = \mathbf{0} \tag{B.18}$$

which contradicts our non-zero assumption on the norm of v. Hence we can obtain layer consistent vectors v and u if and only if f is composed of a periodic function with period u and a non-zero linear term M. We provide no proof as to whether this form of f can occur in practice in a neural network, however our empirical results in the main paper and § B.4 suggest that it does not. Our proof holds generally, however, in the next three sections, we go into more detail for a linear function as it is a special case of Eq. B.13 and for the ReLU and sigmoid functions as they are common activation functions used in neural networks.

Special Case: Linear Function One counter-example that at first look seems to contradict Eq. B.13 is a linear function. If f is a linear function, i.e. it conserves vector addition, then Eq. B.4 trivially holds:

$$v = g_{l_1,l_2}(z + v_{c,l_1}) - g_{l_1,l_2}(z) \tag{B.19}$$

$$v = g_{l_1,l_2}(z) + g_{l_1,l_2}(u) - g_{l_1,l_2}(z) \tag{B.20}$$

$$v = g_{l_1,l_2}(u). \tag{B.21}$$

The general result (Eq. B.13) requires that f be a combination of a periodic function, $g(z)$, and a linear function, $Mz+b$, but we have just shown that $f = Mz+b$ would also hold. This is because a function that outputs some constant value is a special case of a periodic function, where there is no minimal period as all periods are valid. So, for the case where f is linear $g(z) = c$, where $c \in R^{m_{l_2}}$. Or for a more specific example, in the case of $f = Mz + b$, $c = \mathbf{0}$. Hence, the result in Eq. B.13 generally holds and, for the case where f is linear, the only layer consistent vector v in layer l_2 for some vector u in layer l_1 is that same vector projected into layer l_2 by f, i.e. $g_{l_1,l_2}(u)$.

ReLU Function In a neural network, f often involves a rectified linear unit (ReLU), so below we find v when f = ReLU. Let $z_{l_1,i,j}$, v_j and u_j refer to the individual elements of $z_{l_1,i}$, v and u, respectively. By the definition of a ReLU activation:

$$g_{l_1,l_2}(z_{l_1,i,j}) = \max(0, z_{l_1,i,j}) = \begin{cases} z_{l_1,i,j} & z_{l_1,i,j} > 0 \\ 0 & z_{l_1,i,j} \leq 0 \end{cases} \tag{B.22}$$

So, Eq. B.4 becomes:

$$v_j = \max(0, z_{l_1,i,j} + v_{c,l_1,j}) - \max(0, z_{l_1,i,j})$$
$$= \begin{cases} z_{l_1,i,j} + v_{c,l_1,j} - z_{l_1,i,j} & z_{l_1,i,j} + v_{c,l_1,j} > 0, z_{l_1,i,j} > 0 \\ z_{l_1,i,j} + v_{c,l_1,j} + 0 & z_{l_1,i,j} + v_{c,l_1,j} > 0, z_{l_1,i,j} \leq 0 \\ 0 - z_{l_1,i,j} & z_{l_1,i,j} + v_{c,l_1,j} \leq 0, z_{l_1,i,j} > 0 \\ 0 - 0 & z_{l_1,i,j} + v_{c,l_1,j} \leq 0, z_{l_1,i,j} \leq 0 \end{cases} \tag{B.23}$$
$$= \begin{cases} u_j & z_{l_1,i,j} + v_{c,l_1,j} > 0, z_{l_1,i,j} > 0 \\ z_{l_1,i,j} + v_{c,l_1,j} & z_{l_1,i,j} + v_{c,l_1,j} > 0, z_{l_1,i,j} \leq 0 \\ z_{l_1,i,j} & z_{l_1,i,j} + v_{c,l_1,j} \leq 0, z_{l_1,i,j} > 0 \\ 0 & z_{l_1,i,j} + v_{c,l_1,j} \leq 0, z_{l_1,i,j} \leq 0. \end{cases}$$

If $z_{l_1,i,j} + v_{c,l_1,j} > 0, z_{l_1,i,j} \leq 0$ or $z_{l_1,i,j} + v_{c,l_1,j} \leq 0, z_{l_1,i,j} > 0$ for any element j then there does not exist a v such that Eq. B.3 is true for all i, i.e., when either of these statements are true, you cannot have two vectors which have the same effect on the activations across layers for all possible inputs. And if we assume that the elements of z can take any value in practice then there exists no two layer consistent vectors across a ReLU function.

Sigmoid Function In this section, we consider the sigmoid activation: $g_{l_1,l_2}(x) = \frac{1}{1+\exp(-x)}$. For ease of notation, we drop i and j as they do not change, but z_{l_1} and z_{l_2} refer to $z_{l_1,i,j}$ and $z_{l_2,i,j}$, respectively. From Eq. B.3, the concept vectors are consistent iff

$$g_{l_1,l_2}(z_{l_1} + v_{c,l_1}) = g_{l_1,l_2}(z_{l_1}) + v_{c,l_2} \tag{B.24}$$
$$\frac{1}{1+\exp(-z_{l_1} - u)} = \frac{1}{1+\exp(-z_{l_1})} + v_{c,l_2} \tag{B.25}$$

Simplifying Eq. B.25 for v, we get:

$$v = \frac{1}{1+\exp(-z_{l_1} - u)} - \frac{1}{1+\exp(-z_{l_1})} \tag{B.26}$$
$$v = \frac{(1+\exp(-z_{l_1})) - (1+\exp(-z_{l_1} - u))}{(1+\exp(-z_{l_1}))(1+\exp(-z_{l_1} - u))} \tag{B.27}$$
$$v = \frac{\exp(-z_{l_1}) - \exp(-z_{l_1} - u)}{(1+\exp(-z_{l_1}))(1+\exp(-z_{l_1} - u))} \tag{B.28}$$
$$v = \frac{1 - \exp(-u)}{(\exp(z_{l_1}) + 1)(\exp(z_{l_1}) + \exp(-u))} \tag{B.29}$$

This can be simplified further with partial fractions:

$$v = \frac{1 - \exp(-u)}{(\exp(z_{l_1}) + 1)(\exp(z_{l_1}) + \exp(-u))} \tag{B.30}$$

$$= \frac{\exp(-z_{l_1})}{(\exp(z_{l_1}) + 1)} - \frac{\exp(-z_{l_1} - u)}{(\exp(z_{l_1}) + \exp(-u))} \tag{B.31}$$

$$= \frac{\exp(-z_{l_1})}{(\exp(z_{l_1}) + 1)} - \frac{\exp(-z_{l_1})}{(\exp(z_{l_1} + v_{c,l_1}) + \exp(-u))} \tag{B.32}$$

For a single v to exist which is consistent for all z_{l_1} it cannot depend on z_{l_1}. Since the left half of Eq. B.32 depends on z_{l_1}, the only way that v does not depend on z_{l_1} is if the right hand side cancels out the left. This only occurs when $u = v = \mathbf{0}$. Since u and v are directions in activation space, and hence have a non-zero norm, this is a contradiction. Therefore, for the sigmoid function, under no conditions does there exist layer consistent vectors.

B.2 Elements Dataset Versions

Standard The default version contains four objects in each image and the allowed concepts are the primary colours and their pairwise combinations (red, green, blue, yellow, cyan, magenta), five shapes (square, circle, triangle, plus, cross) and three textures (solid, spots and diagonal stripes).

Simple In some experiments, when stated, we use a simpler version which contains fewer shapes and colours. This is to reduce the complexity of the figures. The standard and simple dataset configurations are in Table B.1.

Spatially Dependent For some experiments we use a spatially dependent version of the standard Elements dataset. This has the same configuration but introduces spatially dependent classes. As in the standard dataset, there is a class for all combinations of two and three concepts, e.g., 'striped squares' or 'spotted cyan triangles'. However, there are additional classes which depend on where the element is present in the image. For all classes involving triangles, we add two new classes which depend on if the object is in the top or bottom half of the image. For example, the class 'blue triangles' will now have two additional classes related to it of 'blue triangles on the top' and 'blue triangles on the bottom'. Similarly, we introduce two new classes for all classes involving squares, but for the left/right halves of the image rather than the top/bottom.

Table B.1: Configurations for the standard and simple versions of the Elements dataset. Element size and image size are in pixels. Brightness is the value of the pixels in the element.

Property	Standard	Simple
Colours	red, green, blue, yellow, cyan, magenta	red, green, blue
Shapes	square, circle, triangle, plus, cross	square, circle, triangle, plus
Textures	spots, stripes, solid	spots, stripes, solid
Brightness	153–255	153–255
Element Size / pixels	48–80	48–80
No. Elements per image	4	4
Image Size / pixels	256	256

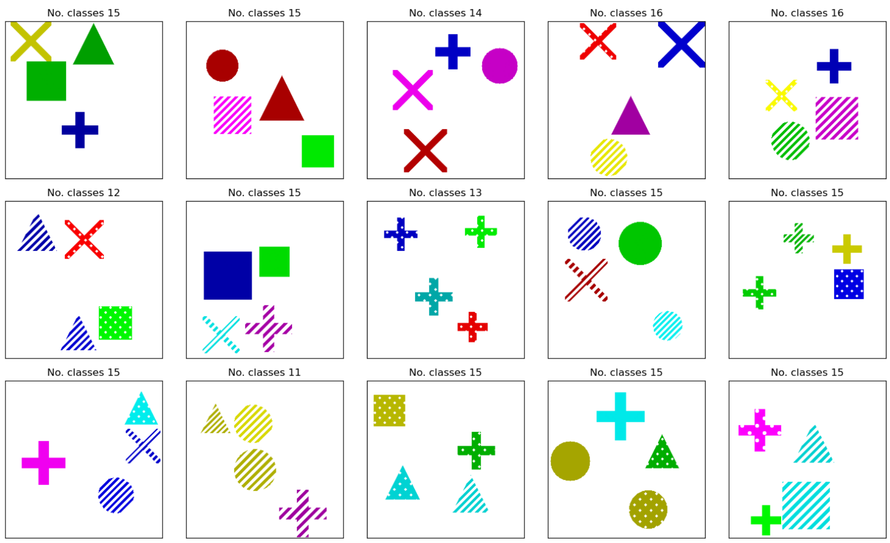


Figure B.1: Example images from the standard elements dataset. The number of classes each image belongs to is displayed above it.

Entangled We use two alternative versions of the simple dataset in the Entanglement experiments: E_2 and E_3. As described in the main text, these are identical to the simple dataset, apart from the association between the red and triangle concepts. In each case, we restrict some of the allowed combinations of concepts that an element can take. This also removes some classes from the dataset which we reflect in any trained models. E_2 does not allow any shape apart from triangles to be red. This removes classes like ‘red circles’ or ‘spotted red squares’. E_3 has this restriction and then places a further restriction that triangles have to red. This removes classes such as ‘blue triangles’ or ‘striped green triangles’.

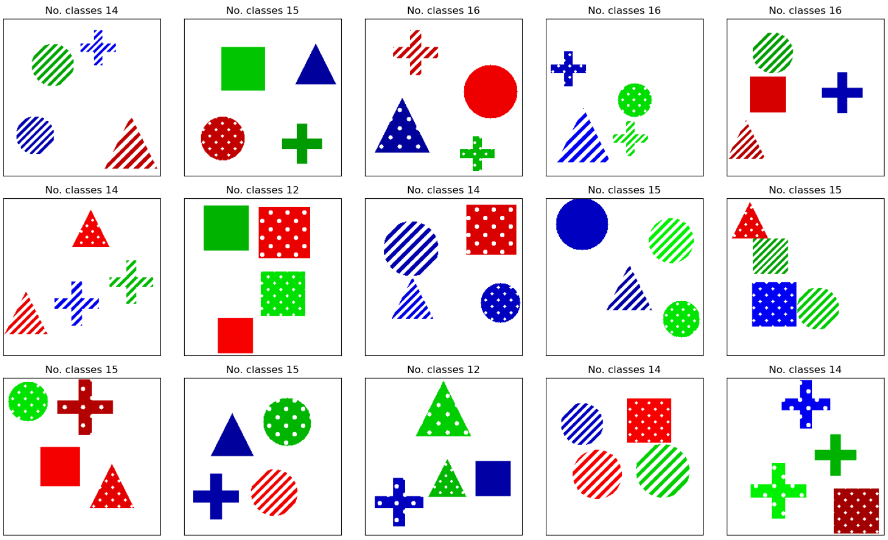


Figure B.2: Example images from the simple elements dataset. The number of classes each image belongs to is displayed above it.

B.3 Implementation Details

In this section, we provide general implementation details applicable to the whole paper. For details relating to individual experiments and additional results, see Sections B.4, B.5 and B.6.

B.3.1 Concept Activation Vectors

Background In [Kim et al., 2018], the statistical test, TCAV, determines whether the model's sensitivity to a concept is significant. The test compares a set of CAV scores found using a concept dataset with CAV scores found using random data. To do this, we must find multiple CAVs for each concept. In practice, each of these CAVs is trained with the same positive set, $\mathbb{X}_c^+$, but a different random set, $\mathbb{X}_c^{r-}$, where $r \in 1, 2 \ldots R$ denotes the random index. A CAV corresponding to a specific random index is labelled $v_{c,l}^r$.

Implementation Details In this work, we create multiple CAVs per training run (30 unless otherwise stated), each using the same positive probe dataset but a different random set. We label a CAV trained with a specific random set as $v_{c,l}^r$, where $r \in 1, 2 \ldots R$ denotes the random index. Random CAVs are generated from pairwise combinations of random data sets, and we conduct a two-sided Welch's t-test to test whether the means of concept and random TCAV scores are equal.

Table B.2: The number of channels for the models trained on the simple, standard, and spatial versions of the Elements dataset.

Layer	Simple	Standard	Spatial
layers.0	64	64	64
layers.1	64	64	64
layers.2	64	64	128
layers.3	64	128	256
layers.4	64	128	256
layers.5	64	128	256

If a set of CAVs passes this test with a p value less than 0.01[1], we consider the concept meaningful. We refer to the mean TCAV score of the random CAVs as the null; it acts as the TCAV score all other CAVs should be compared against to understand their sensitivity to the model. The null is often very close to 0.5, simplifying the interpretation of the TCAV score to the concept having positive sensitivity when greater than 0.5 and negative when less.

B.3.2 Elements

Classification Model The model architecture is a simple convolutional neural network with six layers: each layer contains a convolution, batch norm and ReLU, followed by an average pooling and fully connected layer to give the logit outputs. The first three convolutional layers utilise a max-pooling operation to reduce dimensionality. We train the model using Adam [Kingma and Ba, 2015] with a learning rate of 1e-3 until the training accuracy is greater than 99.99%, giving a validation accuracy of 99.98% for the standard dataset. We use a different number of channels for the models trained on different datasets. This allows us to provide more model capacity when needed. The number of channels per layer for each model/dataset is summarised in Table B.2.

The models for datasets $\mathbb{E}_2$ and $\mathbb{E}_3$ in Section 3.4 are the same architecture as for the simple dataset ($\mathbb{E}_1$).

Probe Dataset For Elements, the probe datasets are generated so that the positive examples for a concept contain only objects with that concept, so, for example, a red concept image will contain four objects with random shapes and textures that occur within the dataset, but all of them will be red. The negative set consists of random samples from the dataset.

[1]we use a threshold of 0.01 to help reduce the false discovery rate

B.3.3 ImageNet

ImageNet is used to demonstrate the experiments on a real-world application.

Classification Model We use the default weights for a ResNet-50 [He et al., 2016] in the TorchVision package in PyTorch, which used a variety of data augmentation techniques including Mixup [Zhang et al., 2018], Cutmix [Yun et al., 2019], TrivialAugment [Müller and Hutter, 2021], and Batch Augmentation [Hoffer et al., 2020].

Probe Dataset Most probe datasets used to train CAVs were collated from the Broden dataset [Bau et al., 2017], particularly focusing on textures such as `striped`, `meshed` or `dotted`, or objects such as `car`, `sea` or `person`. Some concepts were manually curated, such as the `anemone` concept, which was collected from test images of the 'anemone fish' class from ImageNet that were not used elsewhere in the experiments.

B.3.4 Layer Selection

When creating CAVs, we need to choose a model layer. For the small CNN used for Elements, we can create CAVs for all layers, but for larger models, such as the ResNet-50 for ImageNet, it is computationally infeasible. In this paper, we focus on layers near the end of the model. The justification for this is twofold. (1) From an information theory perspective, the activations earlier in the network may contain more irrelevant information, suggesting the activations closer to the output may be more relevant to the prediction [Xu et al., 2020, McGrath et al., 2022b]. We aim to use TCAV to explain the model output, therefore later layers may be more desirable. (2) The model representations may be more complex in later layers. This allows us to create CAVs for more complex concepts. We find that the empirical evidence supports these hypotheses. Figures B.3 and B.4 show the accuracy of the linear classifiers used to create the CAVs on a held-out test set for each probe dataset. The accuracy for each concept tends to increase in later layers. This suggests the CAVs better capture the model representations in later layers. However, we observe variation across the concepts. For example, the colours in Elements are easily classified in all layers, whereas the shapes/textures have lower performance in layers.1. As our goal is to understand the behaviour of concept vectors (when the concept is represented), we focus on CAVs that obtain at least 90% test accuracy. Therefore, we omit layers.1 in our analysis.

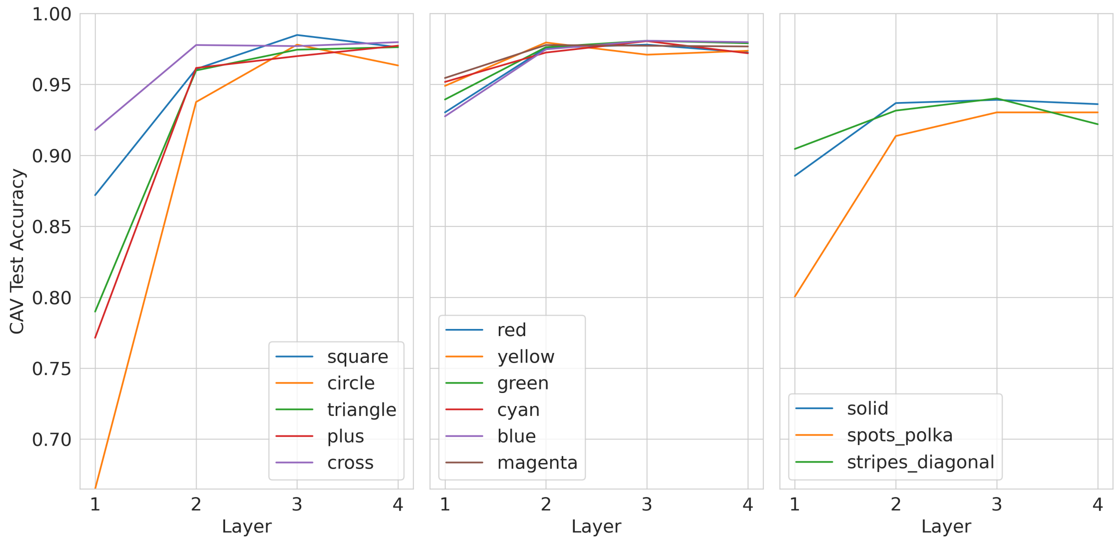


Figure B.3: Mean test accuracy for the linear classifiers from which the CAVs are generated for all concepts in the standard Elements dataset (split by concept type).

We do not create CAVs for the final convolutional layer in either the simple CNNs for Elements (layers.5) or the ResNet-50 for ImageNet (layer4.2) due to the gradient behaviour in these layers. In both cases, the network has no non-linearities after the layer. Therefore, the gradient of the logit with respect to the activations solely depends on the model weights, not the activations. TCAV relies on having a distribution of directional derivatives, which are then thresholded and averaged over different data points. For these layers, the gradient is the same for all inputs, and hence so is the directional derivative. This means the TCAV score for an individual CAV in these layers will be exactly 1 or 0. As such, we do not perform TCAV on layers after which there are no non-linearities.

B.4 Consistency Experiment Details

Figure 3.9 shows the consistency error for various types of CAV. In this section, we describe how we find the different types of CAV. In each case, v_{c,l_1} is a CAV trained as normal using a probe dataset, whereas the creation method for v_{c,l_2} varies for each experiment and is described below:

Optimised CAV We use gradient descent to optimise v_{c,l_2} to minimise the consistency error:

$$\arg\min_{v_{c,l_2}} ||f(z_{l_1} + v_{c,l_1}) - (z_{l_2} + v_{c,l_2})|| \tag{B.33}$$

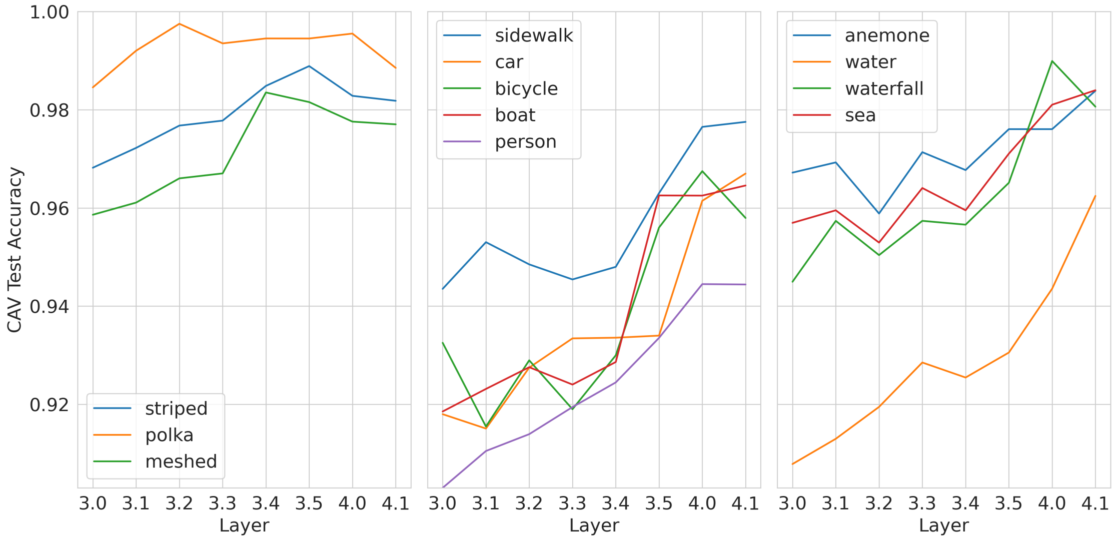


Figure B.4: Mean test accuracy for the linear classifiers from which the CAVs are generated for a selection of concepts in ImageNet.

The starting point for each optimisation process is a CAV in layer l_2, $v^{r_2}_{c,l_2}$, trained on a different random probe dataset. The nearly identical errors for each optimisation process support the likelihood of a global minimum being reached.

Concept CAV These are CAVs trained using a probe dataset containing $\mathbb{X}^+_c$ and $\mathbb{X}^-_c$. The distribution is over r_2 for different random probe datasets $\mathbb{X}_{c,r_2}$ for v_{c,l_2}, where $r_2 \neq r_1$ denotes the random set.

Projected CAV CAVs in layer l_1 projected into layer l_2 using f: $f(v_{c,l_1})$. The distribution is over different CAVs in layer l_1, i.e., v^r_{c,l_1}.

Random CAV CAVs trained using a random probe dataset for both the positive and negative sets:

$$\begin{aligned} \mathbb{X}^-_c &= \mathbb{X}^-_{c,r_1} \\ \mathbb{X}^+_c &= \mathbb{X}^-_{c,r_2}, \quad r_2 \neq r_1 \end{aligned} \tag{B.34}$$

Random Direction Each element of v_{c,l_2} is drawn from a uniform distribution between [-0.5, 0.5], and rescaled to be a unit vector. The distribution is over different random seeds for the random number generator.

B.4.1 Scaling perturbations

To ensure that $z_l + v_{c,l}$ stays in-distribution, we scale the perturbations as follows:

$$\hat{z}_l = z_l + \gamma v_{c,l} \frac{\overline{||z_l||}}{||v_{c,l}||}, \tag{B.35}$$

where γ is a hyperparameter used for perturbation size (typically set to 0.01), $||\cdot||$ the L_2 norm of a vector, and $\overline{||z_l||}$ the average norm of z_l. We scale the perturbation by the mean activation norm to account for the difference in the norms between the activation and concept vector to have consistently sized perturbations across layers.

We performed experiments to explore the sensitivity of consistency error to the size of the perturbation, γ, for various layers and concepts for the ImageNet and Elements datasets. In Figures B.5 and B.6, we show the results for a variety of concepts for both the ImageNet and Elements datasets. Similar patterns were observed across experiments. As we increase $|\gamma|$, the consistency error scales linearly, as a larger perturbation causes a larger difference between $f(\hat{z}_{l_1})$ and $\hat{z}_{l_2}$. The scale of the y axis on the left of Figures B.5 and B.6 is not particularly meaningful without context, so we scale it by the norm of the perturbation in layer l_2, $||\gamma v_{c,l_2} \frac{\overline{||z_{l_2}||}}{||v_{c,l_2}||}||$, on the right of the same figures. Values greater than one mean that the difference between $f(\hat{z}_{l_1})$ and $\hat{z}_{l_2}$ are larger than the perturbation made in layer l_2.

In addition to scaling γ for both perturbations (in layer l_1 and l_2), we explored fixing the size of the perturbation in layer l_1 and varying γ in l_2. We fixed γ_{l_1} to be 0.01 and then varied γ_{l_2} from 0 to 0.02 and measured the consistency error of the two perturbations. In Figure B.7, we show these results for `striped` CAVs from a ResNet-50 trained on ImageNet. We repeated the experiment on two types of CAVs: (1) standard CAVs (2) optimised CAVs (as described in the previous section). In both cases having γ fixed at 0.01 gives a larger error, but the minimum error is not substantially different. We observe a larger difference in error between $\gamma_{l_2} = 0.01$ and the minimum error for the standard CAVs than for the optimised CAVs. This is to be expected, as the directions of the standard CAVs are less aligned than the optimised CAVs.

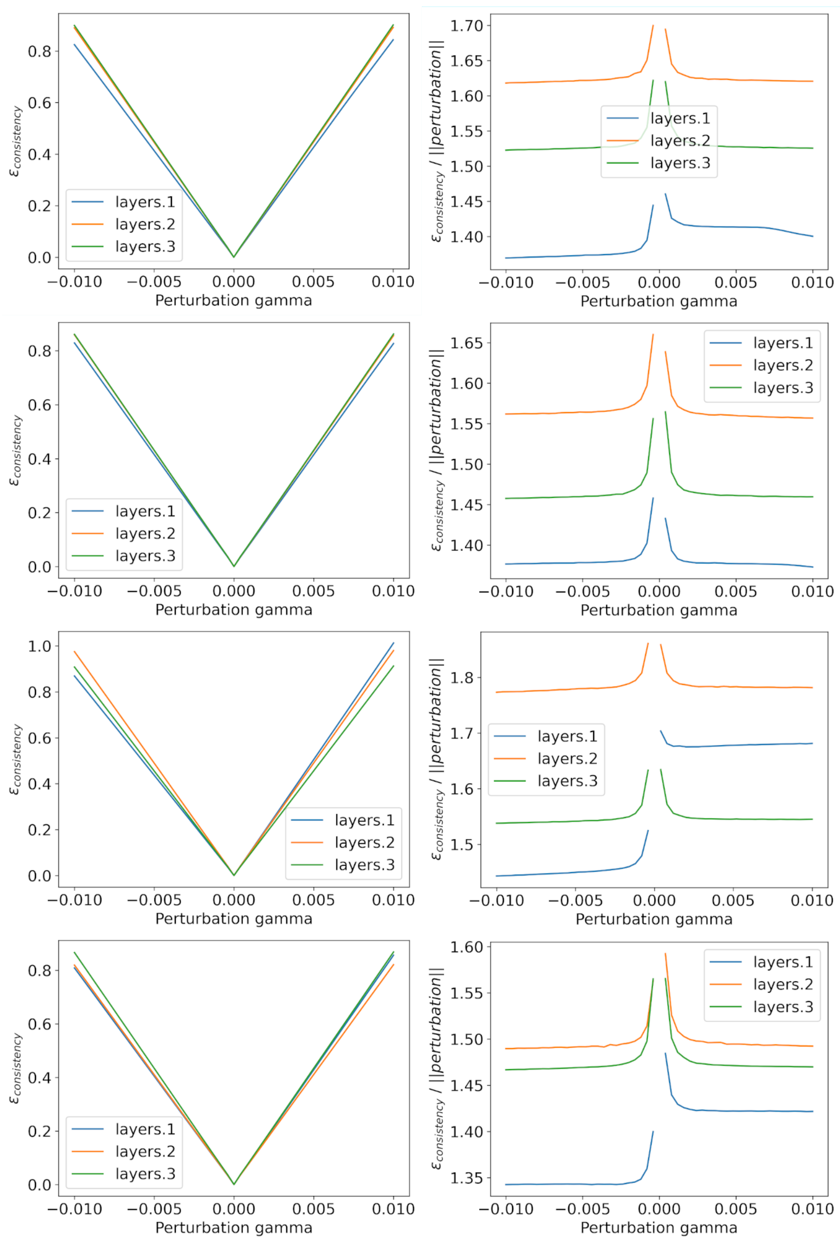


Figure B.5: The mean consistency error for (from top to bottom) `red`, `blue`, `triangle` and `striped` CAVs across layers (left) scaled by the size of the perturbation (right) for the Elements dataset.

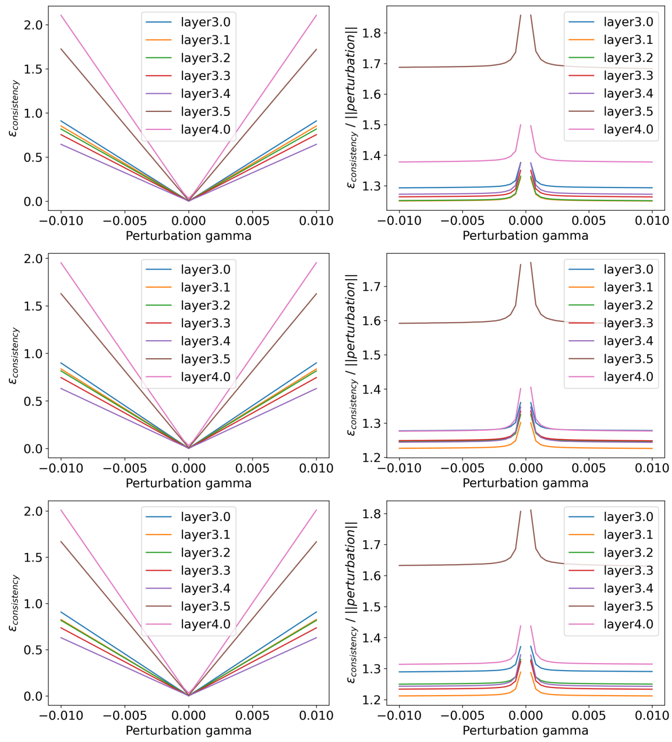


Figure B.6: The mean consistency error across 10 CAVs for `striped` (top), `lined` (middle) and `dotted` (bottom) CAVs across layers (left) scaled by the size of the perturbation (right) for a ResNet-50 trained on ImageNet.

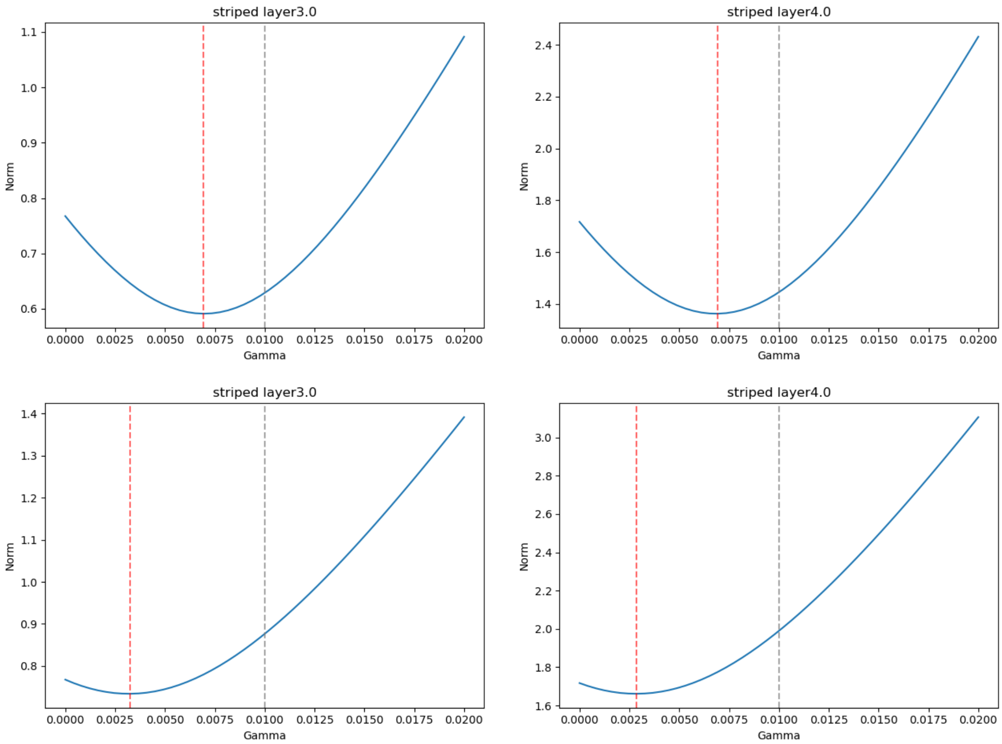


Figure B.7: The sensitivity of the mean consistency error to scaling γ_{l_2} for optimised CAVs (top) and standard CAVs (bottom) where l_1 =layer3.0 (left) and l_1 =layer4.0 (right) of a ResNet-50 trained on ImageNet. $\gamma = 0.01$ is the standard perturbation made in our experiments and is marked by a grey dashed line and the γ which gives a minimal consistency error is marked by a red dashed line.

In order to determine how often TCAV scores are inconsistent across layers, we introduce a new metric, the TCAV layer consistency score. It is a measure of how well the significant TCAV scores for some concept, c, and class, k, agree with each other across different layers $l_i, i \in 1 \dots L$. It is based on how many layers are on the same side of the null TCAV score. The null TCAV score is the mean TCAV score for a set of random CAVs (see Appendix B.3.1 for more detail). Mathematically the metric, $S_{\text{consistent}}$, is defined as

$$S_{\text{consistent}} = |2((\frac{1}{L}\sum_{i} \text{TCAV}_{c,k,l_i} > \text{TCAV}_{r,k,l_i}) - 0.5)| \tag{B.36}$$

where r indicates the TCAV score is calculated using random CAVs. It is difficult to make confident statements about the sensitivity of the model when the layer consistency score is close to 0 as it indicates the TCAV scores from different layers tend to disagree on the direction of the sensitivity.

We obtain a mean TCAV consistency score of 0.841 across all concepts/classes/layers for Elements and a mean TCAV consistency score of 0.868 for a selection of 6 classes and 16 concepts for a ResNet-50 trained on ImageNet (See Appendix B.4.5 for further details). The scores indicate that although complete disagreement across layers is not common (5.5% and 2.1% of scores were equal to 0 for Elements and ImageNet, respectively), it can occur, and a not insubstantial number of concepts/classes have a reasonable amount of disagreement across layers (23% and 14% of scores were ≤ 0.5 for Elements and ImageNet, respectively).

B.4.2 Additional Results

In this section we provide additional results for the different consistency experiments as in Figure 3.9. We include the results for multiple concepts and layers for both the ImageNet and Elements datasets. To allow for better comparison between layers, we normalise the consistency errors in some of the figures. For each layer, the consistency error is divided by the mean error for the optimised CAVs. For the normalised plots, a value of one can be seen as the lowest error possible for that layer. The relative ordering of layers is approximately consistent across the different types of CAV. For example, in Figure B.8, there is a downward trend between layer3.0 to layer3.5 and then an increase for layer4.0.

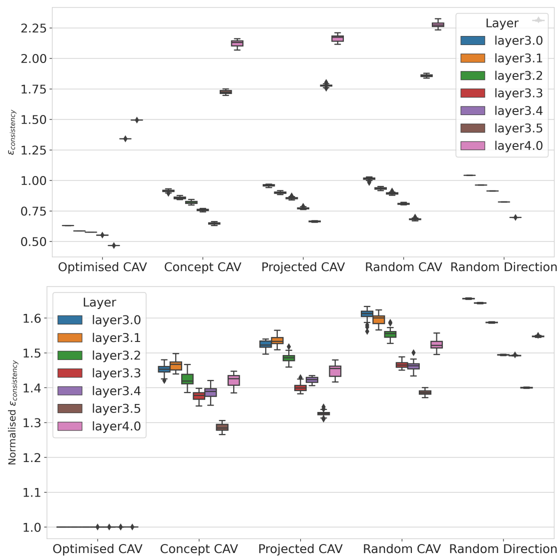


Figure B.8: The distribution of consistency errors (top) and normalised consistency errors (bottom) for different v_{c,l_2} for `striped` in a selection of layers from a ResNet-50 trained on ImageNet. Optimised CAV: The lower bound – a vector optimised to have the minimum error. Concept CAV: `striped` CAVs, trained as normal. Projected CAV: striped CAVs from layer l_1 projected into layer l_2, $f(v_{c,l_1})$. Random CAV: CAVs with random images for the probe dataset. Random Direction: Random vectors drawn from a uniform distribution.

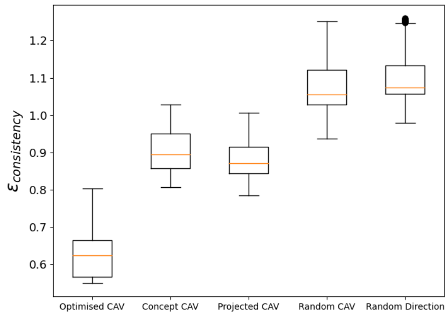


Figure B.9: The distribution of consistency errors for different v_{c,l_2} for `square`, `triangle`, `red`, `green`, `solid` and `stripes` CAVs for 'layers.1', 'layers.2' and 'layers.3' of a CNN trained on the Elements dataset. Optimised CAV: The lower bound – a vector optimised to have the minimum error. Concept CAV: CAVs, trained as normal. Projected CAV: striped CAVs from layer l_1 projected into layer l_2, $f(v_{c,l_1})$. Random CAV: CAVs with random images for the probe dataset. Random Direction: Random vectors drawn from a uniform distribution.

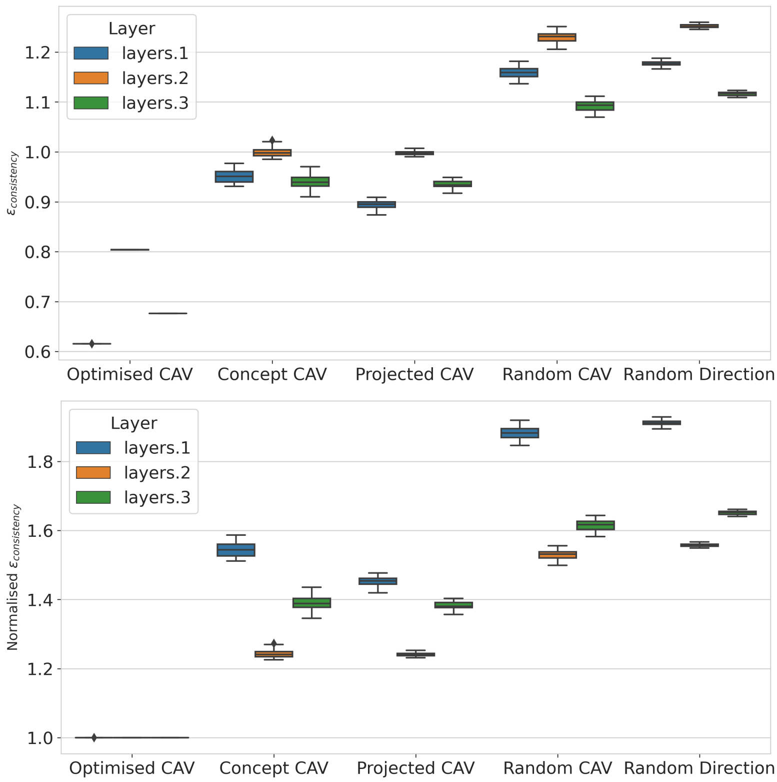


Figure B.10: The distribution of consistency errors (top) and normalised consistency errors (bottom) for different v_{c,l_2} for `square` CAVs for a variety of layers for the Elements dataset. Optimised CAV: The lower bound – a vector optimised to have the minimum error. Concept CAV: CAVs, trained as normal. Projected CAV: striped CAVs from layer l_1 projected into layer l_2, $f(v_{c,l_1})$. Random CAV: CAVs with random images for the probe dataset. Random Direction: Random vectors drawn from a uniform distribution.

B.4.3 DeepDream

DeepDream [Mordvintsev et al., 2015] is a feature visualisation tool. It starts from an image, generated by sampling noise from a random uniform distribution and then iteratively updates the input image to maximise the L2 norm of activations of a particular layer. We use a similar approach but instead maximise the dot product between the activations and a CAV: $v_{c,l} \cdot z_l$. In Figures B.11 and B.12, we show these visualisations for a selection of ImageNet CAVs for successive layers in a ResNet-50.

These visualisations offer qualitative evidence that the CAVs represent different components of the same concept in different layers. For example, the `car` concept in Figure B.11 consists of many square box-like objects in earlier layers, but nothing recognisable as a car, whereas, in later layers, whole car sections can be seen in the visualisations.

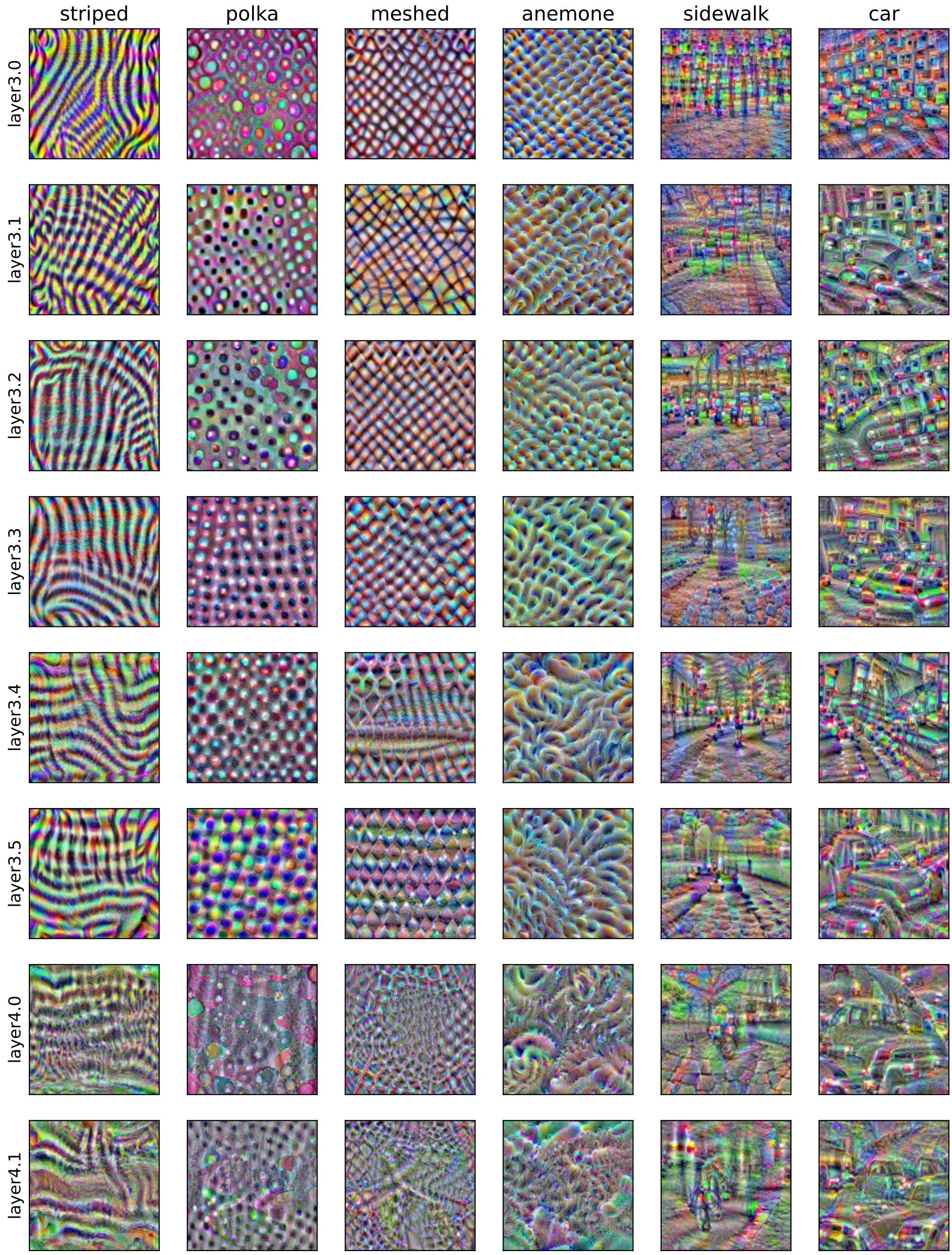


Figure B.11: CAV visualisations using DeepDream for a selection of concepts from ImageNet. Each row corresponds to a layer of a ResNet-50 and each column a different concept.

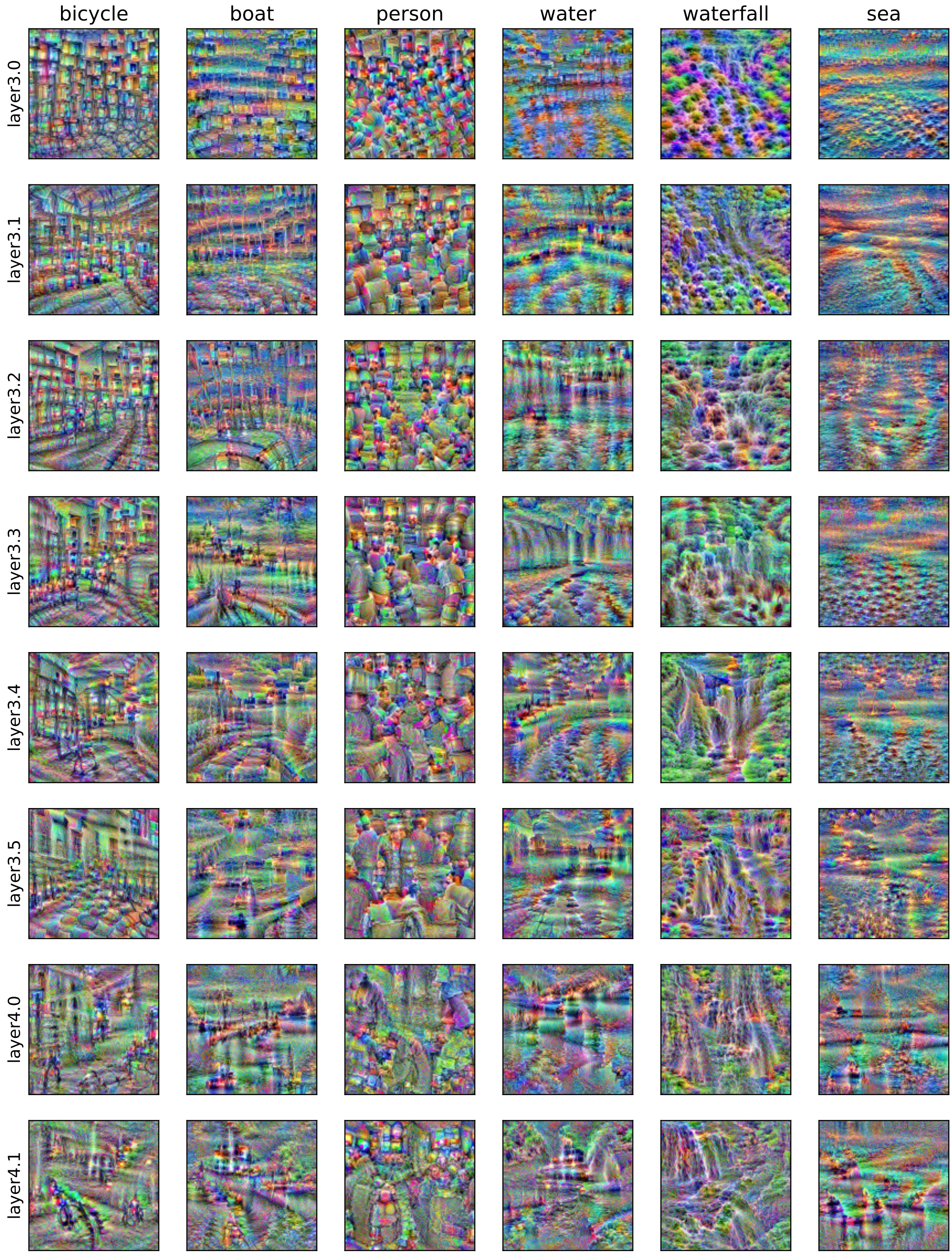
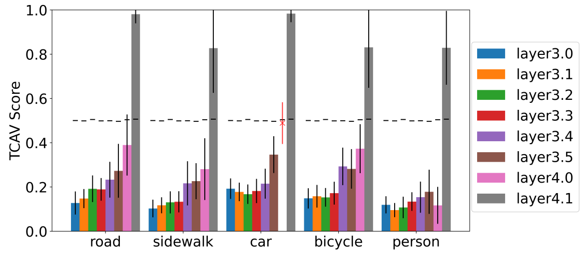


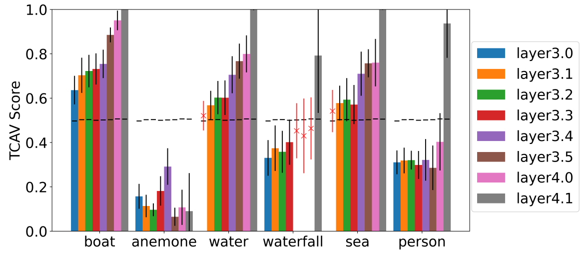
Figure B.12: CAV visualisations using DeepDream for a selection of concepts from ImageNet. Each row corresponds to a layer of a ResNet-50 and each column a different concept.

B.4.4 Inconsistent TCAV Scores

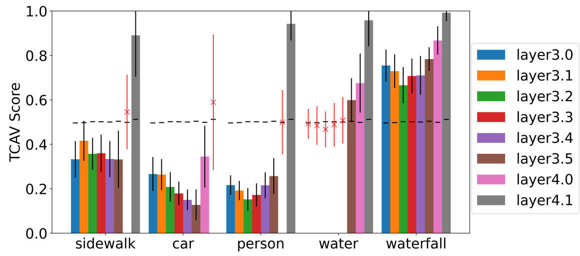
In this section, we display additional examples of inconsistent TCAV scores across layers for ImageNet classes. In Figure B.13 each subfigure contains at least one concept that has inconsistent TCAV scores. We include example images from those classes in Figure B.14.



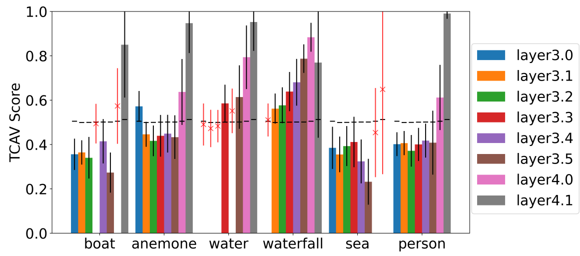
(a) TCAV scores for a selection of concepts for the 'car wheel' class in ImageNet.



(b) TCAV scores for a selection of concepts for the 'dock' class in ImageNet.



(c) TCAV scores for a selection of concepts for the 'sidewalk' class in ImageNet.



(d) TCAV scores for a selection of concepts for the 'lionfish' class in ImageNet.

Figure B.13: Inconsistent TCAV scores for a selection of concepts and classes in ImageNet. The standard deviation is shown in black for significant results and red for insignificant results. The mean TCAV score for random CAVs are shown as horizontal black lines.

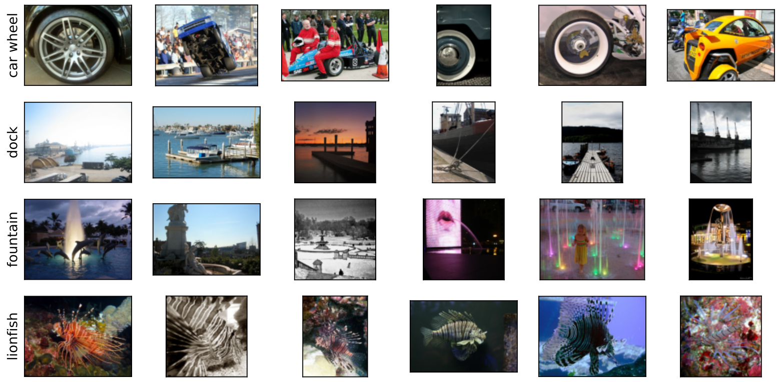


Figure B.14: Example images from a selection of ImageNet classes.

B.4.5 TCAV Layer Consistency Score

At the end of Section 3.6 we defined the TCAV layer consistency score as a measure of how often significant TCAV scores disagree on the direction of sensitivity for a specific concept and class across a set of layers. Here, we provide some example results for both the ImageNet and Elements datasets. In each case we train 30 CAVs for each concept/layer pair to calculate the TCAV score across. For Elements, when analysing all 69 classes from the simple version of the dataset, we obtain a mean TCAV consistency score of 0.841 across all concepts and all layers of the simple NN. 38/690 sets of TCAV scores had a TCAV layer consistency score of less than 0.2 and 158/690 less than 0.5. For ImageNet, we obtain a mean TCAV consistency score of 0.868 across the last 8 convolutional layers (from layer 3.0 to layer 4.1) for a range of concepts (striped, polka, meshed, road, sidewalk, car, bicycle, boat, person, anemone, water, waterfall, sea, sky, snow, tree) and classes (zebra, leopard, tiger, car wheel, acoustic guitar, academic gown). 7/96 sets of TCAV scores had a TCAV layer consistency score of less than 0.2 and 13/96 had a score less than 0.5. See Figure B.15 for the full distributions.

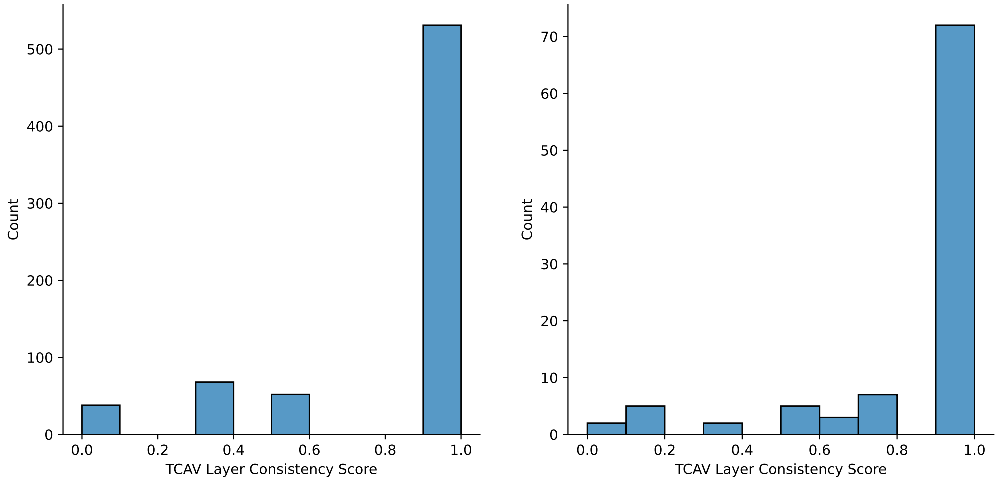


Figure B.15: Distribution of TCAV layer consistency score for the simple Elements dataset (left) and a selection of classes/concepts for a ResNet-50 trained on ImageNet (right).

B.5 Entanglement Experiment Details

We use cosine distance to measure how similar two CAVs are. Assuming the CAVs, $v_{c_1,l}$ and $v_{c_2,l}$, are unit vectors this simplifies to the dot product of the two:

$$\begin{aligned} \text{Cosine Similarity} &= \frac{v_{c_1,l} \cdot v_{c_2,l}}{\|v_{c_1,l}\| + \|v_{c_2,l}\|} \\ &= v_{c_1,l} \cdot v_{c_2,l} \end{aligned} \tag{B.37}$$

In our visualisations (Figures 3.3 and B.5.1) we compare multiple CAVs for each concept, each with a different random probe dataset, denoted by r. This allows us to see how similar CAVs for the same concept are on repeat training runs. Each value in the visualisation is the mean cosine similarity between the concepts on its corresponding x and y axis labels between all CAVs which do not have the same random probe dataset, i.e.:

$$\sum_{r_1}^{R} \sum_{r_2 != r_1}^{R} \frac{v_{c_1,l}^{r_1} \cdot v_{c_2,l}^{r_2}}{R(R-1)} \tag{B.38}$$

where $v_{c_1,l}^{r_1}$ is the CAV corresponding to concept c_1, layer l and random probe dataset $\mathbb{X}_{c,r_1}$.

B.5.1 Additional Results

Elements

In Figure B.16, we show the cosine similarities for all concepts in the standard Elements dataset. The conclusions are similar to the visualisation for $\mathbb{E}_1$ in Figure 3.3, but the negative associations between the mutually exclusive concepts are weaker. We hypothesise that this is because there are more concepts within each group. This is empirically justified as the average cosine similarity of each group approximately corresponds to the number of concepts in the group. If this hypothesis is true, it makes it unlikely that we will find similar groupings in real datasets containing natural images as concepts are rarely partitioned as neatly or in as few possible combinations.

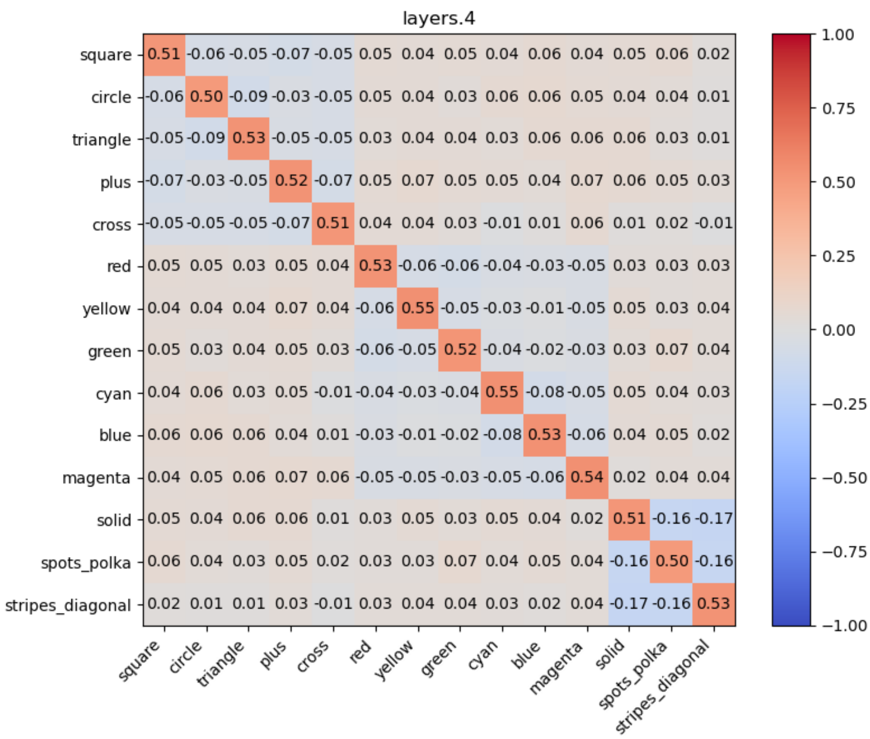


Figure B.16: Mean pairwise cosine similarities between 30 CAVs for different concepts from the standard Element dataset.

ImageNet

In Figure B.17, we show the pairwise cosine similarities for a selection of concepts for a ResNet-50 trained on ImageNet. The associations between concepts are less clear-cut than for Elements, but qualitatively they make intuitive sense. For example, the concepts most similar to `field` are `grass` and `earth`, in the top of Figure B.17, and the concepts most similar to `sidewalk` are `bicycle`, `road` and `hedge`. The latter makes sense as many of the `hedge` exemplars in the probe dataset are next to a path or road. This emphasises the importance of designing your probe dataset to match the concept you want the CAV to represent.

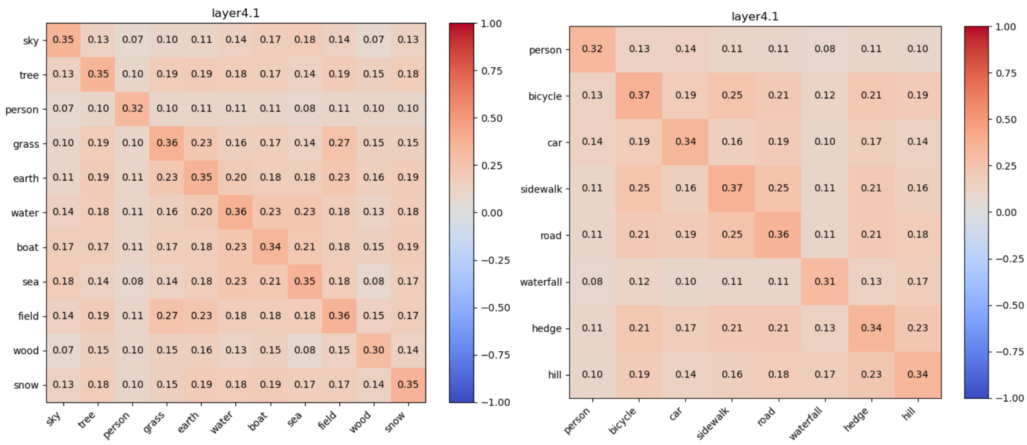
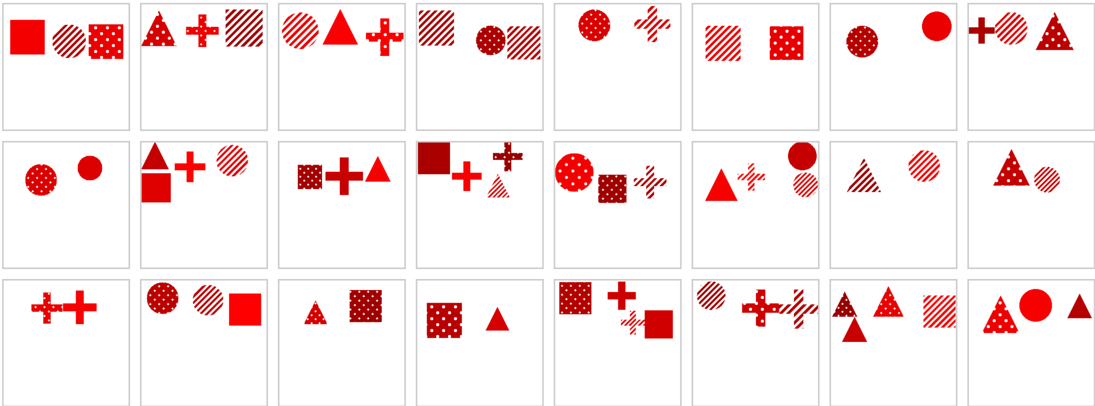


Figure B.17: Mean pairwise cosine similarities between 30 CAVs for different concepts from ImageNet.

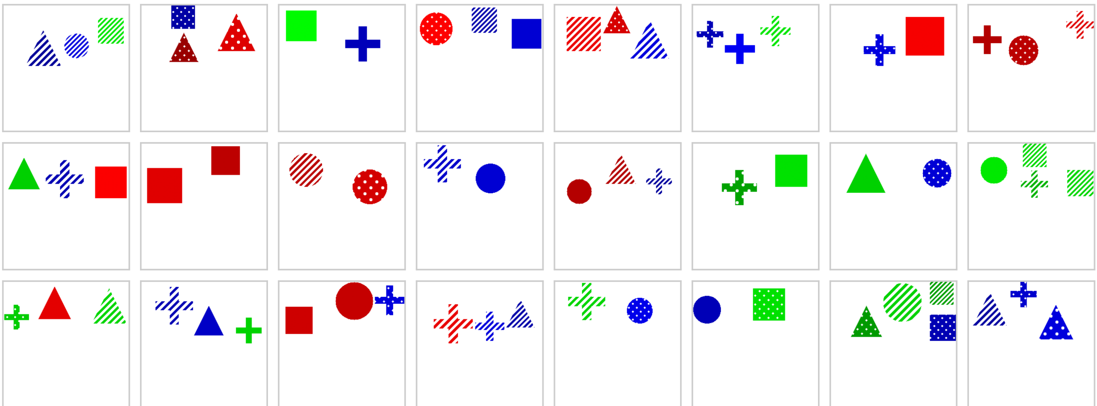
B.5.2 Polysemanticity

In Figures 3.3 and B.16 we find that the vector representations of mutually exclusive concepts are anti-correlated with each other. Each concept vector does not just mean, for example, `red`. It means `red`, `not green` and `not blue`.

Elhage et al. [2022a] also find evidence for polysemantic representations. However, they found individual neurons which were polysemantic, whereas here vectors, i.e., groups of neurons, are polysemantic. In addition, the reasoning is different. The polysemanticity discussed in Elhage et al. [2022a] is caused by sparse features being compressed into fewer neurons than there are features. Here, we do not have sparse features and have more neurons than features. Instead, the polysemanticity is caused by associations between the concepts and the optimisation process favouring negatively correlated representations for mutually exclusive concepts.



(a) The positive set.



(b) The negative set.

Figure B.18: Example images for the positive and negative sets of the probe dataset for the `red top` in the simple elements dataset.

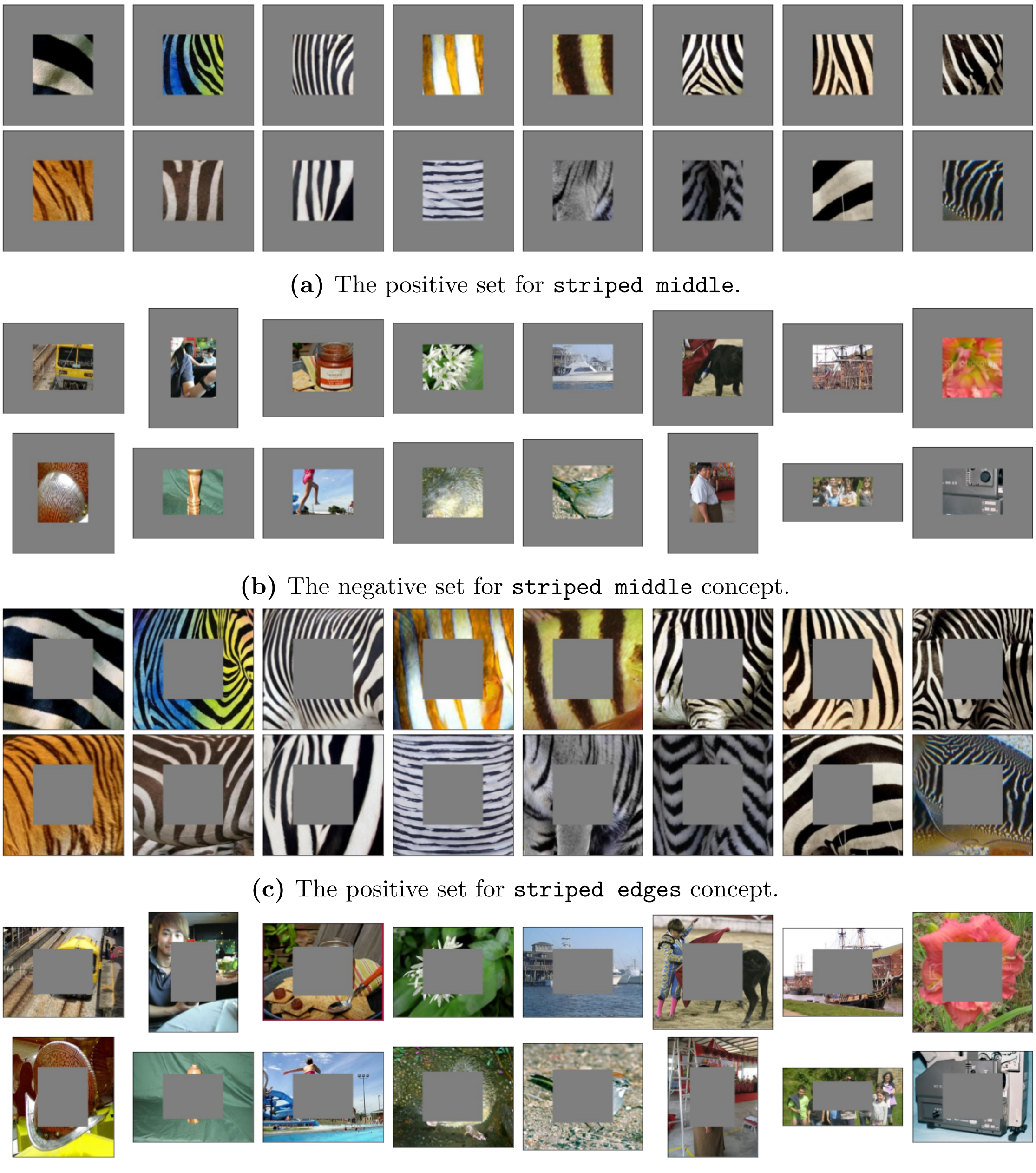
B.6 Spatial Dependency Experiment Details

B.6.1 Spatially Dependent Probe Datasets

For Elements, the probe datasets contained elements that only appear in specified locations – an example is shown in Figure 3.2. For ImageNet, we do not have direct control of where objects can appear. Therefore, we greyed out different regions of the image. For example, we created oppositely dependent concepts by either greying out the middle of the image, or the edges - see Appendix B.6.1 for examples.

B.6.2 Spatial Norms Details

When finding CAVs, we use the activations of layer l in a convolutional neural network, which has shape $H \times W \times D$, where H, W and D are the height, width and number of channels, respectively. When finding the CAV, these activations



(a) The positive set for `striped middle`.

(b) The negative set for `striped middle` concept.

(c) The positive set for `striped edges` concept.

(d) The negative set for `striped edges`.

Figure B.19: Example images from spatially dependent probe datasets for ImageNet.

are flattened to be vectors of length $m = H \times W \times D$. The value of each element in the activations depends on its specific height, width and depth indices. As a result, each corresponding element of the resultant CAV is also index-dependent. We are interested in spatial dependence, so to visualise how a CAV varies across the width and depth dimensions we calculate the L2 norm of each depth-wise slice – the CAV's spatial norms.

B.6.3 Individual Spatial Norms

In Figure B.20, we present the spatial norms for `striped` in a ResNet trained on the ImageNet dataset. Each heatmap is for a different random probe dataset, denoted by r. The different patterns in the heatmaps show that each individual $v^r_{c,l}$ has a spatial dependency which differs across r. However, when we average the norms across multiple CAVs, $\sum_{r=1}^{R} \mathbf{S}^r_{c,l}/R$, we obtain a uniform distribution across the spatial dimensions. This uniformity suggests that the spatial dependencies of each individual CAV cancel out across multiple seeds, as depicted in the top rows of Figure 3.6 in the main text.

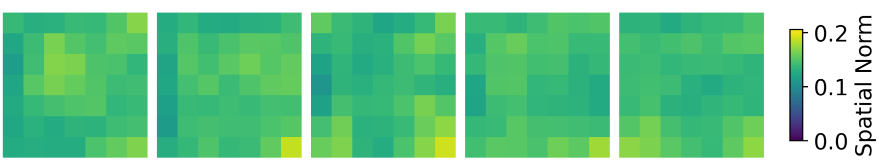


Figure B.20: Individual spatial norms for `striped`, where each CAV was trained on a different negative probe set, for layer4.1 of a ResNet trained on ImageNet.

B.6.4 Additional Spatial Norms

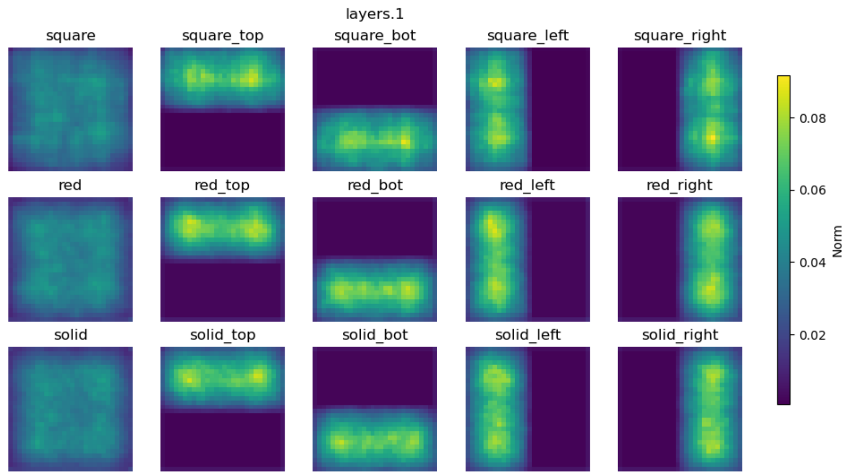


Figure B.21: Mean CAV spatial norms across 30 CAVs for a selection of concepts in the Element dataset for the second convolutional layer.

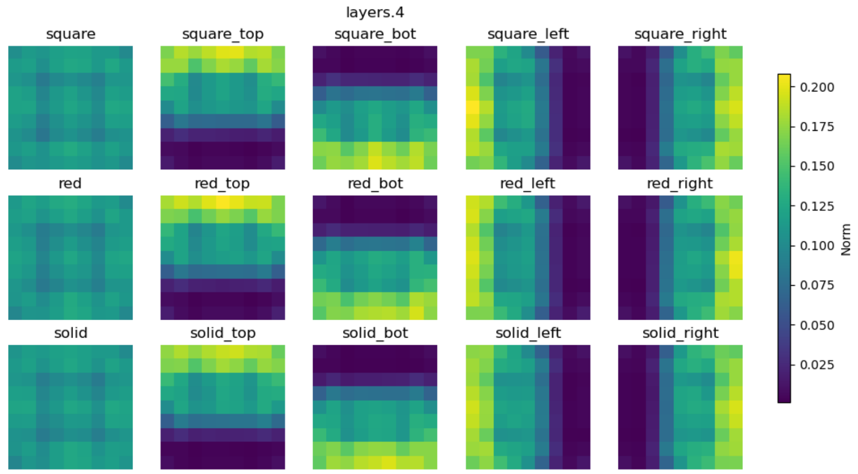


Figure B.22: Mean CAV spatial norms across 30 CAVs for a selection of concepts in the Element dataset for the fifth convolutional layer.

B.6.5 Spatial Means

Instead of visualising the norm of each depth-wise slice, we can visualise the mean. We default to showing the norm because you could have a spatial mean of zero in a region of activation space which has a large effect on the directional derivative. This could occur, for example, if half the elements of the CAV are large and positive and the corresponding gradients are positive, and half the elements of the CAV are large and negative and the corresponding gradients are also negative. This would lead to that spatial region having a large contribution to the directional derivative, but the spatial mean would be close to zero. The spatial norm, however, would be large for this region. This makes the norm a better measure of the effect of each region, but the mean can still be useful to show the direction of that effect.

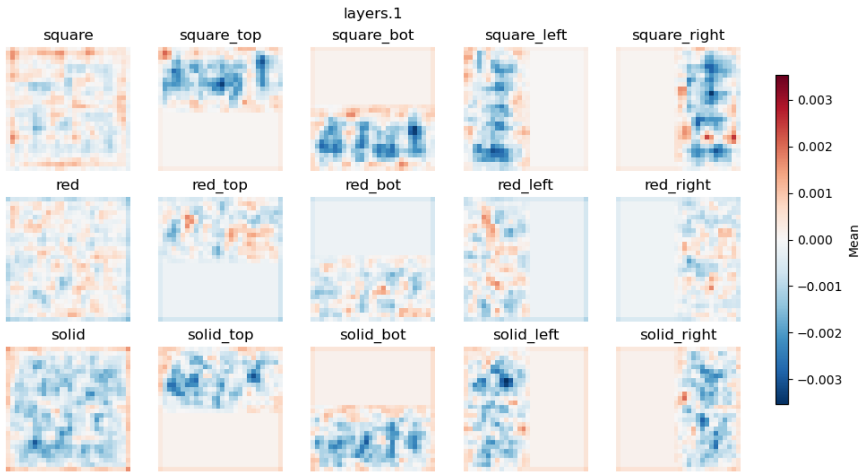


Figure B.23: Mean CAV spatial means across 30 CAVs for a selection of concepts in the Element dataset for the second convolutional layer.

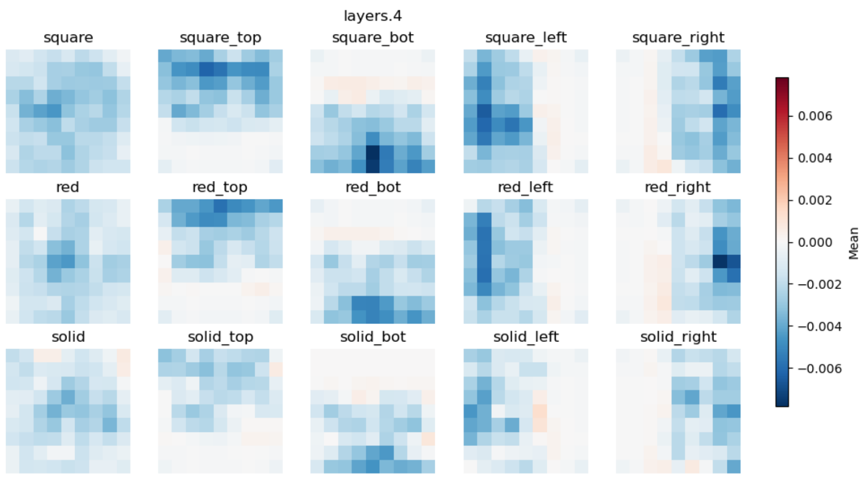
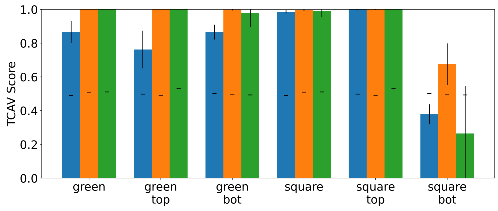


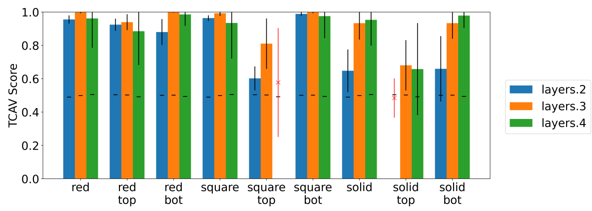
Figure B.24: Mean CAV spatial means across 30 CAVs for a selection of concepts in the Element dataset for the fifth convolutional layer.

B.6.6 Spatially Dependent TCAV Scores

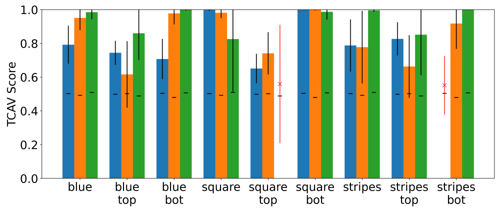
In this section we provide example TCAV scores which differ across complementary spatially dependent CAVs, i.e.„ for at least one concept, the TCAV score is the opposite side of the null for the `edges` version of a concept compared to the `middle` version (or the `left/right` and `top/bottom` versions).



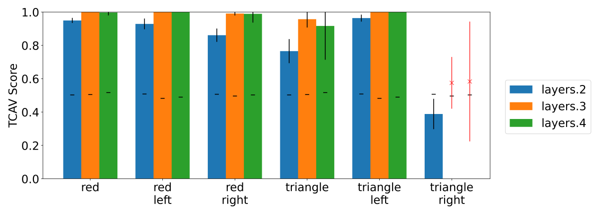
(a) ‘green squares on the top’



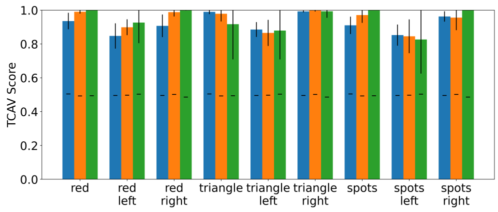
(b) ‘solid red squares on the bottom’



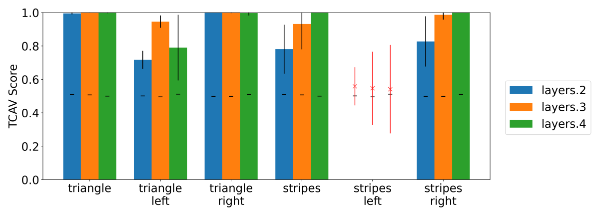
(c) ‘striped blue squares’



(d) ‘red triangles on the left’

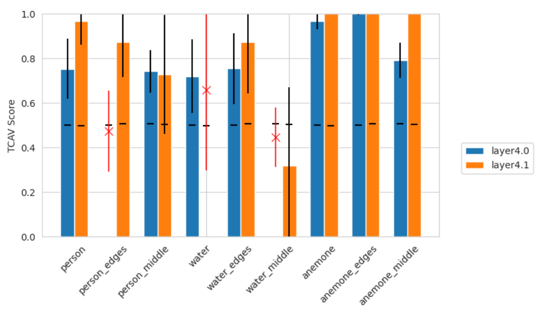


(e) ‘spotted red triangles on the right’

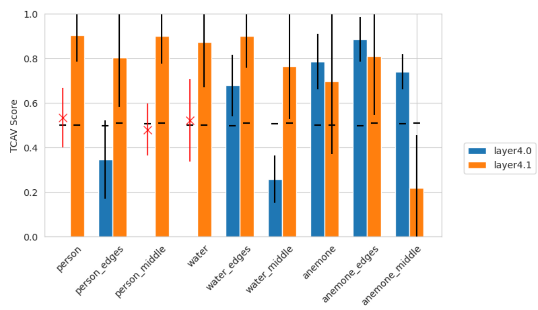


(f) ‘striped triangles on the right’

Figure B.26: Examples of spatially dependent TCAV scores in the spatially dependent version of Elements for a ResNet-50. Each subfigure is a separate class. The standard deviation is shown in black for significant results and red for insignificant results. The mean TCAV score for random CAVs are shown as horizontal black lines.



(a) TCAV scores for a selection of concepts for the ‘anemone fish’ class in ImageNet.



(b) TCAV scores for a selection of concepts for the ‘spiny lobster’ class in ImageNet.

Figure B.25: Examples of spatially dependent TCAV scores in ImageNet. Each subfigure is a separate class. The standard deviation is shown in black for significant results and red for insignificant results. The mean TCAV score for random CAVs are shown as horizontal black lines.

B.6.7 Spatially Dependent CAVs in ViTs

Even though we cannot use spatial norms to visualise the spatial dependence of CAVs in a transformer based architecture, in this section we have some preliminary results suggesting that that spatially dependent CAVs can still be created for a ViT-B16 [Dosovitskiy et al., 2021]. As for the ResNet-50, we test the model on the spatially dependent Elements dataset and found TCAV scores that vary by location.

We finetuned a ViT-B16 model pretrained on ImageNet for 50 epochs on the spatially dependent version of Elements (i.e. there are some classes which depend on the location of the objects as well as which concepts are present). We used an exponentially decaying learning rate with initial learning rate of 0.0001 and a γ of 0.95. Similarly to the previous section, we demonstrate that we obtain different TCAV scores for the `left` versions of a concept than the `right`, although there is substantially less variation in TCAV score than for the ResNet-50 (Figure B.26). Future work should perform more extensive experiments to determine how the properties analysed in this paper are affected by architecture.

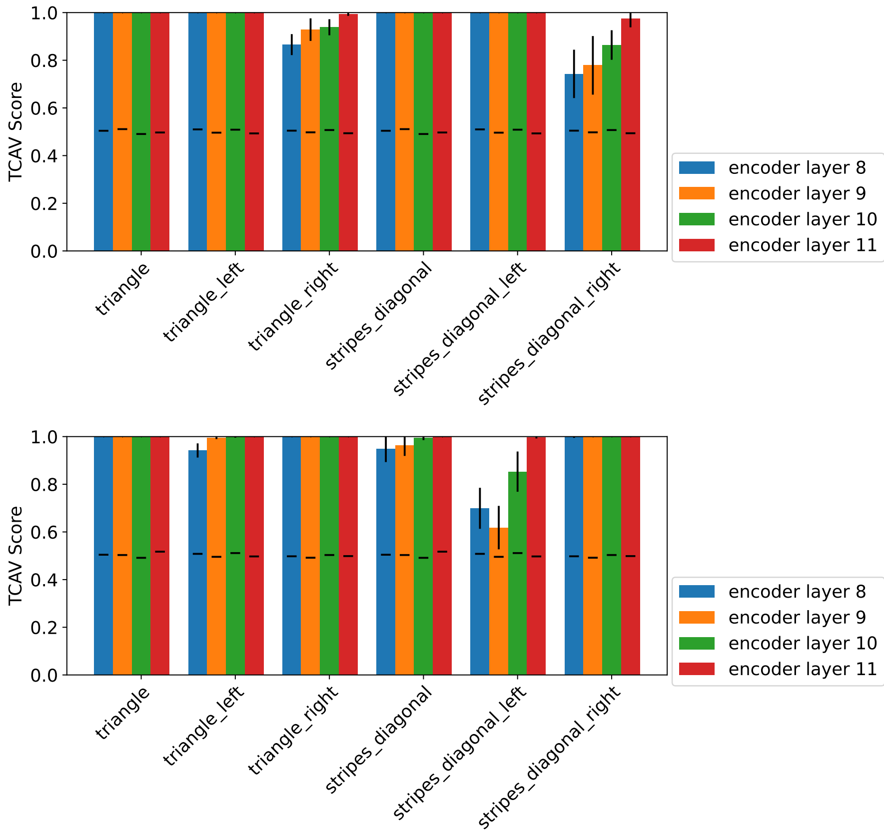


Figure B.27: Spatially dependent TCAV scores in the spatially dependent version of Elements for a ViT-B16. The TCAV scores for the classes "striped triangles on the left" (top) and "striped triangles on the right" (bottom) are shown. The standard deviation is shown in black for significant results and red for insignificant results. The mean TCAV score for random CAVs are shown as horizontal black lines.

B.6.8 Dot Product Distributions

The definition of concept vector spatial dependence in Equation 3.25 compares a CAV, $v_{c,l}$, with the activations of two positive probe datasets with different spatial dependencies, z^+_{c,l,μ_1} and z^+_{c,l,μ_2}, by taking the dot product between them $z^+_{c,l,\mu_x} \cdot v_{c,l}$. In Equation B.28, we show the distribution of dot products for three concepts and three test probe datasets in the spatially dependent version of Elements. The separation between the distributions for the `stripes left` and `stripes right` probe datasets (blue and green bars, respectively) for both the `stripes left` and `stripes right` CAVs (left and right plots, respectively) demonstrate that these CAVs are spatially dependent.

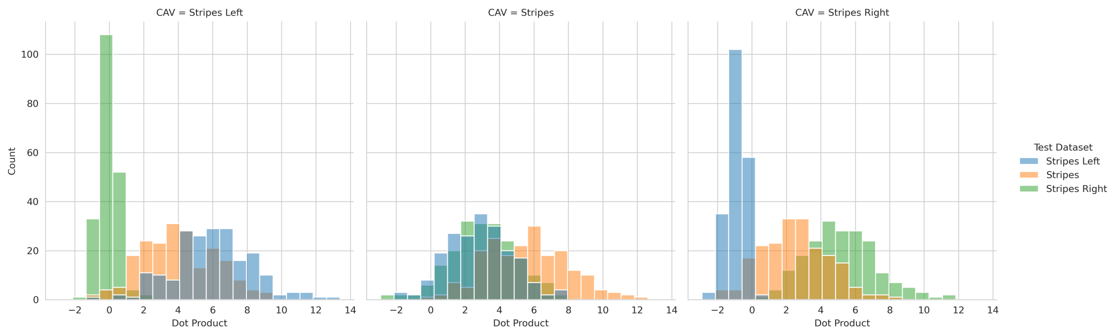


Figure B.28: Distribution of dot products between spatially dependent CAVs and image activations ($z^{+}_{c,l,\mu} \cdot v_{c,l}$) for the spatially dependent Elements dataset. Each column is for different CAVs. From left to right these are: `stripes left`, `stripes`, `stripes right`. For each CAV we show the distribution for three positive probe datasets: `stripes left` (blue), `stripes` (orange), `stripes right` (green).

C
Appendix: Understanding Knowledge Encoding in LLMs

C.1 LLM Training Data Languages

Table C.1 summarizes the languages the different models are trained on.

Table C.1: LLMs: High resource training languages

Model	Languages
Llama-3.1-70B [Dubey et al., 2024]	English, German, French, Italian, Portuguese, Hindi, Spanish, and Thai
Mixtral-8x22B-v0.1 [Jiang et al., 2024]	English, French, Italian, German, and Spanish
Aya-23-35B [Aryabumi et al., 2024]	Arabic, Chinese (simplified and traditional), Czech, Dutch, English, French, German, Greek, Hebrew, Hindi, Indonesian, Italian, Japanese, Korean, Persian, Polish, Portuguese, Romanian, Russian, Spanish, Turkish, Ukrainian, and Vietnamese
Gemma-2-27b [Team, 2024]	English

C.2 LLM-Insight Dataset

Our goal is to generate a dataset that can be used for cross-lingual interpretability. We wanted a dataset that can be used to introspect LLM internal representations and analyze LLM behavior when the internal representations are intervened on. Additionally, the dataset focuses on open-ended sentence generation rather than being restricted to specific tasks like fact recall or sentiment analysis, as text generation is an important real-world application.

Text Generation

We use GPT-4o to generate sentences and prompts. For each target word, we generate:

- 10 unique sentences containing a version of the word – for example, for the verb '(to) see', a suitable sentence is 'She saw a bird in the sky.'
- a list containing the version of the word used in each sentence. In the previous example, the version of the word is 'saw'.
- 10 unique prompts, designed to be completed with the target word.
- a list containing a version of the word used in each sentence

We instruct GPT-4o to generate prompts that can be completed with the target word, as well as semantically distinct words. However, we observe that the model sometimes produces sentences and prompts that do not meet the criteria.

An example of a sentence that does not meet the criteria is:

Target word: bouquet (boeket in Dutch)
Sentence: Het boeket was gevuld met levendige rozen en lelies.
Translation: The bouquet was filled with live roses and lilies.

The issue with this sentence is its unnatural phrasing–the word "live" is not typically used in this context.

An example of a prompt that does not meet the criteria is:

Target word: money
Prompt: He went to the bank to withdraw

In this case, the only plausible continuation is "money." While the prompt is coherent, it lacks the open-endedness needed to analyze how interventions influence model behavior.

An example of a well-constructed prompt is:

Target word: bus
Prompt: She took a

This can be completed with the intended word "bus", as well as semantically different alternatives such as "walk" or "long road trip".

To ensure data quality, we asked native speakers to review and correct the data. The original version of the data and the corrections are provided in the dataset.

Dataset Summary

We selected words that vary in the number of tokens (in non-English languages), whether the word is a homograph with the English version of the word, and the part of speech. Table C.2 summarizes the words used.

Table C.2: Word Translations Across Languages

Word	English	Dutch	French	German	Mandarin
animal	animal, animals	dier, dieren	animal, animaux	Haustier, Tier, Tiere	动物
bad	bad	slecht, slechte	mal, mauvais, mauvaise, mauvaises	schlecht, schlechte, schlechten	不好, 坏了
ballet	ballet	ballet	ballet	Ballett	芭蕾
bank	bank	bank	banque	Bank	银行

Word	English	Dutch	French	German	Mandarin
beautiful	beautiful	mooi, mooie	beau, bel, belle, magnifique	schön, schöne, schönen	美丽
big	big, tall	groot, groots, grote	grand, grande, grands, gros	große, großen, großer, großes	大
bouquet	bouquet	boeket	bouquet	Strauß	花束
brother	brother	broer	frère	Bruder	哥哥, 弟弟, 兄弟
bus	bus	bus	bus	Bus	公交车 / 巴士
cat	cat	kat	chat	Katze, Katzen, Mutterkatze	猫, 小猫, 流浪猫
centre	centre	centrum	centre	Zentrum	中心, 市中心, 研究中心, 社区中心, 艺术中心
chair	chair	stoel	chaise	Stuhl, Stuhls, Stühlen	椅子, 椅子, 摇椅
chauffeur	chauffeur, driver	chauffeur	chauffeur, chauffeuse	Chauffeur	司机
child	child	kind	enfant	Kind	孩子
club	club	club, vereniging	club	Club	俱乐部
cold	cold	ijskoud, kou, koud, koude	froid, froide, froides	kalt	冷, 寒冷
computer	PC, computer, laptop, machine, rig, system	computer	ordinateur	Computer, Computern, Laptop	电脑
culture	culture, cultures	culturen, cultuur	culture, cultures, la culture	Kultur	文化
day	day	dag	jour, journée	Tag	天, 日子
dog	dog	hond	chien	Hund	小狗, 狗
drink	drink	drank, drankje, drinken	boire, boisson, drinken	trinken	饮料
eat	eat	eten	dîner, manger, repas	essen	吃
fast	fast	fast, snel, snelle	rapide, vite	schnell	快
film	film	film, films	film	Film, film	电影, 影片, 纪录片
food	cuisine, cuisines, dish, food, meals	gerecht, gerechten, voedsel	cuisine, nourriture	Essen, Futter, Nahrung	食物

Word	English	Dutch	French	German	Mandarin
fruit	fruit	fruit	fruit, fruits	Frucht, Früchte, Obst	水果
garage	garage	garage, parkeergarage	garage	Garage	车库
give	give	geven, helpen	dire, donnent, donner, partager	geben	给
goal	goal	doel, doelpunt	but, objectif	Tor, Ziel, Ziele	球门, 目标
gobbledygook	buzzwords, gibberish, jargon, nonsense	gebazel, geheimtaal, jargon, koeterwaals, onzin, retoriek, waanzin, wartaal	baragouin, bla-bla, charabia, galimatias, gargouilloux	Kauderwelsch	废话, 难懂的术语
good	delicious, excellent, fun, good, great, helpful	goed, goede	bien, bon, bonne, bons	gut	好
hand	hand	hand, handen	main	Hand, Hände	手
happy	happy	blij	content, contente, contents, enthousiaste, gais, heureux, joyeuse	froh, glücklich	快乐, 高兴, 高兴, 快乐
horse	horse, pony	paard, paarden	cheval	Pferd, Pferde	马
hot	hot	heet, hete	chaud, chaude, chaudes	heiß	热
incomprehensibility	incomprehensibility	onbegrijpelijkheid	incompréhensibilité	Unverständlichkeit	不可理解, 难以理解
information	information	informatie	information, informations	Information, Informationen	信息
land	land	land, landen	atterrir, campagne, nature, terrain, terrains, terre, territoire	Land	土地, 土地, 国家
machine	device, equipment, machine, maker, system	machine	machine	Maschine	机器
menu	menu	menu, menukaart	menu	Menü	菜单
money	money	geld	argent	Geld	钱
no	no	nee	non	nein	不

Word	English	Dutch	French	German	Mandarin
please	please	alsjeblieft	s'il te plaît, s'il vous plaît	bitte	请
police	police	politie	police	Polizei	警察
radio	radio	radio, radiozender	fréquence, radio	Radio	广播, 收音机
read	read	lezen	lire	lesen	阅读
room	room	kamer	chambre	Zimmer	房间
sea	sea	zee	mer, zoo	Meer	大海, 海, 海洋, 海浪, 海风
see	see	zien	voir	sehen	欣赏, 看, 观察
serendipity	serendipity	toeval, toevalstreffer	chance, coïncidence, hasard, sérendipité, éventualité	glückliche Zufälle, glücklicher Zufall	机缘巧合
sister	sister	zus, zusje	soeur	Schwester	姐妹
sleep	asleep, sleep	slaap, slapen	coucher, dormir, s'assoupir, se coucher, se reposer, sommeil, somnoler	Schlaf, schlafen	睡眠, 睡觉
slow	slow, slowly	langzaam, langzame	lent, lente, lentement, lentes, lents	langsam	慢
small	compact, little, small, tiny	klein, kleine	petit, petite	klein	小
speak	speak	spreken	communiquer, parler, s'exprimer	sprechen	发言, 表达, 讲话, 说, 说话
supermarket	grocer, grocery, market, store, supermarket	supermarkt	supermarché	Supermarkt	超市
table	table	tafel	table	Tisch	台, 桌, 桌子
take	take	maken, neemt, nemen, zorgen	prendre	mitnehmen, nehmen	取, 带, 拿
taxi	taxi	taxi	taxi	Taxi	出租车
thermo-dynamics	thermo-dynamics	thermo-dynamica	thermo-dynamique	Thermo-dynamik	热力学
tour	tour	excursie, rondleiding, tour, tournee	Tour, tour, visite	Tour	旅行, 游览
water	water	water	eau	Wasser	水
write	write	schrijven	écrire	schreiben	写, 写字

Word	English	Dutch	French	German	Mandarin
yes	yes	ja	oui	ja	对, 是

We selected words that varied in the number of tokens used to represent the words and selected some that were the same as English words. Table C.3 summarizes the word overlap across the different languages. Tables C.4, C.5, C.6 and C.7 summarize the average tokenization lengths of the words in different languages and models. The average number of tokens per word in English is lower than in other languages.

Table C.3: Word overlap between languages

	English	Mandarin	German	Dutch	French
English	75	0	0	14	16
Mandarin	0	75	0	0	0
German	0	0	75	0	0
Dutch	14	0	0	75	8
French	15	0	0	8	75

Table C.4: Llama-3.1-70B tokenization statistics

Language	1	2	3	4	5+	Mean
English	55	15	2	2	1	1.39
Mandarin	35	18	5	3	3	1.80
German	17	30	13	2	2	2.12
Dutch	19	29	11	2	3	2.09
French	20	32	7	3	2	2.03

Table C.5: Gemma-2-27b tokenization statistics

Language	1	2	3	4	5+	Weighted Mean
English	66	6	1	1	1	1.20
Mandarin	50	7	3	3	1	1.41
German	33	24	5	2	0	1.62
Dutch	34	23	4	2	1	1.64
French	39	20	2	2	1	1.53

Table C.6: Mixtral-8x22B tokenization statistics

Language	1	2	3	4	5+	Mean
English	65	3	3	2	2	1.31
Mandarin	0	22	22	4	16	3.52
German	15	30	13	2	4	2.25
Dutch	17	29	12	3	3	2.19
French	21	28	8	5	2	2.11

Table C.7: Aya-23-35B tokenization statistics

Language	1	2	3	4	5+	Mean
English	59	11	3	2	0	1.31
Mandarin	47	11	1	3	2	1.47
German	20	35	7	1	1	1.88
Dutch	29	24	7	1	3	1.83
French	32	25	4	1	2	1.70

C.2.1 Parts of Speech Analysis

In this experiment, we analyze how often a word is first 'selected' in English, for each part of speech. To identify the part of speech, we used `spacy` models [Honnibal and Montani, 2017]. To identify English words, we use `enchant.Dict("en_US")`. We use `nl_core_news_sm`, `de_core_news_sm`, `fr_core_news_sm` and `zh_core_web_sm`. In general, we can use spaces to identify words in sentences. For Mandarin, we use the package `jieba`.

C.3 Logit Lens Quantitative Evaluation

To evaluate whether a word is chosen in English, we use GPT-4o. We considered alternative evaluation procedures. We tested various translation packages but found issues with both word- and sentence-level approaches. When used on a word level, this caused problems with colexification and did not allow for close synonyms often only providing a single translation per word. When using translation on a sentence level, it was difficult to map tokens to each word (due to changes in the sentence structure). We also considered WordNet [Miller, 1994], but it only covers nouns, verbs, adjectives, and adverbs, making it unsuitable for other parts

Table C.8: Part of Speech Abbreviations, Terms, and Examples

Abbreviation	Term	Examples
ADJ	Adjective	
ADP	Adposition	in, to, during
ADV	Adverb	very, everywhere
AUX	Auxiliary	has (done), was (done)
CCONJ	Coordinating Conjunction	and, or, but
DET	Determiner	their, her, some
INTJ	Interjection	ouch
NOUN	Noun	place, thing, idea
NUM	Number	10, 200
PRON	Pronoun	he, she, they
PROPN	Proper Noun	specific name, place
SCONJ	Subordinating Conjunction	that, if, while
SYM	Symbol	
VERB	Verb	see, run

of speech. Ultimately, we chose GPT-4o and manually verified 100 samples to ensure the evaluation was accurate.

We ask GPT-4o to score words as follows:

- 5: An exact translation.
- 4: A close synonym.
- 3: A word with a similar but distinct meaning.
- 2: A word whose meaning is at best weakly related.
- 1: A word whose meaning is not related.

When a word receives a score of 4 or higher, we evaluate the word as chosen in English.

An example of the command we use is:

Dutch Instructions

Below, you will be given a reference word in Dutch and a context (i.e., phrase or sentence) in which the word is used. You will then be given another list of English words or subparts of words/phrases. You should respond with the word from the list that is most similar to the reference word, along with a grade for the degree of similarity. Special Note on Contextual Translations: If an English word could form a common phrase or idiomatic expression that accurately translates the reference word, it should be rated highly. For example, if a phrase like "turned out" perfectly matches a Dutch verb, the word "turned" alone would receive a high score due to its idiomatic fit. Special Note on Tenses: Do not penalize for different tenses. For example, the word 'want' matches 'wilde' and should receive a 5.
Degrees of Similarity: Similarity should be evaluated from 1 to 5, as follows:
5: An exact translation.
4: A close synonym.
3: A word with a similar but distinct meaning.
2: A word whose meaning is at best weakly related.
1: A word whose meaning is not related.
Consider the following examples:
Example 1
Reference Word in Dutch: 'waarop' Context: 'Ze had een hekel aan de manier waarop hij zijn'
English Word List: ['hicks', 'mild', 'rut', 'sens', 'spiral', 'hometown', 'how', 'manner', 'van', '101', 'ward']
Analysis: 'waarop' means "on which" in Dutch. The word 'how' is most similar to this in the list, while the other options are unrelated.
Answer Word: 'how'
Similarity Score: 4 - a close synonym
Example 2
Reference Word in Dutch: 'bleek'
Context: 'Ze adopteerde een zwerfdier, maar het bleek een wolf te zijn'
English Word List: ['cup', 'freed', 'freeman', 'laurent', 'turns', 'turned', 'van', '348', 'i', 'ken', 'oms']
Analysis: 'bleek' means "turned out" in Dutch, making 'turned' the most similar option.
Answer Word: 'turned'
Similarity Score: 5 - an exact translation

Dutch Instructions (continued)

Example 3
Reference Word in Dutch: ‘vaas’
Context: ‘Ze schikte een prachtig boeket bloemen in een vaas.’
English Word List: [’tucker’, ’van’, ’container’, ’opp’, ’van’, ’vessel’, ’-g’, ’-t’, ’397’, ’art’, ’as’, ’ed’, ’ion’, ’let’]
Analysis: ‘vaas’ means "vase" in Dutch. The word ‘vessel’ is somewhat similar, as vases are vessels for holding items like flowers.
Answer Word: ‘vessel’
Similarity Score: 3 - a word with a similar but distinct meaning
Example 4
Reference Word in Dutch: ‘werd’
Context: ‘Ze ging geld opnemen bij de bank en werd overvallen.’
English Word List: [’dee’, ’lafayette’, ’bank’, ’bu’, ’herself’, ’kw’, ’met’, ’ramp’, ’return’, ’returning’, ’113’, ’347’]
Analysis: ‘werd’ means ‘was’ in Dutch. None of these words are related.
Answer Word: None
Similarity Score: 1 - a word whose meaning is not related
Example 5
Reference Word in Dutch: ‘vrienden’
Context: ‘Ze bracht het weekend door met haar vrienden in een huisje in de Ardennen.’
English Word List: [‘sag’, ‘sat’, ‘tween’, ‘bro’,‘families’, ‘family’, ‘her’, ‘herself’, ‘mo’, ‘own’, ‘parents’, ‘weekend’, ‘666’, ‘elf’]
Analysis: ‘vrienden’ means "friends" in Dutch. The closest word here is ‘families’, which is weakly related but distinct.
Answer Word: ‘families’
Similarity Score: 2 - a word whose meaning is at best weakly related
The examples are complete. Now it is your turn. The reference word will be in Dutch, and you must find the most similar English word and assess the degree of meaning similarity on a scale from 1 to 5.

The commands for other languages are similar but adapted to provide examples in the language.

C.3.1 Other Language-specific Phenomena

Explicit Text

We observe explicit vocabulary in the latent space of LLMs (examples can be found in Appendix C.3.1). Table C.9 shows the frequency of vulgar words in the latent space, with Llama-3.1-70B showing the highest count. This model is safety-tuned in eight languages [Dubey et al., 2024], including English, German, and French, but not Dutch. This may suggest that explicit terminology is a language- and model-specific feature.

Table C.9: Frequency of explicit words decoded in the latent space across LLMs. Llama-3.1-70B has the highest proportion of explicit terms.

Model	Explicit words (%)				
	ENG	FR	NL	DE	ZH
Llama-3.1-70B	6.25	11.25	18.35	8.13	7.19
Mixtral-8x22B	1.56	3.91	5.21	4.53	10.19
Aya-23-35B	2.50	2.97	3.44	2.89	2.69
Gemma-2-27b	0.00	0.31	0.31	0.39	0.38

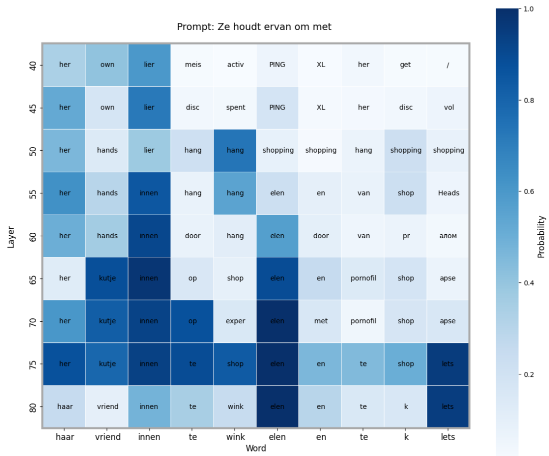


Figure C.1: Example 1: Logit Lens applied to Llama-3.1-70B.

Terms such as kutje (pussy), pornofilm (porn film), lul (dick), and knull (fuck) appear in various contexts, including inappropriate sentences about children. For example, in Figure C.1 during the generation:

> Ze houdt ervan om met **haar vriendinnen te winkelen en te klets(en)** ...
> English translation: She enjoys shopping and talking with her friends ...

We find the explicit words 'kutje' and 'pornofil(m)' when decoding the latent space using the logit lens. This behavior is consistent across other examples (see Figure C.2).

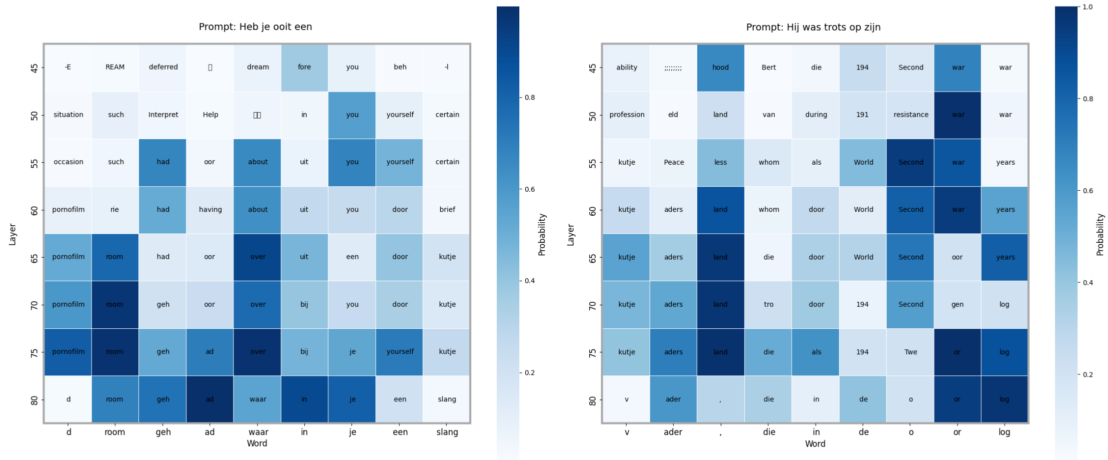


Figure C.2: Logit Lens applied to Llama-3.1-70B.

C.4 Steering

For the steering experiment, we use the LLM-Insight dataset. We compute two steering vectors:

- a topic steering vector: this is a steering vector that captures the intended topic. For example, for the topic 'love, we create a steering vector that is $v_l^t = h_l(\text{love}) - h_l(\text{hate})$, where h_l is the hidden state in layer l.
- a language steering vector: we add a steering vector that captures the intended output language. For example, for the target language Dutch, we can create a steering vector $v_l^l = h_l(\text{Dutch}) - h_l(\text{English})$.
- For each steering vector, we take the difference between sets of sentences containing the topic.

Currently, we consider steering successful if (1) the generated sentence contains the target word and (2) does not lead to output collapse (stuttering). We set the steering vector weights by using a hold-out set of 5 words (50 prompts). We found that 5 was optimal for the topic steering vector, and 10 was optimal for the language steering vector. For the layers, we considered every 5-th layer of the model for the topic steering vector. We considered every 2nd layer for the language vector. We reported the results across the best layers. On average, we found that 20-40 % of layers allowed for successful steering, with English steering vectors being the least sensitive to layer selection.

C.5 Causal Tracing

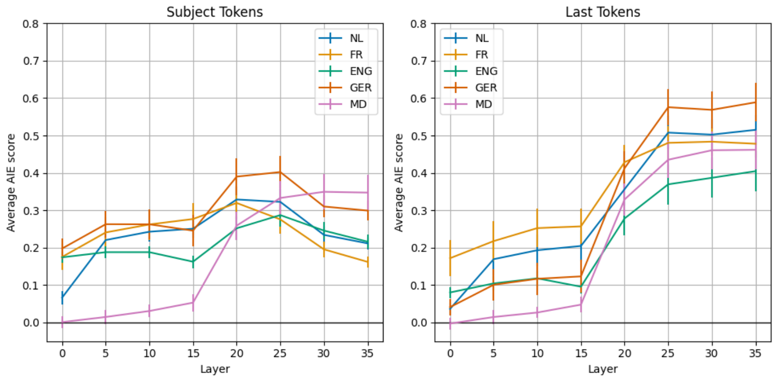


Figure C.3: The causal traces of the city facts in Aya-23-35B.

Figures C.3, C.4, C.5 and C.6 show the causal traces, averaged over different country-city pairs for Aya-23-35B, Llama-3.1-70B, Mixtral-8x22B and Gemma-2-27b respectively. Across all models, we find that facts are generally localized in similar layers, regardless of the language. Two main traces emerge: a mid-layer trace on the subject token(s), which may correspond to entity resolution, and a later trace when attribute recollection occurs (as suggested by Nanda et al. [2023b]). Overall, these plots suggest that facts are approximately stored in the same parts of the model.

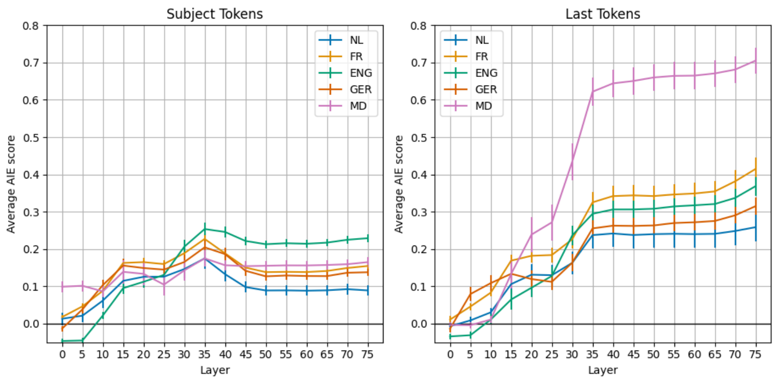


Figure C.4: The causal traces of the city facts in Llama-3.1-70B.

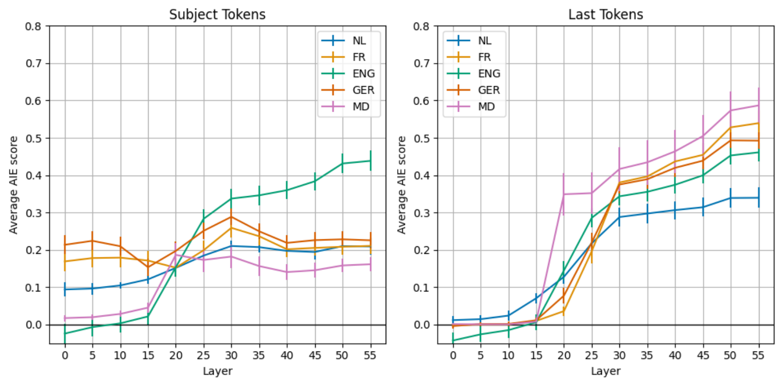


Figure C.5: The causal traces of the city facts in Mixtral-8x22B.

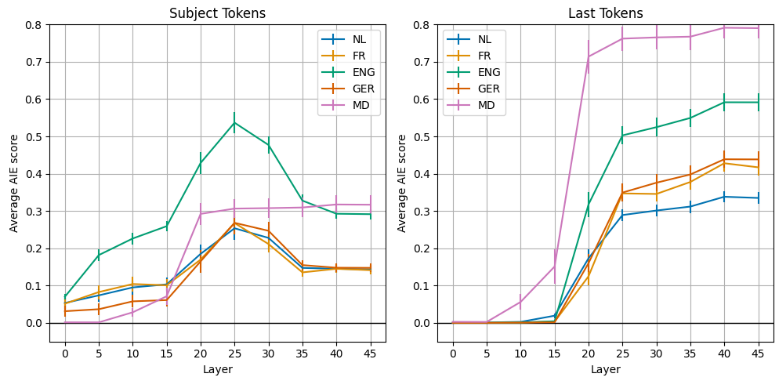


Figure C.6: The causal traces of the city facts in Gemma-2-27b.

C.6 Hidden State Interpolation

These figures can be found in the Appendix of Schut et al. [2025a].

D

Appendix: Leveraging Uncertainty for Realistic Prototypes

D.1 Derivations

Below, we provide a formal derivation of the intuition introduced in Section 5.3.2.

We start by introducing the notation, definitions and assumptions. Let $\mathcal{X}$ be the observation space, and $\mathcal{Y}$ the set of classes. For $x' \in \mathcal{X}$ and $y' \in \mathcal{Y}$, let $p_f(y'|x')$ denote the softmax probability assigned to class y' for input x' by the classifier f.

We assume $0 \cdot \log(0) := 0$; $\log(0) := -\infty$; and $p_f(y|x) > 0, \forall x \in \mathcal{X}, y \in \mathcal{Y}$. We also assume that there exists an $x \in \mathcal{X}$ such that $p_f(y'|x) = 1$.

Proposition 1. Under the above assumptions, $\arg\min_{x'} \ell(f, x', y') \in \arg\min_{x'} h(x')$, where $l(\cdot)$ is cross-entropy and $h(\cdot)$ is predictive entropy.

Proof. For simplicity, we provide the derivation for a single input $x' \in \mathcal{X}$ with target class y'. However, the proof can be easily extended to multiple observations. When generating a CE x' targeted to class y' we minimize the cross-entropy loss, defined as

$$\ell(x', y') = -\log p_f(y'|x'). \tag{D.1}$$

We observe that the function obtains a minimum at $p_f(y'|x) = 1$ and $p_f(y|x) = 0 \quad \forall y \in Y \backslash y'$ for which $\ell(x', y') = 0$. This is a unique minimum because [1] cross-entropy is bounded below by 0 and [2] it is monotonically decreasing in $p_f(\cdot)$.

Predictive entropy, $h(x')$, is defined as

$$h(x') = -\sum_{y \in \mathcal{Y}} p_f(y|x') \log p_f(y|x'). \tag{D.2}$$

If $p_f(y'|x) = 1$, and $p_f(y|x) = 0 \quad \forall y \in Y \backslash y'$, then $h(x') = 0$. This is a minimum because predictive entropy is also bounded below by 0. $\square$

We note that the assumption that there exists an $x \in \mathcal{X}$ such that $p_f(y'|x) = 1$ may not hold in practice. However we find in our experiments that our method performs well even if we allow a tolerance of $p_f(y'|x) = \gamma$, for γ close to 1. This is likely because minimizing the cross-entropy results in a near-zero value for both the cross-entropy and predictive entropy.

D.2 Counterfactual Explanation Generation Algorithm

Algorithm 2 Generating Counterfactuals

Input original observation x; target class y'; ensemble of models $\{f_m\}_{m=1}^{M}$; maximum number of iterations N; minimum confidence of target class γ; perturbation size δ; maximum number of times each feature is changed n; optional: a function that clips the values to a permitted pre-defined range clip.
Output counterfactual x'
$x' \leftarrow x$
$c \leftarrow 0$
Create a n_p vector, where n_p, the number of input features, with all values initialized to zero, $P = 0_{n_p}$ This vector will ensure keep track of the number of times each feature has been changed.
while $\bar{p}(y'|x') < \gamma$ and $c < N$ **do**
    Compute forward derivative: $S(x', y') = \nabla_{x'} \frac{1}{M} \ell(f_m, x', y')$
    Select the most salient feature: $i = \text{argmax}_{i:i \in P, P[i] < n} S(x', y')[i]$
    Update the most salient feature by δ: $x'[i] = x'[i] + \text{sign}(S(x', y')[i]) \cdot \delta$
    Clip the counterfactual so that it stays within the pre-defined range: $x' = \text{clip}(x')$
    $P[i] \leftarrow P[i] + 1$
    $c \leftarrow c + 1$
end while
return x'

D.3 Adversarial Training

Recent work in adversarial literature has linked adversarial robustness to improving model interpretability [Tsipras et al., 2018]. Improving adversarial robustness can be achieved through adversarial training, corresponding to minimizing the loss:

$$\min_{\theta} \mathop{\mathbb{E}}_{p_\theta(y|x)} \left[\max_{\delta \in \Delta} \ell(x + \delta, y|\theta) \right], \tag{D.3}$$

where θ are the model parameters, ℓ is the loss function, x is the original input, y is the original class, δ is a perturbation, and Δ is set of possible perturbations.

In practice, adversarial training is often implemented by augmenting the dataset with adversarial examples during training. These additional images ensure that the model does not focus on noise when learning features for classification. Thus, the model is more likely to learn features that are not noise, and therefore are more interpretable [Tsipras et al., 2018, Ilyas et al., 2019]. This means that adversarial training can also be used to improve the performance of models, outside of the adversarial literature setting.

Further, Chalasani et al. [2020] show a connection between feature-attribution explanations and adversarial training, finding that it leads to more sparse and stable explanations from [1] an empirical perspective for image data and [2] a theoretical perspective for single-layer models.

Lastly, adversarial training can be used to improve uncertainty estimation. Lakshminarayanan et al. [2017] show that adversarial training is a computationally efficient solution for smoothing the predictive distribution, and can improve the accuracy and calibration of classifiers in practice.

The aforementioned work motivates the use of adversarial training to generate more interpretable, stable explanations.

D.4 Evaluation

We omit IM2 evaluation as we found that it failed to pass a sanity check that compares the metric for in- and out-of- distribution images. We perform the check using MNIST, as it made it easier to visually verify test results.

We use MNIST as in-distribution data, and EMNIST as out-of-distribution data. This means that the autoencoders are trained on MNIST images. For the 'in-distribution' CEs, we take MNIST training data in the target class – these images can be considered as 'gold standard' CEs. For the 'out-of-distribution' CEs, we select a random image from EMNIST. We expect to see a clear different in the IM2 scores for the in- and out-of-distribution data as we are comparing the gold standard CEs with random images from a different dataset.

However, the right-hand plot of D.1 demonstrates that the IM2 scores for out-of-distribution did not differ significantly than IM2 scores of in-distribution data at a 5% significance level. On the contrary, for IM1, we find a significant difference (measured using a t-test) for in- and out-of distribution IM1 score at a 5% significance level. Visually, the difference can be observed in the left box-plot in Figure D.1.

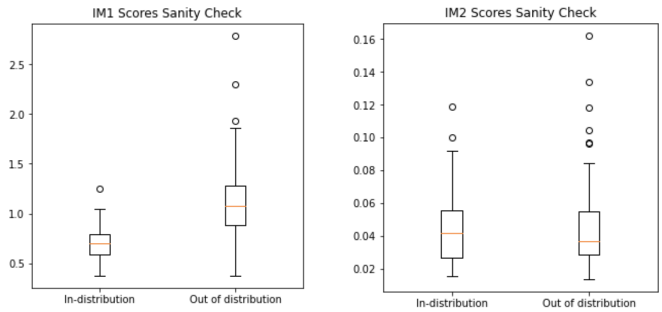


Figure D.1: IM1 and IM2 scores for in-distribution data versus out-of-distribution data

D.5 Experiment Setup

Below, we summarize the experiment setup for the different experiments. All experiments are implemented in PyTorch. The code repository contains the instructions for reproducing each result and generating example CEs, alongside the details of the environment setup including the versions of dependencies.

D.5.1 MNIST

Dataset Configuration For our method and JSMA we normalize the inputs to $[0, 1]$. For Van Looveren and Klaise [2019] we normalize to $[-0.5, 0.5]$, as suggested. We train the classifier on the training set, and generate CEs on the test set. We tune the hyperparameters of our method by generating CEs on the training set. We did not tune the hyperparameters of Van Looveren and Klaise [2019], as we could select them from the original paper.

For each point in the test set, we select the target class for the explanation by randomly selecting from a set of classes specific to the class of the input image. This allows us to avoid selecting target classes which are difficult for a particular input class, for example transforming a 6 into a 2. For a complete list of the possible target classes for each input class, see the code release.

Model Architecture We use a three-layer MLP with 200, 200 and 10 nodes per layer, respectively. The first two layers have ReLU activations and batch normalization after the activation. The final layer has a softmax activation. We use an ensemble of 50 models.

Optimisation We train the network using mini-batches of size 128 and Adam [Kingma and Ba, 2014] (with default PyTorch hyper-parameters) to optimize the weights. We train the model for 50 epochs.

Our Method: adversarial training We implement adversarial training as follows: For each iteration, perform a training step a (clean) batch of data (X, y) followed by a training step on an adversarially-augmented batch of data. To create the latter, we use FGSM [Goodfellow et al., 2014] to generate adversarial examples X'. The augmented dataset is $(X, y) \cup (X', y)$. We choose $\varepsilon = 0.15$ as the perturbation is large enough to fool a trained classifier 90% of the time (and therefore will improve robustness), however does not change the true classification.

Our Method: hyperparameters The maximum number of permitted changes is 5 and the maximum number of iterations is 3,920. The confidence level, γ, is set to 0.99. The perturbation (δ in the pseudo-code) is set to 0.2 (which is equal to $1/n$, where n is the number of times each feature can be changed).

IM1 Evaluation We implement IM1 evaluation following Van Looveren and Klaise [2019], and include the configuration here for completeness. Table D.1 shows the architecture of the all class and single class autoencoders. We train both autoencoders using the mean-squared objective and Adam with a batch size of 128. For the all class autoencoder we train for 4 epochs, for the single class autoencoder we train for 30 epochs.

Results aggregation The results in Table 5.1 are aggregated as follows:

- For each method and dataset pair we choose 100 points from the test set.
- For 10 random seeds
  - Generate a CE for each of the 100 test points
  - Compute the mean IM1 score and L_1 distance
- Compute the mean of the means, and the standard deviation of the means.

Configuration of Van Looveren and Klaise [2019] We use the implementation released by the authors in the ALIBI library [Klaise et al., 2020], largely in its default configuration as given in the documentation. We use the same classifier architecture as above. As the classifier is a white-box model, we follow the ALIBI documentation and use loss function D from Van Looveren and Klaise [2019]. Thus, the hyperparameter configuration is $c = 1$, $\kappa = 0$, $\beta = 0.1$, $\gamma = 100$, $\theta = 100$, max iterations = 2000. As in Van Looveren and Klaise [2019] we use the encoder to find the class prototypes, and following the ALIBI documentation set K = all instances.

D.5.2 Wisconsin Breast Cancer Dataset and Boston Housing Dataset

We generally use the same setup for both the Wisconsin Cancer and the Boston datasets, except where we clearly indicate differences below.

Dataset Configuration For our method and JSMA we normalize the inputs to $[0, 1]$. For Van Looveren and Klaise [2019] we standardize the inputs to mean 0 and standard deviation 1, as suggested. We randomly select 70% of each dataset as the train set, 10% as the validation set, and 20% as the test set. We train the classifier on the training set, tune hyperparameters on the validation set, and generate CEs on the test set.

Model Architecture We use a three-layer MLP with 80, 80 and 2 nodes per layer, respectively. The first two layers have ReLU activations and batch normalization after the activation. The final layer has a softmax activation. We use an ensemble of 20 models.

Optimisation We follow the same setup as for MNIST.

Our Method: adversarial training We perform adversarial training similarly to MNIST. Contrary to MNIST, each feature has a different scale and distribution. A list of perturbation sizes can be found in the code in the configuration file in demo_data, after the word perturbation.

Our Method: hyperparameters The maximum number of permitted changes is 5 and the maximum number of iterations is 150. The confidence level, γ, is set to 0.99. The perturbation sizes (δ in the pseudo-code) are feature-specific and can be found in the file breast_cancer_config.txt in the code.

IM1 Evaluation For the Wisconsin Breast Cancer dataset we follow Van Looveren and Klaise [2019], and include the configuration here for completeness. We use the same procedure for the Boston Housing Dataset. We use the same autoencoder architecture for both the all class and single class autoencoders, and Table D.2 shows the architecture. We train both autoencoders using the mean-squared objective and Adam with a batch size of 128 for 500 epochs.

Results aggregation The same as for MNIST.

Configuration of Van Looveren and Klaise [2019] We use the implementation released by the authors in the ALIBI library [Klaise et al., 2020], largely in its default configuration as given in the documentation. We use the same classifier architecture as above. As the classifier is a white-box model, we follow the ALIBI documentation and use loss function B from Van Looveren and Klaise [2019] for both datasets. For both datasets we use the kd-tree approach to find the prototypes. For the Wisconsin Breast Cancer dataset the hyperparameter configuration is $c = 1$, $\kappa = 0$, $\beta = 0.1$, $\gamma = 0$, $\theta = 100$, $k = 1$, max iterations $= 2000$. We set θ by examining the grid search shown in Van Looveren and Klaise [2019, Figure 7]. We set $k = 1$ as this is the default value in ALIBI. For the Boston Housing dataset we use the same configuration as for the Wisconsin Breast Cancer dataset. The exception are k and θ for which we perform a grid search similar to that run by Van Looveren and Klaise [2019, Appendix A] for the Breast Cancer dataset, based on which we choose $k = 10$ and $\theta = 100$.

We note that the IM1 scores we report for Van Looveren and Klaise [2019] are worse than those in their paper. They report an IM1 score which is not significantly different to ours. However, their results are not directly comparable because we use a test set of 100 points while the original paper uses one of 19 points. To try and ensure our results are correct, we took the following steps:

- Used the implementation of the generation algorithm provided by the authors
- Ensured inputs to the generation algorithm are standardized as in the original paper
- Ensured we use the same hyperparameters as in the original paper
- Visually examined the CEs generated by the method to ensure they were reasonable

We repeated the experiment using a test set of 19 points rather than 100, but this also did not reproduce the results in the original paper.

D.6 Additional Figures

D.6.1 Uncertainty And Calibration

Calibration

Calibration is important as we assume that our classifier outputs $p(y|x)$. We investigate the effect of the number of components in the ensemble by performing a similar experiment to [Lakshminarayanan et al., 2017, Figure 6], and considering accuracy as a function of confidence. We train our network (same configuration as in D.5) on MNIST. We test the network on a dataset formed by combining MNIST and FashionMNIST, where we increase the proportion of the dataset taken from FashionMNIST until the confidence of the classifier falls to the desired level. Ideally the network will have a low-confidence for incorrect predictions. Figure D.2 shows the performance of a single classifer, and ensembles with between 5 and 50 components. We can see that the ensembles perform better than a single classifier as their accuracy is higher.

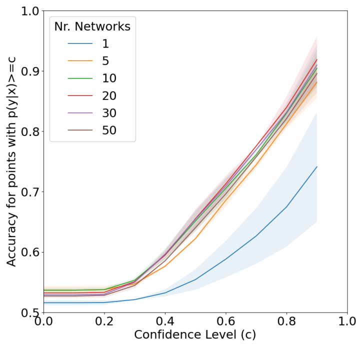
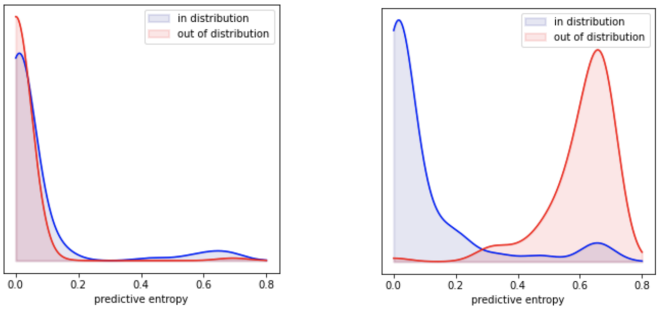


Figure D.2: Accuracy of the model as a function of confidence, for ensembles containing between 1 and 50 models. The solid line shows the mean, and the shaded area shows the 95% confidence level over 10 random seeds. A higher line is better.

Out-of-Distribution Uncertainty

Our method is reliant on the quality of the uncertainty estimates offered by the classifier, thus we investigate the effect of ensembling and adversarial training on the uncertainty estimates. We measure epistemic and aleatoric uncertainty using predictive entropy. Ideally, we observe a high predictive uncertainty for out-of-distribution data and a low predictive uncertainty for inputs similar to the training data.

We consider in-distribution to be test samples from the Breast Cancer Dataset, and out-of-distribution is tabular data sampled from a normal distribution. Figure D.3 shows the effect of using adversarial training and ensembling on predictive uncertainty. While a single model cannot distinguish between in- and out-of-distribution inputs, when using ensembling and adversarial training we see a clear separation of the predictive entropy of both sets of inputs. This goes some way to explain why we see improved performance in the ablation study as we increase the number of models in the ensemble and include adversarial training.



No Adversarial Training or Ensembling

Adversarial Training and Ensembling

Figure D.3: Predictive entropy of the model for in- and out-of distribution inputs. The left plot shows a single model without adversarial training or ensembling. The right plot shows an ensemble of 50 models with adversarial training.

D.6.2 Qualitative Analysis

Figure D.4 shows more qualitative examples of counterfactuals generated by our algorithm on MNIST. The first column shows the original class; the second column shows the counterfactual; the third column shows the proposed change. In the third image in each tuple, black denotes *"painting the pixels black in the original image"*, white denotes *"painting the pixels white in the original image"* and gray denotes *"no change"*. For example, consider the first row in Figure D.4. The goal is to change a 0 into 3. Our model proposes:

- adding an additional stroke in the middle (shown by the white pixels in the third image),
- removing some pixels so that the from the left and right part of the 0 (shown by the black pixels in the their images).

Both changes are required to create a realistic 3. We observe that, in general, our model can grasp high-level changes that are required. These changes suffice for explanatory purposes. However, our model does not capture stylistic properties, which be seen from the examples in the third row, left side in Figure D.4. Further work is required if we want to employ our method as a generative model. An example of a 'failure case' is on left side of the last row – this example is particularly challenging for our model. However, on a high-level we can see that the algorithm roughly understands the required changes.

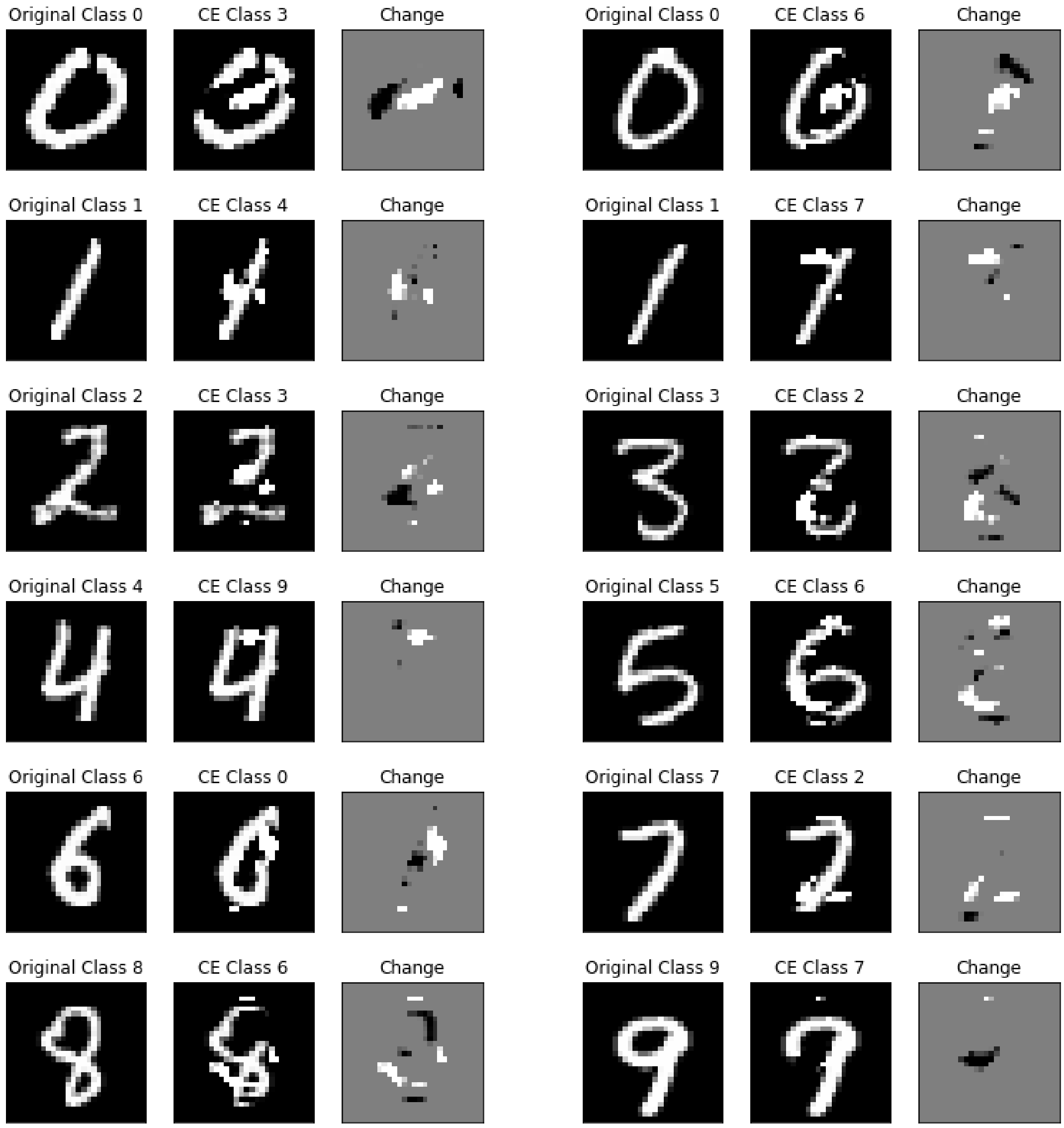


Figure D.4: Qualitative Examples of Generated CE on MNIST. The first column shows the original image, the second column the counterfactual, and the third column the proposed change (to the original image to generate the counterfactual).

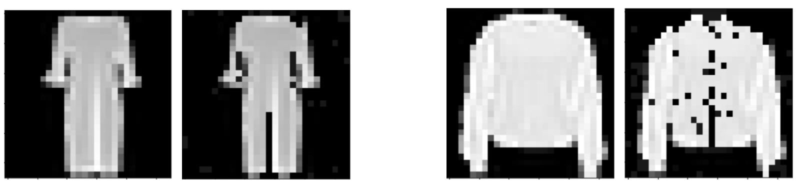


Figure D.5: Qualitative Examples of Generated Counterfactuals for FashionMNIST. For each pair, the left image shows the original images: a dress (left pair) and pullover (right). The right images below show generated counterfactuals: pants (left pair) and a coat (right pair).

Figure D.5 shows more qualitative examples of counterfactuals generated by our algorithm on FashionMNIST. Again, we observe that our model is able to grasp high-level changes required, such as adding a split to the dress or a zipper to the coat.

D.6.3 Feature Analysis

The importance of a feature j can be roughly estimated by $\sum_i x_{i,j} - x'_{i,j}$, where i denotes the observation. Using this, we determine which features are most frequently changed. In Figure D.7, we show examples for MNIST. Here features denote pixels that are changed.

For each triple, the left image shows the average input image from the training dataset – this represents a prototype of the MNIST digit. The blurriness is caused by the natural variation of the class within the dataset. The second image in each triplet is the average CE generated for the target class shown above the image. The third row shows the most average pixel change, i.e. $1/n \sum_i x_{i,j} - x'_{i,j}$. In the third image, black denotes *"painting the pixels black in the original image"*, white denotes *"painting the pixels white in the original image"* and gray denotes *"no change"*.

Overall, the proposed changes appear to be realistic. Let consider a specific example: a counterfactual explanation that changes the predicted class from 0 to 6, as shown in the first row of Figure D.7. In Figure D.6, we highlight the key changes that can be read from Figure D.7. The red circles show the parts that have been *removed* from the original input (left image) to create the counterfactual (shown in the middle image). To change a zero into a 6, we need to paint some pixels black at the top right – these pixels are shown in black in the average change plot (i.e., right image). The green circles show the parts that have been *added* to the original input (left image) to create the counterfactual (shown in the middle image). To change a zero into a 6, we need to add a white diagonal stroke – these pixels are shown in white in the average change plot (i.e., right image).

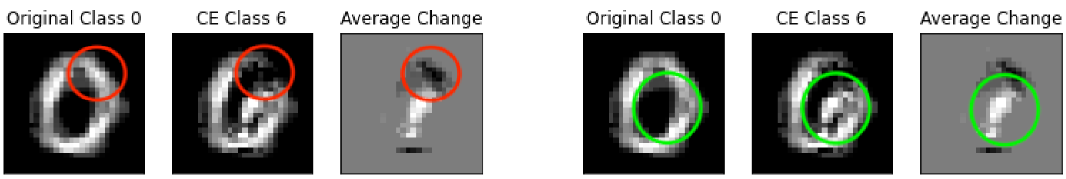


Figure D.6: Average Feature Perturbation for MNIST. In each tuple: the left image shows the average original input image; the middle image is the average CE; the right image is the average change.

Figure D.8 shows the most frequently changed cell nuclei properties, and Figure D.9 shoes the average perturbation size per changed feature. Further interpretation of these results requires expert knowledge, which we intend to look into for future work.

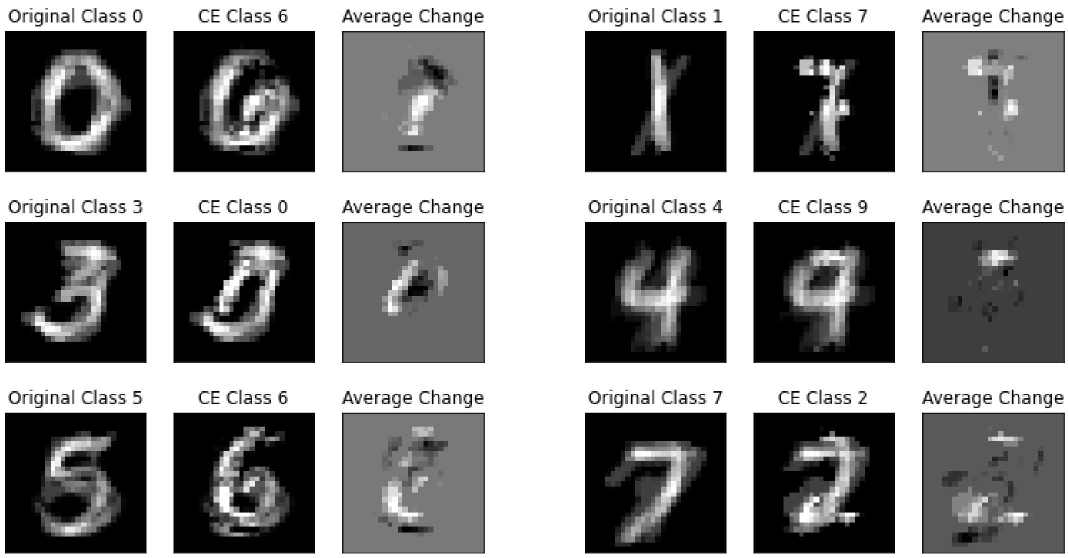


Figure D.7: Average Feature Perturbation for MNIST. The left image shows the average original input image. The middle image is the average CE. The right image is the average change.

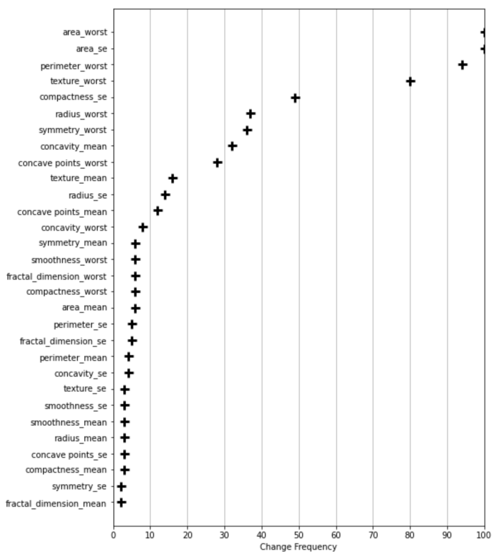


Figure D.8: Frequency at which features are changed in counterfactual explanations

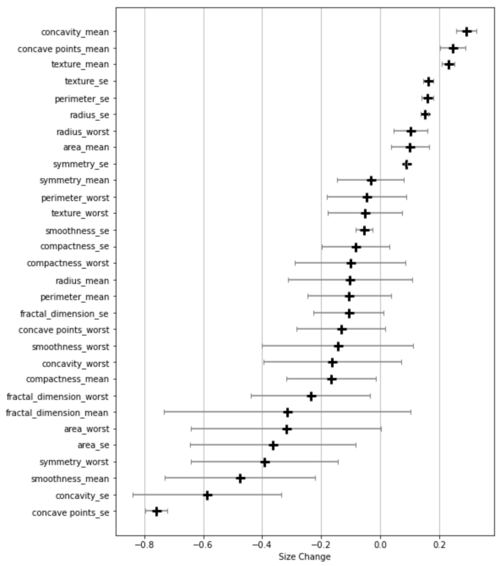


Figure D.9: Average Feature Perturbation (of altered features)

Table D.1: IM1 MNIST autoencoder architectures. Top-down in the table corresponds to input-output in the architecture. The hyperparameters are output channels c, kernel size k, padding p, stride s, scale sc.

All Class	
Layer	Parameters
Encoder	
2D Convolution	$c = 16, k = 3, p = 1$
ReLU activation	
2D Convolution	$c = 16, k = 3, p = 1$
ReLU activation	
Max-pool 2D	$k = 2, s = 2$
2D Convolution	$c = 1, k = 3, p = 1$
Decoder	
2D Convolution	$c = 16, k = 3, p = 1$
ReLU activation	
Upsample	$sc = 2$, mode = "nearest"
2D Convolution	$c = 16, k = 3, p = 1$
ReLU activation	
2D Convolution	$c = 1, k = 3, p = 1$

Single Class	
Layer	Parameters
Encoder	
2D Convolution	$c = 16, k = 3, p = 1$
ReLU activation	
2D Convolution	$c = 16, k = 3, p = 1$
ReLU activation	
Max-pool 2D	$k = 2, s = 2$
2D Convolution	$c = 1, k = 3, p = 1$
Decoder	
2D Convolution	$c = 16, k = 3, p = 1$
ReLU activation	
Upsample	$sc = 2$, mode = "nearest"
2D Convolution	$c = 16, k = 3, p = 1$
ReLU activation	
2D Convolution	$c = 1, k = 3, p = 1$

Table D.2: IM1 VAE architecture for the Wisconsin Breast Cancer and Boston Housing datasets. The top of the table corresponds to the input of the model, and the bottom the output. The hyperparameter n is the number of hidden units.

Layer	Parameters
Encoder	
Linear	$n = 20$
ReLU activation	
Linear	$n = 10$
ReLU activation	
Linear	$n = 6$
Decoder	
Linear	$n = 6$
ReLU activation	
Linear	$n = 10$
ReLU activation	
Linear	$n = 20$

E

Appendix: Concept Discovery in Chess

E.1 Background: How Humans and AZ Play Chess

In this section, we provide further intuition on how humans play chess, and how this relates to AZ's system. This section provides context as to why concepts should explain the policy value network and MCTS to provide a holistic view of chess.

Figure E.1: Simplified Summary of How Humans Play Chess

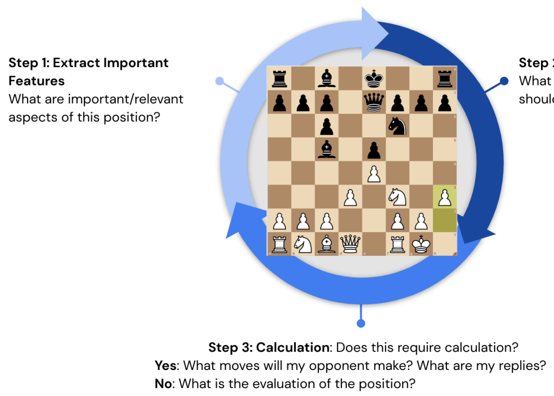


Figure E.1 shows a simplified summary of how human chess players analyse a chess position. They generally ask the following questions:

1. What are the critical aspects of this chess position? E.g., on which side of the chess board do I have more space? How do I want to develop my pieces? What are my opponent's weaknesses?

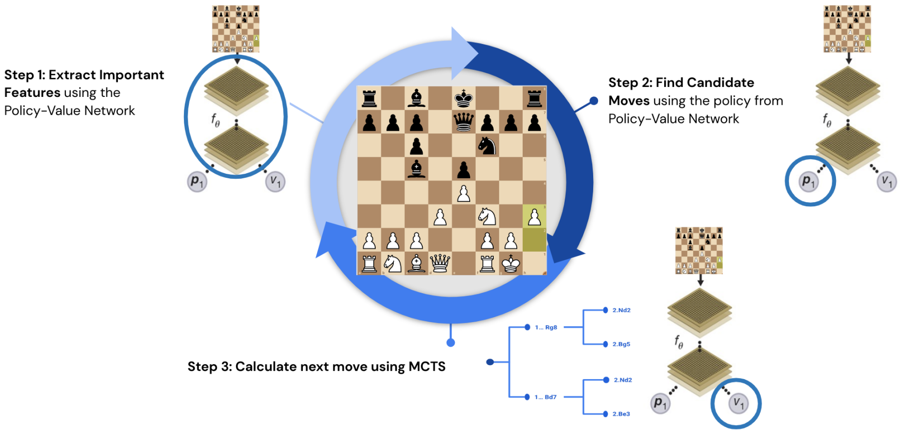
2. Based on step 1, a chess player will find a *couple candidate moves* – actions they could play in the chess position.

3. For each move, they may calculate a likely continuation – i.e., what is the likely sequence of moves to follow?

This process loops until the player has considered all candidate moves, calculated the relevant move sequences, and determined the optimal trajectory.[1] To play well, chess players must understand the important features of a chess position and calculate move sequences to understand the correct evaluation of the chess position. However, there are several different types of chess positions (e.g., endgames or attacking chess positions) where principles alone are insufficient to determine the optimal continuation, and calculation is necessary.

[1]This is a simplified model - in practice, other factors such as time are important.

Figure E.2: Simplified Summary of How AZ Plays Chess



AZ uses a similar approach. In a given chess position, AZ extracts features using the layers in the policy-value network and outputs a policy and value estimate. The policy weighs the different possible actions, and the moves with the most probability mass can be interpreted as the *candidate moves*. Next, the policy is passed on as an input to MCTS, a search algorithm that calculates the optimal move. By drawing parallels between AZ's system and how humans play chess, we highlight the importance of each component of AZ. This motivates our method's design, which incorporates all components of AZ.

E.2 Further Details: Convex Optimisation Formulation for Concepts

This section provides further details on our convex optimisation framework to find concepts. Subsection E.2.1 describes how we set the dynamic concept hyperparameters. Section E.2.3 explains how the convex optimisation formulations were implemented for the different datasets.

E.2.1 Dynamic Concept Hyperparameters

As AZ learnt to play chess through self-play, the latent representations alternates between the player's and the opponent's perspective within a rollout. For concepts, we may want to find a concept that influences a single player or both players. Therefore, we consider two different ways of using rollouts.

1. For a rollout $\{z_t\}_{t=0}^{T}$, we use every other latent representation, i.e., $\{z_{2t}\}_{t=0}^{\text{floor}(T/2)}$, to find concepts for a single player (i.e., for the player to move, or their opponent). These concepts are referred to as 'single'.

2. We use every latent representation to find concepts for both players, i.e. $\{z_t\}_{t=0}^{T}$. These concepts will be referred to as 'both'.

For the rollout depth used in our dynamic concept formulations, we consider $T = 5$ and $T = 10$. For the subpar variations, we required AZ to estimate a minimum value difference of 0.20 and/or a visit count difference of 10% (of the most visited move).

E.2.2 Datasets

We use labelled and unlabelled datasets to construct concept constraints for the convex optimisation formulation. We use a different convex optimisation formulation for each concept we want to discover. Therefore, for each concept, we need a set of chess positions $\mathbb{X}^+$ that contains a concept. We leverage educational resources designed to teach humans chess, which contains themed chess puzzles: positions designed to encapsulate single important concepts and test the degree to which a chess player can deploy them in realistic situations. We go beyond chess puzzles, delving into chess positions arising from different openings and searching for AZ-specific concepts to find new ones.

Factors in Datasets. The datasets we use vary in the following ways:

- **Concept type** e.g., the concepts can be strategic or tactical (see Wikipedia contributors [2023a] and Wikipedia contributors [2023b] for a further explanation on strategy and tactics), or correspond to different periods of the game (opening/middle game/endgame).

- **Degree of human knowledge** some datasets contain human games while others contain AZ's games. The degree of human knowledge (or style of play) may vary across the datasets.

- **Complexity** some concepts are elementary (i.e., can be learned by beginner-level chess players), whereas others are highly complex (i.e., can only be understood by top-level grandmasters).

Below, we briefly describe the different data sets used

1. **Piece dataset** We construct a labelled dataset that contains paired chess positions that either contain or do not contain a chess piece. These simple concepts indicate the presence of a piece - queen, rook, bishop, knight, or pawn. We exclude the concept of 'king' as this piece is always present in a chess game.

2. **Stockfish dataset** We construct a labelled dataset for each concept in the Stockfish engine [Stockfish Community, 2018]. Each dataset contains two sets of chess positions - with and without the concept. Humans use Stockfish concepts in chess position evaluation (such as piece placement, open files, etc.)

3. **STS puzzles dataset** This is a labelled dataset containing 15 different categories (see Corbit et al. [2014] for further details). These puzzles capture different types of strategic themes.

4. **Chess openings** (e.g., concepts in Grünfeld vs. Najdorf vs. Queens Gambit) Openings (i.e., the first couple of moves) determine the pawn structure that acts as the backbone of a chess position – it determines crucial aspects of the game, such as optimal piece placement, square weaknesses and strengths, and more generally, plans. We use the Encyclopedia of Chess Openings (ECO) to create a set of labelled chess opening positions [LiChess, 2023]. For each chess position, we use AZ's MCTS rollouts to construct sequences of the opening moves.

5. **AZ self-play games** We construct an unlabelled dataset. We sample $30,000$ chess positions from AZ's games. To ensure that the chess positions we analyse contain complex concepts, we only select chess positions where two versions of AZ at different points in training select a different move. These versions differ by 61 Elo points.

Table E.1 briefly summarises the different datasets.

Table E.1: Datasets Summaries: from concepts more known to humans (top rows) to AZ (bottom rows)

Name	**Concept Type**		**Type of**	**Complexity**
	Strategic vs. Tactical	**Game Phase**	**Knowledge**	
Piece	N/A	All	Human	Low
Stockfish	Both	All	Human	Varies
STS	S	Middle/End	Human	Medium
Opening	S	Begin	Human/AZ	Varies
AZ	Both	All	AZ	High

The human crafted datasets are used to (1) validate the convex optimisation framework and (2) explain the novel concepts by relating discovered concepts to something humans know (by learning a graph).

E.2.3 Convex formulation for different datasets

In this section, we provide further details on the convex optimisation formulations for each dataset. Unless otherwise stated, we use a 90-10 train-test split for the supervised datasets. In our analysis, we generally consider layers 19, 20, 21 and 23. However, as layers 19 and 23 showed the largest difference in rank compared to human data in our analysis (Section 6.5.2), we use layer these layers to discover the concepts shown to grandmasters.

Piece. We artificially created a piece dataset by sampling chess positions from grandmaster games sampled from ChessBase [2021]. For each concept, we sampled 100 chess positions with the concept to create the set $\mathbb{X}^+$ and then created 100 chess positions without the concept to create the set $\mathbb{X}^-$(by removing the piece).

We formulated the convex optimisation problem as follows. For pair of chess positions, $\mathbf{x}_i^+ \in \mathbb{X}^+$ and $\mathbf{x}_i^- \in \mathbb{X}^-$, we find the corresponding latent representations in layer l to create $\mathbb{Z}_l^+$ and $\mathbb{Z}_l^-$, respectively. Using these latent representations, we can search for the piece presence concept using the following formulation

$$\min \quad \|v_{c,l}\|_1 \tag{E.1}$$

$$\text{such that} \quad v_{c,l} \cdot \mathbf{z}_{i,l}^+ \geq v_{c,l} \cdot \mathbf{z}_{i,l}^- \quad \text{for all} \quad \mathbf{z}_{i,l}^+ \in \mathbb{Z}_l^+, \mathbf{z}_{i,l}^- \in \mathbb{Z}_l^- . \tag{E.2}$$

Note that this is the same formulation as for static concepts more generally (see Section 6.4.1).

Stockfish. Following McGrath et al. [2022a], we extract the concepts encoded in the Stockfish engine. We use the Stockfish engine code to extract a concept value for each position. We sampled $30,000$ chess positions from ChessBase [2021]. For each of these chess positions, we found the chess positions that were in the top 5th percentile (assumed to contain the concept) to construct $\mathbb{X}^+$ and the bottom 95th percentile (assumed not to contain the concept) for the concept score to construct $\mathbb{X}^-$. We randomly paired the chess positions that contained the concept, $\mathbf{x}_i^+$, with chess positions that scored low for the concept $\mathbf{x}_j^-$. Similarly to before, we extracted the latent representations for each concept. Then, using the formulation for static concepts, we found the concept vector using the following formulation

$$\min \quad \|v_{c,l}\|_1 \tag{E.3}$$

$$\text{such that} \quad v_{c,l} \cdot \mathbf{z}_{i,l}^+ \geq v_{c,l} \cdot \mathbf{z}_{j,l}^- \quad \text{for all} \quad i : \mathbf{x}_i \in \mathbb{X}^+,\ j : \mathbf{x}_j \in \mathbb{X}^- . \tag{E.4}$$

Table E.2: Hyperparameter selection for β

	0.05	0.01	0.025	0.05	0.1	0.25	0.5	1.0	2.0
Layer 18	58.42	58.58	58.50	58.42	58.17	58.17	57.42	56.75	55.00
Layer 19	60.17	60.42	60.42	60.17	60.08	60.17	59.75	58.00	55.83
Layer 23	58.83	59.42	59.33	58.83	58.58	56.25	53.92	47.75	40.33

Strategic. We use the strategic test suite to extract strategic concepts [Corbit et al., 2014]. In this dataset, there are 15 different concepts. We omit the 12th concept due to irregular data formatting. The remaining concepts are undermine, open files, knight outposts, square vacancy, bishop vs. knight, recapture, offer of simplification, fgh-pawn, abc-pawn, simplification, king activity, pawn push center, 7th rank, avoiding an exchange [see Corbit et al., 2014, for further details].

Each concept has a set of 100 chess positions, $\mathbb{X}$, and the solution (move) requires applying a strategic concept in each chess position. In our analysis, we run MCTS on the chess position and store the search statistics. We store the optimal trajectory for each chess position $\mathbf{x}_i$, denoted as $\mathbb{X}^+_{i,\leq T}$, where T is the maximum rollout depth. Similarly we select a subpar rollout $\mathbb{X}^-_{i,\leq T}$. To find the subpar rollout, we find a rollout in the MCTS tree with the most visits where (1) the estimated difference in value is at least 0.2, and (2) the visit difference is at least 10%. As in the main text, for $\mathbb{X}^+_{i,\leq T}$ and $\mathbb{X}^-_{i,\leq T}$, we find the corresponding latent representations in layer l: $\mathbb{Z}^+_{i,\leq T}$ and $\mathbb{Z}^-_{i,\leq T}$, respectively. Then, we can find the concept vector using the dynamic concept formulations as follows

$$\min \quad \|v_{c,l}\|_1 \tag{E.5}$$

$$\text{such that} \quad v_{c,l} \cdot \mathbf{z}^+_{i,t} \geq v_{c,l} \cdot \mathbf{z}^-_{i,t} \quad \text{for all} \quad t \leq T,\ i : \mathbf{x}_i \in \mathbb{X}. \tag{E.6}$$

In our analysis, we use a maximum depth of $T = 5$.

Openings. For the openings, we focus on the English, Dutch, Scandinavian, Najdorf, French, Tarrasch, Winawer, Ruy Lopez, Grünfeld, King's Indian, Queen's Gambit Declined and Queen's Gambit Accepted. We consider a subset of all openings due to computational costs. For each opening, we use the encyclopedia of chess openings to find relevant starting chess positions to construct $\mathbb{X}^+$ (see the encyclopedia of chess openings). For each chess position, we ran MCTS to obtain the search statistics and used the formula in Equation 6.14 to find a concept for each opening. Further, we create a concept set $\mathbb{X}^+$ for each ECO index belonging to one of the aforementioned openings.

Other Implementation Details. We solve the convex optimisation problem using a standard solver in the package `cvxpy` [Diamond and Boyd, 2016, Agrawal et al., 2018].

E.2.4 Beta hyperparameter tuning

In this section, we provide the validation values for the concept amplification experiments in Section 6.4.2. For the values in Table E.2, we randomly chose 2 concept sets from the STS dataset and estimated the amplification results for different values of β. Overall, we observe that the results are not very sensitive to the value of β.

E.3 Extra results

E.3.1 Concept Constraint Satisfaction

Table E.3: Constraint Satisfaction Results for Piece Dataset

Concept	layer 19	layer 20	layer 21	layer 23
rook	1.0	1.0	1.0	1.0
knight	1.0	1.0	1.0	1.0
bishop	1.0	1.0	1.0	1.0
queen	1.0	1.0	1.0	1.0
pawn	1.0	1.0	1.0	1.0

In the table below, the concepts are extracted from Stockfish 8's public API. Further details can be found in Appendix A of McGrath et al. [2022a].

Table E.4: Concept constraint Satisfaction Results for Stockfish Dataset. w/b/t denotes White/Black/total difference; mg/eg/ph denotes middle game/ endgame/phased value, where the phased value is the weighted sum between the middle and endgame values, depending on the phase of the game.

Concept		layer 19	layer 20	layer 21	layer 23
bishop	[b,eg]	1.0	1.0	1.0	1.0
bishop	[b,mg]	1.0	0.9	0.9	1.0
bishop	[b,ph]	0.9	0.9	0.9	1.0
bishop	[t,eg]	0.7	0.7	0.4	0.5
bishop	[t,mg]	0.6	0.7	0.8	0.6
bishop	[t,ph]	0.7	0.6	0.7	0.4
bishop	[w,eg]	1.0	1.0	0.9	1.0
bishop	[w,mg]	1.0	1.0	1.0	1.0
bishop	[w,ph]	1.0	1.0	0.5	1.0
imbalance	[t,eg]	0.6	0.2	0.6	0.4
imbalance	[t,mg]	0.6	0.2	0.6	0.4
imbalance	[t,ph]	0.5	0.4	0.5	0.8
king safety	[b,eg]	0.9	0.9	0.9	0.8
king safety	[b,mg]	1.0	1.0	1.0	1.0
king safety	[b,ph]	1.0	1.0	1.0	1.0
king safety	[t,eg]	0.7	0.9	0.6	0.7
king safety	[t,mg]	0.8	0.8	0.7	0.7
king safety	[t,ph]	0.6	0.6	0.7	0.6
king safety	[w,eg]	0.6	0.7	0.6	0.6
king safety	[w,mg]	1.0	1.0	1.0	1.0
king safety	[w,ph]	0.0	1.0	1.0	1.0
knights	[b,eg]	0.0	0.0	0.9	0.0
knights	[b,mg]	0.9	0.9	0.5	0.9
knights	[b,ph]	0.0	0.0	1.0	0.8
knights	[t,eg]	0.6	0.6	0.7	0.7
knights	[t,mg]	0.9	0.5	0.9	0.9
knights	[t,ph]	0.8	0.5	0.4	0.8
knights	[w,eg]	0.9	0.8	0.8	0.9
knights	[w,mg]	1.0	0.8	0.6	0.8
knights	[w,ph]	0.8	0.6	0.3	0.8
material	[t,eg]	0.8	0.7	0.3	0.7
material	[t,mg]	0.6	0.8	0.5	0.6
material	[t,ph]	0.6	0.6	0.6	0.7
mobility	[b,eg]	1.0	1.0	1.0	1.0
mobility	[b,mg]	1.0	1.0	1.0	1.0
mobility	[b,ph]	1.0	1.0	1.0	1.0
mobility	[t,eg]	0.8	0.9	0.8	0.7
mobility	[t,mg]	0.7	0.4	0.6	0.6
mobility	[t,ph]	0.8	0.7	0.7	0.8

Table E.5: Concept constraint Satisfaction Results for Stockfish Dataset. w/b/t denotes White/Black/total difference; mg/eg/ph denotes middle game/ endgame/phased value, where the phased value is the weighted sum between the middle and endgame values, depending on the phase of the game.

Concept		layer 19	layer 20	layer 21	layer 23
mobility	[w,eg]	1.0	1.0	1.0	1.0
mobility	[w,mg]	1.0	1.0	1.0	1.0
mobility	[w,ph]	1.0	1.0	1.0	1.0
passed pawns	[b,eg]	0.8	0.8	0.9	0.8
passed pawns	[b,mg]	0.8	0.9	0.7	0.7
passed pawns	[b,ph]	0.9	1.0	0.9	0.9
passed pawns	[t,eg]	0.8	0.9	0.7	0.7
passed pawns	[t,mg]	0.6	0.7	0.3	0.8
passed pawns	[t,ph]	0.6	0.5	0.6	0.7
passed pawns	[w,eg]	1.0	1.0	1.0	1.0
passed pawns	[w,mg]	0.6	0.8	0.6	0.7
passed pawns	[w,ph]	1.0	1.0	0.9	0.9
pawns	[t,eg]	0.6	0.5	0.4	0.5
pawns	[t,mg]	0.3	0.7	0.5	0.4
pawns	[t,ph]	0.8	0.7	0.7	0.7
phase		1.0	1.0	1.0	1.0
queens	[b,eg]	0.3	0.3	0.4	0.3
queens	[b,mg]	0.9	0.9	0.8	0.8
queens	[b,ph]	0.9	0.9	0.7	0.9
queens	[t,eg]	0.3	0.3	0.4	0.3
queens	[t,mg]	0.6	0.4	0.2	0.7
queens	[t,ph]	0.6	0.5	0.6	0.6
queens	[w,eg]	0.3	0.3	0.4	0.3
queens	[w,mg]	0.8	0.7	0.6	0.9
queens	[w,ph]	0.9	0.8	0.7	0.7
rooks	[b,eg]	1.0	0.9	1.0	0.9
rooks	[b,mg]	1.0	1.0	0.9	1.0
rooks	[b,ph]	0.9	1.0	1.0	0.9
rooks	[t,eg]	0.6	0.5	0.3	0.7
rooks	[t,mg]	0.7	0.8	0.7	0.7
rooks	[t,ph]	0.7	0.5	0.5	0.8
rooks	[w,eg]	1.0	1.0	0.9	1.0
rooks	[w,mg]	1.0	1.0	1.0	1.0
rooks	[w,ph]	0.9	0.9	0.8	0.9
scale factor		0.7	0.8	0.8	0.6

Table E.6: Concept constraint Satisfaction Results for Stockfish Dataset. w/b/t denotes White/Black/total difference; mg/eg/ph denotes middle game/ endgame/phased value, where the phased value is the weighted sum between the middle and endgame values, depending on the phase of the game.

Concept		layer 19	layer 20	layer 21	layer 23
space	[b,eg]	0.3	0.3	0.4	0.3
space	[b,mg]	1.0	1.0	1.0	1.0
space	[b,ph]	1.0	1.0	0.9	1.0
space	[t,eg]	0.3	0.3	0.4	0.3
space	[t,mg]	0.8	0.8	1.0	0.9
space	[t,ph]	0.9	0.9	0.7	0.9
space	[w,eg]	0.3	0.3	0.4	0.3
space	[w,mg]	1.0	1.0	0.9	1.0
space	[w,ph]	1.0	1.0	1.0	1.0
threats	[b,eg]	1.0	1.0	0.9	0.9
threats	[b,mg]	1.0	0.7	0.8	1.0
threats	[b,ph]	1.0	0.8	0.9	1.0
threats	[t,eg]	0.7	0.9	0.7	0.7
threats	[t,mg]	0.4	0.7	0.4	0.3
threats	[t,ph]	0.4	0.6	0.7	0.4
threats	[w,eg]	0.9	0.9	0.8	0.9
threats	[w,mg]	1.0	0.8	0.9	0.9
threats	[w,ph]	1.0	0.9	1.0	1.0
total score		0.7	0.8	0.4	0.6
total	[t,eg]	0.5	0.7	0.5	0.6
total	[t,mg]	0.8	0.6	0.6	0.8
total	[t,ph]	0.8	0.7	0.9	0.8

Further descriptions of the STS concepts can be found at this website.

Table E.7: Concept constraint Satisfaction Results for STS Dataset

Concept	layer 19	layer 20	layer 21	layer 23
Undermine	1.0	1.0	1.0	0.9
Open files	1.0	1.0	1.0	1.0
Knight outposts	1.0	1.0	1.0	0.9
Square vacancy	1.0	1.0	1.0	1.0
Bishop vs. knight	1.0	0.9	0.9	1.0
Recapture	1.0	1.0	1.0	1.0
Offer of simplification	0.9	1.0	1.0	0.8
fgh-pawn	1.0	1.0	1.0	1.0
abc-pawn	1.0	1.0	1.0	1.0
Simplification	0.9	1.0	1.0	0.9
King activity	1.0	1.0	1.0	1.0
Pawn push center	1.0	1.0	1.0	1.0
7th rank	1.0	1.0	1.0	1.0
Avoid exchange	1.0	1.0	1.0	1.0

Table E.8: Concept Constraint Satisfaction Results for Opening Concept Configurations

Hyperparameters	layer 19	layer 20	layer 21	layer 23
single	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
both (k=10)	1.00 (0.00)	0.99 (0.00)	0.99 (0.01)	1.00 (0.00)
both (k=5)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)

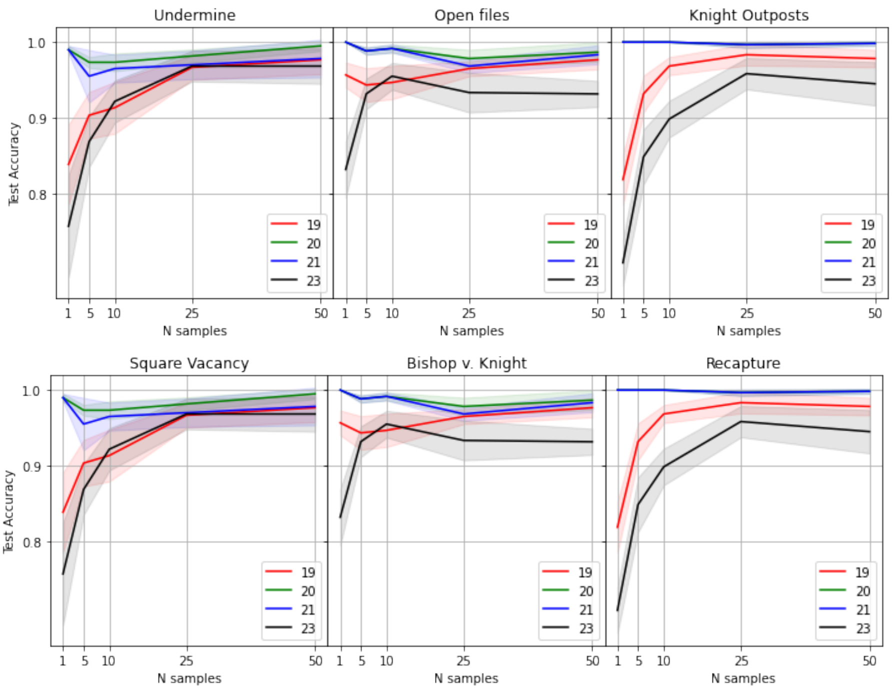
Table E.9: Concept Constraint Satisfaction Results for Opening Concepts

Concept	layer 19	layer 20	layer 21	layer 23
English	1.0	1.0	1.0	1.0
Dutch	1.0	1.0	1.0	1.0
Scandinavian	1.0	1.0	1.0	1.0
Sicilian	1.0	1.0	1.0	1.0
Najdorf	1.0	1.0	1.0	1.0
French	1.0	1.0	1.0	1.0
Tarrasch	1.0	1.0	1.0	1.0
Winawer	1.0	1.0	1.0	1.0
Grünfeld	1.0	1.0	1.0	1.0

E.3.2 STS: Sample efficiency

Figure E.3 shows the result for sample efficiency experiment (Section 6.4.2) on the STS dataset. As with the piece dataset, we find that the convex optimisation method reaches close to full-set accuracy with as little as 10 or 25 data points. However, contrary to the piece dataset, we find that the performance is relatively similar across all layers, and slightly lower for the policy head (layer 23). One reason may be that these concepts are more important for the value than the policy.

Figure E.3: Sample efficiency of convex optimisation framework across 10 seeds for the bottleneck layer (19), value head (20) and policy head (21, 23).



E.4 Human Experiment Setup

Recruitment. We recruited four chess players based on their Elo rating. All participants hold the grandmaster title, and currently/previously the World Champion title; one of our participants is rated 2600-2700, and three are rated 2700-2800.

Experiment Instructions. Each grandmaster was asked to spend two hours on Phase 1, one hour on Phase 2 and two hours on Phase 3. We ask the grandmasters to provide (1) the move they would play or their ranked candidate moves and (2) a thought record – the idea is to capture any thoughts about the chess position. The grandmasters were sent the chess positions to solve at home, in their own time. We explained that the chess positions could vary in nature. The chess positions could be better, equal or worse for the player to move. Similarly, the continuation may require calculation or finding a general plan.

Evaluation. We evaluated how often the grandmasters find the move selected by AZ. Note that if grandmasters made the right move but incorrect reasoning appeared in their free-form comments, we counted this as an incorrect answer.

Prototype Filtering. To ensure the quality of the prototype selection, we filter them according to the following criteria:

- **Quality of the value estimate.** We ensure that the AZ value estimate is close to the correct assessment of the prototype by running self-play and computing the expected score. If the expected score and the value estimate are in concordance, the prototype chess position is kept, otherwise, it is discarded from consideration for the human study.
- **Chess position complexity.** For the concepts to be sufficiently complex to be of interest to the top grandmasters, we use prototypes where the policies of the 512K step checkpoint and fully trained models disagree on their top move. The 512K checkpoint model is 75 Elo points weaker than the final model, and therefore, if the policies differ, AZ learned the continuation during a late stage of training when it was already strong.
- **Solution complexity.** We manually remove *trivial* chess positions where the solution is theoretically known (e.g., present as an entry in pre-computed tablebases such as the Syzygy tablebase [Bojun Guo, 2023]). Tablebases are sets of chess positions where the ground truth evaluation (outcome with ideal play) is known.
- **Reliability.** We reject chess positions where AZ's limited compute budget may lead to an unreliable chess position evaluation (i.e., where we observe abrupt changes in the predicted outcome). Therefore, we require that the evaluation stays approximately consistent (i.e., the predicted outcome (win/loss/draw) does not change) throughout the provided lines.

We did not filter based on the difference in the value or the policy probability mass of the optimal move compared to other moves. The reasons are that (1) AZ's value estimate is noisy, and (2) either filter could remove potentially interesting chess positions. For example, requiring a small entropy and large value estimate difference (between moves) would result in predominantly tactical chess positions, thereby omitting interesting strategic puzzles.

While there was some overlap between study participants in terms of puzzles shown, different participants were shown different concepts.

AZ's calculations. In the second stage of the human experiment, we provide part of the MCTS statistics. We ran MCTS without a depth limit for a maximum of $10,000$ simulations. We pruned the MCTS tree to avoid providing too many lines, or lines that were insufficiently explored. We provided the main line (most frequently visited), second and third moves, ranked according to visits. We did this for depth $t \leq 2$.

E.5 Concept Puzzle Examples

In this section, we provide more examples of concept puzzles, along with the grandmasters' analysis of the puzzles and the provided AZ suggestions can be found in the Appendix of Schut et al. [2025c]. The puzzles are plotted using the `chess` python package [Fiekas, 2023].

Concept Example: Positive Knowledge Transfer

On a high level, this concept appears to be intrinsically related to centre control and improving piece activity. However, a more detailed analysis unveils a nuanced dimension to this concept, as AZ leverages unconventional manoeuvres to achieve these goals. The grandmaster improved performance by +2/4 between Phases 1 and 3, suggesting that this concept was not part of their existing knowledge and is human-learnable.

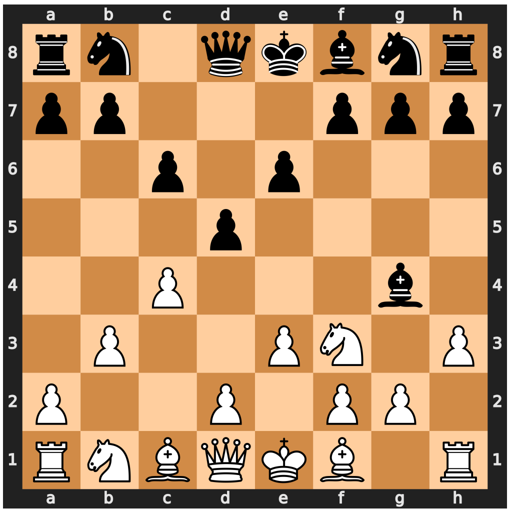
Figure E.4 shows a concept puzzle that was shown in Phase 1. Here, AZ plays the move `5...Bf5` to control the square e4. The grandmaster chose `5...Bh5` while also considering the moves `5...Bxf3` and `5...Qf6`. After seeing the solution, the grandmaster commented

> "`5...Bh5` line looks quite natural ... [however, AZ's move] `5... Bf5` with the concept in mind **is very interesting** as after `8.d4` [the continuation for Black of] `Nbd7-Bd6` is more natural but **`Bb4` is something new**. I was curious about the idea after `11.Bb2 Nd7 12. Bd3` where `h5!?` was probably the point."

We explore the question posed by the grandmaster – what happens after `11.Bb2`? The ideas is

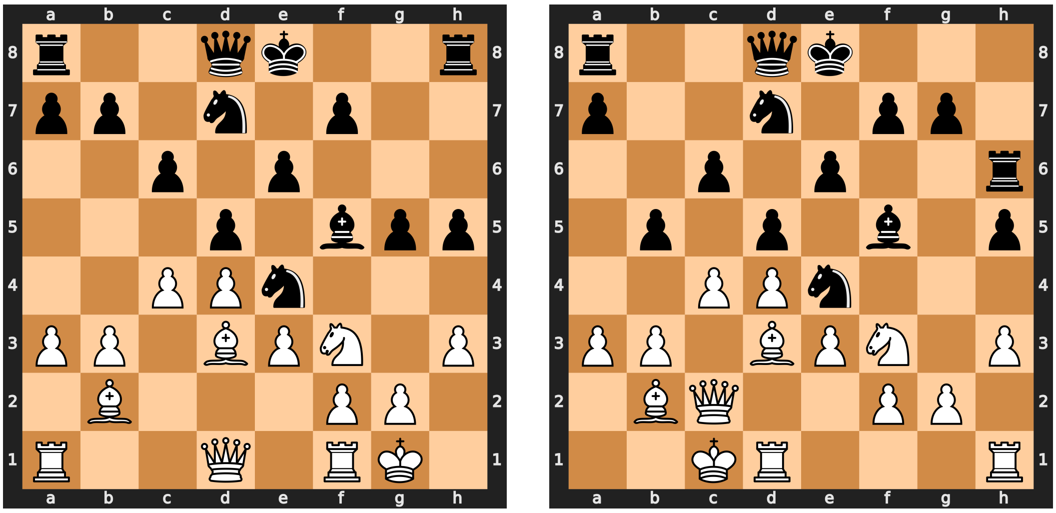
```
5...Bf5 6.Nc3 h6 7.Bb2 Nf6 8.d4 Bb4 9.a3 Bxc3+ 10.Bxc3 Ne4 11.Bb2
Nbd7 12.Bd3 h5 13.0-0 g5!
```

Figure E.4: Concept Puzzle 1: Black is to move.



AZ's calculations: 5...Bf5 (5...Bh5 6.Bb2 Nd7 7.cxd5 cxd5 8.Nc3 a6 9.Rc1 Ngf6 10.g4 Bg6 11.Nh4 Be4 12.Nxe4 Nxe4 13.Nf3) 6.Nc3 h6 7.Bb2 Nf6 8.d4 Bb4 9.a3 Bxc3+ 10.Bxc3 Ne4 11.Rc1 Nxc3 12.Rxc3 Qe7 13.Qc1 0-0 14.Be2 Nd7 Approximately equal

Figure E.5: Digging Deeper into the Concept in Figure E.4. In both positions, White is to play.

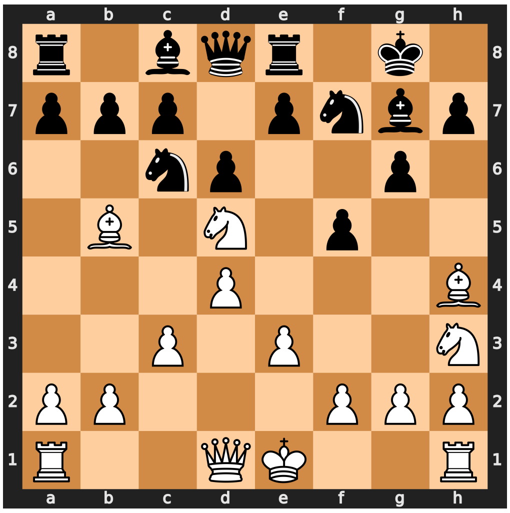


This resulting chess position is shown on the left of Figure E.5. If White tries to castle kingside with 13.Qd2 instead of 13.0-0, Black can pursue a kingside pawn advance:

13.Qc2 Rh6 14.0-0-0 b5!?.

The resulting chess position is shown in the right chess position in Figure E.5. Both continuations are unorthodox; conventional human-designed chess principles emphasise completing piece development, securing the king's safety and maintaining the bishop pair over trading it for a knight, as outlined in King [2000]. However, AZ deviates from these principles favouring a continuation that prioritises a strong control of the centre, space, and piece activity.

Figure E.6: Concept Puzzle 2: White is to play.



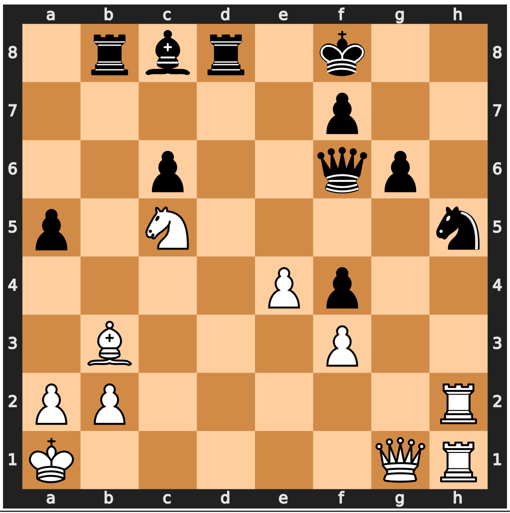
Another puzzle from the same concept is shown in Figure E.6, given in Phase 3. Here, the grandmaster found the best continuation according to AZ: `10.Ndf4` threatening `d5`. The ideas are

```
10...a6 11.Qa4
10...Bd7 11.Bc4 gaining control over the square e6
10...d5? 11.Qb3 and the pawn on d5 is lost
10...Bf6 Black's best option 11.Bxf6 exf6 (12.d5? a6 13.Qa4
then Black has the intermediate move 13...Re4!)  12.Ng1 a6 13.Ba4
Bd7 14.Nge2
```

The knight manoeuvres `10.Ndf4` and `12.Ng1` are against the common rules which advocate for finishing piece development and bringing the king to safety, above further improving a developed piece [King, 2000, Brunia and van Wijgerden, 2021, Hansen, 2021]. As in the previous puzzle, AZ prioritises controlling the centre and piece activity.

The grandmaster missed the idea 12.Ng1, although did appreciate it remarking that "`Ng1` [is] quite nice actually, [knight] on `h3` is gone, and then we probably go for `h4` at some point."

Figure E.7: Queen Manoeuvre. White is to move.



AZ's calculations: `37.Qc1 Kg7 (37...Rb5 38.a4 Rb4 39.Ka2; 37...Qe5 38.Qc4 Be6 39.Nxe6+ fxe6 40.Qxc6) 38.Re1 Qe5 39.Rc2 Rb4 40.Ba4 Qd6 41.a3 Rd4 42.e5 Qd5 43.Bxc6 Qxc6 44.Nb3 White is better`

Informative Puzzles

In some puzzles, we observe manoeuvres from AZ. We provide a few examples here.

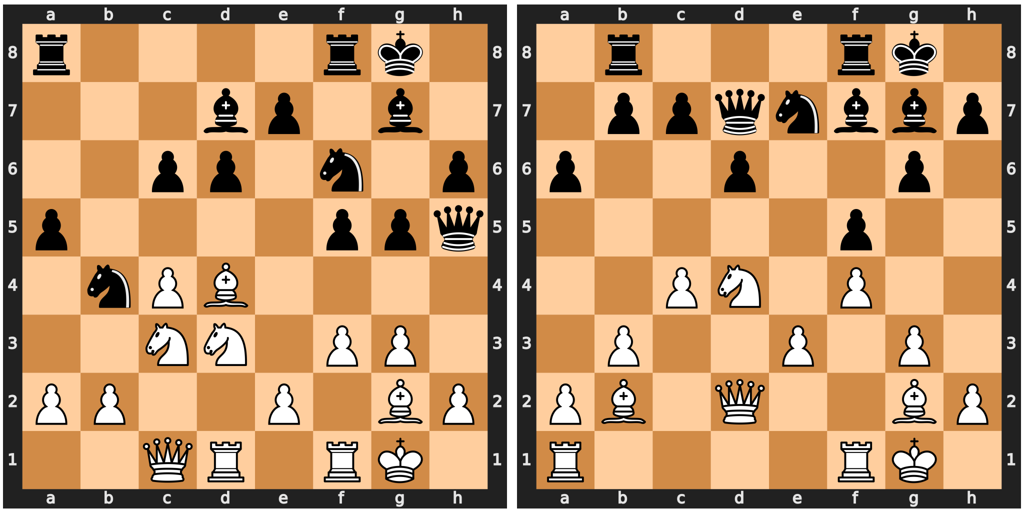
In the chess position in Figure E.7, AZ plays `Qc1` with the idea of manoeuvring it to `c4`. Most human chess players would find this idea unconventional, as White's pieces seem to be active on the kingside. However, there is no way to break through Black's position. AZ's idea is the only way to maintain an advantage. The plan is to re-position the pieces to the queenside, with ideas like `Re1, Ba4 and e5`.

When analysing this position (and only spending a fixed amount of time), the grandmaster misted the idea and opted for `Rxh5`, which was the only way for White to equalise according to the grandmaster. We speculate that the difference between AZ and humans is because AZ is more flexible in changing its plan. In this position, humans are likely primed to continue playing on the kingside.

The puzzles in Figure E.8 correspond to the same concept. In both positions, AZ uses tactics to obtain a positional advantage by maintaining a space advantage. In the left puzzle, the best move is `21.Nc5`, which stops Black from advancing their *c*5 pawn to control the center. The tactic behind the idea is `21.Nc5 dxc5 22.Bxf6 Bxf6 23.Rxd7`.

In the puzzle on the right of Figure E.8, Black plays `18.Rad1` which is prophylactic against `18...b5` as White has tactic `19.c5 dxc5 20.Nc6`. Both of these continuations were found by the grandmaster, who appreciated the importance of `Nc5`. The grandmaster explained that they took a long time to analyse this position

Figure E.8: Positional Tactics. White is to move in both positions.



AZ's calculations (left): `21.Nc5 dxc5 (21...Bc8 22.N5a4 Be6 (22...c5 23.Bxf6 Bxf6 24.a3 Ra6 25.f4) (22...f4 23.Nb6 Rb8 24.Nxc8 Rbxc8 25.a3 Na6 26.e3) 23.a3 Na6 24.c5 dxc5 25.Nxc5 Nxc5 26.Bxc5) 22.Bxf6 Bxf6 23.Rxd7 Bd4+ 24.Kh1 f4 (24...Qe8 25.Rb7 Rf7 26.f4 Qc8 27.Rb6 Qd8 28.Na4) 25.Ne4 White is slightly better`

AZ's calculations (right): `18.Rad1 Rbe8 (18...b5 19.c5 Rbe8 20.Ba3 Nd5 21.Rfe1 b4 22.Bxb4 Nxb4 23.Qxb4) 19.Nf3 Bxb2 20.Qxb2 Nc6 21.Rfe1 h6 22.Qc3 Re7 23.b4 Rfe8 24.a4 White is better`

as they found it complicated. While recognising Black's threat of c5 (followed by `Bc6` or `Nc6`), they first explored several other options, including moves such as `a3`, `c5`, `Bf2` or `Na4`. However, after exploring other moves, they found `Nc5`. The grandmaster commented that the idea was very strong and 'by far the best move'.

In this puzzle, the grandmaster opted for the tactical move `Bxh7` but also considered quieter moves such as `Re3`, `Be4` or `Be1`. When analysing this position, they commented

> "This is tricky, if White decides to protect the pawn, it's clearly better due to the weakness in Black's structure, but somehow getting addicted to more forced attacking lines. `Bxh7` is hard to figure out, maybe `Kxh7` [followed by] `Qh5-Qf7` then `Rd3`, `Bxg2` ... [I] didn't calculate until clear much better position, but thought even with some play, h-file the long problem, maybe certain chances [to win]."

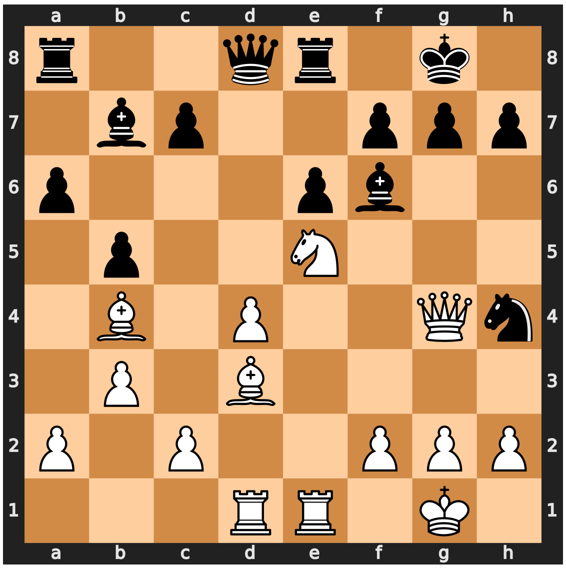
However, this sequence ends in a draw, and AZ instead opts for `f3`, maintaining the advantage for White. When reading AZ's analysis, the grandmaster commented:

> "Wow, it's a completely positional play. Well, my decision to [sacrifice] is too emotional. To be honest, this choice `f3` ... [makes] sense as Black does not have any breakthrough idea, so if White successful [in] controlling both `c5` and `e5` square then its clearly much better. [My conclusion is] technically strong but again within [the] human perspective."

Here, we see a difference in style between humans and AZ. AZ opts for a slower, longer-dominance play in chess positions where grandmasters tend to consider more forcing sequences.

Human vs AI Play: AZ opts for Less Forced Lines than Humans

Figure E.9: White is to move.

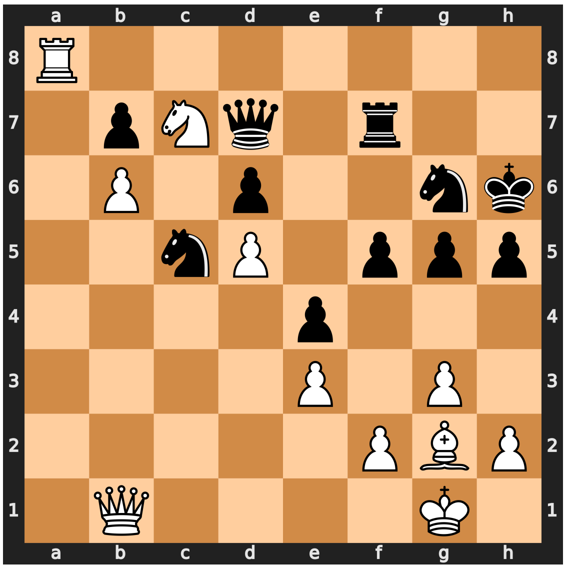


AZ's calculations: `18.f3 (18.Bf1 Ng6 (18...Nf5 19.c4 g6 20.cxb5 axb5 21.Bxb5 h5 22.Qh3 Rxa2 23.Bxe8 Qxe8) 19.Bd3 Nh4 (19...Bh4 20.Bxg6 hxg6 21.Rd3) 20.f3) 18...Ng6 19.Bc5 Bh4 20.Bxg6 hxg6 21.g3 White is better`

Difficult or Non-instructive Puzzles

There may exist concepts that are intrinsically hard to understand and learn for human chess players due to differences in ways of abstract thinking, overall capabilities, and their computational budgets. The example in Figure E.10 highlights that humans and AI have different computational capacity, allowing AZ to make moves that appear risky to humans.

Figure E.10: AZ takes on more risk than humans due to computational capacity. White is to move.



AZ's calculations: `31.Qa1 f4 (31...Qe7 32.Re8 Qf6 33.Qxf6 Rxf6 34.Rd8 Ne5 35.Ne8) (31...h4 32.Re8 f4 33.exf4 gxf4 34.Bxe4 Nxe4 35.Rxe4 Qf5 36.Qe1 Ne5 37.Qe2 hxg3 38.hxg3 fxg3 39.fxg3) 32.exf4 gxf4 33.Ne6 Nxe6 34.Bh3 Qb5 35.dxe6 Rg7 36.Ra5 White is slightly better`

In the puzzle in Figure E.10, AZ plays `Qa1` to activate the queen. This move requires calculating carefully to ensure that Black has no counterplay due to an attack on the kingside. As such, humans may perceive this move as risky, and it was not chosen by the grandmaster. When seeing AZ's calculations, they remarked

> "I would be really worrie[d] to keep the queens on the board because of the threat with `f4` but AZ has a tactical solution. `33.Ne6 Nxe6 [34.]Bh3` is a very nice idea which is quite hard to spot. Black should probably stay still and try to hold with `31...h4 [32].Re8 Re7`."

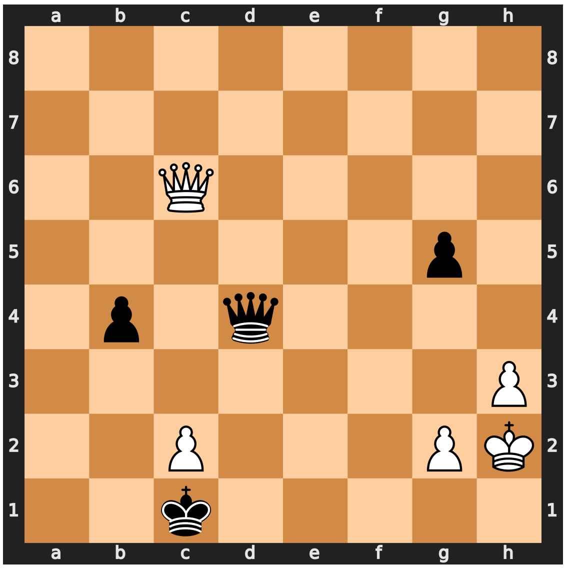
Here, we see that humans are more risk-averse than AZ. This is logical, given that AZ has a much larger computational capacity and can calculate more/deeper than humans can to more accurately assess the chess position (and thereby take on less risk).

Differences in Motivation in Human and AI Play

The next example shows that AZ and humans play chess with different motivations. AZ forces the draw with `g4`. Upon seeing AZ's calculations, the grandmaster commented:

> "... this is a very clear and important theme to understand. So, g4, the move it proposes, in a practical sense it's a very big move, because you

Figure E.11: AZ simplifies the chess position for the draw whereas humans would continue to try to win. Black is to move.



AZ's calculations: `58...g4 59.hxg4 (59.Qc7 gxh3 60.Kxh3 Qc3+) 59...Qxg4 60.Qc5 Qh4+ 61.Kg1 Qe1+ 62.Kh2 Qc3 (62...Qh4+) Draw`

> see, in such situations, the engine already knows the final result. For engine it doesn't matter which move it plays because it calculated it's a draw, but g4 is basically forcing it. After `g4` Black has no winning chances, but otherwise I have a feeling that after Black plays let's say `Qd2`, it's not ... easy practically for White to make the draw. For an engine it's ok, but practically no one would play it because `g4` is basically offering a draw - and with other moves Black is running zero risk, yet has practical chances to win the game if White makes a mistake. An engine doesn't understand the concept of practical play - while this is a draw, it's not an easy draw for White. `g4` is one of many moves leading to a draw, but in a practical sense the worst one as it gives Black zero chances to win. So that is my understanding, that it's not the objective best move. Practically definitely a wrong move."

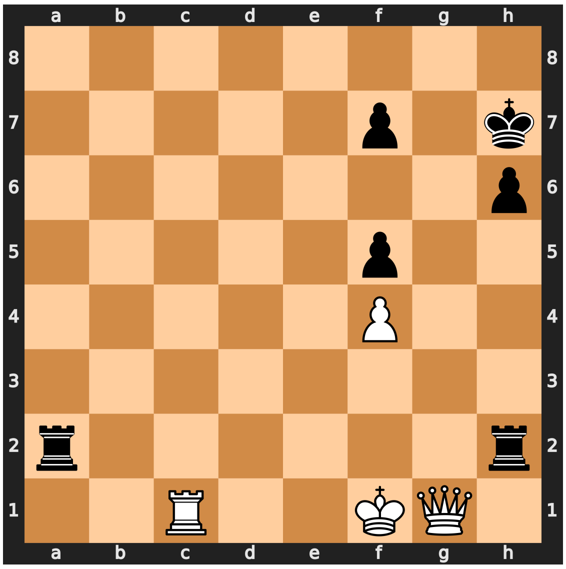
This underscores a fundamental difference between AZ's playing style and human's playing style. AZ was trained to obtain the *expected outcome* without an explicit term in the loss function, encouraging it to win. The incentive to find the best move comes from the exploration and move selection criteria in MCTS. Further, AZ assumes an equally strong opponent. For AZ, there is no difference between different equalising moves, even if one move requires a much more precise sequence of moves to equalise. In contrast, humans assume that their opponent may make suboptimal moves. As human chess players play competitively (i.e., their goal is to maximise the outcome), they will try to leverage these chances. This example highlights how the difference in objectives and assumptions may lead to different behaviour of AZ and humans when playing chess.

Shortcomings of Method for Generating Prototypes

In this puzzle, AZ chooses `Rd1` whereas the grandmaster wanted to play `Re1` or `Ke1`. When seeing AZ's calculations, the grandmaster commented:

> "Drawing position? At first trying to find winning moves for White, but really didn't see any plan to make improvements. In the meantime, considering the possibility for Black to push h pawn to h3, maybe tiny chances, its better to plan `Ke1-Qf1-Qf3` at the beginning, or moving the rook to e1 with the idea `Re7`, forcing ... `Ra1` check then White rook retreat to `e1`, ... [draw by] repetition."

Figure E.12: White is to move



AZ's calculations:

```
65.Rd1 Rh4 (65...Rac2 66.Rb1 Rh4 67.Ra1 Rxf4+ 68.Ke1 Rg4 69.Qa7
Kg7) 66.Ke1 (66.Rb1 Rxf4+ 67.Ke1 Re4+ 68.Kd1 Rf4 69.Qh1 Rff2)
66...Rhh2 67.Rc1 Rag2 68.Qf1 h5 69.Rc7 Kg7 Draw
```

This puzzle can be seen as a shortcoming of our method for finding prototypes. We only filter positions based on the criteria described in Section E.4; however, this position does not fall under one of our categories. This puzzle is not informative for humans as there are many other viable options. As such, it is more difficult to understand the concept from the sequence of moves.