Preserving Knowledge During Learning

Lukas Braun

University of Oxford
Department of Experimental Psychology

Submitted for the degree of Doctor of Philosophy

October 2024

"All change is a miracle to contemplate,
but it is a miracle which is taking place every instant."

— Henry David Thoreau, Walden

Contents

Acknowledgements

I would like to acknowledge the immense privilege of having had the support and lucky starting conditions which allowed me to pursue a PhD in a world in which many face struggles and hardships that make such pursuits unattainable. I am deeply grateful and have valued this opportunity every day.

It took my initiative to begin this journey, but it took many for me to arrive.

I feel immensely grateful to my supervisors, Andrew and Chris, who were kind, understanding, and always aware of their own limits. They consistently had a keen understanding of my work, were quick and reliable in their support, and unconditionally stood by me both as a person and in my journey to becoming a scientist. Andrew and Chris, you are true gems, and I feel honoured to have had the chance to work with you. Thank you for believing in me, even during times when I struggled to believe in myself.

This work would not have been possible without the generous funding provided by a Medical Sciences Graduate School Studentship, jointly funded by the UK Medical Research Council and the Department of Experimental Psychology Oxford.

I would like to thank the seemingly endless number of helping minds and hands around me for their inspiration, constructive criticism, editing, feedback, and moral support. Clémentine, thank you for sharing the joys and struggles of writing a paper with me. Thank you Steph, I had such great fun stitching knowledge with you.

My dear Vicaroos, you were the shelter and home for my soul. Anna, you were my MSG. Lydia, you were my shooting star. Thank you for the love, support, and the time we shared. Tyler, you were the raft when my ship was sinking. My dear brothers, I love you. Dear Mum and Dad, there are no words to express my gratitude for your unconditional love and support. I would not be here without you.

Abstract

Humans and non-human animals live and learn in a continuous stream of information, and rarely experience catastrophic interference between currently and previously acquired knowledge. In contrast, artificial neural networks suffer from catastrophic forgetting and typically remain static once trained. In this work, we provide an analytical study of continual learning and its effects on neural representations in deep linear networks. We present a mathematical analysis of the solution manifold, illustrating how different areas result in different neural representations and computational properties; and obtain an exact expression for the continual learning dynamics under gradient descent. Based on these analyses, we derive exact continual learning rules for categorical and relational knowledge, and demonstrate how common replay- and regularisation-based algorithms either require exploding memory and computational capacity or fail to learn representations that capture the structure of experiences. Our theory predicts that any successful continual learning rule requires representational drift of previously acquired knowledge, and it makes testable predictions about the geometry and temporal evolution of drifting neural representations. Empirically, we confirm this prediction by studying relational learning in humans through functional magnetic resonance imaging, showing that a single experience can cause systematic reorganisation of representations to preserve previously acquired knowledge. Taken together, our results provide a mathematical framework for studying neural representations during continual learning in both biological and artificial neural networks.

List of Acronyms

ANN artificial neural network

dmPFC dorsomedial prefrontal cortex

ECL exact continual learning

MDS multidimensional scaling

PPC posterior parietal cortex

RSM representational similarity matrix

SVD singular value decomposition

List of Tables

List of Figures

General Introduction

The ability to continuously acquire new skills and knowledge is a hallmark of intelligent behaviour. Crucially, continual learning happens throughout a lifetime and without major interference with previously acquired knowledge. This is achieved even though experiences are being processed by shared sensory streams and encoded within the same nervous system. Despite the large body of research on human and animal learning and learning in artificial neural networks, a quantitative theory that explains and can make accurate predictions about continual learning in minds and machines is missing.

Theoretically, this thesis presents exact mathematical analysis of the solution manifold (Chapter 3) and exact continual learning dynamics (Chapter 4) in deep linear networks. These results are then used to derive provably exact continual learning rules for categorical and relational knowledge (Chapter 5). Empirically, the theoretical results are then tested against changes in neural representations during relational learning in humans (Chapter 6).

1.1 A connectionist perspective on action and cognition

Action and cognition emerge in the nervous system: an intricate network of highly interconnected neurons. Therefore, unravelling the functional underpinnings of the distributed and parallel computations within a nervous system are key to understanding cognition (Thagard, 1986; Rumelhart, Hinton, & McClelland, 1986).

However, the sheer size and complexity of these neural networks, even in tiny organisms like *Caenorhabditis elegans* (Cook et al., 2019) or *Drosophila melanogaster* (Scheffer et al., 2020), render reverse engineering the functionality of the nervous system a tremendous challenge. Data has to be acquired from delicate organic tissue and then mapped to function across the spatial scales of nanometer-sized synaptic processes to entire brains and temporal scales that range from millisecond-long individual neuronal spikes to processes that can last a lifetime (Sejnowski et al., 2014). This challenge is further aggravated by the fact that as new experiences unfold, new knowledge and skills are acquired, and complex processes renew and adjust the neural tissue, the underlying structure and function of the nervous system is constantly changing. So how can we understand how this intricate and ever changing network gives rise to action and cognition?

1.1.1 Early developments of connectionism

Already in the 40s McCulloch & Pitts (1943) proposed that the functionality of nervous systems could be understood through the use of logical calculus. Based on the all-or-none response of biological neurons, they studied the computational properties of feed-forward and recurrent networks in which artificial neurons compute binary outputs based on weighted inputs from the environment or other neurons. They analytically showed that such networks can calculate any first-order logic function. A key assumption in their analysis was that weights do not change, meaning that no learning occurs. Thus, this left the question unanswered of how such networks can acquire a certain computation from experience, a challenge that any nervous system faces. In his seminal work, Rosenblatt (1958) formalised ideas from Hebb (1949) and Hayek (1952) into the perceptron learning rule, which can be used to optimise the synaptic weights of a single artificial neuron with binary output. Given a set of inputs and target outputs, the learning rule iteratively adds

or subtracts a fraction of a given input to the synaptic weights, based on whether its output was correct or incorrect. While the perceptron learning rule can be used to learn any set of linearly separable patterns, and is guaranteed to converge to a solution in a finite amount of training steps (Novikoff, 1962), single binary neurons are incapable of learning patterns that are not linearly separable like XOR (Minsky & Papert, 1969). Although it was already known from the work of McCulloch & Pitts (1943) that by combining multiple binary neurons, any first-order logical function, such as XOR, could be implemented, the question of how such a network of multiple neurons could learn the synaptic weights that would implement these functions remained unresolved. The conceptual framework of Parallel Distributed Processing (Rumelhart, Hinton, & McClelland, 1986) was pivotal in bridging this gap. It proposed that action and cognition arise from simple, parallel computations in individual neurons, which collectively generate meaningful distributed representations across the network and that these distributed representations are essential for enabling networks to perform flexible, robust, and scalable information processing. This marked a paradigm shift from traditional logic-based and symbolic representations to a distributed perspective, where knowledge is represented and processed by the network as a whole rather than by individual neurons. As summarised by Schmidhuber (2014), it was only with the combination of efficient reverse-mode automatic differentiation (Linnainmaa, 1970) and neural networks with differentiable activation functions (Werbos, 1982) that it became technically possible to train such networks (Rumelhart, Hinton, & Williams, 1986), sparking the rise of modern connectionism (Rumelhart, Hinton, & McClelland, 1986) and deep learning (LeCun et al., 2015).

1.1.2 Deep learning

The realisation that complex, non-linear, and inherently non-convex artificial neural networks (ANNs) can be effectively optimised using first-order methods such as gradient descent, started a revolution in the field of machine learning. Throughout the early 2000s, the massive increase in computational power (Amodei et al., 2018; Sevilla et al., 2022), advances in network architecture (Schmidhuber & Hochreiter, 1997; LeCun et al., 1989; Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; Goodfellow et al., 2014; Kingma & Welling, 2014; Szegedy et al., 2015; K. He et al., 2016; Vaswani et al., 2017), activation functions (Maas et al., 2013; K. He et al., 2015; Clevert, 2015; Hendrycks & Gimpel, 2016; Klambauer et al., 2017), optimisation algorithms (Duchi et al., 2011; Zeiler, 2012; Kingma, 2014; Schulman et al., 2017), network initialisation techniques (LeCun et al., 2002; Glorot & Bengio, 2010; Saxe et al., 2013; K. He et al., 2015) and novel regularisation methods (N. Srivastava et al., 2014; Ioffe & Szegedy, 2015) led to a series of breakthroughs. In 2012 Krizhevsky et al. (2012) combined convolutional neural networks (Fukushima, 1980; LeCun et al., 1989), max-pooling layers for downsampling (Weng et al., 1993), rectified linear units as activation function (Fukushima, 1969) and parallel optimisation on multiple GPUs using CUDA (Nickolls et al., 2008), to win the ImageNet competition (Russakovsky et al., 2015), in which images have to be assigned to one of a thousand categories. Their neural network AlexNet was trained on 1.2 million images, and achieved a top-5 classification error of 15.3%, outperforming competitors by a margin of more than 10% (Russakovsky et al., 2015). Over the next few years, the error rate rapidly dropped further (K. He et al., 2015, 4.94%) and further (K. He et al., 2016, 3.57%), quickly surpassing human level performance (Russakovsky et al., 2015, $\sim 5\%$). At the same time, researchers combined Q-learning (Watkins & Dayan, 1992) with ANNs to achieve human-level performance across 49 Atari games (Mnih et al., 2015). Silver et al. (2016) combined

convolutional networks, policy gradient algorithms (Williams, 1992; Schulman et al., 2017) and Monte-Carlo tree search to a network that defeated the best human Go players, a breakthrough that was believed to be at least a decade away. Follow-up work, replaced the dependence on human training data by self-play (Silver et al., 2017), and reduced training time to surpass human performance from 3 weeks to less than 24 hours (Silver et al., 2018). Models were developed to create captions for images (Karpathy & Fei-Fei, 2015), machine translation (Cho et al., 2014; Sutskever et al., 2014) or map text to speech (Van Den Oord et al., 2016). Generative models, quickly evolved from generating low-resolution, low-quality images (Goodfellow et al., 2014; Kingma & Welling, 2014; Sohl-Dickstein et al., 2015) to text-to-image models (Reed et al., 2016), which generate photo-realistic, high-resolution images (Saharia et al., 2022; Esser et al., 2024). The development of advanced self-supervised learning algorithms (De Sa, 1993) eliminated the need for large amounts of human-annotated training data (T. Chen et al., 2020; Caron et al., 2020; Grill et al., 2020; X. Chen & He, 2021; Zbontar et al., 2021). Finally, the emergence of the transformer architecture (Vaswani et al., 2017) with its self-attention mechanism, which in contrast to earlier recurrent models that process data sequentially, allows models to process entire sequences of data in parallel, led to a breakthrough in natural language processing (Devlin et al., 2018; Brown et al., 2020; Touvron et al., 2023). These large language models are first trained in an autoregressive fashion to predict missing parts of large text corpora (Bengio et al., 2000), allowing them to learn grammar, context, and meaning without explicit human labelling, and are subsequently fine-tuned to adapt their knowledge to specialised tasks like sentiment analysis or question answering, closing the gap towards passing the Turing Test (Turing, 1950). Scaling these models (Kaplan et al., 2020), led to the emergence of in-context learning, in which models are capable of solving tasks from examples without updating their parameters (Brown et al., 2020).

While this summary only scratches the surface of the advances made during the last two decades, the impact of deep learning on society, industry, and research is irrefutable. With ANNs now performing tasks that were previously believed to be only attainable with human intelligence, what are the implications for connectionism? Can ANNs aid our understanding of the brain and how can our understanding of the brain aid research in ANNs?

1.1.3 Gradient-based learning and the brain

The great success of deep learning has not only led to a minor revolution in machine learning but also in the neuro- and cognitive sciences (Richards et al., 2019; Saxe et al., 2021; Doerig et al., 2023). With ANNs performing on par with humans on specific tasks, do they employ similar computational mechanisms, and can we gain insights into one by studying the other?

We begin by exploring work which has studied whether gradient-based optimisation, which is used to optimise complex distributed and parallel computations in deep networks, could also be the underlying algorithm driving learning in the brain. There are several key aspects on the implementational level of the backpropagation algorithm that raise questions regarding its biological plausibility (Crick, 1989; Whittington & Bogacz, 2019; Lillicrap et al., 2020). For example, the same weights are used during forward propagation of information and backpropagation of errors. It is unclear how the information to preserve this symmetry could be shared across neurons in the brain. This concern has been partially addressed by work that showed that this symmetry may not be required for the backpropagation algorithm to work (Lillicrap et al., 2016; Nøkland, 2016; Launay et al., 2020) due to symmetries in the solution space (Refinetti et al., 2021). Further, during backpropagation, a global error signal is propagated backwards through multiple layers and requires synchronous updates to all weights, but the brain lacks an obvious method

for globally distributing such an error signal across distant neurons and cortical layers. Again, this concern has been partially addressed by algorithms like (difference) target propagation (Le Cun, 1986; Bengio, 2014; Meulemans et al., 2020; D.-H. Lee et al., 2015) and energy-based approaches (Friston, 2010; Scellier & Bengio, 2017; Song et al., 2024) like predictive coding (Friston, 2005; Whittington & Bogacz, 2017; Sacramento et al., 2018). The limitation that the all or none response of spiking is non-differentiable has for example been addressed by surrogate-gradient methods (Courbariaux et al., 2016; Zenke & Ganguli, 2018; Zenke & Vogels, 2021; Braun & Vogels, 2021).

Although the brain may not implement the backpropagation algorithm exactly, the neural computations, representations, and learning dynamics observed in deep networks could be closely related to those used in the brain (Richards et al., 2019; Doerig et al., 2023; Haxby et al., 2014). This idea gave rise to a line of research that has focused on examining the similarities between computational mechanisms and neural representations in artificial and biological networks. For example, the structure of receptive fields and the layer-wise hierarchical feature extraction are computational properties that are observed both in the ventral visual pathway as well as convolutional neural networks (Hubel & Wiesel, 1962; Erhan et al., 2009; Zeiler & Fergus, 2014; DiCarlo et al., 2012; Lindsay, 2021). Importantly, this observation is consistent across species. For example, training a convolutional neural network with an architecture similar to that of the *Drosophila* retina results in analogous spatial and temporal receptive fields as observed in the fly (McIntosh et al., 2016). Albeit differences exist and neural networks do not account for many results from psychological research (Bowers et al., 2023). For example, they exhibit a strong preference for texture over shape (Geirhos et al., 2018) and are susceptible to minimal image perturbations known as adversarial attacks (Szegedy et al., 2014, though see Veerabadran et al. (2023)). One way to understand how neural networks encode information is to study and compare the computational properties

of single cells. However, action and cognition are encoded across populations of neurons throughout the brain. Given a certain stimulus or task, the corresponding experience elicits patterns of neural activity, which can be analysed and compared across experiences, individuals and even between artificial and biological networks (Haxby et al., 2014). More formally, a pattern of activity can be thought of as a single point in a high-dimensional space, in which each dimension represents the level of activity of a single neuron. Then, analysing and comparing patterns is a matter of studying the relation of points in this high dimensional space.

For example, similar low-dimensional population dynamics were observed during context-dependent decision Mante et al. (2013) and delayed match-to-category tasks Chaisangmongkon et al. (2017) in artificial recurrent neural networks and monkey prefrontal cortex. In their seminal work, Yamins et al. (2014) used the activity patterns of the final layer of convolutional neural networks to predict single-cell and population-level neural responses in monkey inferior temporal cortex when exposed to the same natural stimuli. Critically, neural networks were not trained to model or predict neural data, but rather were either untrained or trained on image classification tasks. They found that in both cases, a network's categorisation performance strongly correlated with their ability to predict neural activity. In a similar line of work Khaligh-Razavi & Kriegeskorte (2014) compared neural representations in ANNs to neural activity patterns in human and monkey inferior temporal cortex using representational similarity analysis (Kriegeskorte, Mur, & Bandettini, 2008). Specifically, they compared the representations of networks that where trained using supervised and unsupervised training algorithms. They also found that better performing models have larger representational similarity and that only supervised learning algorithms gave rise to categorical clustering, which is observed in inferior temporal cortex. Accordingly, later work showed similar activity patterns across the layers of neural networks and the visual hierarchy in humans (Eickenberg et al., 2017). These observations gave rise to a whole new

field of research, which aims at facilitating comparisons between the dynamical and representational features of artificial and biological neural networks (Schrimpf et al., 2018). However, this methodology has been criticised on the basis that overlap of representational similarities does not foster understanding of perceptual and cognitive processes (Bowers et al., 2023). Furthermore, theoretical work has demonstrated that partial observations can result in network models exhibiting spurious attractor structures, which fail to accurately capture similarities in neural dynamics (Qian et al., 2024). Additionally, it has been shown that the observed similarity between representations is highly contingent upon the similarity measure used (Soni et al., 2024). In summary, a quantitative theory that explains why, when, and under what circumstances neural representations in brains and machines align is essential for advancing our understanding of neural computation.

One key aspect of these comparisons between mind and machine is that they often treat both as static systems. However, humans and non-human animals live and learn in a continuous stream of information, meaning their brains are constantly evolving. This is in contrast to artificial neural networks, which are typically trained once and then remain static in their computations. Incorporating the dimension of time however, is challenging, as the underlying computational systems are already highly non-linear and complex, even without considering the dynamics of learning. While some progress has been made, such as relating learning dynamics in linear networks to concept learning in children (Saxe et al., 2019), understanding how neural computations and representations unfold over time is crucial to understanding how the brain gives rise to action and cognition.

1.2 Continual Learning

Humans and non-human animals stand as living proof for the existence of a system that can continuously acquire new knowledge and skills throughout a life-

time. At first glance, this ability may appear unsurprising, as it arises naturally without conscious effort or major interference with previously acquired knowledge. However, while biological neural networks excel at continual learning, the underlying computational mechanisms remain poorly understood and the subject remains an unsolved problem for ANNs. Continual learning can occur over various time scales, from brief moments to a lifetime and requires the integration of experiences to one coherent whole. It can involve mastering new skills, refining prior knowledge, or adapting to novel environments and contexts. As such, continual learning fundamentally differs from traditional optimisation of ANNs, which are typically trained all at once on a fixed dataset. Already in the late 1980s, McCloskey & Cohen (1989) and Ratcliff (1990) explored this fundamental difference, observing that when ANNs were trained continuously using gradient descent, earlier acquired knowledge was often completely overwritten by parameter updates during the acquisition of new knowledge, a phenomenon they termed catastrophic forgetting. Specifically, drawing inspiration from how children learn, McCloskey & Cohen (1989) trained ANNs to learn addition involving ones (e.g., $1 + 1 = 2$, $1 + 2 = 3$, ...) and subsequently trained them on additions involving twos (e.g., $2 + 1 = 3$, $2 + 2 = 4$, ...). They observed that after training on the twos-task, the network's performance on the ones-task had declined back to the level of random guessing. This is in stark contrast to how human and non-human animals learn, as they not only do not suffer from catastrophic forgetting, but even more so, benefit from structured learning curricula (Krueger & Dayan, 2009; Bengio et al., 2009; Flesch et al., 2018; Dekker et al., 2022). Continual learning quickly developed into its own research field (French, 1999), which experienced a resurgence with the rise of deep learning (Goodfellow et al., 2013), with a vast literature investigating and developing algorithms for continual learning. In the following, we give an overview over common algorithms and theoretical insights into the continual learning problem.

1.2.1 Overcoming catastrophic forgetting

A plethora of algorithms have been proposed to circumvent catastrophic forgetting (as reviewed in Parisi et al., 2019; Hadsell et al., 2020; Wang et al., 2024), which can be broadly separated into four groups. Architecture and optimisation-based methods adjust the network structure and calculation of gradients, while regularisation-based approaches control how and which synapses change, and replay-based methods store previous training examples and combine them with new data for subsequent learning. Most algorithms are motivated by either a mathematical or biological insights (Kudithipudi et al., 2022). For example, optimisation and regularisation-based approaches are often motivated by overparameterisation (C. Zhang et al., 2021) or Bayes-optimal learning (Huszár, 2017) and are closely related to the stability-plasticity dilemma in neuroscience (Grossberg, 1987; Abraham & Robins, 2005). Whereas, replay-based approaches aim at approximating full-batch learning and are motivated by complementary learning systems theory (McClelland et al., 1995; Kumaran et al., 2016).

Solving the stability-plasticity dilemma

The stability-plasticity dilemma hypothesises that a core challenge of learning is the need for synaptic weights to remain stable enough to preserve previously acquired knowledge, while also maintaining enough plasticity to encode new information (Grossberg, 1987; Abraham & Robins, 2005). If we view the continual learning problem through this conceptual framework, an effective continual learning algorithm must be capable of identifying synapses essential for encoding previous knowledge and restricting plasticity therein while allowing plasticity in others. For ANNs, this involves analysing the shape of the error landscape. If we identify directions in which parameters can be updated without increasing the loss, those represent permissible plasticity, while directions where the loss increases rapidly indicate

updates that should be restricted. To this end, one can use a Bayesian approach, combined with Laplace's method, to get a point estimate of the error landscape for the current set of network parameters (Huszár, 2017). Since the Bayesian approach requires the computation of the Hessian matrix, which is computationally expensive and scales quadratically in the number of parameters, continual learning algorithms often approximate the Hessian using the diagonal of the empirical Fisher information (Kirkpatrick et al., 2017) or approximations thereof (Zenke et al., 2017; Aljundi et al., 2018; Benzing, 2022). These algorithms then regularise parameter updates in proportion to the estimated curvature. Alternatively, it has been suggested to use techniques such as dropout, learning rate decay, and an appropriate batch size to converge to solutions that are both wide and flat (Mirzadeh et al., 2020). However, point estimates of the error surface may miss important non-local structure like manifolds and symmetries in the solution space (Baldi & Hornik, 1989; Sussmann, 1992; Fukumizu & Amari, 2000; Dinh et al., 2017; Simsek et al., 2021). Furthermore, controlling synaptic plasticity will change how information is processed and encoded in the network, possibly leading to ineffective neural representations (Prabhu et al., 2024). However, a clear understanding of the relationship between the stability-plasticity dilemma, the structure and symmetry of solution manifolds and its implications for continual learning, necessitates a robust theoretical framework that goes beyond heuristics and numerical simulations.

Minimising interference

The idea to mitigate catastrophic forgetting by minimising interference between internal representations of old and new information is nearly as old as the discovery of catastrophic forgetting itself (Kortge, 1989; French, 1991). While early approaches primarily focused on explicitly finding disjoint representations (R. K. Srivastava et al., 2013), newer methods primarily focus on exploiting that ANNs are highly overparamterised. They aim at orthogonalising gradient updates or directly learning

in orthogonal subspaces to minimise interference. Gradients can be orthogonalised with respect to previous inputs (Zeng et al., 2019) or previous gradient updates (Farajtabar et al., 2020), and networks can be separated explicitly into low-rank orthogonal subspaces (Chaudhry et al., 2020) or indirectly by projecting gradient updates into activity-defined (Duncker et al., 2020) or gradient-defined (Saha et al., 2021) subspaces within the network. Prior work has argued that biological networks indeed use orthogonal task representations, however, for tasks in which an explicit context is provided (Roy et al., 2010; Flesch et al., 2022; Koay et al., 2022). Almost as old as the idea to mitigate catastrophic forgetting by minimising interference, is the concern that, when using distributed representations, networks might lose their highly desirable generalisation properties (Lewandowsky, 1994). For example, simulations showed that training networks on orthogonal subspaces can indeed fail to find and exploit structure that is shared across tasks (Duncker et al., 2020). However, a quantitative theory that can explain the impact of learning in orthogonal subspaces on generalisation and neural representations is missing.

Complementary learning systems and replay

The complementary learning systems theory (McClelland et al., 1995; Kumaran et al., 2016) suggests that there are two interdependent systems for memory and learning in the brain. In this framework, the hippocampus and adjacent cortical areas like the entorhinal and perirhinal cortex in the medial temporal lobe are responsible for rapidly memorising newly arriving information with minimal interference. Over time, these memories are then gradually integrated and consolidated into generalised long-term knowledge in the neocortex. The theory emerged from the observation that damage to the medial temporal lobe, like in patient HM (Scoville & Milner, 1957), results in temporally graded retrograde amnesia, where memories formed before the injury become inaccessible, with more recent memories being more affected than older memories (Alvarez & Squire, 1994). According to

the theory, to combine and consolidate memories, they first have to be retrieved and replayed. It has been shown that neural activity representing past experiences are in fact reoccurring in hippocampus and cortex, particularly during sleep and rest (Wilson & McNaughton, 1994; Carr et al., 2011), and that interrupting these reactivations impairs memory consolidation (Girardeau et al., 2009; Ego-Stengel & Wilson, 2010). While it remains an active area of research how replay facilitates learning, planning and memory consolidation (reviewed in Z. S. Chen & Wilson, 2023; Wittkuhn et al., 2021), and when and what memories exactly should be replayed to facilitate these functions (Sun et al., 2023), a system that memorises and integrates newly arriving information is an ideal candidate mechanism to support continual learning.

The simplest way to address catastrophic forgetting in ANNs is to store the training data from previous tasks and then train jointly on both past and current data (Robins, 1995). However, this naïve approach demands significant memory or may be unfeasible when past information is inaccessible. A large body of work thus investigated how to select items for memorisation and replay (Wittkuhn et al., 2021). Approaches include choosing a small, random subset of training data (Chaudhry et al., 2019), data that covers the task's distribution (Rebuffi et al., 2017; Isele & Cosgun, 2018), or the distribution of gradients induced by the data (Aljundi, Lin, et al., 2019), data whose retrieval is most at risk of being disrupted by parameter updates (Aljundi, Belilovsky, et al., 2019), data that maximises sample diversity (Bang et al., 2021), and tricks to tackle class imbalance between memorised and current data (Wu et al., 2019). One way to avoid explicitly storing training data is to generate and replay pseudo training data instead (Robins, 1995). This pseudo data can for example be random (Robins, 1995), a compressed version of the original data (Wang et al., 2022; Caccia et al., 2020), or come from a generative model (Goodfellow et al., 2014) that generates pseudo inputs that are then replayed (Shin et al., 2017; Van de Ven et al., 2020; Atkinson et al., 2021).

However, the effectiveness of methods that try to improve over naïve replay has been called into question, as many of them are outperformed by a simple baseline model that greedily stores a small subset of memories and then retrains a model from scratch after each task rather than continuing to retrain an existing model (Prabhu et al., 2020). This observation may be explained by the loss of plasticity throughout the acquisition of knowledge (Dohare et al., 2024). In summary, replay-based algorithms seek to mimic training over the full data distribution to integrate newly arriving knowledge. However, a quantitative theory that provides limitations and guidance on item selection and data compression is missing.

1.3 Representational drift

A major difference between artificial and biological neural networks is that biological neural networks are ever changing, while artificial networks become static once trained. But when, why and how do biological neural networks undergo change? Intuitively, one would assume that under behavioural and perceptual stability for a given stimulus, associated neural computations and therefore the underlying neural representations should remain stable too. This intuition, however, is challenged by a phenomenon called representational drift: the observation that neural representations are fully reorganised over the course of weeks without any noticeable changes in performance or behaviour (Rule et al., 2019; Driscoll et al., 2022). Representational drift has been observed in many brain regions and modalities. For example in neurons coding for spatial navigation in mouse hippocampus (Ziv et al., 2013), and mouse parietal cortex (Driscoll et al., 2017), representations of odours in the mouse primary olfactory cortex (Schoonover et al., 2021; Chapman et al., 2024), representations of simple and complex visual stimuli in mouse visual cortex (Marks & Goard, 2021; Deitch et al., 2021), representations of whisker touch in barrel cortex (Alisha et al., 2023), monkey motor cortex (Gallego et al., 2020)

and even human visual cortex (Roth & Merriam, 2023). Further, there is strong evidence that representational drift cannot be simply attributed to measurement error as similar rates and patterns of drift are observed regardless of whether electrophysiological or optical recording methods were used (Driscoll et al., 2022) and significant efforts and technological advances have been made to reliably identify the same cells across long time scales (e.g. Dhawale et al., 2017; Schoonover et al., 2021). Also, studies that argue that drifting neural representations may be a mere consequence of the behavioural state of the animal (Sadeh & Clopath, 2022), which is known to modulate neural activity (Musall et al., 2019), are challenged by long-term studies showing that time is a stronger predictor of representational drift than behavioural factors (Driscoll et al., 2017) and that distinct aspects of representational drift depend on time and experience (Geva et al., 2023). But what are the causes and consequences of drifting neural representations? Are they simply the consequence of noise-driven processes similar to the constant shedding and renewal of skin that leaves it broadly unchanged, or must representational drift be attributed to learning and the acquisition of new knowledge? A large body of theoretical work and simulation studies has tried to clarify the nature of representational drift. In the following, we will categorise them into epiphenomenal and functional explanations.

Representational drift as epiphenomenon

As elaborated by Rule et al. (2019), if representational changes occur along dimensions not accessed by upstream neurons during a specific task, these alterations would lead to drifting neural representations without affecting behaviour (Kaufman et al., 2014; Rokni et al., 2007). However, while such drift might not disrupt network function within this given task, it could change how other tasks are processed. Thus, one possibility is that all representational drift occurs along directions that do not affect upstream processing across all tasks. Another possibility is that

changes in neural circuitry that create drift in neural representations are compensated for by upstream neural circuits, maintaining overall computational stability while allowing neural representations to change. This compensation could occur through mechanisms such as local Hebbian plasticity (Rule et al., 2020). Formally, this relates to the analytical concept of solution manifolds: If there exist many combinations of synapses that all implement the same function, neural representations can change without changing the underlying function. For example, stochastic gradient descent can induce drift along this manifold, resulting in drifting neural representations (Pashakhanloo & Koulakov, 2023; Avidan et al., 2023; Li et al., 2024). Similarly, leveraging permutation symmetries within the manifold (Simsek et al., 2021), a combination of Hebbian and anti-Hebbian learning in two-layer nonlinear neural networks causes drifting neural representations by inducing sudden transitions between different points on the manifold (Qin et al., 2023). In this view of representational drift, neural representations change as a result of system noise, which either has no effect on upstream computations, necessitates a computational mechanism that counterbalances these changes, or causes a diffusion on the solution manifold with inherently stable underlying computations.

Functional representational drift

If drifting neural representations were indeed driven by noisy processes, such as synaptic turnover, it might be anticipated that all neural representations would change over time. The rate of this change, however, would likely vary depending on regional differences in physiology, with some regions exhibiting faster or slower turnover rates. However, certain neural circuits, such as those involved in encoding task-relevant information in the mouse medial prefrontal cortex (Muysers et al., 2024), the generation of the courtsong in zebra finches (Katlowitz et al., 2018), and face-selective cells in the macaque inferotemporal cortex (McMahon et al., 2014), produce neural representations that are remarkably stable, remaining virtually un-

changed over weeks to months. This suggests that drifting neural representations may not be driven by random processes but depend on perceptions (Bauer et al., 2023) and support an underlying function. For example, drift along the solution manifold may occur in directions that regularise and consolidate underlying computations (Aitken et al., 2022; Masset et al., 2022; Micou & O'Leary, 2023; Ratzon et al., 2024), or newly arriving information could lead to changes in the neural representations of previously acquired knowledge to minimise interference (Devalle et al., 2024). Furthermore, while it has been suggested that continual learning may be the underlying cause of representational drift (Driscoll et al., 2022; Micou & O'Leary, 2023; Anthes et al., 2024), the quantitative relationship between the two remains illusive, as the continual learning problem is still only partially understood and largely unresolved.

In summary, a unifying theory that can distinguish between epiphenomenal and functional causes of representational drift and generates testable predictions is missing. Throughout this thesis, we will use deep linear networks to study and dissociated how neural representations can drift as an epiphenomena or during functional refinement and reveal that representations must drift during continual learning.

General Methods

Here, we introduce mathematical notation and key analytical concepts used throughout this thesis. While each chapter begins with a brief preliminaries section, this section offers formal definitions and more detailed explanations.

2.1 Mathematical notation

Throughout this thesis, scalars are denoted by letters (e.g. a, τ, L), matrices by bold uppercase letters (e.g. $\mathbf{X}$, $\mathbf{\Gamma}$), column vectors are denoted by bold lowercase letters (e.g. $\mathbf{x}$, $\boldsymbol{\lambda}$), and row vectors by the transpose of a column vector (e.g. $\mathbf{x}^T$, $\boldsymbol{\lambda}^T$). The vector dot product and vector outer product are denoted by $\mathbf{a}^T\mathbf{b}$ and $\mathbf{a}\mathbf{b}^T$ respectively. The identity matrix of size n is denoted by $\mathbf{I}_n$. $\mathbf{A}^{-1}$ and $\mathbf{A}^+$ denote the inverse and Moore–Penrose pseudoinverse of a matrix. The ℓ_2-norm of a vector is then expressed by $||\mathbf{x}||_2$ and the Frobenius norm of a matrix is expressed by $||\mathbf{A}||_F$. The Kronecker product is denoted by $\otimes$. Column-wise and row-wise vectorisation of a matrix are denoted by $\text{vec}(\mathbf{A})$ and $\text{vec}_r(\mathbf{A})$ respectively. Given a univariate or multivariate function f, we write their respective derivatives and partial derivatives with respect to a variable $\bullet$ as $\frac{df}{d\bullet}$ and $\frac{\partial f}{\partial \bullet}$.

2.2 Supervised learning

This thesis focuses on supervised learning, a process in which a human, non-human animal, or machine learning model learns to generate correct responses from examples with known answers. Formally, a supervised learning task $\mathcal{T}$ consists of

a set of P training pairs $\mathcal{D} = \{(x_n, y_n)\}_{n=1...P}$, with input vectors $\mathbf{x}_n \in \mathbb{R}^{N_i}$ and associated target output vectors $\mathbf{y}_n \in \mathbb{R}^{N_o}$. Here, N_i and N_o are the input and output dimension respectively. We denote by $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_P] \in \mathbb{R}^{N_i \times P}$ the matrix containing all inputs, and by $\mathbf{Y} = [\mathbf{y}_1, ..., \mathbf{y}_P] \in \mathbb{R}^{N_o \times P}$ the matrix containing all targets. The objective of a learner is then to make predictions $\hat{\mathbf{y}}_n$ that resemble the respective target $\mathbf{y}_n$ when presented with input $\mathbf{x}_n$.

2.2.1 Covariance matrices

We define the input, output, and input-output covariance matrices of the training data as

$$\mathbf{\Sigma}_{xx} = \frac{1}{P} \sum_{i=1}^{P} \mathbf{x}_n \mathbf{x}_n^T \in \mathbb{R}^{N_i \times N_i}, \tag{2.1}$$

$$\mathbf{\Sigma}_{yy} = \frac{1}{P} \sum_{i=1}^{P} \mathbf{y}_n \mathbf{y}_n^T \in \mathbb{R}^{N_o \times N_o}, \tag{2.2}$$

and

$$\mathbf{\Sigma}_{yx} = \frac{1}{P} \sum_{i=1}^{P} \mathbf{y}_n \mathbf{x}_n^T \in \mathbb{R}^{N_o \times N_i} \tag{2.3}$$

respectively.

2.3 Continual learning

During continual learning, a learner has to incrementally acquire knowledge from a sequence of supervised learning tasks $\mathcal{T}^1$, $\mathcal{T}^2$, $\mathcal{T}^3$, ... The learning objective is then to maximise performance on the current task while maintaining performance on all previous tasks. Crucially, the learner has access to a single learning task at a time and is presented with each task exactly once.

2.4 Deep linear networks

Let

$$\hat{\mathbf{y}}_n = \left(\prod_{l=1}^{L} \mathbf{W}_l \right) \mathbf{x}_n \tag{2.4}$$

denote a deep linear network model (Saxe et al., 2013) of depth L, where $\mathbf{W}_l \in \mathbb{R}^{N_l \times N_{l+1}}$ is the l-th weight matrix of the network with input dimension N_l and output dimension N_{l+1}. We note that the input and output dimension of the overall network function are equivalent to the dimensionality of the training data, i.e., $N_1 = N_i$ and $N_{L+1} = N_o$. The entries of the weight matrices are also referred to as parameters or weights, of which the model has a total number of

$$N = \sum_{l=1}^{L} N_l N_{l+1}. \tag{2.5}$$

We write $\theta \in \mathbb{R}^N$ to denote the vector that contains a collection of all parameters of the model.

2.4.1 Two-layer linear network

We call a linear network with $L = 2$ a two-layer linear network and write

$$\hat{\mathbf{y}}_n = \mathbf{W}_2 \mathbf{W}_1 \mathbf{x}_n \tag{2.6}$$

for its network function. In this special case, we denote the dimensions of the weight matrices as $\mathbf{W}_1 \in \mathbb{R}^{N_o \times N_h}$ and $\mathbf{W}_2 \in \mathbb{R}^{N_h \times N_i}$, where $N_h = N_2$ is the dimension of the hidden layer representation

$$\mathbf{h}_n = \mathbf{W}_1 \mathbf{x}_n. \tag{2.7}$$

2.5 Loss function

A loss function is then used to quantify the difference between predicted outputs $\hat{\mathbf{y}}_n$ and respective target outputs $\mathbf{y}_n$. We write ℓ and $\mathcal{L}$ for loss functions that quantify a single or all training pairs respectively. Specifically, we use the ℓ_2-loss

$$\ell_2 = \frac{1}{2}||\hat{\mathbf{y}}_n - \mathbf{y}_n||_2^2. \tag{2.8}$$

and the mean squared error loss, i.e., the average ℓ_2-loss

$$\mathcal{L}_{\text{MSE}} = \frac{1}{2P}\sum_{n=1}^{P}||\hat{\mathbf{y}}_n - \mathbf{y}_n||_2^2 \tag{2.9}$$

to quantify the performance of linear neural network models. We note that the mean squared error can also be written as

$$\mathcal{L}_{\text{MSE}} = \frac{1}{2P}||\hat{\mathbf{Y}} - \mathbf{Y}||_F^2. \tag{2.10}$$

2.6 Gradient descent

To minimise a loss function $\mathcal{L}$, i.e., optimise a model's performance across all training pairs, the gradient descent algorithm adjusts the network's parameters by taking steps in the opposite direction of its gradient. Given the gradient with respect to the parameters of a scalar-valued (partially) differentiable loss function

$$\nabla_\theta \mathcal{L} = \left[\frac{\partial \mathcal{L}}{\partial \theta_1}, ..., \frac{\partial \mathcal{L}}{\partial \theta_N}\right]^T, \tag{2.11}$$

the parameters at time t are iteratively updated by

$$\theta(t+1) \leftarrow \theta(t) - \eta \nabla_{\theta(t)} \mathcal{L}, \tag{2.12}$$

where the learning rate η is an adjustable hyperparameter, which determines the step size of the parameter updates. Gradient descent is a first-order method, it converges to local minima. Since the gradient is calculated over all training pairs in each step, this parameter update rule is also called batch or full-batch gradient descent.

2.6.1 Stochastic and online gradient descent

The computational burden of full-batch gradient descent increases with the size of the dataset. Thus, the gradient in each training step is often approximated by calculating it over a random subset of the training data. This is referred to as mini-batch gradient descent. In the extreme case, where the gradient is calculated for a single training pair only

$$\theta(t+1) \leftarrow \theta(t) - \eta \nabla_{\theta(t)} \ell, \tag{2.13}$$

we speak of stochastic gradient descent, which can model online learning.

2.6.2 Gradient descent in deep linear networks

In case of neural networks, the gradient update is usually derived as a partial derivative with respect to the weight matrices instead of the vectorised parameters θ, which leads to parameter updates

$$\mathbf{W}_j(t+1) \leftarrow \mathbf{W}_j(t) - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}_j} \quad \forall\ j \in \{1, ..., L\}. \tag{2.14}$$

For deep linear networks, the partial derivative of the mean squared error with respect to the j-th weight matrix is

$$\frac{\partial \mathcal{L}_{\text{MSE}}}{\partial \mathbf{W}_j} = \left(\prod_{l=j+1}^{L} \mathbf{W}_l \right)^T \left(\left(\prod_{l=1}^{L} \mathbf{W}_l \right) \boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{yx} \right) \left(\prod_{l=1}^{j-1} \mathbf{W}_l \right)^T. \tag{2.15}$$

In the special case of two-layer linear networks this is

$$\frac{\partial \mathcal{L}_{\text{MSE}}}{\partial \mathbf{W}_1} = \mathbf{W}_2^T \left(\mathbf{W}_2 \mathbf{W}_1 \boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{yx} \right) \tag{2.16}$$

and

$$\frac{\partial \mathcal{L}_{\text{MSE}}}{\partial \mathbf{W}_2} = \left(\mathbf{W}_2 \mathbf{W}_1 \boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{yx} \right) \mathbf{W}_1^T. \tag{2.17}$$

2.6.3 Gradient flow

The stepwise gradient descent parameter updates turn into a continuous time differential equation, in the limit of small learning rates

$$\frac{d}{dt}\mathbf{W}_j = \lim_{\eta \to 0} \frac{\mathbf{W}_j(t+1) - \mathbf{W}_j(t)}{\eta} = -\frac{\partial \mathcal{L}}{\partial \mathbf{W}_j}. \tag{2.18}$$

This is called the gradient flow. For non-infinitesimal but sufficiently small learning rates, the discrete time steps of the gradient descent parameter updates can be related to the continuous time dynamics by rescaling time by the time constant

$$\tau = \frac{1}{\eta}, \tag{2.19}$$

which yields

$$\tau \frac{d}{dt}\mathbf{W}_j = -\frac{\partial \mathcal{L}}{\partial \mathbf{W}_j}. \tag{2.20}$$

2.7 Compact singular value decomposition

Throughout this thesis, we will extensively employ the so called compact singular value decomposition (SVD) to analyse the structure of training data and examine how the weight matrices project and process the training data within the network. The compact SVD (Strang, 2014, p. 396), decomposes any matrix $\mathbf{A} \in \mathcal{R}^{m \times n}$ with rank r into a product of three matrices

$$\mathrm{SVD}\left(\mathbf{A}\right) = \mathbf{U}\mathbf{S}\mathbf{V}^T. \tag{2.21}$$

In contrast to the full SVD, the singular vectors $\mathbf{U} \in \mathbb{R}^{m \times r}$ and $\mathbf{V} \in \mathbb{R}^{n \times r}$ are not generally square orthogonal matrices but may be semi-orthogonal dependent on the relationship between m, n and r (Table 2.1). The matrix $\mathbf{S} \in \mathbb{R}^{r \times r}$ is a

	$m = n$		$m < n$		$m > n$	
	$r = m$	$r < m$	$r = m$	$r < m$	$r = n$	$r < n$
$\mathbf{U}^T\mathbf{U}$	$= \mathbf{I}_m$	$= \mathbf{I}_r$	$= \mathbf{I}_m$	$= \mathbf{I}_r$	$= \mathbf{I}_n$	$= \mathbf{I}_r$
$\mathbf{U}\mathbf{U}^T$	$= \mathbf{I}_m$	$\neq \mathbf{I}_m$	$= \mathbf{I}_m$	$\neq \mathbf{I}_m$	$= \mathbf{I}_m$	$\neq \mathbf{I}_m$
$\mathbf{V}^T\mathbf{V}$	$= \mathbf{I}_m$	$= \mathbf{I}_r$	$= \mathbf{I}_m$	$= \mathbf{I}_r$	$= \mathbf{I}_n$	$= \mathbf{I}_r$
$\mathbf{V}\mathbf{V}^T$	$= \mathbf{I}_m$	$\neq \mathbf{I}_m$	$\neq \mathbf{I}_n$	$\neq \mathbf{I}_n$	$= \mathbf{I}_n$	$\neq \mathbf{I}_n$

Table 2.1: Orthonormality of singular vectors

square matrix with ordered, non-zero eigenvalues on its diagonal. Thus, $\mathbf{S}^{-1}$ is well defined. Using that $(\mathbf{BC})^+ = \mathbf{C}^+\mathbf{B}^+$ if $\mathbf{B}$ has orthonormal columns or $\mathbf{C}$ has orthonormal rows (Greville, 1966), we can then show that

$$\mathbf{A}^+ = \left(\mathbf{U}\mathbf{S}\mathbf{V}^T\right)^+ = \mathbf{V}\left(\mathbf{U}\mathbf{S}\right)^+ = \mathbf{V}\mathbf{S}^+\mathbf{U}^T = \mathbf{V}\mathbf{S}^{-1}\mathbf{U}^T \tag{2.22}$$

is the Moore-Penrose inverse of any matrix $\mathbf{A}$.

2.8 Representational similarity matrices

A common tool to understand neural computations and representations is to analyse how similar the neural representations of stimuli are (Kriegeskorte, Mur, & Bandettini, 2008; Haxby et al., 2014). Let

$$\mathbf{h}_n = \mathbf{W}_1 \mathbf{x}_n \tag{2.23}$$

denote the hidden-layer representation of the n-th input. Then, a representational similarity matrix (RSM) is a symmetric $P \times P$ matrix, where the entry at position (i, j) represents the degree of (dis)similarity between objects i and j. The simplest measure of representational similarity is the inner product covariance matrix of the hidden-layer representations, i.e.,

$$\text{RSM}_{ij} = \mathbf{x}_i^T \mathbf{W}_1^T \mathbf{W}_1 \mathbf{x}_j. \tag{2.24}$$

Other measures of representational similarity include euclidean distance

$$\text{RSM}_{ij} = ||\mathbf{h}_i - \mathbf{h}_j||_2 \tag{2.25}$$

and the Pearson correlation coefficient

$$\text{RSM}_{ij} = \frac{\text{cov}(\mathbf{h}_i, \mathbf{h}_j)}{\sigma_i \sigma_j}, \tag{2.26}$$

where cov is the covariance and σ_i and σ_j denote the standard deviation of $\mathbf{h}_i$ and $\mathbf{h}_j$ respectively.

Not all solutions are created equal: Understanding the solution manifold of deep linear networks

---

This chapter presents original work. Contributions to the research presented in this chapter are as follows:

Supervision Professor Andrew M. Saxe, Professor Christopher Summerfield

Derivations, simulations, and figures Lukas Braun

3.1 Introduction

To understand learning in biological and artificial neural networks one has to answer two principal questions. First, how does a neural computation solve a certain task? Second, how is this computation attained? The first question is concerned with neural circuitry, representations and computation, while the second one is concerned with the changes in neural circuitry and neural representations that are required to attain this computation. While ultimately, these two components are inextricably intertwined, in this chapter we focus on the former. In particular, we are going to study how a linear network can solve a certain task and how these solutions differ from each other. This has the advantage that we can derive statements about neural computations and representations that are independent from the learning algorithm that achieve them. While the computations performed by

the nervous system are highly nonlinear and most real-world tasks require nonlinear systems to solve them (LeCun et al., 2015), patterns of neural activity often lie on a low dimensional manifold (Gao et al., 2017; Gallego et al., 2017; Jazayeri & Ostojic, 2021; Kriegeskorte & Wei, 2021; Langdon et al., 2023) that sometimes can be decoded using linear methods (Graf et al., 2011; Mante et al., 2013). We note, that any deep linear network can be simplified to a single linear projection, as the composition of multiple linear transformations is equivalent to a single linear transformation and therefore, that the best a deep linear neural network can do is to perform linear regression (Laurent & Brecht, 2018). However, in contrast to simple linear regression, deep linear networks perform multistage distributed computations, giving rise to hidden-layer neural representations, much like the brain. Linear networks retain a non-convex optimisation problem with a high-dimensional solution manifold that in detail depends on the statistics of the training data and the shape of the neural network (Baldi & Hornik, 1989; Saxe et al., 2013; Arora et al., 2018) and therefore can be used as an analytically tractable model of representation learning (Saxe et al., 2019; Braun et al., 2022). As such, they are insightful not despite, but because of their simplicity.

We can think of the computations performed by a two-layer linear network as follows: In a first step, a stimulus is mapped to a neural representation and in the second, this neural representation is read out to a target output (Figure 3.1A). As such, they are minimal models of biological connectionist systems that project stimuli through an intricate and highly nonlinear network to a neural representation, which is then read out by a higher order brain area to facilitate action and cognition (Figure 3.1B). Accordingly, they are minimal models for multivariate pattern analysis (Haxby et al., 2014) in which neuroscientist try to map neural representations to brain function (Figure 3.1C).

Deep linear networks are highly overparametrised and as a consequence, there exist many network configurations that all have optimal performance. In this

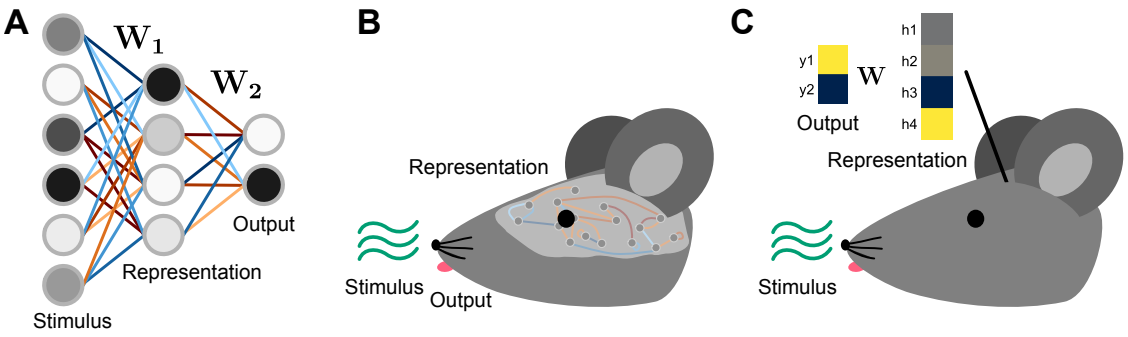

Figure 3.1: Linear networks as a model for neural computation. (**A**) Schematic of a two-layer linear network. A stimulus is mapped to a neural representation via the first-layer weights, and this representation is subsequently read out by the second-layer weights to produce an output. (**B**) Schematic of a mouse performing a lick in response to an odour. As in A, a stimulus is mapped to a a neural representation in the mouse brain which is then mapped to an output response. (**C**) Schematic of a neuroscientist's setup. As in A, a stimulus is mapped onto a neural representation via neural recordings, which is then analysed using a linear model.

chapter, we examine the variety of solutions in two-layer networks and explore how they differ. Thinking about differences of solutions may seem counter intuitive, as per definition, they all obtain optimal linear performance on the training data. However, as we will see, they differ in how they implement the global optimum, which has significant consequences for their sensitivity to input and parameter noise and furthermore on the hidden-layer neural representations and therefore for multivariate pattern analysis.

3.1.1 Related work

The solution manifold of two-layer neural networks was first described by Baldi & Hornik (1989). Subsequent work showed, under a small set of assumptions, that all minima in deep linear networks are global and equivalent to those in linear regression (Laurent & Brecht, 2018). The dissociation between general linear solutions and minimum-norm solutions has been previously studied under a set of strong assumptions (Saxe et al., 2013). The sensitivity to noise for task-specific and task-agnostic rich and lazy solutions has been previously studied numerically in

non-linear neural networks (Flesch et al., 2022) and the generalisation and transfer performance of deep linear networks have been previously investigated (Lampinen & Ganguli, 2018; Advani et al., 2020) using a teacher-student paradigm (Biehl & Schwarze, 1995; Saad & Solla, 1995). The relation between representational drift and drift on the solution manifold that resulted from stochastic parameter updates during gradient descent (Chaudhari & Soatto, 2018), has been studied on a subpart of the solution manifold in linear networks (Pashakhanloo & Koulakov, 2023) and in non-linear networks, again, relying on the teacher-student paradigm (Avidan et al., 2023; Li et al., 2024).

3.2 Preliminaries and setting

Connectionist neural systems employ a multi-step process to perform the computations required to solve a task. First, a stimulus is mapped to a neural representation and subsequently, this neural representation is read out. Here, we consider the setting in which a neural network has already learned this mapping. More formally, the parameters of the model are already optimised such that a set of input vectors $\mathbf{x}_n \in \mathbb{R}^{N_i}$ from a dateset with a total of P training pairs $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1,\dots,P}$ are mapped to their corresponding target values $\mathbf{y}_n \in \mathbb{R}^{N_o}$, such that the mean-squared error

$$\mathcal{L}_{\text{MSE}} = \frac{1}{2P} \sum_{n=1}^{P} ||\hat{\mathbf{y}}_n - \mathbf{y}_n||_2^2 \tag{3.1}$$

is at its minimum. We use two-layer linear neural networks to model connectionist neural computations

$$\hat{\mathbf{y}}_n = \mathbf{W}_2 \mathbf{W}_1 \mathbf{x}_n. \tag{3.2}$$

The input weights $\mathbf{W}_1 \in \mathbb{R}^{N_h \times N_i}$ map stimuli to a hidden-layer neural representation

$$\mathbf{h}_n = \mathbf{W}_1 \mathbf{x}_n, \tag{3.3}$$

and the readout weights $\mathbf{W}_2 \in \mathbb{R}^{N_o \times N_h}$ map the neural representations to a target output. For our analysis it is essential that any change to $\mathbf{W}_1$ that lies in the left null space of $\mathbf{X}$ does not affect the projection $\mathbf{H} = \mathbf{W}_1\mathbf{X}$. The left null space of $\mathbf{X}$ consists of vectors $\mathbf{u}$ such that $\mathbf{u}^T\mathbf{X} = \mathbf{0}$. So, modifying $\mathbf{W}_1$ by adding any linear combination of vectors from this null space leaves the projection from inputs to hidden-layer representations unchanged, as

$$(\mathbf{W}_1 + \mathbf{a}\mathbf{u}^T)\mathbf{X} = \mathbf{W}_1\mathbf{X} + \mathbf{a}\mathbf{u}^T\mathbf{X} = \mathbf{W}_1\mathbf{X}. \tag{3.4}$$

Accordingly, we can add any vector $\mathbf{v}$ that lies in the left null space of $\mathbf{H}$ to $\mathbf{W}_2$ without affecting the projection from hidden representations to outputs, as

$$(\mathbf{W}_2 + \mathbf{b}\mathbf{v}^T)\mathbf{H} = \mathbf{W}_2\mathbf{X} + \mathbf{b}\mathbf{v}^T\mathbf{H} = \mathbf{W}_2\mathbf{H}. \tag{3.5}$$

Further, it is essential to our analysis, that matrix multiplication can never increase, but only decrease the rank of its product. Specifically,

$$\text{rank}(\mathbf{AB}) \leq \min(\text{rank}(\mathbf{A}), \text{rank}(\mathbf{B})), \tag{3.6}$$

and therefore, whenever, the rank of a neural representation has collapsed, there is no way to expand it again. We begin our analysis by investigating the solution manifold of deep linear networks.

3.3 Results

3.3.1 Types of linear solutions

In our setting, two-layer linear networks are highly overparametrised. As a consequence, there exist many combinations of $\mathbf{W}_1$ and $\mathbf{W}_2$ that minimise the mean

squared error for a given set of training pairs. Laurent & Brecht (2018) showed that under

Assumption 1. *The loss function is convex and differentiable, e.g., the mean-squared error loss*

Assumption 2. *The network is not bottlenecked, i.e.,* $\min(N_i, N_o) \geq N_h$

all minima of deep linear networks over the training data are global and identical to the global optimum of the equivalent convex single-layer optimisation problem. In other words, the only minima that exist in two-layer linear neural networks, are identical to single-layer linear regression. As we show in Appendix A.1, it follows for our setting that any combination of network weights $\mathbf{\Omega}_1 = \mathbf{W}_1$ and $\mathbf{\Omega}_2 = \mathbf{W}_2$ for which

$$\mathbf{\Omega}_2\mathbf{\Omega}_1\mathbf{\Sigma}_{xx} = \mathbf{\Sigma}_{yx} \tag{3.7}$$

is a general linear solution, where

$$\mathbf{\Sigma}_{xx} = \frac{1}{P}\sum_{i=1}^{P}\mathbf{x}_n\mathbf{x}_n^T \quad \text{and} \quad \mathbf{\Sigma}_{yx} = \frac{1}{P}\sum_{i=1}^{P}\mathbf{y}_n\mathbf{x}_n^T \tag{3.8}$$

are the input and input-output correlation matrices of the training data. We emphasise that this result holds without any assumptions regarding the structure of the training data. We further emphasise, that the absence of local minima does not imply that there are no saddle points or that gradient-based algorithms converge. The collection of all combinations of input and output weights that implement a solution is the so called solution manifold

$$\mathcal{M} = \left\{\mathbf{\Omega}_2\mathbf{\Omega}_1 : \mathbf{\Omega}_2\mathbf{\Omega}_1\mathbf{\Sigma}_{xx} = \mathbf{\Sigma}_{yx}\right\}. \tag{3.9}$$

Describing all points on this manifold is challenging due to the complexity of constructing a single parameterised equation that captures the manifold's intricate

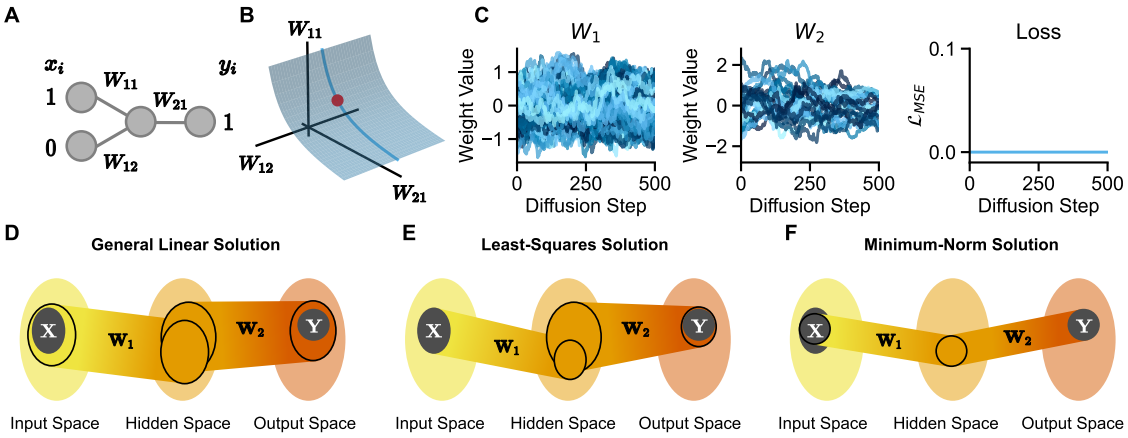

Figure 3.2: Solution manifold. (**A**) Schematic of a two-layer linear network with a single training pair, featuring three synaptic weights connecting the input, hidden, and output neurons. (**B**) Solution manifold for the network in A (blue plane). Weight $\mathbf{W}_{12}$ lies in the input null-space of the input and can be set arbitrarily, while $\mathbf{W}_{11}$ and $\mathbf{W}_{22}$ are interdependent. The least-squares solution (blue line) and minimum-norm solution (red dot) are special regions of the manifold. (**C**) A random walk (Methods 3.4.2) on the manifold of a network with many input, hidden and output neurons modifies all weights without affecting the network output, keeping the loss at minimum. (**D**) Schematic of linear projections by weight matrices: $\mathbf{W}_1$ projects from the input to hidden space, and $\mathbf{W}_2$ projects from the hidden to output space. Grey solid circles depict the range of $\mathbf{X}$ and $\mathbf{Y}$, while open circles depict the column spaces of $\mathbf{W}_1$ and $\mathbf{W}_2$. (**E**) Same as D but for least-squares solutions and for (**F**) minimum-norm solutions.

geometric structure. For example, one can scale the input weights by a factor of α and the readout weights by α^{-1} or one can add any transformation to $\mathbf{W}_1$ that is in the left null space of $\mathbf{X}$. While the solution manifold can be visualized for very small networks with a single training pair (Figure 3.2A-B), grasping the degrees of freedom in larger networks with intricate training data becomes increasingly difficult (Figure 3.2C). This difficulty arises from a combination of factors such as transformations in the null space, scaling, and permutation invariances within and across network layers. Some intuition on the structure of the solution manifold can be gained by thinking of it as points that are connected through a linear transformation. Let $\mathbf{\Omega}_1$ and $\mathbf{\Omega}_2$ be a pair of optimal network weights and let $\mathbf{Q}$ be an

arbitrary invertible matrix, such that $\mathbf{Q}^{-1}\mathbf{Q} = \mathbf{I}$, then any pair of network weights

$$\tilde{\mathbf{\Omega}}_2 = \mathbf{\Omega}_2\mathbf{Q}^{-1} \text{ and } \tilde{\mathbf{\Omega}}_1 = \mathbf{Q}\mathbf{\Omega}_1 \tag{3.10}$$

implements the same input-output map and is therefore a solution too. While helpful, this perspective on the solution manifold is limited to the case where $N_i = N_o$ and $\mathbf{\Sigma}_{yx}$ has full rank (Appendix A.4). Otherwise, the rank of either of the weight matrices may not be full, and since matrix multiplication can not increase rank, full-rank solutions cannot be attained by means of $\mathbf{Q}$. Without loss of generality, we will now define different parts of the solution manifold and subsequently, analyse and compare their computational properties.

General linear solutions

We call any combination of network weights $\mathbf{\Omega}_1 = \mathbf{W}_1$ and $\mathbf{\Omega}_2 = \mathbf{W}_2$ for which

$$\mathbf{\Omega}_2\mathbf{\Omega}_1\mathbf{\Sigma}_{xx} = \mathbf{\Sigma}_{yx} \tag{3.11}$$

a general linear solution (Appendix A.1).

Least-squares solutions

As we derive in Appendix A.2, any linear solution for which the norm of the network function is minimised, i.e.,

$$\underset{\mathbf{\Omega}_1,\mathbf{\Omega}_2}{\operatorname{argmin}} ||\mathbf{\Omega}_2\mathbf{\Omega}_1||_F^2 \quad \text{s.t.} \quad \mathbf{\Omega}_2\mathbf{\Omega}_1\mathbf{\Sigma}_{xx} = \mathbf{\Sigma}_{yx} \tag{3.12}$$

is a least-squares linear solution. All least-squares solutions obey

$$\mathbf{\Omega}_2\mathbf{\Omega}_1 = \mathbf{\Sigma}_{yx}\left(\mathbf{\Sigma}_{xx}\right)^{+}. \tag{3.13}$$

Minimum-norm solutions

Any least-squares solution for which the norm of the weight matrices is minimised, i.e.,

$$\underset{\mathbf{\Omega}_1,\mathbf{\Omega}_2}{\operatorname{argmin}} ||\mathbf{\Omega}_1||_F^2 + ||\mathbf{\Omega}_2||_F^2 \quad \text{s.t.} \quad \mathbf{\Omega}_2\mathbf{\Omega}_1 = \mathbf{\Sigma}_{yx}\left(\mathbf{\Sigma}_{xx}\right)^+ \tag{3.14}$$

is a minimum-norm solution. We show that the network weights of a minimum-norm solution obey

$$\mathbf{\Omega}_2 = \mathbf{U}\sqrt{\mathbf{S}}\mathbf{R}^T \text{ and } \mathbf{\Omega}_1 = \mathbf{R}\sqrt{\mathbf{S}}\mathbf{V}^T, \tag{3.15}$$

where

$$\text{SVD}\left(\mathbf{\Sigma}_{yx}\left(\mathbf{\Sigma}_{xx}\right)^+\right) = \mathbf{U}\mathbf{S}\mathbf{V}^T \tag{3.16}$$

is the compact SVD of any least-squares solution and $\mathbf{R}$ is an arbitrary (semi-)orthonormal matrix of appropriate shape. For a proof see Appendix A.3. We note that this relation has been previously derived under a set of strong assumptions, namely that $N_i = N_o$ and that $\mathbf{\Sigma}_{yx}$ has full rank (Saxe et al., 2019, Appendix S14-S15).

3.3.2 Dissociating computational properties of varying types of linear solutions

General, least-squares and minimum-norm solutions all achieve the same minimal error on the training data, however, they differ in how they implement the solution. We begin by developing an intuition behind the operations performed by the two weight matrices. In a two-layer linear network, $\mathbf{W}_1$ projects from the input space to the space of hidden-layer representations and $\mathbf{W}_2$ projects from the space of hidden-layer representations to the output space. Importantly, depending on the exact configuration of the weight matrices, the subspace which is covered by each

of the matrices varies. In particular, general solutions can have $\mathbf{W}_1$ that project from parts of the input space that are not actually covered by inputs of the training data, i.e., the left null space of $\mathbf{X}$. Similarly, $\mathbf{W}_2$ may project from parts of the hidden space that are not covered by hidden-layer representations, i.e., the left null space of $\mathbf{H}$. In other words, weight matrices of a general linear solution may project from and to parts of the input, hidden and output space that is not encoded by the task (Figure 3.2D). In contrast, $\mathbf{W}_1$ of least-squares solutions are guaranteed to only project from parts of the input space that are actually covered by inputs of the task. However, the projection of $\mathbf{W}_2$ remains unrestricted (Figure 3.2E). Finally, minimum-norm solutions are guaranteed to have $\mathbf{W}_1$ that only project from task relevant parts of the input to the hidden space and $\mathbf{W}_2$ project from exactly this part of the hidden space to only task relevant parts of the output space (Figure 3.2F). We note, that the space covered by task-relevant parts of the inputs or outputs may be smaller than the space covered by the inputs or outputs themselves, as the rank of the task is

$$\operatorname{rank}\left(\mathbf{\Sigma}_{yx}\right) \leq \min(\operatorname{rank}(\mathbf{X}), \operatorname{rank}(\mathbf{Y})). \tag{3.17}$$

Therefore, $\mathbf{W}_1$ of a minimum-norm solution immediately discards all task irrelevant information; and $\mathbf{W}_2$ only project into the subspace of outputs that is task relevant. We depict random walks (Method 3.4.2) on the general, least-squares and minimum-norm solution and their respective network functions in Figure 3.3A-C. Next, we discuss implications of these differences.

Norm of network function and weight matrices

Since for a general linear solution, weights that are associated with the null space of the input and hidden space can be adjusted arbitrarily without affecting how inputs are processed, there is no upper limit on the norm of the network function or

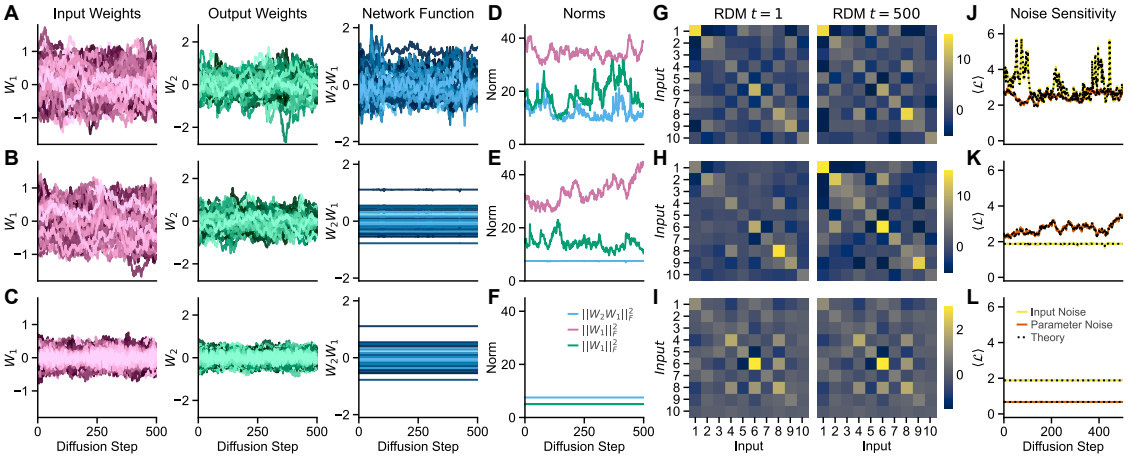

Figure 3.3: Computational properties of varying types of linear solutions. (**A**) Input and output weights and resulting network function during a random walk on the general, (**B**) least-squares and (**C**) minimum-norm solution manifold. (**D**) Temporal evolution of norms during a random walk on the general, (**E**) least-squares and (**F**) minimum-norm solution manifold. (**G**) Representational similarity of hidden-layer neural representations at the first and last time step of a random walk on the general, (**H**) least-squares and (**I**) minimum-norm solution manifold. Note how the representational similarity remains steady for minimum-norm solutions despite changes in the input and output weights. (**J**) Expected loss under input and parameter noise throughout the random walk on the general, (**K**) least-squares and (**L**) minimum-norm solution manifold.

individual weight matrices (Figure 3.3D). In contrast, for least-squares solutions, input-weights are restricted to project from the space that is covered by training data. As a consequence, any change that is happening to $\mathbf{W}_1$ has to be compensated by $\mathbf{W}_2$ and the norm of the network function has to remain constant and minimal. However, changes to one weight matrix that are compensated by the other and perturbations to $\mathbf{W}_2$ that lie in the null space of the hidden space are still possible and therefore there is no restriction on the norm of the individual weight matrices (Figure 3.3E). Finally, for minimum-norm solutions, changes to the projections in the hidden space must be tracked precisely by the readout weights, and the norm of the network weights and the network function must remain minimal and identical (Figure 3.3F).

Hidden-layer representations

General linear solutions and least-square solutions do not constrain the shape of hidden-layer representations as long as they preserve sufficient information for the task to be solvable by the readout weights. As a consequence, a general linear solutions can achieve almost arbitrary representational similarity (Figure 3.3G-H). Again, while limited, an intuition why this is true can be gained by thinking of the representational similarity by means of $\mathbf{Q}$. We can choose any invertible $\mathbf{Q}$ to modify the representational similarity

$$\mathrm{RSM}_{ij} = \mathbf{x}_i^T \mathbf{\Omega}_1^T \mathbf{Q}^T \mathbf{Q} \mathbf{\Omega}_1 \mathbf{x}_j. \tag{3.18}$$

In contrast, minimum-norm solutions reduce the degrees of freedom in the hidden layer representation to a rotational invariance, which necessitate neural representations that obey

$$\begin{aligned} \mathrm{RSM}_{ij} &= \mathbf{x}_i^T \mathbf{\Omega}_1^T \mathbf{\Omega}_1 \mathbf{x}_j \\ &= \mathbf{x}_i^T \mathbf{V} \sqrt{\mathbf{S}} \mathbf{R}^T \mathbf{R} \sqrt{\mathbf{S}} \mathbf{V}^T \mathbf{x}_j \\ &= \mathbf{x}_i^T \mathbf{V} \mathbf{S} \mathbf{V}^T \mathbf{x}_j. \end{aligned} \tag{3.19}$$

Here, the (semi-)orthonormal matrix $\mathbf{R}$ cancels out, leaving the representational similarity determined entirely by the left-singular vectors and singular values of the least-squares solution, which depend solely on the training data (Equation 3.16). Thus, hidden-layer neural representations of minimum-norm solutions are task-specific, and reflect the underlying correlational structure of the task (Figure 3.3I).

Sensitivity to noise

Biological neural systems are subject to a multitude of internal and external sources of noise, which range from fluctuations in synaptic efficacy and spontaneous neural activity to variability in incoming sensory signals. Therefore, solutions that exhibit

robustness to such noise are advantageous, as they enable the neural circuitry to maintain reliable function. In Appendix A.5, we show that the expected loss of a linear solution under additive, independent and identically distributed, zero-centred input noise $\xi_{\mathbf{x}_n}$ with variance σ_x^2 is

$$\left\langle \frac{1}{2P} \sum_{n=1}^{P} ||\mathbf{\Omega}_2\mathbf{\Omega}_1 \left(\mathbf{x}_n + \xi_{\mathbf{x}_n}\right) - \mathbf{y}_n||_2^2 \right\rangle = \frac{\sigma_{\mathbf{x}}^2}{2} ||\mathbf{\Omega}_2\mathbf{\Omega}_1||_F^2 + c, \tag{3.20}$$

where c is a noise-independent constant that only depends on the statistics of the training data. Since every minimum-norm solution is a least-squares solution and least-squares solutions are solutions that minimise the norm of the network function, both minimise sensitivity to input noise (Figure 3.3J-L). Furthermore, we show that the expected loss under additive, independent and identically distributed, zero-centred noise in the parameters, i.e., in the first and second layer weights, with variance σ_1^2 and σ_2^2 is

$$\begin{aligned} &\left\langle \frac{1}{2P} \sum_{n=1}^{P} || \left(\mathbf{\Omega}_2 + \mathbf{\Xi}_2\right) \left(\mathbf{\Omega}_1 + \mathbf{\Xi}_1\right) \mathbf{x}_n - \mathbf{y}_n||_2^2 \right\rangle \\ &= \frac{1}{2} \left(\sigma_1^2 ||\mathbf{\Omega}_2||_F^2 \operatorname{Tr}(\mathbf{\Sigma}_{xx}) + N_o \sigma_2^2 \operatorname{Tr}(\mathbf{\Omega}_1^T \mathbf{\Omega}_1 \mathbf{\Sigma}_{xx}) + N_h \sigma_1^2 N_o \sigma_2^2 \operatorname{Tr}(\mathbf{\Sigma}_{xx}) \right). \end{aligned} \tag{3.21}$$

Both general and least-squares solutions are sensitive to parameter noise, whereas minimum-norm solutions exhibit a low and constant expected loss (Figure 3.3J-L). Thus, adopting a minimum-norm solution with task-specific neural representations offers a clear computational advantage.

3.3.3 Consequences for neural analyses and the brain

In the previous section, we established the existence of a solution manifold and differentiated between computational properties of general, least-squares, and minimum-norm solutions. Specifically, our analysis shows that for general and least-squares solutions, changes in the network function are not necessarily reflected in hidden-

layer neural representations. Since any change in $\mathbf{W}_1$ can be compensated by adjustments in $\mathbf{W}_2$, changes in hidden-layer neural representations can occur without changes in the underlying network function. Vice versa, learning can occur entirely within $\mathbf{W}_2$, leaving neural representations unchanged while changing the underlying network function. However, in the minimum-norm regime, changes in the underlying function must lead to corresponding changes in neural representations. We now explore the implications of these findings for multivariate pattern analysis of neural recordings, the synaptic stability-plasticity dilemma, and representational drift.

Linear predictivity

Linear predictivity, as for example used in Yamins et al. (2014), assesses how well the activation patterns from one model or recording can linearly predict the activations of another model or recording. When two models exhibit high linear predictivity, a common conclusion is that they process information similarly, or that one serves as a good model for the other. However, as our analysis reveals, hidden-layer neural representations of two-layer linear networks have many degrees of freedom and therefore, high linear predictivity does not imply functional alignment. For instance, linear decoding of the least-squares solution from the minimum-norm solution (or vice versa) yields an $R^2 \approx 0.5$, despite both networks implementing the same function (Figure 3.4A). Similarly, a network that predicts $-\mathbf{y}$ can use identical hidden-layer representations, resulting in perfect linear predictivity, even though it implements the inverse function of the model under comparison (Figure 3.4A).

Representational similarity analysis

Representational similarity analysis aims to decode computational similarity of two networks by examining how neural responses within or across artificial and biological networks relate to one another (Kriegeskorte, Mur, & Bandettini, 2008;

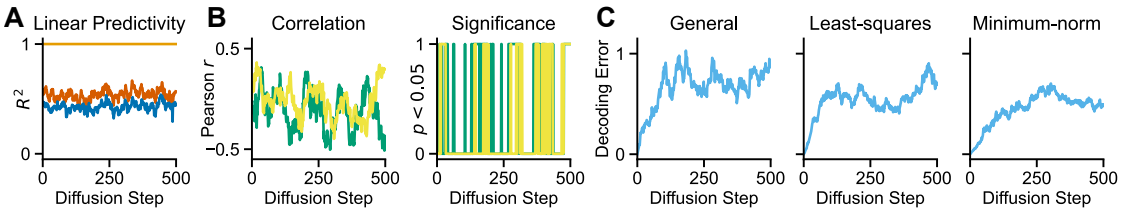

Figure 3.4: Consequences for neural analyses and the brain. (**A**) Linear predictivity of hidden-layer representations of the least-squares solution on the representations of the general (blue) or least-squares solution (red) throughout a random walk on the solution manifold. (**B**) Pearson correlation between the hidden-representations of the least-squares solution and the general (green) and minimum-norm solution (green), and whether their representational similarity is statistically significant. (**C**) Error of linear decoder that was fit on the hidden-representation of the first time-step of the random walk. The neural representations are drifting and therefore, the performance of the decoder deteriorates, while the underlying function remains unchanged.

Haxby et al., 2014). From our analytical results it follows that in the general and least-squares regimes, the geometric relationships between representations in linear networks do not carry information about the underlying computation. Hidden-layer neural representations are only determined by the underlying function if the network operates in the minimum-norm regime. This has profound implications when using representational similarities to compare computations across networks. For example, performing representational similarity analysis between the least-squares solution and the general linear or minimum-norm solution yield results that randomly fluctuate between statistically significant and insignificant similarity, despite the fact that they implement the same function (Figure 3.4B).

Synaptic stability

The stability-plasticity dilemma states that an underlying challenge of learning is that neural systems must remain plastic enough to integrate new knowledge while maintaining stability to preserve previously learned information (Grossberg, 1987; Abraham & Robins, 2005). This dilemma has driven the development of algorithms designed to balance stability and plasticity at the level of individual synapses during

continual learning (Kirkpatrick et al., 2017; Zenke et al., 2017). However, from a network perspective, synaptic stability is not necessary for preserving knowledge (see e.g. Figure 3.3A-C). Due to the existence of a solution manifold, knowledge is not encoded in specific synapses but through their interactions within the network. Furthermore, in the case of minimum-norm solutions, all synapses can be altered without affecting the representational similarity of stimuli (see e.g. Figure 3.3C,I).

Drifting neural representations

From an intuitive standpoint, one might infer that if a stimulus ensures both behavioural and perceptual stability, that the associated neural representations should remain stable too (Rule et al., 2019; Driscoll et al., 2022). However, this intuition has been challenged by the observation of drifting neural representations. Our analysis shows that the existence of a solution manifold in two-layer linear networks enables hidden-layer neural representations to change without a change in network function and therefore that behavioural and perceptual stability in theory are dissociated from stable neural representations. Therefore, an optimal linear decoder on the neural representation may quickly become inaccurate even though the underlying function remains unchanged (Figure 3.4C). We note that if the network implements a minimum-norm solution, representations may drift but that representational similarity must remain stable.

3.4 Methods

3.4.1 Network and task

To study our analytical results, we used two-layer linear networks with $N_i = 12$, $N_h = 20$, and $N_o = 9$ neurons. Networks implemented a teacher-student task for which inputs are uncorrelated Gaussian noise and targets are generated from these

samples using a random linear projection. Specifically,

$$\mathbf{X} \sim \mathcal{N}(\mu = 0, \sigma = 1/2) \in \mathcal{R}^{N_i \times P} \tag{3.22}$$

and

$$\mathbf{Y} = \mathbf{A}\mathbf{X} \in \mathcal{R}^{N_o \times P} \quad \text{with} \quad \mathbf{A} \sim \mathcal{N}(\mu = 0, \sigma = 1/\sqrt{N_i}) \in \mathcal{R}^{N_i \times N_o}. \tag{3.23}$$

To ensure that the left null space of the inputs was not empty, we set $P = 10 < N_i$.

3.4.2 Random walk

Given a two-layer linear network with weight matrices $\mathbf{W}_1$ and $\mathbf{W}_2$, we implemented random walks on the solution manifold on the general linear solution as follows. First, we randomly initialised weight matrices using a random normal initialisation with zero mean and standard deviation $1/\sqrt{N_i}$ and $1/\sqrt{N_h}$ respectively. For each step, we first diffused both weight matrices by

$$\mathbf{W}(t+1) = (1-\alpha)\mathbf{W}(t) + \alpha\xi \quad \text{with} \quad \xi \sim \mathcal{N}(\mu = 0, \sigma = 5), \tag{3.24}$$

and $\alpha = 0.0125$, and then performed gradient descent on the mean squared error of the task with learning rate $\eta = 0.01$ to return back to the solution manifold. Similarly, for the random walk on the least-squares solution, we sampled initial $\mathbf{W}_1$ and $\mathbf{W}_2$ randomly, then, restricted $\mathbf{W}_1$ to the space covered by the inputs

$$\mathbf{W}_1 = \mathbf{W}_1\mathbf{G}\mathbf{G}^T \quad \text{with} \quad \text{SVD}(\mathbf{X}) = \mathbf{G}\mathbf{B}\mathbf{H}^T, \tag{3.25}$$

and subsequently performed gradient descent on $\mathbf{W}_2$ only to return to the solution manifold. The random walk on the minimum-norm solution was implemented by

computing the compact SVD of the least-squares solution

$$\text{SVD}\left(\mathbf{\Sigma}_{yx}(\mathbf{\Sigma}_{xx})^{+}\right) = \mathbf{USV}^T \tag{3.26}$$

and sampling a random semi-orthogonal matrix $\mathbf{R}$ from the random orthogonal group. Then, for each step, we defused $\mathbf{R}$ with the same operation as described for the weight matrices and then re-orthogonalised by

$$\mathbf{R}(t+1) = \mathbf{R}_1\mathbf{R}_2^T \quad \text{with} \quad \text{SVD}(\mathbf{R}(t)) = \mathbf{R}_1\mathbf{C}\mathbf{R}_2^T. \tag{3.27}$$

Network weights were then set to

$$\mathbf{W}_1(t) = \mathbf{R}(t+1)\sqrt{\mathbf{S}}\mathbf{V}^T \quad \text{and} \quad \mathbf{W}_2(t) = \mathbf{U}\sqrt{\mathbf{S}}\mathbf{R}(t+1)^T. \tag{3.28}$$

3.4.3 Linear predictivity

For Figure 3.4A, we fit a linear model to predict the hidden-layer activations of one model by the hidden-layer activations of another for each time step of the random walk, and subsequently calculated the R^2 score of that fit.

3.4.4 Representational similarity analysis

Representational similarity matrices in Figure 3.3G show the inner product covariance matrix of the hidden-layer representations

$$\text{RSM}_{ij} = \mathbf{x}_i^T\mathbf{W}_1^T\mathbf{W}_1\mathbf{x}_j. \tag{3.29}$$

We performed representational similarity analysis between two models by calculating the euclidean distance matrix of their respective hidden-layer representations

$$\mathrm{RSM}_{ij} = ||\mathbf{h}_i - \mathbf{h}_j||_2 \tag{3.30}$$

and subsequently compared their off-diagonal values using Pearson correlation coefficients.

Drift analysis

To analyse drift, we fit a linear decoder on the hidden-layer representation of a model at the beginning of the random walk

$$\tilde{\mathbf{W}} = \mathbf{\Sigma}_{yx}\mathbf{W}_1(1)^T(\mathbf{W}_1(1)\mathbf{X}\mathbf{X}^T\mathbf{W}_1(1)^T)^+ \tag{3.31}$$

and subsequently calculated the mean squared error of that classifier, given the hidden-layer representation at each time-step.

3.5 Discussion

Our analysis is limited to the solution manifold of linear networks and therefore it is not immediately apparent how it applies to non-linear networks and the brain. However, similar degrees of freedoms exist in non-linear networks (Sussmann, 1992; Fukumizu & Amari, 2000), particularly in those with ReLU activation functions (Dinh et al., 2017). While non-linearities enable networks to decorrelate inputs and capture complex relationships, and with enough layers, can map inputs to almost arbitrary hidden-layer representations and representational similarities, they still broadly operate in the two regimes discussed in this chapter, using task-specific or task-agnostic representations, often referred to as the *rich* and *lazy* regimes

(Chizat et al., 2019; Woodworth et al., 2020; Flesch et al., 2022; Farrell et al., 2023). In neuroscience, it is common practice to first extract low-dimensional representations from high-dimensional neural activity patterns and then apply a linear decoder. Thus, even though out two-layer linear model is a stark simplification of brain function, it closely mirrors this common practice. Finally, we note that if inputs are one-hot encoded, each input can be assigned to any hidden layer representation, as each input is essentially a column in $\mathbf{W}_1$. In fact, as we show in Appendix A.7, in this case any non-linear network is achievable by the identical linear network. In conclusion, while the solution manifold of two-layer linear networks are a simple model of the non-linear manifolds of artificial and biological systems, their analytical tractability offers value insights, where our intuitions about these more complex systems fall short.

Exact learning dynamics of deep linear networks with prior knowledge

---

This chapter is based on published work (Braun et al., 2022). If not indicated otherwise, contributions to the research presented in this chapter are as follows:

Supervision Professor Andrew M. Saxe, Professor James E. Fitzgerald

Derivations, simulations, and figures Lukas Braun

Additional results on the dynamics of alignment, reversal learning and knowledge revision that were derived in collaboration with Cleméntine C.J. Dominé can be found in the original publication.

4.1 Introduction

Learning is known to critically depend on prior knowledge in humans, non-human animals, and artificial neural networks. This impact of prior knowledge on learning in mind and machine becomes evident in a wide range of learning paradigms, including transfer learning (Perkins & Salomon, 1992; Zhuang et al., 2020; Lampinen & Ganguli, 2018; Gerace et al., 2022), curriculum learning and shaping (Krueger & Dayan, 2009; Bengio et al., 2009; Dekker et al., 2022; Saglietti et al., 2022; Hwa Lee et al., 2024), meta-learning (Schmidhuber, 1987; Finn et al., 2017; Javed & White, 2019), and in particular during continual learning (McCloskey & Cohen, 1989).

That is, how exactly does learning new things depend on the initial state of the underlying neural system, which has been shaped throughout the acquisition of prior knowledge; and furthermore, how well does the neural system perform on prior tasks after learning? Although the importance of prior knowledge in learning is evident, our theoretical understanding remains limited, and fundamental questions remain about the implicit inductive biases of neural networks trained from structured initial states. Here, we derive exact solutions to the dynamics of learning for two-layer linear networks for a broad class of initial conditions (Figure 4.1A), by generalising Fukumizu's matrix Riccati solution (Fukumizu, 1998). We obtain explicit expressions for the evolving network function, hidden representational similarity, and neural tangent kernel over training for a broad class of initialisations. Our analytical solution reveals an intricate and systematic dependence of a network's initial state on the learning dynamics (Figure 4.1B-C), and generalises to the continual learning setting in which the initial weights encode previously acquired knowledge (Figure 4.1D). Taken together, our results provide a mathematical toolkit for understanding the impact of initial conditions on learning in two-layer linear neural networks.

4.1.1 Prior work

Our work builds upon a large body of analytical studies, which have investigated the intricate fixed point structure (Baldi & Hornik, 1989; Saxe et al., 2013; Laurent & Brecht, 2018), nonlinear learning dynamics (Saxe et al., 2013; Fukumizu, 1998; Tarmoun et al., 2021), convergence properties (Arora et al., 2018; Min et al., 2021), generalisation (Lampinen & Ganguli, 2018; Advani et al., 2020), and implicit bias of gradient descent (Gunasekar et al., 2018; Ji & Telgarsky, 2018; Arora et al., 2019; Varre, Vladarean, et al., 2024) in deep linear networks.

Other work has focused on the influence of different random initialisations on

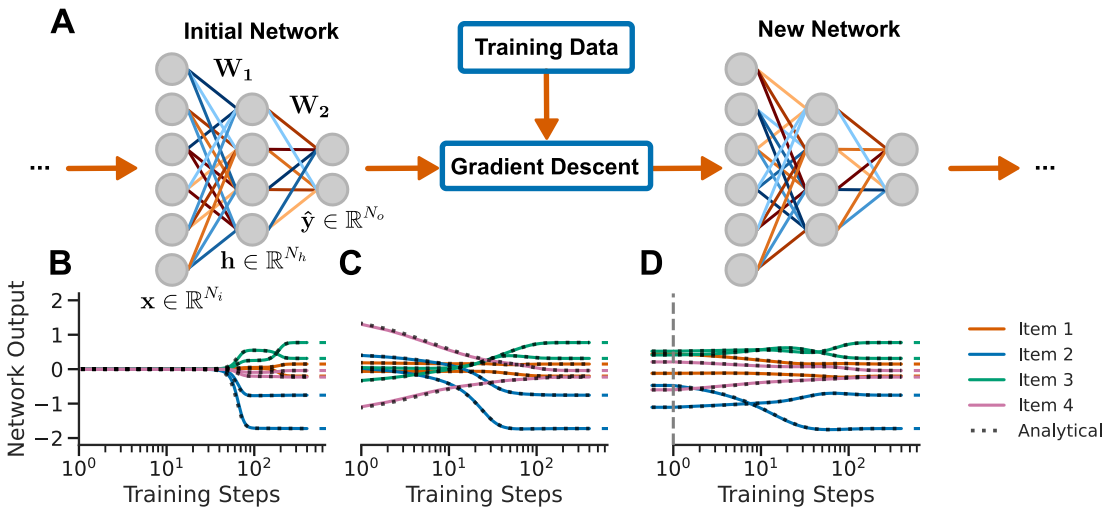

Figure 4.1: Learning with prior knowledge. (**A**) A deep linear network with N_i input, N_h hidden, and N_o output neurons is trained from a particular initialisation using gradient descent. The exact learning dynamics depend in detail on the initialisation: (**B**) Network output throughout training, starting from small random weights, (**C**) large random weights, (**D**) or network weights derived from prior training on another task. Solid and dotted lines indicate simulations and analytical solutions we derive in this work respectively. Short lines indicate target values.

learning dynamics in linear and nonlinear neural networks. Guided by the central limit theorem, they unravelled the relationship between initialisation and vanishing and exploding gradients, leading to enhanced initialisation techniques (LeCun et al., 2002; Glorot & Bengio, 2010; Saxe et al., 2013; K. He et al., 2015; Mishkin & Matas, 2015; Pennington et al., 2017; Xiao et al., 2018; H. Zhang et al., 2018; Burkholz & Dubatovka, 2019; Lu et al., 2019; Huang et al., 2020; Bachlechner et al., 2021). While these approaches overcome the vanishing and exploding gradient problem, they do not provide insights into exact learning dynamics, convergence properties, and structure of neural representations.

It has been shown that the scale of the random initial weights mediates a transition from the so called *rich* to *lazy* regime in both linear as well as nonlinear networks (Chizat et al., 2019; Woodworth et al., 2020; Flesch et al., 2022; Farrell et al., 2023; Varre, Vladarean, et al., 2024; Tu et al., 2024). These regimes are usually associated with small and large initial weights, respectively, leading to

qualitatively different learning dynamics and neural representations. Specifically, small initial weights have been linked to slow, step-like nonlinear learning dynamics that result in *rich*, task-specific representations (Saxe et al., 2013; Chizat et al., 2019; Woodworth et al., 2020; Varre, Vladarean, et al., 2024). In contrast, large initial weights have been associated with fast, exponential learning dynamics that exploit the initial, high-dimensional random projections of the data, leading to *lazy*, task-agnostic representations (Du et al., 2018; Jacot et al., 2018; J. Lee et al., 2019; Allen-Zhu et al., 2019; Du et al., 2019; Varre, Vladarean, et al., 2024). However, an exact link between the network's initial state, learning dynamics, and neural representations remains unclear.

Another line of theoretical work has derived ordinary differential equations to describe the exact learning dynamics during online-learning in the teacher-student paradigm in linear and nonlinear networks (Biehl & Schwarze, 1995; Saad & Solla, 1995; Goldt et al., 2019), and applied those equations to the continual learning setting (S. Lee et al., 2021; Shan et al., 2024). However, solving these equations requires numerical integration.

Contributions

- We derive an explicit solution for the temporal dynamics of the network function, representational similarity, and finite-width neural tangent kernel of two-layer deep linear networks for a broad class of initial conditions.

- We characterise a set of random initial network states that exhibit fast, exponential learning dynamics and yet converge to rich neural representations. Dissociating fast and slow learning dynamics from the *rich* and *lazy* learning regimes.

- We provide exact solutions to the learning dynamics in the continual learning setting.

4.2 Preliminaries and setting

Consider a supervised learning task in which input vectors $\mathbf{x}_n \in \mathbb{R}^{N_i}$ from a set of P training pairs $\mathcal{D} = \{(x_n, y_n)\}_{n=1\dots P}$ have to be associated with their target output vectors $\mathbf{y}_n \in \mathbb{R}^{N_o}$. We learn this task with a two-layer linear network model (Fig. 4.1A) that produces the output prediction

$$\hat{\mathbf{y}}_n = \mathbf{W}_2\mathbf{W}_1\mathbf{x}_n, \tag{4.1}$$

with weight matrices $\mathbf{W}_1 \in \mathbb{R}^{N_h \times N_i}$ and $\mathbf{W}_2 \in \mathbb{R}^{N_o \times N_h}$, where N_h is the number of hidden units. The network's weights are optimised using full batch gradient descent (General Methods 2.6) with learning rate η (or respectively time constant $\tau = 1/\eta$) on the mean squared error loss

$$\mathcal{L}_{\text{MSE}} = \frac{1}{2P}||\hat{\mathbf{Y}} - \mathbf{Y}||_F^2. \tag{4.2}$$

The input and input-output correlation matrices of the dataset are

$$\mathbf{\Sigma}_{xx} = \frac{1}{P}\sum_{i=1}^{P}\mathbf{x}_n\mathbf{x}_n^T \in \mathbb{R}^{N_i \times N_i} \quad \text{and} \quad \mathbf{\Sigma}_{yx} = \frac{1}{P}\sum_{i=1}^{P}\mathbf{y}_n\mathbf{x}_n^T \in \mathbb{R}^{N_0 \times N_i}. \tag{4.3}$$

Our goal is then to derive a solution for the time trajectory of the network function and its resulting output and internal representations throughout training as a function of the network's initial state and the statistics of the task.

To this end, we build upon the seminal work of Fukumizu (1998), which showed that the gradient flow (General Methods 2.6.3) of the matrix

$$\mathbf{Q}\mathbf{Q}^T = \begin{bmatrix} \mathbf{W}_1^T\mathbf{W}_1 & \mathbf{W}_1^T\mathbf{W}_2^T \\ \mathbf{W}_2\mathbf{W}_1 & \mathbf{W}_2\mathbf{W}_2^T \end{bmatrix} \tag{4.4}$$

can be written as a matrix Riccati equation

$$\tau \frac{d}{dt}\left(\mathbf{Q}\mathbf{Q}^T\right) = \mathbf{F}\mathbf{Q}\mathbf{Q}^T + \mathbf{Q}\mathbf{Q}^T\mathbf{F} - \left(\mathbf{Q}\mathbf{Q}^T\right)^2, \tag{4.5}$$

with known solution (Yan et al., 1994; Fukumizu, 1998), where

$$\mathbf{F} = \begin{bmatrix} \mathbf{0} & \mathbf{\Sigma}_{yx}^T \\ \mathbf{\Sigma}_{yx} & \mathbf{0} \end{bmatrix}. \tag{4.6}$$

Instead of directly solving for the temporal dynamics of the weight matrices $\mathbf{W}_1(t)$ and $\mathbf{W}_2(t)$, this approach is tracking the temporal dynamics of the network function (and its transpose) as the off-diagonal blocks of $\mathbf{Q}\mathbf{Q}^T$ and the correlation structure of the weight matrices as its on-diagonal blocks. In particular, defining the initial state

$$\mathbf{Q}(0) = \begin{bmatrix} \mathbf{W}_1^T(0) \\ \mathbf{W}_2(0) \end{bmatrix}, \tag{4.7}$$

the solution to the Ricatti equation is then

$$\mathbf{Q}\mathbf{Q}^T(t) = e^{\mathbf{F}\frac{t}{\tau}}\mathbf{Q}(0)\left[\mathbf{I} + \frac{1}{2}\mathbf{Q}(0)^T\left(e^{\mathbf{F}\frac{t}{\tau}}\mathbf{F}^{-1}e^{\mathbf{F}\frac{t}{\tau}} - \mathbf{F}^{-1}\right)\mathbf{Q}(0)\right]^{-1}\mathbf{Q}(0)^T e^{\mathbf{F}\frac{t}{\tau}}, \tag{4.8}$$

if the following five assumptions hold:

Assumption 3. *The inputs are whitened, that is,* $\mathbf{\Sigma}_{xx} = \mathbf{I}$

Assumption 4. *The initial network weights are zero-balanced, that is,* $\mathbf{W}_1(0)\mathbf{W}_1(0)^T = \mathbf{W}_2(0)^T\mathbf{W}_2(0)$

Assumption 5. *The input and output dimension are identical, that is,* $N_i = N_0$

Assumption 6. *The input-output correlation of the task has full rank, that is* $\operatorname{rank}\left(\mathbf{\Sigma}_{yx}\right) = N_i = N_o$

Assumption 7. *The initial network weights* $\mathbf{Q}(0)$ *have full rank, that is,*
$\text{rank}(\mathbf{W}_1(0)) = \text{rank}(\mathbf{W}_2(0)) = \min(N_o, N_h, N_i)$

Assumptions 3 and 4 are required to derive the matrix Riccatti equation as shown in Appendix A of Braun et al. (2022). Assumptions 5 and 6 are required for $\mathbf{F}$ to be non-singular. And finally, Assumption 7 is required during the derivation of the solution to the Ricatti equation (Yan et al., 1994; Fukumizu, 1998).

The solution provided by Fukumizu (1998) comes with limitations. First, it involves general matrix exponentials and inverses, which tend to be numerically unstable. Second, the asymptotic behaviour of the equation as time goes to infinity cannot be explicitly derived. Third, the equation holds true only for equal input and output dimensions, limiting the variety of learning tasks that can be analysed. And lastly, the temporal dynamics of the individual quadrants of the $\mathbf{QQ}^T$ matrix, e.g., the network function, can't be expressed independently of the other quadrants. We will address these limitations in the following.

4.3 Results

4.3.1 Exact learning dynamics

Here, we present an exact and numerically stable solution for $\mathbf{QQ}^T(t)$ that reveals the learning dynamics, convergence behaviour and generalisation properties of two-layer linear networks from a broad class of initial conditions. First, we note that if the inputs are whitened (Assumption 3), any combination of network weights for which

$$\mathbf{W}_2\mathbf{W}_1 = \mathbf{\Sigma}_{yx} \tag{4.9}$$

is a solution (Appendix A.1). Then, let

$$\text{SVD}\left(\mathbf{W}_2(0)\mathbf{W}_1(0)\right) = \mathbf{USV}^T \text{ and } \text{SVD}\left(\mathbf{\Sigma}_{yx}\right) = \tilde{\mathbf{U}}\tilde{\mathbf{S}}\tilde{\mathbf{V}}^T \tag{4.10}$$

denote the compact SVD (General Methods 2.7) of the initial network function and the input-output correlation of the task. Further, let

$$\mathbf{B} = \mathbf{U}^T\tilde{\mathbf{U}} + \mathbf{V}^T\tilde{\mathbf{V}} \quad \text{and} \quad \mathbf{C} = \mathbf{U}^T\tilde{\mathbf{U}} - \mathbf{V}^T\tilde{\mathbf{V}} \tag{4.11}$$

and define

Assumption 8. $\mathbf{B}$ *is non-singular.*

Assumption 8 implies that $\mathbf{B}$ must be square, which in turn requires the input-output correlation of the task to have full rank (Assumption 6). Additionally, the assumption requires that the network is not bottlenecked, meaning $N_h \geq \min(N_i, N_o)$.

Theorem 4.1. *Under the assumptions of whitened inputs (Assumption 3), zero-balanced and full-rank initial weights (Assumptions 4 and 7), and* $\mathbf{B}$ *non-singular (Assumption 8), the temporal dynamics of* $\mathbf{QQ}^T$ *are*

$$\begin{aligned}\mathbf{QQ}^T(t) = \mathbf{Z}\Bigg[&4e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{B}^{-1}\mathbf{S}^{-1}\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}} + \left(\mathbf{I} - e^{-2\tilde{\mathbf{S}}\frac{t}{\tau}}\right)\tilde{\mathbf{S}}^{-1} \\ &- e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{B}^{-1}\mathbf{C}\left(e^{-2\tilde{\mathbf{S}}\frac{t}{\tau}} - \mathbf{I}\right)\tilde{\mathbf{S}}^{-1}\mathbf{C}^T(\mathbf{B}^T)^{-1}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}} \\ &+4\frac{t}{\tau}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{B}^{-1}\left(2\mathbf{I} - \mathbf{V}^T\tilde{\mathbf{V}}\tilde{\mathbf{V}}^T\mathbf{V} - \mathbf{U}^T\tilde{\mathbf{U}}\tilde{\mathbf{U}}^T\mathbf{U}\right)\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\Bigg]^{-1}\mathbf{Z}^T\end{aligned} \tag{4.12}$$

with

$$\mathbf{Z} = \begin{bmatrix}\tilde{\mathbf{V}}\left(\mathbf{I} - e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{C}^T\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\right) + 2\left(\mathbf{I} - \tilde{\mathbf{V}}\tilde{\mathbf{V}}^T\right)\mathbf{V}\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}} \\ \tilde{\mathbf{U}}\left(\mathbf{I} + e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{C}^T\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\right) + 2\left(\mathbf{I} - \tilde{\mathbf{U}}\tilde{\mathbf{U}}^T\right)\mathbf{U}\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\end{bmatrix}. \tag{4.13}$$

For a proof of Theorem 4.1 please refer to Appendix B.1.

Given our solution, the four quadrants of the $\mathbf{QQ}^T$ matrix are fully separable by means of the upper and lower parts of $\mathbf{Z}$. Therefore, the temporal dynamics of the lower-left quadrant $(\mathbf{W}_2\mathbf{W}_1)(t)$ provide exact temporal dynamics of the network

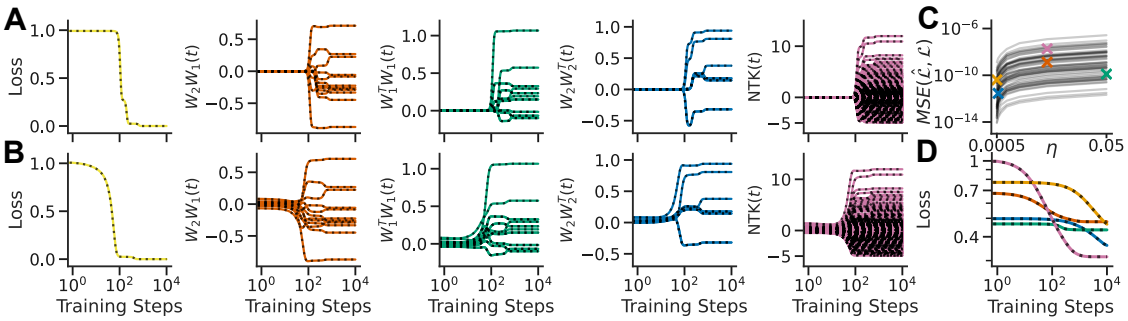

Figure 4.2: Exact learning dynamics. (**A**) Temporal dynamics of the the loss, network function, correlation of input and output weights and the neural tangent kernel for small initial weights and (**B**) large initial weights. Solid and dotted lines indicate simulations and analytical solutions respectively. (**C**) Deviation from analytical loss from the numerical loss for $n = 50$ independent simulations with varying network architecture and training data across a range of learning rates. The deviation decreases with decreasing learning rate. Coloured markers indicate five example networks with corresponding (**D**) numerical and analytical loss curves.

function

$$\hat{\mathbf{Y}}(t) = (\mathbf{W}_2\mathbf{W}_1)(t)\mathbf{X}, \tag{4.14}$$

and the mean square loss

$$\mathcal{L}_{\text{MSE}}(t) = \frac{1}{2P} ||(\mathbf{W}_2\mathbf{W}_1)(t)\mathbf{X} - \mathbf{Y}||_F^2 . \tag{4.15}$$

Whereas, the on-diagonal entries $(\mathbf{W}_1^T\mathbf{W}_1)(t)$ and $(\mathbf{W}_2\mathbf{W}_2^T)(t)$, provide exact solutions for the temporal dynamics of the network's RSM (Kriegeskorte, Mur, & Bandettini, 2008) of the hidden layer representations

$$\text{RSM}(t) = \mathbf{X}^T(\mathbf{W}_1^T\mathbf{W}_1)(t)\mathbf{X} \tag{4.16}$$

and the network's finite-width neural tangent kernel (Jacot et al., 2018)

$$\text{NTK}(t) = \mathbf{I}_{N_o} \otimes \mathbf{X}^T(\mathbf{W}_1^T\mathbf{W}_1)(t)\mathbf{X} + (\mathbf{W}_2\mathbf{W}_2^T)(t) \otimes \mathbf{X}^T\mathbf{X}. \tag{4.17}$$

For a derivation of these quantities refer to Appendix B.2. In Figure 4.2A-B

we show that our analytical solution closely matches numerical simulations (Methods 4.4.3) for these quantities. As the solution only contains negative exponentials, it is numerically stable and provides high precision across a wide range of learning rates and network architectures (Figure 4.2C-D). Moreover, we can derive the steady state solution.

Theorem 4.2. *Under the assumptions of Theorem 4.1, the network function converges to the global optimum* $\mathbf{W}_2\mathbf{W}_1 = \tilde{\mathbf{U}}\tilde{\mathbf{S}}\tilde{\mathbf{V}}$ *and acquires a minimum-norm solution, that is,* $\mathbf{W}_1^T\mathbf{W}_1 = \tilde{\mathbf{V}}\tilde{\mathbf{S}}\tilde{\mathbf{V}}^T$ *and* $\mathbf{W}_2\mathbf{W}_2^T = \tilde{\mathbf{U}}\tilde{\mathbf{S}}\tilde{\mathbf{U}}$.

The proof of Theorem 4.2 is in Appendix B.3. We now turn to several implication of these results.

4.3.2 Convergence behaviour

In Chapter 3 we established that the solution manifold of two-layer linear networks is partitioned into qualitatively different regions. In particular, general linear solutions that use task-agnostic hidden-layer representations, are sensitive to noise in the inputs and parameters, and have suboptimal generalisation properties. Whereas, minimum-norm solutions have hidden-layer representations that are task-specific and are most robust to noise in the inputs and parameters. This raises the question of how and when different regions of the solution manifold are reached when the parameters of a network are optimised with gradient descent.

The so called *rich* learning regime has been associated with small initial weights, task-specific representations and slow, step-like nonlinear learning dynamics (Saxe et al., 2013; Woodworth et al., 2020; Varre, Vladarean, et al., 2024). Whereas the *lazy* regime has been associated with large initial weights, task-agnostic neural representations and fast, exponential learning dynamics (Chizat et al., 2019; Woodworth et al., 2020; Varre, Vladarean, et al., 2024). We illustrate these phenomena with the help of a semantic hierarchy learning task in a two-layer linear network

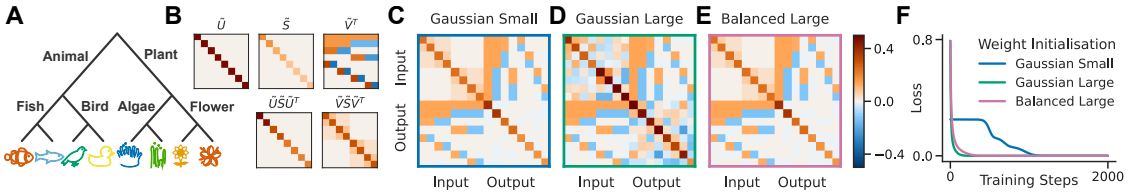

Figure 4.3: Rich and lazy learning. (**A**) In the semantic learning task, items have to be assigned to their properties. (**B**) SVD of the input-output correlation of the task (top) and the respective minimum-norm solution (bottom). (**C**) Final $\mathbf{QQ}^T$ matrices after training converged when initialised from random small weights, (**D**) random large weights (note how the upper left and lower right quadrant differ from the task's minimum-norm solution) and (**E**) large zero-balanced weights. (**F**) Learning curves for the three different initialisations as in C (blue), D (green) and E (pink). While both large weight initialisations lead to fast exponential learning curves, small weight initialisations lead to a slow step-like decay of the loss.

(Figure 4.3A). The minimum-norm solution for the input weights

$$\mathbf{W}_1^T\mathbf{W}_1 = \tilde{\mathbf{V}}\tilde{\mathbf{S}}\tilde{\mathbf{V}}^T \tag{4.18}$$

reveals the task's inherent structure (Fig. 4.3B). For example, the representations of the two fish are most similar to each other, less similar to birds and least similar to plants. Likewise, the minimum-norm solution of the readout weights

$$\mathbf{W}_2\mathbf{W}_2^T = \tilde{\mathbf{U}}\tilde{\mathbf{S}}\tilde{\mathbf{U}}^T \tag{4.19}$$

reveals the three levels among which items in the hierarchical tree are organised (Fig. 4.3B). The neural representations of the minimum-norm solution are thus task-specific. They obey the correlational structure of the task. Crucially, $\mathbf{QQ}^T(t)$ contains the temporal dynamics of the weights' correlation and therefore can be used to study if a network finds a *rich* task-specific or *lazy* task-agnostic solution. We note that more generally, knowledge is often organised within an underlying shared structure, which can be learned and represented in deep linear networks (Saxe et al., 2019). In Figure 4.3C we show that the on-diagonal entries of $\mathbf{QQ}^T$ at

convergence when training the network from random small initial weights converges to a minimum-norm solution with task-specific neural representations. Whereas when training from large initial weights, the network converges to a general solution with task-agnostic neural representations (Figure 4.3D). In both cases, the off-diagonal entries of $\mathbf{QQ}^T$ reveal that the network function is identical and optimal (Figure 4.3C-D). As expected, small initial weights lead to slow, step-wise learning dynamics, whereas large initial weights result in fast, exponential dynamics (Figure 4.3F).

Our results challenge the association of initial weight scale, shape of learning dynamics and type of solution at convergence. In particular, from Theorem 4.2 it follows that there exist large scale initial weights that converge to a minimum-norm solution with task-specific neural representations (Figure 4.3E) and exponential learning dynamics (Figure 4.3F). From this it further follows that convergence to a minimum-norm solution and the scale of the initial weights are in fact dissociated. We note, that the connection between large zero-balanced initial weights and convergence to a minimum-norm solution has been made before (Min et al., 2021). In summary, our equations describe the emergence of learning dynamics for the whole range of initial weights, from step-like to exponential, and dissociate this dynamical phenomenon from the structure of internal representations.

4.3.3 Exact continual learning dynamics

Understanding the dynamics of continual learning is challenging because the learning dynamics for any given task depends on the state of the synaptic weights after training on previous tasks. Prior work describing exact learning dynamics either relied on the initial weights sharing singular vectors with the training data, i.e., $\mathbf{U} = \tilde{\mathbf{U}}$ and $\mathbf{V} = \tilde{\mathbf{V}}$, and therefore that the learning dynamics can be fully described by a change in singular values (Saxe et al., 2013), or that learning begins

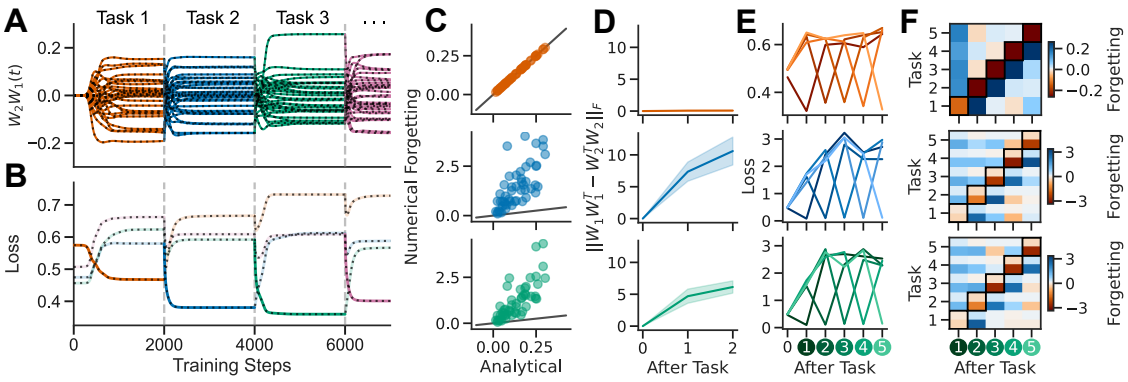

Figure 4.4: Exact continual learning dynamics. (**A**) Learning dynamics during training of a network from small zero-balanced initial weights on a sequence of four tasks and (**B**) corresponding loss for each task. Solid and dotted lines indicate simulations and analytical solutions respectively. (**C**) Comparison of numerical and analytical forgetting on the first task after training on a second task for $n = 50$ independent simulations of a linear (red), tanh (blue), or ReLU (green) network. (**D**) Zero-balancedness before and after training on a sequence of two tasks for the same networks as in B. Shaded areas show $\pm$ std. (**E**) Evaluation loss for each of 5 tasks during training a linear (red), tanh (blue) and ReLU (green) network. (**F**) Same data es in D but evaluated as forgetting, i.e., relative change in loss. The top half of each square shows the precomputed analytical amount of forgetting and the bottom half the numerical value.

from small initial weights (Atanasov et al., 2021). However, both these assumptions are violated in the continual learning setting. In contrast, the assumptions for Theorem 4.1 and Theorem 4.2 are met at any time during training. In particular, if the network was initialised zero-balanced, the weights remains zero-balanced during gradient flow (Arora et al., 2018). That means that the network weights at any point during training on one task are a valid starting point for the equation to describe the learning dynamics for another, making our solution applicable to task switching. Therefore, our equation provides a solution to the exact learning dynamics in the continual learning setting and provide the exact temporal dynamics of the network function and the performance on each individual task throughout training (Figure 4.4A-B).

Forgetting

When switching the task, the learning dynamics evolve according to Theorem 4.1 and are therefore guaranteed to converge in accordance with Theorem 4.2. Consequentially, the network converges to the global optimum of the current task independent of prior tasks' training and structure. Thus, forgetting is truly catastrophic: No information about prior tasks is preserved.

Formally, let the input-output correlation of tasks $\mathcal{T}^i$, $\mathcal{T}^j$, and $\mathcal{T}^k$ be denoted by $\mathbf{\Sigma}^i_{yx}$, $\mathbf{\Sigma}^j_{yx}$, and $\mathbf{\Sigma}^k_{yx}$. Then, according to Theorem 4.2, training on task $\mathcal{T}^j$ completely overwrites the initial network function at convergence

$$\lim_{t\to\infty} \left(\mathbf{W}_2\mathbf{W}_1\right)(t) = \mathbf{\Sigma}^j_{yx}. \tag{4.20}$$

At that time, the loss of any prior or future task $\mathcal{T}^i$ is then determined by the difference of the input-output correlations

$$\mathcal{L}^i_{\text{MSE}} = \frac{1}{2}\left|\left|\mathbf{\Sigma}^j_{yx} - \mathbf{\Sigma}^i_{yx}\right|\right|^2_F + c, \tag{4.21}$$

where c is a constant offset that only depends on the structure of the training data of task $\mathcal{T}^i$ (Appendix B.4). As a consequence, the amount of forgetting, i.e., the relative change of loss, is fully determined by the similarity of the input-output correlations of tasks. In particular, the amount of forgetting for task $\mathcal{T}_i$ when training on task $\mathcal{T}_k$ after having previously trained the network on task $\mathcal{T}_j$ is

$$\mathcal{F}^i_{j\to k} = \frac{1}{2}\left|\left|\mathbf{\Sigma}^k_{yx} - \mathbf{\Sigma}^i_{yx}\right|\right|^2_F - \frac{1}{2}\left|\left|\mathbf{\Sigma}^j_{yx} - \mathbf{\Sigma}^i_{yx}\right|\right|^2_F. \tag{4.22}$$

For a derivation of this quantity see Appendix B.4. It follows that the performance and amount of forgetting for any task and network can be fully precomputed without any training (Figure 4.4C-F). We show numerically that our results depend

on the network to be linear and that networks with *tanh* or *ReLU* nonlinearities show different behaviour. Typically, in nonlinear networks, weights quickly become unbalanced and the performance on each individual task improves, while the amount of forgetting on all other tasks worsens relative to their linear counterparts (Figure 4.4C-F).

In summary, our results link exact learning dynamics with the exact temporal dynamics of forgetting during continual learning in two-layer linear networks and thus provide an analytical tool to study its mechanisms, and potential countermeasures like early stopping or replay.

4.4 Methods

4.4.1 Zero-balanced initial weights

For the initial network weights to be zero-balanced, the left singular vectors of the first-layer weights have to be equal to the right singular vectors of the second-layer weights, and both weight matrices have to share the same singular values. Let the compact SVD of the initial network weights be denoted by

$$\text{SVD}(\mathbf{W}_1(0)) = \mathbf{R}\sqrt{\mathbf{S}}\mathbf{V}^T \quad \text{and} \quad \text{SVD}(\mathbf{W}_2(0)) = \mathbf{U}\sqrt{\mathbf{S}}\mathbf{R}^T \tag{4.23}$$

then

$$\begin{aligned} \mathbf{W}_1(0)\mathbf{W}_1(0)^T &= \mathbf{W}_2(0)^T\mathbf{W}_2(0) \\ \mathbf{R}\sqrt{\mathbf{S}}\mathbf{V}^T\mathbf{V}\sqrt{\mathbf{S}}\mathbf{R}^T &= \mathbf{R}^T\sqrt{\mathbf{S}}\mathbf{U}^T\mathbf{U}\sqrt{\mathbf{S}}\mathbf{R}^T \\ \Leftrightarrow \mathbf{R}\mathbf{S}\mathbf{R}^T &= \mathbf{R}\mathbf{S}\mathbf{R}^T. \end{aligned} \tag{4.24}$$

Therefore, if we denote sampling from the random orthogonal group by $\mathbf{X} \sim \mathcal{G}$, and sampling from a random normal distributed with mean μ and standard deviation

σ by $\mathbf{X} \sim \mathcal{N}(\mu, \sigma)$, then zero-balanced network weights with initial norm

$$a = ||\mathbf{W}_1(0)||_F = ||\mathbf{W}_2(0)||_F \tag{4.25}$$

can be generated using the following algorithm

Algorithm 1 Zero-balanced weight initialisation

Require: N_i, N_h, N_o, a
$N_m \leftarrow \min(N_i, N_o)$
$\mathbf{R} \sim \mathcal{G} \in \mathbb{R}^{N_h \times N_m}$
$\mathbf{U} \sim \mathcal{G} \in \mathbb{R}^{N_o \times N_m}$
$\mathbf{V} \sim \mathcal{G} \in \mathbb{R}^{N_m \times N_i}$
$\mathbf{a} \sim \mathcal{N} \in \mathbb{R}^{N_m}$
$\sqrt{\mathbf{S}} \leftarrow \text{diag}\,(a\mathbf{a}^2/||\mathbf{a}^2||_2)$
$\mathbf{W}_1 \leftarrow \mathbf{R}\sqrt{\mathbf{S}}\mathbf{V}^T$
$\mathbf{W}_2 \leftarrow \mathbf{U}\sqrt{\mathbf{S}}\mathbf{R}^T$
return $\mathbf{W}_1\mathbf{W}_2$

4.4.2 Learning tasks

The following tasks were used during network simulations:

Random regression task

A random regression task consists of network inputs and target outputs that are uncorrelated Gaussian noise. Specifically, inputs are sampled from a normal distribution $\mathbf{X} \sim \mathcal{N}(\mu = 0, \sigma = 1) \in \mathcal{R}^{N_i \times P}$ and subsequently whitened, such that $\mathbf{\Sigma}_{xx} = \mathbf{I}$; and the corresponding target values are sampled from a random normal distribution $\mathbf{Y} \sim \mathcal{N}(\mu = 0, \sigma = 1/\sqrt{N_o}) \in \mathcal{R}^{N_o \times P}$.

Teacher-student task

A teacher-student task (Biehl & Schwarze, 1995; Saad & Solla, 1995) consists of inputs $\mathbf{X}$ that are sampled and whitened as in the random regression tasks. The corresponding target values are generated by $\mathbf{Y} = \mathbf{A}\mathbf{X}$, where $\mathbf{A} \sim \mathcal{N}(\mu = 0, \sigma =$

$1/\sqrt{N_i}) \in \mathcal{R}^{N_o \times N_i}$. We note that, unlike a random regression task, the teacher-student setting ensures the existence of a linear solution with zero error.

Semantic hierarchy

Each item in the semantic hierarchy task (Figure 4.3) is encoded as a scaled one-hot vector, e.g., $[0, 0, 0, 0, \sqrt{8}, 0, 0, 0]$ for the fifth out of the eight items, such that $\mathbf{\Sigma}_{xx} = \mathbf{I}$. Corresponding target vectors encode for the position in the hierarchical tree. A one and minus one encode for being a left or right child of a node, and a zero encodes that the item is not a child of a node. For example, the blue fish is a left child of the root node, a left child of the animal node, not part of the plant branch, a right child of the fish node, and not part of the bird, algae, or flower branch, leading to the label $[1, 1, 1, 0, -1, 0, 0, 0]$. The complete target matrix for all eight items is then

$$\mathbf{Y} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}. \tag{4.26}$$

4.4.3 Simulation details

All simulation studies were implemented using Jax (Bradbury et al., 2018) and a learning rate of $\eta = 0.05$.

Figure 4.1

Simulations for panels B-D are based on a teacher-student task with $P = 10$, $N_i = 5$, $N_h = 10$ and $N_o = 2$. The norm of the initial zero-balanced weights was set to $a = 0.05$, $a = 0.9$, and $a = 1$ respectively.

Figure 4.2

Simulations for panels A-B are based on a teacher-student task with $P = 10$, $N_i = 4$, $N_h = 5$ and $N_o = 3$. The norm of the initial zero-balanced weights was set to $a = 0.002$, and $a = 0.9$ respectively. Panel C and D were generated by running $n = 50$ independent simulations for each of which we sampled $N_i \in [2, 50]$, $N_o \in [2, 50]$, $N_h = [\min(N_i, N_o), 50]$, and $P \in [2\max(N_i, N_h, N_o), 3\max(N_i, N_h, N_o)]$. Networks were then trained on a random regression task until convergence. Each of the networks was trained from the same initial weights and task for learning rates in $\eta \in [0.005, 0.05]$.

Figure 4.3

Simulations for panels C-F are based on the semantic hierarchy task. A network with $N_i = 8$, $N_h = 14$ and $N_o = 8$ was trained on the $P = 8$ items. The initial weights for panel C, D and E were sampled from a random normal distribution with $\sigma = 0.0001$ and $\sigma = 0.42$ and zero-balanced weights with $a = 4.3$ respectively.

Figure 4.4

Simulations for panel A and B were generated by training a network with $N_i = 5$, $N_h = 10$ and $N_o = 6$ on a sequence of four random regression tasks with $P = 25$ training examples. Panels C and D were generated by running $n = 50$ independent simulations on a sequence of two random regression tasks with $P = 100$. We run the simulations three times, with a linear, tanh, and ReLU activation function in the

hidden layer. Network dimensions were randomly sampled such that $N_i \in [2, 50]$, $N_o \in [2, 50]$, and $N_h = [\min(N_i, N_o), 50]$. Likewise, we trained $n = 50$ independent simulations on a sequence of five random regression tasks with $N_i = 15$, $N_h = 18$, $N_o = 21$, and $P = 50$ for panels D and E.

4.5 Discussion

Our equations for the exact learning dynamics in two-layer linear networks are constrained by several assumptions. Specifically, the network must not have bottlenecks, so these equations are not applicable to autoencoder-like architectures (Hinton & Zemel, 1993; Kingma & Welling, 2014). Additionally, the requirement that inputs are whitened and that input-output correlations are of full rank limits the applicability to settings where training tasks have non-trivial correlations in the inputs or are underconstrained. This is especially relevant for studying continual learning scenarios with varying degrees of task overlap. Even when the task has full rank and the network is not bottlenecked, the matrix $\mathbf{B}$ may still be non-invertible, as for example seen during reversal learning (Braun et al., 2022). Moreover, the zero-balancedness assumption, which causes input and output norms to be identical during initialisation and training, prevents us from examining the impact of norm imbalances, such as those in the neural tangent kernel regime (Jacot et al., 2018), on the *rich* and *lazy* regimes. Followup research has partially addressed these limitations (Tarmoun et al., 2021; Dominé et al., 2024). Furthermore, our analysis is confined to full-batch gradient flow, disregarding the stochastic effects of stochastic gradient descent with finite learning rates (F. He et al., 2019; Varre, Sagitova, & Flammarion, 2024). Finally, our study is restricted to linear networks. Recent work on gated deep linear networks, which mimic ReLU networks, has introduced a tractable nonlinear architecture beyond the finite-width thermodynamic limit (Saxe et al., 2022).

Exact continual learning and the necessity of drifting neural representations in linear networks

---

This chapter presents original work. Contributions to the research presented in this chapter are as follows:

Supervision Professor Andrew M. Saxe, Professor Christopher Summerfield

Derivations, simulations, and figures Lukas Braun

5.1 Introduction

During continual learning, the connections of a neural system have to be optimised to incrementally acquire knowledge from a continuous stream of information. While humans and non-human animals are capable of acquiring new knowledge and skills throughout their lifetime, artificial neural networks suffer from catastrophic forgetting. In the previous chapter we investigated how the initial state of the network influences catastrophic forgetting and analytically derived under a set of assumptions that knowledge during learning is not preserved, but rather, that networks simply acquire the current task, leading to truly catastrophic forgetting of previously acquired knowledge. This poses the question, how previously acquired knowledge can be preserved during the acquisition of new knowledge? The plethora of continual learning algorithms that have been proposed to overcome catastrophic

forgetting are often based on heuristics. While these heuristics are typically mathematically or biologically motivated, a full understanding usually remains difficult due to the complexity and intractability of nonlinear networks and the brain. In this chapter we analytically study popular algorithms that have been proposed to circumvent catastrophic forgetting, using two-layer linear networks. The well defined solution manifold of linear networks (Chapter 3) makes their connectionist computations ideally suited to study and compare the algorithms' performance, success and failure modes. Based on our observations, we subsequently derive exact continual learning rules for categorical and relational knowledge. Further, we show that exact continual learning to a solution with task-specific neural representations requires drifting neural representations. Based on our observations, we argue, that perceptual and behavioural stability is not observed despite, but because of drifting neural representations.

5.2 Preliminaries and setting

Formally, a learning algorithm has to optimise the parameters of a model such that input vectors $\mathbf{x}_n \in \mathbb{R}^{N_i}$ from a set of P training pairs $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1,\dots,P}$ are mapped to their corresponding target values $\mathbf{y}_n \in \mathbb{R}^{N_o}$. In the case of continual learning, the algorithm has access to a single training pair at a time and is presented with each training pair example exactly once. The learning objective during continual learning is then to maximise the performance on the current training example while maintaining performance across all previously encountered examples. This is in contrast to mini-batch, multi-epoch learning in which the learning algorithm has access to multiple training pairs at each training step and can revisit training data multiple times. We use two-layer linear neural networks as a model

for connectionist neural computations

$$\hat{\mathbf{y}}_n = \mathbf{W}_2\mathbf{W}_1\mathbf{x}_n, \tag{5.1}$$

with first $\mathbf{W}_1 \in \mathbb{R}^{N_h \times N_i}$ and second layer $\mathbf{W}_2 \in \mathbb{R}^{N_o \times N_h}$ network weights. Throughout this chapter we make

Assumption 9. *The network is not bottlenecked, i.e.,* $N_h \geq \min(N_i, N_o)$.

We quantify the performance of learning algorithms with the mean squared error on all $K \leq P$ training examples that the model has encountered so far

$$\mathcal{L}^K_{\text{MSE}} = \frac{1}{2K}\sum_{n=1}^{K} ||\mathbf{W}_2\mathbf{W}_1\mathbf{x}_n - \mathbf{y}_n||_2^2. \tag{5.2}$$

After having successfully learned about the first $K-1$ training pairs, starting from initial network weights $\mathbf{W}_1^{K-1}$ and $\mathbf{W}_2^{K-1}$, learning can then be defined as finding additive parameter updates $\Delta\mathbf{W}_1$ and $\Delta\mathbf{W}_2$, which minimise the loss across all K items. While in practice, we are going to apply gradient-descent for T training steps, with learning rate η on a loss function $\mathcal{L}$ to find parameter updates

$$\Delta\mathbf{W}_1 = -\eta\sum_{t=0}^{T}\frac{\partial\mathcal{L}}{\partial\mathbf{W}_1(t)} \quad \text{and} \quad \Delta\mathbf{W}_2 = -\eta\sum_{t=0}^{T}\frac{\partial\mathcal{L}}{\partial\mathbf{W}_2(t)}, \tag{5.3}$$

we are interested in the shape and geometry of the solution manifold and less so by which algorithm a solution is reached. In particular, we are interested in if and how the hidden-layer neural representations of stimuli

$$\mathbf{h}_n = \mathbf{W}_1\mathbf{x}_n \tag{5.4}$$

and their representational similarity

$$\text{RSM}_{ij} = \mathbf{h}_i^T\mathbf{h}_j \tag{5.5}$$

change during the continuous acquisition of knowledge. Intuitively, an algorithm that integrates newly arriving information should keep the network's input-output mapping of previously learned neural computation stable, i.e.,

$$\mathbf{W}_2^{K-1}\mathbf{W}_1^{K-1}\mathbf{x}_n \approx (\mathbf{W}_2^{K-1} + \Delta\mathbf{W}_2)(\mathbf{W}_1^{K-1} + \Delta\mathbf{W}_1)\mathbf{x}_n. \tag{5.6}$$

One question that then arises is, if the hidden-layer neural representations of previously acquired knowledge should remain stable too, i.e.,

$$\mathbf{W}_1^{K-1}\mathbf{x}_n \stackrel{?}{=} (\mathbf{W}_1^{K-1} + \Delta\mathbf{W}_1)\mathbf{x}_n. \tag{5.7}$$

To analyse if and under which conditions a change in the neural representations is to be expected during the continuous acquisition of knowledge, we have to identify and study the shape and geometry of the solution manifold and respective parameter updates. In our setting, the best the neural network can do is to perform linear regression (Laurent & Brecht, 2018). However, in contrast to simple linear regression, deep linear networks are highly overparametrised. Specifically, after exposure to K training pairs, any combination of network weights $\mathbf{\Omega}_2^K$ and $\mathbf{\Omega}_1^K$ for which

$$\mathbf{\Omega}_2^K\mathbf{\Omega}_1^K\mathbf{\Sigma}_{xx}^K = \mathbf{\Sigma}_{yx}^K \tag{5.8}$$

is a linear solution, where

$$\mathbf{\Sigma}_{xx}^K = \frac{1}{K}\sum_{n=1}^{K}\mathbf{x}_n\mathbf{x}_n^T \quad \text{and} \quad \mathbf{\Sigma}_{yx}^K = \frac{1}{K}\sum_{n=1}^{K}\mathbf{y}_n\mathbf{x}_n^T \tag{5.9}$$

are the input and input-output correlation matrices of the K training pairs. In Chapter 3, we investigated how these solutions differ from each other despite having optimal linear performance on the training data. In the following, we are going to investigate which consequences this has for continual learning.

5.3 Results

5.3.1 Representation learning

In Chapter 3 we established that the solution manifold of two-layer linear networks can yield almost arbitrary representational similarities. However, neural representations observed in neural recordings are generally not arbitrary but are often specific to the structure of the environment and the task (O'Keefe & Dostrovsky, 1971; Kiani et al., 2007; Kriegeskorte, Mur, Ruff, et al., 2008; Knudsen & Wallis, 2021; Flesch et al., 2022; Muhle-Karbe et al., 2023).

For instance, in the mouse olfactory bulb, neural representations adhere to the structure of categorical learning tasks (Kudryavitskaya et al., 2021). In this experiment, mice learned to classify intermediate mixtures of two odours using a go-no-go paradigm, initially according to a 5-boundary categorisation task and subsequently a 1-boundary categorisation task (Figure 5.1A-C). During the acquisition of the 5-boundary task, neural representations of the six different odour mixtures in the olfactory bulb became linearly separable (Figure 5.1D). However, during the acquisition of the 1-boundary task, these representations collapsed, making odours within each of the two groups linearly inseparable (Figure 5.1E). Importantly, both downstream categorisation tasks can be solved using the linearly separable representations, but not vice versa.

This can be illustrated using a two-layer linear network. The 1-boundary decision task can be solved with a network that employs a general linear solution (Section 3.3.1). This results in task-agnostic and linearly separable hidden-layer neural representations (Figure 5.1E) that are qualitatively similar to the neural representations observed during the 5-boundary decision task in mice. Alternatively, the 1-boundary task can be solved using a network that adopts a minimum-norm solution (Section 3.3.1). In this case, the hidden-layer representations are

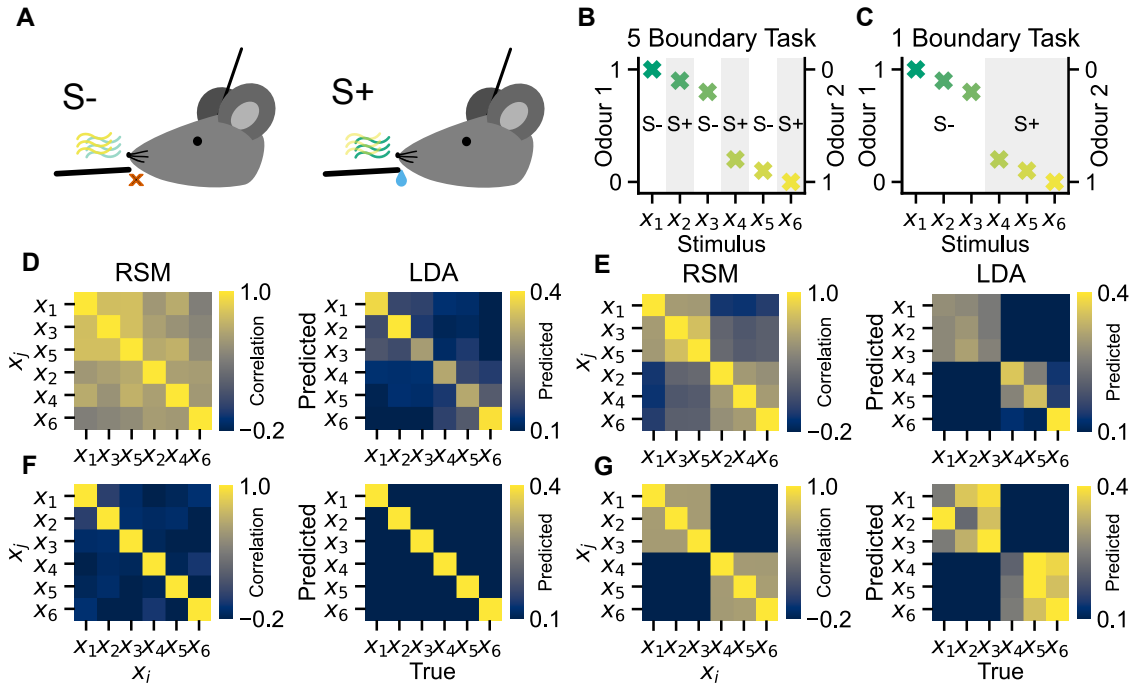

Figure 5.1: Representation learning. (**A**) Schematic of the go-no-go task in Kudryavitskaya et al. (2021). The mouse is presented with a mixture of two odours. Depending on the exact mixing ratio a trial is or is not rewarded. (**B**) Assignment of odour mixtures to rewards in the 5-boundary and (**C**) 1-boundary discrimination task. (**D**) Average representational similarity matrix and confusion matrix following linear discriminant analysis of neural representations of odours in olfactory bulb mitral cells after training on the 5-boundary task. Higher correlations indicate greater similarity between the neural encodings of two odours. Diagonal values in the confusion matrix represent correct decoding, while off-diagonal values reflect misclassification. (**E**) Same as D but after training on the 1-boundary task. Note how odours within each of the two groups are now similar and are confused during decoding. (**F**) Representational similarity and confusion matrix of neural representations in a two-layer linear network after training to a task-agnostic solution. The 1-boundary task can be solved with linearly-separable neural representations as observed in the 5-boundary task in mice. (**G**) Same as in E but using a solution with task-specific neural representations as observed after the 1-boundary task observed in mice. *Panels D and E are reproduced from Kudryavitskaya et al. (2021) with permission.*

task-specific and linearly inseparable (Figure 5.1F) and model the neural representations observed during the 1-boundary decision task in mice. Given that the 1-boundary decision task could have been solved using the linearly separable neural representations, why did the stimulus representations instead collapse into a low-rank, task-specific form during the acquisition of the 1-boundary task?

To this end, we refer to Chapter 3, where we analytically demonstrated that points on the solution manifold with task-specific hidden-layer representations are least sensitive to input and parameter noise. This computational advantage, combined with the experimental data, suggests that neural circuits may leverage learning mechanisms that favour minimum-norm solutions with task-specific neural representations. Therefore, in the following, we will not only examine whether a continual learning algorithm can acquire new knowledge without catastrophic forgetting but also determine whether the algorithm converges to a task-agnostic or task-specific solution. We measure the distance to the minimum-norm solution using

$$D = ||\mathbf{Q}^T\mathbf{Q} - \mathbf{I}||_F \quad \text{with} \quad \mathbf{Q} = \mathbf{W}_1\mathbf{V}\sqrt{\mathbf{S}^{-1}} \tag{5.10}$$

where $\text{SVD}(\mathbf{\Sigma}_{yx}) = \mathbf{USV}^T$ is the compact SVD of the task. Further, since the standard deviation of initial weights is known to influence the type of solution found (Woodworth et al., 2020), we test algorithms across a wide range of initial values.

5.3.2 Continual learning in two-layer linear networks

During continual learning, a learning algorithm has to change the connections of a neural system to integrate the information contained in the current training pair without eradicating knowledge about all previous training pairs. Splitting the mean

squared error into the contribution of the current and all previous items

$$\mathcal{L}_{\text{MSE}} = \underbrace{\frac{1}{2K}||\mathbf{W}_2\mathbf{W}_1\mathbf{x}_k - \mathbf{y}_k||_2^2}_{\text{Current pair}} + \underbrace{\frac{1}{2K}\sum_{n=1}^{K-1}||\mathbf{W}_2\mathbf{W}_1\mathbf{x}_n - \mathbf{y}_n||_2^2}_{\text{Previous pairs}}, \tag{5.11}$$

makes it apparent why continual learning is difficult, even in the simple, linear setting. Optimising the weight matrices based on the current training pair may also change how previous training pairs are processed by the network and thus change their contribution to the loss. For example, when training a two-layer linear network on a semantic hierarchy task (Figure 5.2A-B, Section 4.4.2) using naïve gradient descent, performance on previously learned items deteriorates during the acquisition of new items. This phenomenon, known as catastrophic forgetting, persists regardless of the size of the initial weights, and the network converges to hidden-layer neural representations that deviate significantly from the minimum-norm solution (Figure 5.2C). As described before (Mirzadeh et al., 2020), adding weight decay (Hinton, 1987) further deteriorates the situation as it actively removes previously acquired knowledge from the network (Figure 5.2D). This raises the question of how the naïve objective can be altered with a correction term $\mathcal{R}$ such that minimising that loss leads to continual learning without forgetting

$$\mathcal{L} = \frac{1}{2K}||\mathbf{W}_2\mathbf{W}_1\mathbf{x}_k - \mathbf{y}_k||_2^2 + \mathcal{R}. \tag{5.12}$$

In the following, we use the simplicity of our setting, the linear solution manifold and corresponding neural representations to dissociate the functionality and failure modes of continual learning algorithms.

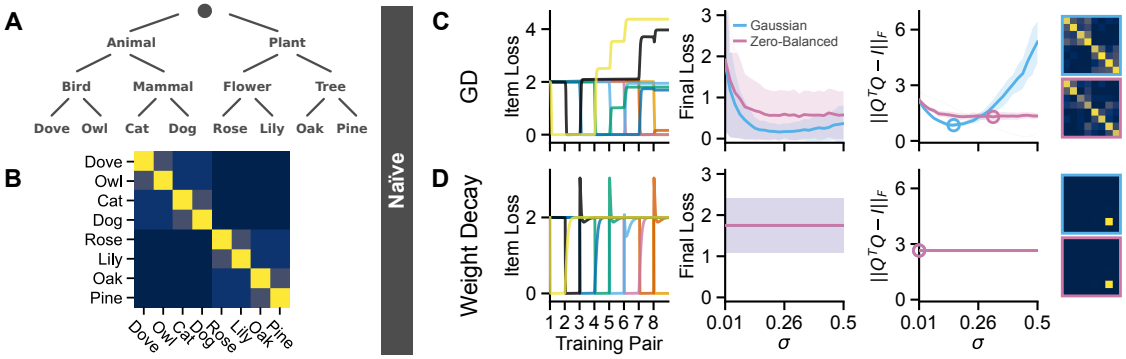

Figure 5.2: Catastrophic forgetting. (**A**) Schematic of a semantic hierarchy task. The items at the lowest level of the hierarchy have to be mapped to their corresponding properties. (**B**) Task-specific neural representation of the semantic hierarchy. The closer items are within the hierarchy, the more similar their representation is. (**C**) Temporal evolution of per item loss during continual learning with gradient descent (left), final mean squared error at the end of training for a range of initial weight standard deviations (centre), distance to task-specific neural representation and example representational similarity for best-performing network as indicated by circle (right). (**D**) Same as C but for gradient descent with weight decay. Note how all knowledge except the current training item is erased. Solid lines and shaded area indicate mean and ± standard deviation across 50 random initialisations.

Replay based algorithms

One proposed mechanism to prevent catastrophic forgetting is to first memorise and subsequently replay previous items during learning (McClelland et al., 1995). The idea behind this is persuasive: To find parameter updates that accurately integrate newly arriving information, experiences are memorised and subsequently, the loss is jointly minimised across all items

$$\mathcal{R}_{\text{Replay}} = \frac{1}{2K} \sum_{n=1}^{K-1} ||\mathbf{W}_2\mathbf{W}_1\mathbf{x}_n - \mathbf{y}_n||_2^2. \tag{5.13}$$

However, while memorisation and replay of all previously encountered knowledge resembles full-batch learning and thus successfully circumvents catastrophic forgetting and converges to a rich representation for small Gaussian or zero-balanced initial weights, *full replay* (Figure 5.3A) has computational and memory storage

requirements that linearly increase with the amount of training examples (Figure 5.3B). While the runtime of such an algorithm may be optimised by finding an optimal learning rate, this does not alter its inherent memory or computational complexity per gradient step. To circumvent the computational demands of *full replay*, a plethora of alternative algorithms have been put forward (Section 1.2.1). Reminiscent of stochastic gradient descent, during *random replay* (Figure 5.3C) only a single item is randomly selected from memory and replayed in each training step, drastically reducing the computational complexity (Figure 5.3D). However, as expected by a stochastic algorithm, the time until convergence increases with the number of memorised items. In contrast, *prioritised replay* (Figure 5.3E) algorithms select the most informative item instead of a randomly selected one. For example, if an algorithm has access to an oracle to select the memorised item that is contributing to the loss the most, convergence time can be kept approximately constant, while keeping computational complexity on par with *random replay* (Figure 5.3F). Finally, during *compressed replay* (Figure 5.3G) instead of replaying memorised items, pseudo input-output pairs are being assembled and replayed. We show analytically, that in our setting, the minimum number of pseudo input-output pairs depends on the rank of the input-output correlation matrix and is therefore directly related to the statistics of the task (Appendix C.1, Figure 5.3H). However, such an algorithm would predict neural activations during replay that are statistically distinct from previous experiences, potentially evading detection by the techniques commonly used to identify replay events in neural recordings. In summary, replay based algorithms either suffer from adverse computational and memory requirements or require access to an oracle to provide item selection or item compression.

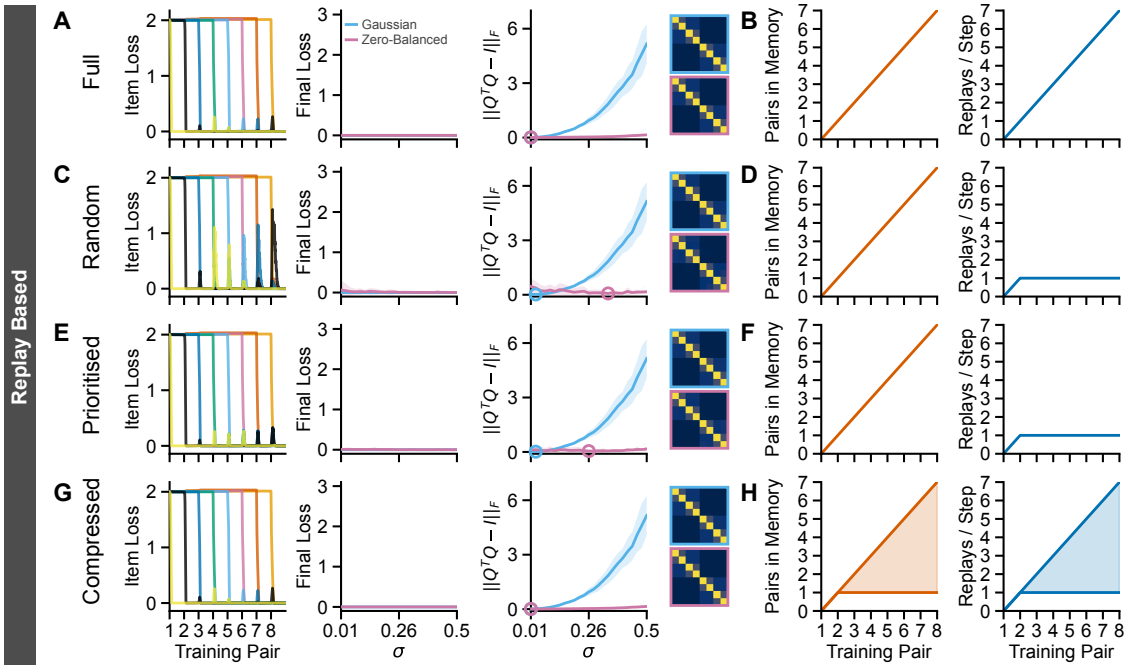

Figure 5.3: Replay based algorithms. (**A**) Continual learning a semantic hierarchy with eight objects (first column), results in a zero mean squared error when trained from random Gaussian (blue) or zero-balanced (red) initial weights across a range of standard deviations (second column), and task-specific neural representations for small Gaussian and zero-balanced weights (third column), with example representational similarity for best performing networks, for full replay. Solid lines and shaded area indicate mean and ± standard deviation across 50 random initialisations. (**B**) Memory and computational complexity increases linearly with the number of training pairs for full replay. (**C**) Same as in A but for random replay. (**D**) During random replay only one item is replayed per training step. (**E**) Same as in A but for prioritised replay. Note how the loss across all previously learned items stays low despite (**F**) only replaying a single training pair per training step. (**G**) Same as in A but for compressed replay. (**H**) Memory and computational complexity depends on the statistics of the task and can lie in between a single item and a linear relationship as in full replay.

Solving the stability-plasticity dilemma

Another class of algorithms (Kirkpatrick et al., 2017; Zenke et al., 2017; Aljundi et al., 2018) is motivated by the stability-plasticity dilemma (Grossberg, 1987; Abraham & Robins, 2005). These algorithms create a point estimate of the network's error surface (Benzing, 2022) and then penalise changes in weights through a regularisation term for which the curvature of the loss is high. Again, the idea behind this is persuasive: To mitigate catastrophic forgetting, first identify synaptic weights that are crucial for encoding previously acquired knowledge and subsequently, during the acquisition of new knowledge, penalise changes to those weights.

However, as we have established in Chapter 3, there exists a manifold of solutions and therefore that the network function can be fully reparameterised without changing the underlying computation. Thus, preserving knowledge does not require synapses to be stable Abraham & Robins (2005), nor is knowledge encoded in specific synapses but through their interplay within the network. While we believe that this interpretation of the stability-plasticity dilemma, in which synaptic weights have to stay stable to continuously encode previous knowledge is ill-posed and has been driving misconceptions of learning in neural networks and particular about the continual learning problem, this does not exclude the possibility that there may exist a solution to the continual learning problem within this framework.

Applying a Bayesian approach along with Laplace's method to the continual learning problem results in a loss function that penalises changes to synapses considered critical for preserving prior knowledge (Huszár, 2017; Kirkpatrick et al., 2017). Following this approach fully mitigates catastrophic forgetting, however, does generally not converge to a minimum-norm solution (Figure 5.4A). This is coherent with the dilemma's hypothesis: Using disjoint sets of synapses to encode knowledge should lead to disjoint hidden-layer representations. The regularization term in the Bayesian approach depends on the Hessian matrix, which makes its

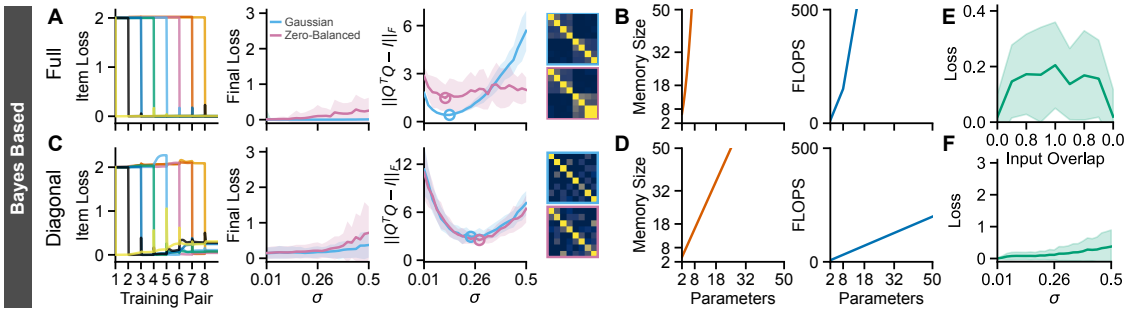

Figure 5.4: Bayes based algorithms. (**A**) Continual learning a semantic hierarchy with eight objects (first column) with full Hessian regularisation, results in (close to) zero mean squared error when trained from random Gaussian (blue) or zero-balanced (red) initial weights across a range of standard deviations (second column). However, neural representations are completely task-agnostic except for a very narrow band of Gaussian initial weights, where they are less task-agnostic, with example RSMs for best performing networks (third column). Solid lines and shaded areas indicate mean and ± standard deviation across 50 random initialisations. (**B**) The memory and floating point operations per training step quickly rise as a function of network parameters. (**C**) Same as in A but for diagonal Hessian regularisation. Note how training generally not converge to a zero-error solution and neural representations are very task-agnostic. (**D**) Same as in B, but for diagonal Hessian regularisation. (**E**) Loss as a function of overlap in the inputs for $\sigma = 0.13$. (**F**) Reparameterisation to reduce overlap in the hidden-layer representations after each training pair improves training loss, especially when starting from very small initial weights.

effects difficult to interpret. In Appendix C.3 we use the analytical tractability of our setting to show that learning using the Bayesian approach is identical to minimising

$$\tilde{\mathcal{R}}_{\text{Bayesian}} = \frac{1}{2K} \sum_{n=1}^{K-1} ||\mathbf{W}_2^{K-1}\mathbf{W}_1\mathbf{x}_n + \mathbf{W}_2\mathbf{W}_1^{K-1}\mathbf{x}_n - 2\mathbf{y}_n||_2^2. \tag{5.14}$$

Thus, the error landscape of the Bayesian approach is identical to the error landscape of full-replay through a network with pairwise combinations of current weight matrices ($\mathbf{W}_2$ and $\mathbf{W}_1$) and weight matrices at training onset ($\mathbf{W}_2^{K-1}$ and $\mathbf{W}_1^{K-1}$). The Bayesian approach requires the computation and storage of the full Hessian matrix which scales quadratically in the number of synaptic weights in the network (Figure 5.4B). To circumvent the costly calculation of the Hessian matrix, in

practice, the Hessian matrix is approximated by the empirical Fisher information or approximations thereof (Benzing, 2022) To further reduce complexity, usually only the diagonal or an approximation of the empirical Fisher information matrix is used (Kirkpatrick et al., 2017; Zenke et al., 2017; Aljundi et al., 2018). In our setting, instead of relying on an approximation we directly use the approximation's upper bound: The diagonal of the Hessian. When using the diagonal of the Hessian matrix, training suffers from mild forgetting, but converges to network functions that are far from the minimum-norm solution (Figure 5.4C). Note however, how the memory and the computational complexity are drastically reduced in comparison to full Hessian regularisation (Figure 5.4D). In Appendix C.4, we show that under the assumption that training on previous training pairs has converged to a minimum-norm solution, minimising

$$\begin{aligned}\tilde{\mathcal{R}}_{\text{Diagonal}} = \left(\mathbf{W}_1 - \mathbf{\Omega}_1\right)^2 \odot \operatorname{diag}\left(\sum_{i=1}^{N_o} \mathbf{W}_{2i}^2\right) \left(\frac{1}{K-1}\sum_{n=1}^{K-1} \mathbf{x}_n^2\right)^T \\ + \left(\mathbf{W}_2 - \mathbf{\Omega}_2\right)^2 \odot \operatorname{diag}\left(\mathbf{I}\right) \left(\frac{1}{K-1}\sum_{n=1}^{K-1} \mathbf{h}_n^2\right)^T\end{aligned} \tag{5.15}$$

is gradient equivalent to regularising with the diagonal of the Hessian. Here, $\mathbf{W}_{2i}$ denotes the i-th column of $\mathbf{W}_2$. Therefore, the stability-plasticity trade-off in two-layer linear networks is not a function of overparameterisation, cf. Kirkpatrick et al. (2017), but is governed by the overlap of inputs, hidden-layer representations, and readout weights. As a result, training converges to a solution when inputs are non-overlapping (e.g., one-hot inputs, i.e., vectors that are all 0 except for one entry that identifies the object that is equal to 1), but performance deteriorates as input overlap increases (Figure 5.4E). Since, the hidden-layer representation in linear networks can be chosen almost arbitrarily, the amount of overlap in the hidden-layer representation, and therefore convergence to a soltion, can be controlled by reparameterising the network after each training pair (Figure 5.4F). This gives

further insights why these methods are particularly successful in tasks with non-overlapping inputs like the permuted-MNIST task (Goodfellow et al., 2013) but struggle otherwise (Farquhar & Gal, 2018), and why reparameterisation of network weights can improve their performance (X. Liu et al., 2018).

5.3.3 Preserving categorical knowledge during learning

Humans and non-human animals excel in learning new categorical information from minimal exposure. For example, a child that went to the zoo for the first time in her life will afterwards report with ease that an elephant is grey and that a cheetah is lightning fast. And a mouse that is footshocked while being exposed to an odour will immediately learn to avoid that odour. Crucially, this acquisition of knowledge usually occurs without catastrophic interference with previously acquired knowledge. Formally, during the acquisition of categorical knowledge, a specific input $\mathbf{x}_n$ has to be related to a set of properties $\mathbf{y}_n$. In the following, we make

Assumption 10. *Inputs are decorrelated and of unit length, i.e.,* $\mathbf{x}_i^T \mathbf{x}_j = 1$ *for* $i = j$ *and* 0 *otherwise,*

to ensure that inputs are distinct. Notably, we make no further assumptions about the properties $\mathbf{y}_n$ and they may be (partially) shared for different inputs. For example, a giraffe and an elephant are distinct inputs but they share properties, e.g., they are mammals. We note that one special instantiation of Assumption 10 that is often used when studying categorical learning are one-hot vectors. Interestingly, if inputs are encoded as one-hot vectors, any solution to the learning problem that is achievable by a two-layer neural network with any desired nonlinear activation function is also achievable by a two-layer linear network (Appendix A.7).

The objective of continuous categorical learning is to find parameter updates that minimise the loss across all training pairs, while having access to the current training pair only. As defined, the learning problem is trivial. For example, it can

be solved in a one-shot manner by a one-layer linear neural network (Appendix C.5). This becomes apparent in the special case of one-hot inputs, where each new $\mathbf{y}_n$ can simply be stored as the i-th column of the weight matrix. However, learning becomes surprisingly difficult in a multi-layered connectionist system: Naïve optimisation of a linear two-layer network with gradient-descent on the loss of the current training example fails catastrophically despite the simplicity of the task and network architecture (Figure 5.2C).

Exact categorical learning

Next, we show that exact continual learning (ECL) for categorical learning does not require memorisation and replay of previously acquired knowledge, nor has to address the stability-plasticity dilemma. Rather, using correction term

$$\mathcal{R}_{\text{Exact}} = ||\mathbf{W}_2\mathbf{W}_1 - \mathbf{W}_2^{K-1}\mathbf{W}_1^{K-1}||_F^2 - ||\mathbf{W}_2\mathbf{W}_1\mathbf{x}_k||_2^2, \tag{5.16}$$

leads to a loss function that when being minimised, minimises the distance to the least-squares solution. The ECL algorithm thus perfectly integrates knowledge without catastrophic forgetting and converges to a rich solution if training begins from small initial weights or random zero-balanced weights (Figure 5.5A). The ECL loss does only depend on the current state of the synaptic weights ($\mathbf{W}_1$ and $\mathbf{W}_2$), the current training pair $(\mathbf{x}_k, \mathbf{y}_k)$ and the state of the synaptic weights at the onset of training ($\mathbf{W}_1^{K-1}$ and $\mathbf{W}_2^{K-1}$). As such, ECL has tractable computational and memory complexity (Figure 5.5B) and does not require resource intensive calculations in between training pairs, like derivation of the Hessian matrix or empirical Fisher information. In brief, the presented ECL algorithm uses the fact that after successful acquisition of some knowledge, this knowledge is fully encoded into the network's synapses and thus readily available during training. As a consequence, we have to set $\mathbf{W}_2^0\mathbf{W}_1^0 = \mathbf{0}$ independent of the actual initial values, as at the begin-

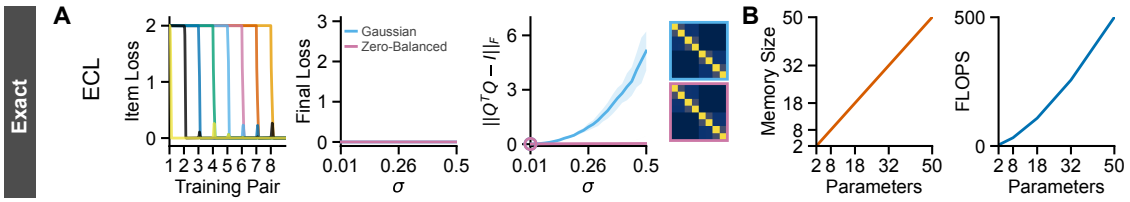

Figure 5.5: Exact categorical learning. (**A**) Exact continual learning of semantic hierarchy with eight objects (first column) results in zero mean squared error when trained from random Gaussian (blue) or zero-balanced (red) initial weights across a range of standard deviations (second column). Neural representations are task-specific for small Gaussian initial weights and zero-balanced weights. Example RSMs for best performing networks (third column). Solid lines and shaded areas indicate mean and ± standard deviation across 50 random initialisations. (**B**) The memory and floating point operations per training step as a function of network parameters.

ning of the training, there is no knowledge encoded in the network. It is reminiscent of previously suggested function normalisation for continual learning (Benjamin et al., 2018) while being analytically exact. For a full derivation see Theorem C.6.

5.3.4 Preserving relational knowledge during learning

A remarkable aspect of intelligent behaviour is the ability to continuously integrate information about relationships between objects, places, and events into a coherent understanding of the world. This process must effectively adapt and integrate newly arriving information into an coherent worldview. A single experience, such as a plot twist in a work of fiction, may require previous relations to be reorganized instantly to accommodate the new information. However, it remains unclear how relational knowledge can be continuously learned in both biological and artificial neural networks, given that changes in connectivity could overwrite previously acquired knowledge.

Relational knowledge can be represented as a connected graph, in which nodes represent objects and edges represent relationships. For example, transitive and spatial relationships, and hierarchies can all be represented as a graph (Figure 5.6A-B). Relations can be either learned by being exposed to two nodes (e.g. giraffe and

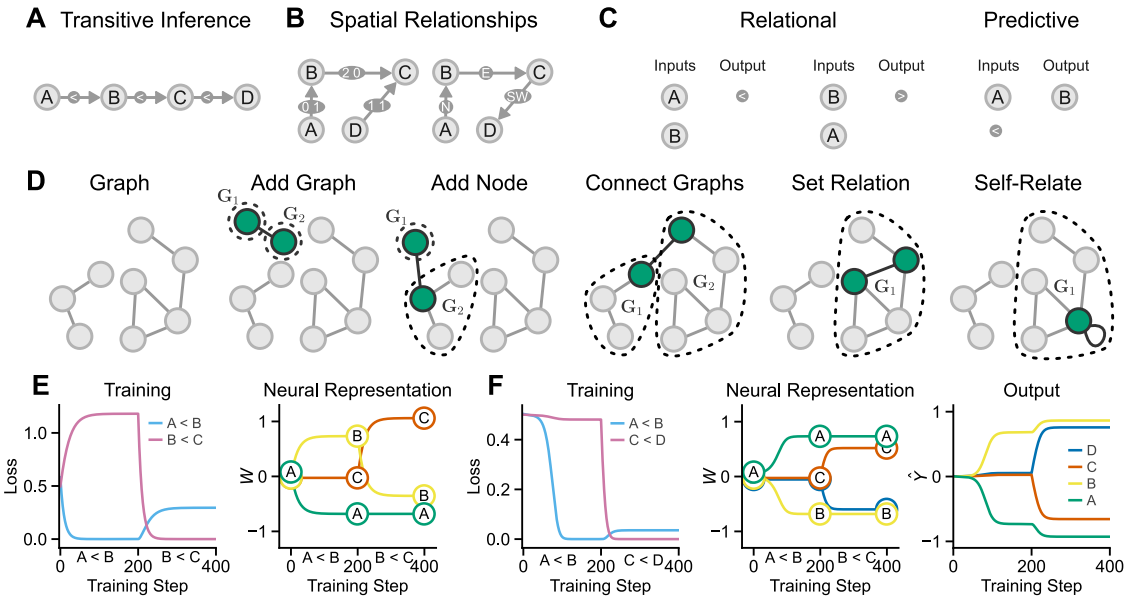

Figure 5.6: Relational knowledge. (**A**) Relational knowledge like transitive inference tasks and (**B**) spatial relationships using euclidean or abstract cardinal relationships can be represented as graphs. (**C**) Learning can either be relational, in which two nodes have to be mapped to their relation, or predictive, in which a node and a relation have to be mapped to the corresponding node. (**D**) New experiences can add a new graph, add a node to an existing graph, connect two existing graphs, connect two graphs, add a new relation within a graph, or add a self-relation. (**E**) Loss and neural representations throughout training on a transitive inference task in a one-layer linear network. During the acquisition of $B < C$, the update to the neural representation of B is corrupting the relation between $A < B$. Note how the neural representation A remains static during the acquisition of the second relation. (**F**) Loss, hidden-layer neural representations, and network output throughout training on a transitive inference task in a two-layer linear network. During the acquisition of $C < D$, the hidden-layer representation of A and B remains unchanged, however their output and loss change due to updates in the readout-layer.

elephant) and their edge has to be learned (e.g. larger than) or through predictive coding, in which a node and an edge (e.g. Bob and sister) have to be related to the corresponding node (e.g. Alice) (Figure 5.6C). In both cases, learning can be exhaustively studied as a sequence of alterations to a graph and its connections. A new experience may (a) establish a new graph, (b) add a node to an existing graph, (c) connect two graphs, (d) add a new relation within a graph or (e) add a self-relation (Figure 5.6D). For example, when first entering your neighbours house (a), you could learn how their foyer is connected to the backyard (b), discover that their backyard is connected to yours (c), find a second door that connects the garden and the foyer (d), and notice the high ceiling of their foyer (e).

Formally, during the acquisition of relational knowledge, two objects i and j have to be mapped to a set of relationships $\mathbf{y}_{ij}$. To encode pairs of objects, we make

Assumption 11. *Inputs are encoded as hot-cold vectors of unit length, i.e.,* $\mathbf{x}_{ij}^T\mathbf{x}_{ij} = 1$, $\mathbf{x}_{ij}^T\mathbf{x}_{jk} = \frac{1}{2}$, *and* $\mathbf{x}_{ij}^T\mathbf{x}_{kl} = 0$.

For example, the hot-cold vector that can encode relationships of up to 4 objects and specifically encodes for object 1 and 3 would be $\mathbf{x}_{1,3} = [0, \sqrt{1/2}, 0, -\sqrt{1/2}, 0]$. As a consequence, inputs are not generally decorrelated and tasks cannot be learned in a one-shot manner in a single-layer neural network anymore. For example, when learning two transitive relationships $A < B$ and $B < C$, while learning the second relationship, knowledge about the first is corrupted as the representation for B is updated (Figure 5.6E). This also applies to two-layer neural networks, where naïve gradient descent optimization leads to catastrophic forgetting. In this case however, there is also catastrophic forgetting when there is no overlap between inputs, for example when learning transitive relationships $A < B$ and $C < D$, due to the readout weights being shared across all inputs (Figure 5.6F). Specifically, during the acquisition of the second relation the hidden-layer neural representations

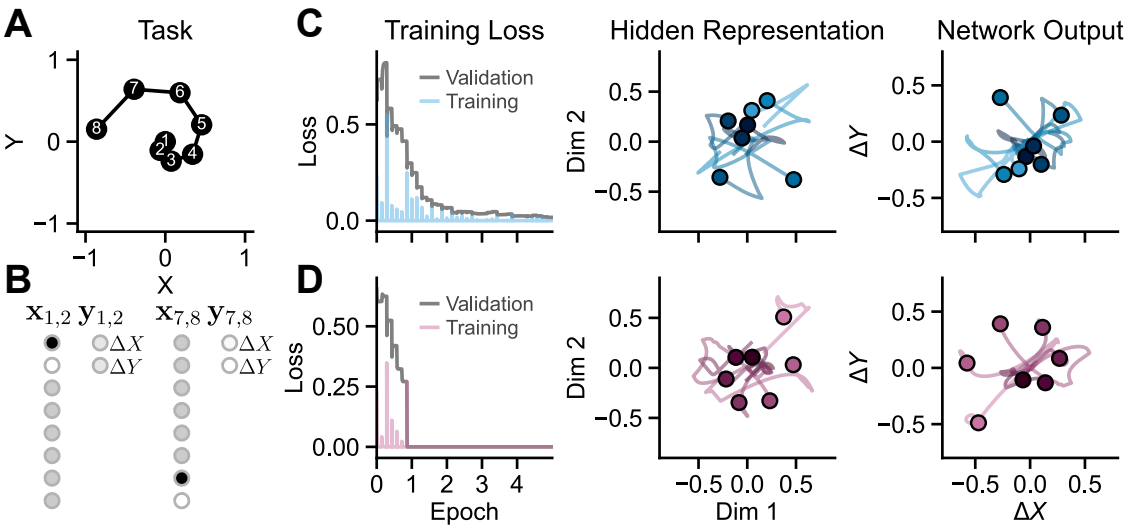

Figure 5.7: Exact relational learning. (**A**) Given a graph with 8 nodes and 7 edges that are arranged as a spiral. The task consist of the 7 pairwise relationships between nodes. (**B**) Example training pairs. Pairwise relationships have to be mapped to their relative distance in two dimensions. (**C**) Training with naïve gradient descent does not converge to the global optimum even after multiple training epochs. After one epoch, the hidden layer representation and network output are disorganised and incoherence with the provided information. (**D**) ECL seamlessly integrates information provided during one epoch of training to a task-specific hidden layer representation and accurate network outputs.

of the first pair remain unchanged. Meanwhile, the readout weights are updated, altering how the first relation is mapped to the output and thus causing catastrophic forgetting. In summary, forgetting occurs not despite but because hidden-layer neural representations of the first pair remain unchanged.

Exact relational learning

In the following, we present a general ECL rule for relational learning that can be formulated as connected graphs. Let's consider the case in which the network has to learn to map the k-th relational input $\mathbf{x}_{ij}$ to its corresponding output $\mathbf{y}_{ij}$ and make

Assumption 12. *All inputs $\mathbf{x}_{ij}$ either establish a new graph, add a new node to a graph or connect two graphs.*

Further, let object i belong to graph $\mathbf{G}_1$ and object j belong to graph $\mathbf{G}_2$, then, we define the entries of vector $\mathbf{h}$ for the $n = 1, ..., P$ objects as

$$\mathbf{h}(n) = \begin{cases} \frac{1}{\sqrt{2}|\mathbf{G}_1|} & \text{if object } n \in \mathbf{G}_1, \\ -\frac{1}{\sqrt{2}|\mathbf{G}_2|} & \text{if object } n \in \mathbf{G}_2, \\ 0 & \text{otherwise,} \end{cases} \tag{5.17}$$

where $|\mathbf{G}_1|$ and $|\mathbf{G}_2|$ denote the size of the two graphs respectively. Additionally, we define

$$\mathbf{y}_0 = \mathbf{W}_2^{K-1}\mathbf{W}_1^{K-1}\mathbf{x}_{ij}. \tag{5.18}$$

Then, we propose that if

$$\mathbf{x}_k = \mathbf{h} \quad \text{and} \quad \mathbf{y}_k = \mathbf{y}_0 - \mathbf{y}_{ij}, \tag{5.19}$$

minimising $\mathcal{R}_{\text{Exact}}$ is gradient equivalent to minimising the distance to the least-squares solution across all training data. Therefore, as in the case of ECL for relational knowledge, learning is only depended on the current state of the synaptic weights and the state of the synaptic weights at the onset of training on the current training pair. Crucially, the only knowledge required to generate $\mathbf{h}$ is if objects are part of the same graph. The approach does not require memorizing previously acquired knowledge, nor does it address the stability-plasticity dilemma. For example, while learning the pairwise relationship of items that are connected as a spiral (Figure 5.7A-B), when using naïve optimisation, even after multiple epochs of repeated exposure, the linear two-layer network does not converge to a solution (Figure 5.7C). During the acquisition of the current training pair, previously acquired relations are not preserved, leading to catastrophic forgetting. In contrast, using the ECL rule, the neural network preserves previously acquired knowledge and seamlessly assembles incoming information to one coherent whole, reaching

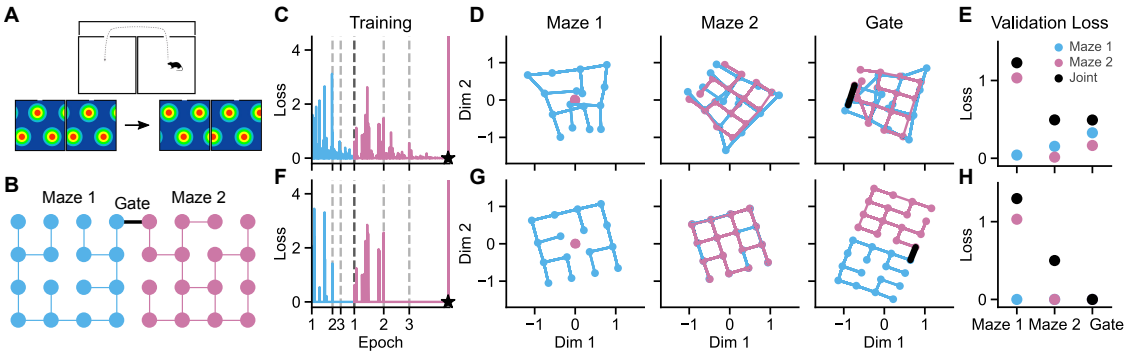

Figure 5.8: Building a cognitive map. (**A**) Experimental setup from Carpenter et al. (2015). Rats are placed in two identically looking rooms that are connect via a corridor (top). Initially, grid cells form two identical patterns for each of the rooms (bottom left) and over time they form one coherent whole (bottom right). Panel is reproduced from Carpenter et al. (2015) with permission. (**B**) Schematic of the experimental setup. Two independent mazes that are connected by a gate. (**C**) Training with naïve gradient descent on the first maze (blue), second maze (red), and connecting corridor (black star). (**D**) Hidden-layer neural representations of positions within the maze (dots), visualised using multidimensional scaling, after training on the first and second maze, and on the relationship that is connecting the two mazes. Lines visualise relationships between locations. (**E**) Validation loss on items of the first, second and joint maze. (**F**) Same as in C but for ECL. Knowledge about each maze is integrated within a single epoch. (**G**) Same as D but for ECL. The algorithm seamlessly forms and assembles neural representations into one coherent whole as new information arrives. (**H**) Validation loss on items of the first, second and joint maze. Knowledge is integrated perfectly.

perfect accuracy after a single exposure to each pairwise relation.

Building a cognitive map

The idea of a cognitive map refers to a mental representation that helps to organise and navigate relational knowledge. Tolman (1948) demonstrated the existence of such mental representations through experiments with rats in mazes, showing that they rely on an internal map rather than a simple mapping from visual cues to behaviour. But how are these cognitive maps assembled from experience (Whittington et al., 2022)? To assess the ECL algorithm's ability to build cognitive maps, we conducted an experiment that was inspired by the work of Carpenter et al. (2015); they observed that grid cells initially form two overlapping grids, which

gradually merge into a unified whole when rats are exposed to two connected, identical rooms (Figure 5.8A). To this end, we placed an artificial agent in a maze and trained it to predict relative directions from adjacent locations (Figure 5.8B). After visiting each location in the maze at least three times, the agent was moved to a second maze. Once the agent had similarly explored the second maze, a gate connecting the two mazes was opened. While naïve optimisation with gradient descent fails to form neural representations that are coherent with the experience of the agent (Figure 5.8C-E). In contrast, ECL seamlessly integrates the information within each maze and stitches knowledge into one coherent whole upon discovery of the gate (Figure 5.8F-H). This is in contrast to the Tolman-Eichenbaum machine (Whittington et al., 2020), which requires prior training on tasks with identical structure to integrate knowledge and has difficulty assembling multiple pieces of knowledge (Hosoya, 2024). Crucially, the ECL rule is general and not limited to spatial relationships, but generalises to any kind of relational knowledge, as long as inputs are encoded as hot-cold vectors (Assumption 11) and knowledge acquisition only relies on operations that establish new graphs, add new nodes or connect graphs (Assumption 12).

5.3.5 Exact continual learning requires drifting neural representations

The central challenge of continual learning it to preserve the input-output map of previously acquired knowledge stable while adjust neural connectivity to integrate new information. This might lead to the assumption that under stable behavioural and perceptual stability for a stimulus, the associated neural structure and thus the corresponding neural representations, should remain stable too. Since two-layer linear neural networks are highly overparameterised, continual learning with stable neural representations in our setting is possible. This can be intuitively

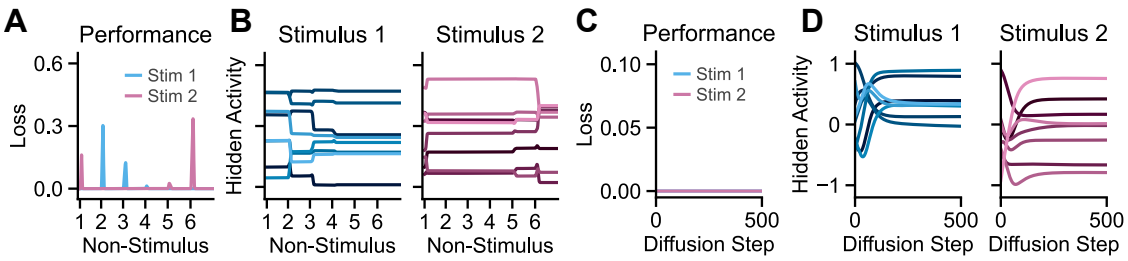

Figure 5.9: Exact continual learning requires drifting neural representations. (**A**) Network performance on two stimuli that are used during an experiment during the acquisition of new knowledge about six items that are not part of the experiment themselves. (**B**) The hidden-layer representation of experimental stimulus one (left) and two (right) changes during the integration of knowledge about non-stimuli outside of the experiment with the ECL rule. (**C**) The loss for the two stimuli remains zero during a diffusion on the solution manifold from a general to a minimum-norm solution, however, (**D**) the hidden-layer neural representation is drifting for both stimuli.

understood by considering an extreme case: setting $\mathbf{W}_1 = \mathbf{I}$ and performing all learning in $\mathbf{W}_2$. However, under the additional constraint that learning must integrate knowledge into task-specific neural representations that reflect the structure of the world, we must revise this line of reasoning. As new information is incorporated, the representations of related knowledge must be updated too. As a result, we would predict that an experimenter may observe representational drift and changes in representational similarity for stimuli under investigation, even when no new information about the stimuli themselves is acquired, simply because they share structure with other experiences within or outside of the experiment that are not under investigation (Figure 5.9A-B). This becomes particularly evident in the case of relational knowledge, where changing the neural representations of one item necessitates corresponding updates to all related items to preserve their relationships. The example of building a spatial cognitive map in Figure 5.8 provides a clear illustration of this idea. One situation where learning can occur without drifting representations is when inputs and target outputs are in fact decorrelated, i.e., $\mathbf{x}_i^T \mathbf{x}_j = \mathbf{y}_i^T \mathbf{y}_j = 0$. In this cases, task-specific representations are decorrelated and therefore can be implemented by separate parts of the network. We note, that

in addition to changes induced by learning, drift on the solution manifold may occur at any time, as for example induced by gradient noise (Chaudhari & Soatto, 2018; Avidan et al., 2023; Li et al., 2024) or as we show in Figure 5.9C-D, when refining representations from task-agnostic to task-specific. In summary, we argue that behavioural and perceptual stability during continual learning is observed not despite, but because of drifting neural representations.

5.3.6 Preserving knowledge with intricate correlational structure

In the previous sections we presented exact and efficient algorithms for continuous learning without catastrophic forgetting. However, these algorithms relied on stimuli being encoded as decorrelated vectors of unit length in the case of categorical learning or as hot-cold vectors in the case of relational knowledge. Therefore, they only work once stimuli can be dissociated and identified. In particular, the reason why these algorithms exist is that in both cases, the pseudo-inverse of the input correlations are tractable. This is generally not the case for inputs with higher order correlational structure. For example, images have intricate correlational structure and are neither categorical nor relational. Even in the simple, linear case, to learn without catastrophic forgetting from correlated information, an exact algorithm would have to incrementally compute, store and use the pseudo-inverse or an approximation thereof. Even though incremental algorithms to compute the pseudo-inverse exist (Sherman & Morrison, 1950; Meyer, 1973), this approach seems computationally prohibitive for a continuous stream of sensory information with intricate correlational structure.

We propose that continual learning must therefore rely on a two-fold process. First, a neural network has to learn to reliably distinguish objects from sensory streams and then, these representations can be utilised to learn in a continuous

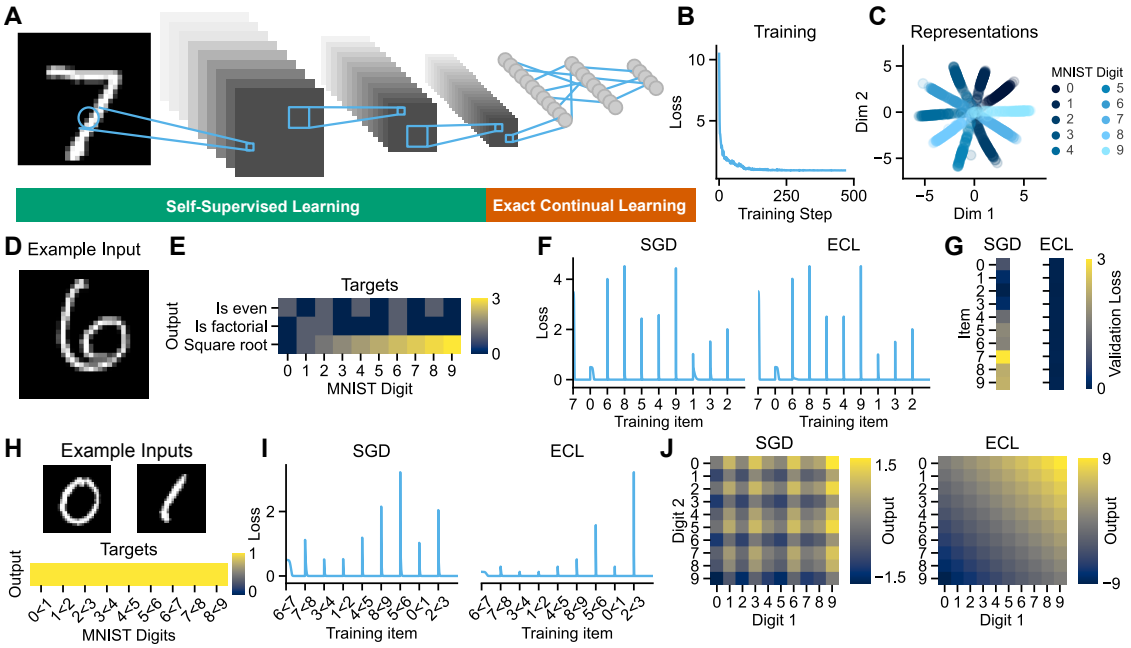

Figure 5.10: Exact continual learning from knowledge with intricate correlational structure. (**A**) Schematic of network architecture in which a convolutional neural network is trained using a self-supervised learning algorithm and readouts from the learned representations are learned using exact continual learning. (**B**) Loss over time during training a convolutional neural network on MNIST digits using self-supervised learning. (**C**) multidimensional scaling (MDS) of hidden-layer neural representations of MNIST digits at the end of self-supervised training. Representations for different image classes are orthogonal. (**D**) Example network input and (**E**) target outputs for a categorical learning task. (**F**) Training loss during continual learning. The network is trained on one digit of each class using either gradient descent or exact continual learning. (**G**) Validation loss after training on unseen MNIST digits. Gradient descent has failed to preserve knowledge throuhgout learning. (**H**) Example network input and target outputs for a relational learning task. (**I**) Training loss during continual learning. The network is trained on each relation using either gradient descent or exact continual learning. (**J**) Average network output for all possible relations. Gradient descent failed to preserve relationships during learning whereas exact continual learning seamlessly integrated all information.

fashion without catastrophic forgetting. In this chapter, we showed that the second step can be achieved using fast and accurate algorithms. In the following, we investigate how ECL may be combined with a sensory stream.

Self-supervised learning algorithms, such as Barlow Twins (Zbontar et al., 2021), are capable of projecting stimuli with complex correlational structures, such as MNIST digits, into orthogonal representations without requiring supervision (Figure 5.10A). Once objects are projected to dissociate representations like this, we can apply the ECL algorithm to learn categorical and relational knowledge, one-shot and without catastrophic forgetting and with perfect generalisation to previously unseen images of a known class.

For example, after a convolutional neural network has been successfully trained to map MNIST digits to dissociated, orthogonal representations using the Barlow Twins algorithm (Figure 5.10B-C), ECL can seamlessly use these representations to integrate facts about numbers, like if it is even, a factorial and its square root (Figure 5.10D-E). In particular, if the network is trained on the aforementioned facts on a single instance of each MNIST digit (Figure 5.10F), as expected, naive gradient descent suffers from catastrophic forgetting, whereas ECL perfectly integrates the knowledge and generalises to previously unseen images of the same class (Figure 5.10G).

Similarly, relational knowledge like if one number is larger or smaller than another (i.e., $3 < 4$) can be acquired without catastrophic forgetting (Figure 5.10H). Again, the convolutional network maps MNIST images to orthogonal representations, which then can be used by the ECL algorithm for relational knowledge to seamlessly integrate information. After a single exposure to a learning example of each relation, the network trained with the ECL algorithm perfectly generalises to previously unseen relations, and unseen images of the same class (Figure 5.10i-J).

We would predict that the first process, which maps sensory stimuli to unique representations, is slow and requires extended exposure to a sensory stream with

varied experiences, whereas the second, which acquires knowledge about facts and relations, is fast and seamless. This proposal is coherent with the observation that neural representations in sensory areas like the visual cortex are constantly changing, while cells that encode categorical knowledge like face-cells stay stable for months (McMahon et al., 2014). In summary, in such a network, a self-supervised learning algorithms would generate neural representations that allow to uniquely distinguish between items of a class, and an ECL algorithm would continuously integrate new information about and relations between classes.

Neural knowledge assembly in humans and neural networks

This chapter is based on published work (Nelli et al., 2023). If not indicated otherwise, contributions to the research presented in this chapter are as follows:

Supervision Professor Christopher Summerfield, Professor Andrew M. Saxe

Experimental design Stephanie Nelli, Christopher Summerfield

Human data acquisition Stephanie Nelli, Tsvetomira Dumbalska

Human data analysis Stephanie Nelli

Derivations, simulations, and figures Lukas Braun

6.1 Introduction

Human understanding of the world must adapt swiftly and continuously to make sense of how objects, people, and places related to each other. This ability to assemble and reassemble knowledge as new information emerges, often requires few-shot reorganisation of neural codes. However, how this rearrangement of relations can emerge in the brain, while preserving other relations intact, remains largely unknown. In this chapter, we examine the process of knowledge assembly in humans and neural networks through a transitive inference task. Human participants were first trained to learn a transitive ordering among novel objects within two distinct sets through pairwise comparisons. Following this training, participants were ex-

posed to a small number of trials that explicitly revealed a link between the two sets, revealing a single unified transitive ordering. Blood-oxygen-level-dependent (BOLD) signals (Logothetis, 2003) in dorsal frontoparietal cortical areas revealed that the neural representation of the objects within the two groups were rapidly rearranged in consistence with the connecting information. We show that human neural representations are task-specific and that the changes of neural representations during knowledge assembly are in line with the predictions made by the exact continual learning rule for relational knowledge in Chapter 5. To capture inter-individual differences in between participants, we extend the exact algorithm for relational knowledge to account for uncertainty in the relationships between objects.

6.2 Preliminaries and setting

6.2.1 Experiment

Human participants ($n = 34$) learned the transitive ordering of arbitrary-looking visual objects (Horst & Hout, 2016) through a visual decision-making task. Twelve objects were randomly assigned a ground truth rank along a nonsense dimension called *brispiness*, with object 1 being the most *brispy* and object 12 the least *brispy* (Figure 6.1A). The objects were arbitrary-looking and the dimension was intentionally meaningless to prevent participants from using familiar attributes or preconceived associations. The experiment consisted of two training phases, each followed by a test phase (Figure 6.1B).

During the first training phase, participants were trained in alternating blocks of 60 trials using objects from either the first half (objects 1–6) or the second half (objects 7–12) of the hierarchy. In the following, we refer to objects 1-6 as the first and objects 7-12 the second set. During each trial (Figure 6.1D), participants

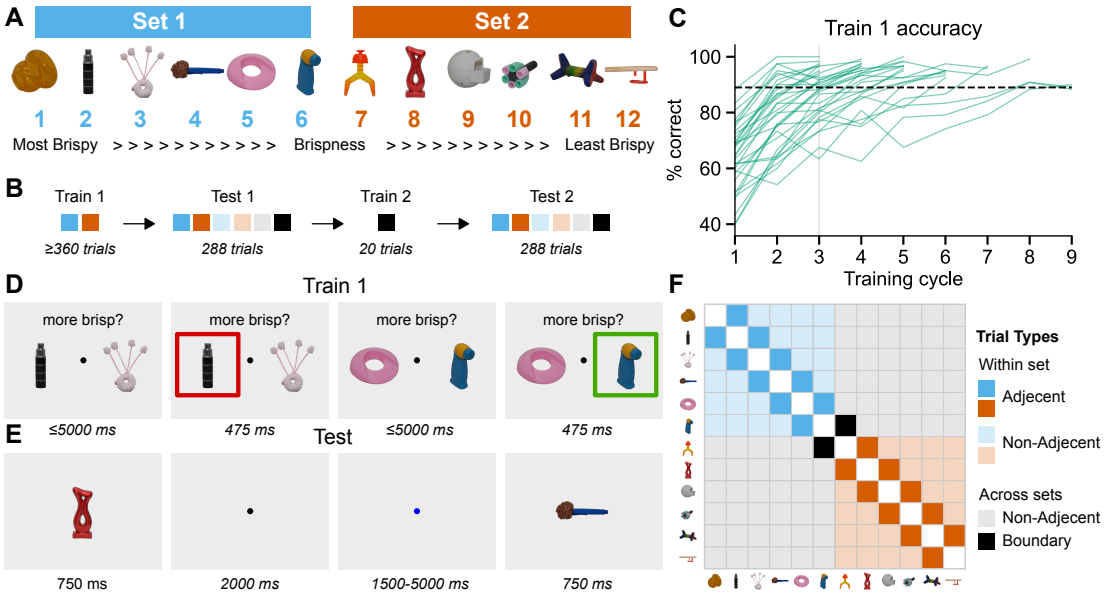

Figure 6.1: Experimental design. (**A**) Schematic of object assignment to ground truth rank of *brispiness*, and object separation into two sets. We note that each participant viewed a unique, randomly sampled set of objects. (**B**) Schematic of experimental design. Coloured squares refer to trial tyes as detailed in panel F. (**C**) Percentage accuracy during training for each participant (green), 90% stopping criterion (black), and minimum amount of trials (grey). A training cycle consist of two blocks, one for each set. (**D**) Example sequence of two trials during the first training phase. In each trial, participants were presented with a pair of objects and received feedback after providing a response. (**E**) Example sequence of a test trial. Participants indicated whether each object was more or less *brispy* than the previous one, followed by a brief, randomly timed interval for fMRI scanning, which was indicated by a blue fixation dot. (**F**) Schematic of trial types. Each square indicates a pair of stimuli as defined by their row and column. Adjacent pairs within sets (dark blue and red) were used during training phase one. The two objects at the boundary between the two sets (black) were used during training phase two. All pairs, within and across sets were used during testing.

were shown two objects with adjacent ranks (e.g. objects 3 and 4), and pressed a button to indicate which object was more (or less) *brispy*. After responding, feedback was provided with a green box (correct) or a red box (incorrect) around the selected object. Participants were trained on the sets in alternating blocks until they reached at least 90% accuracy on both sets (Figure 6.1C). Since only objects with adjacent rank were presented during training phase one, participants had to use transitive inference to infer relations of objects within each set. Importantly, no information about the relationship between the two sets was provided during the first training phase.

After the first training phase, participants entered the fMRI scanner for a test phase where they viewed all possible pairwise combinations of the 12 objects in random order. They performed a 1-back task, indicating the *brispiness* of each object relative to the preceding one via button press (Figure 6.1E). The test phase included comparisons of trained adjacent pairs (e.g., 2 and 3), untrained non-adjacent pairs within each set (e.g., 2 and 5), and untrained pairs spanning the two sets (e.g., 2 and 7) (Figure 6.1F). No feedback was provided during the test phase.

Immediately after the first test phase, participants began the second training phase while remaining in the scanner. This phase consisted of 20 training trials with feedback. During each trial, only the least *brispy* object from the first set (object 6) and the most *brispy* object from the second set (object 7) were presented. Note, that this relation had not been included in the first training phase and revealed that the two sets were connected, forming a single unified hierarchy. The second training phase was followed by a second test phase, identical to the first.

6.2.2 Model

During the experiment participants had to do pairwise comparisons between two objects i and j, one shown on the left-hand side and the other one on the right-hand

side of the screen. We model these comparisons using hot-cold input vectors $\mathbf{x}_{ij}$ of length 12 that are all 0 except at the i-th and j-th position, where they are $\sqrt{1/2}$ and $-\sqrt{1/2}$ respectively. If $i > j$, the target output is $\mathbf{y}_{ij} = 1$, indicating that object i is more *brispy* than object j, and if $i < j$, the target output is $\mathbf{y}_{ij} = -1$, indicating that object i is less *brispy* than object j respectively (Figure 6.2A). Like in the experiment, we do not perform training or comparisons between identical objects, i.e., $i = j$. This results in a dataset of $P^1 = 20$ training pairs for the first training phase, a dataset of $P^2 = 2$ training pairs for the second training phase, and a combined dataset with a total of $P = 22$ training pairs. We use two-layer linear neural networks

$$\hat{\mathbf{y}}_{ij} = \mathbf{w}_2^T \mathbf{W}_1 \mathbf{x}_{ij}, \tag{6.1}$$

with $N_h = 6$ hidden neurons as a network model, where $\mathbf{W}_1 \in \mathbb{R}^{6\times 12}$ and $\mathbf{w}_2^T \in \mathbb{R}^{1\times 6}$ denote the first and second layer network weights respectively (Figure 6.2B). We quantify the performance of the network using the mean squared error

$$\mathcal{L}_{\text{MSE}} = \frac{1}{2P} \sum_{n=1}^{P} ||\hat{\mathbf{y}}_{ij} - \mathbf{y}_{ij}||_2^2 \tag{6.2}$$

and use the representational similarity matrix

$$\text{RSM}_{ij} = \mathbf{x}_i^T \mathbf{W}_1^T \mathbf{W}_1 \mathbf{x}_j \tag{6.3}$$

to compare neural representations. In our setting, it follows from Laurent & Brecht (2018) that any combination of optimal network weights $\mathbf{\Omega}_2$ and $\mathbf{\Omega}_1$ for which

$$\mathbf{\Omega}_2 \mathbf{\Omega}_1 \mathbf{\Sigma}_{xx} = \mathbf{\Sigma}_{yx} \tag{6.4}$$

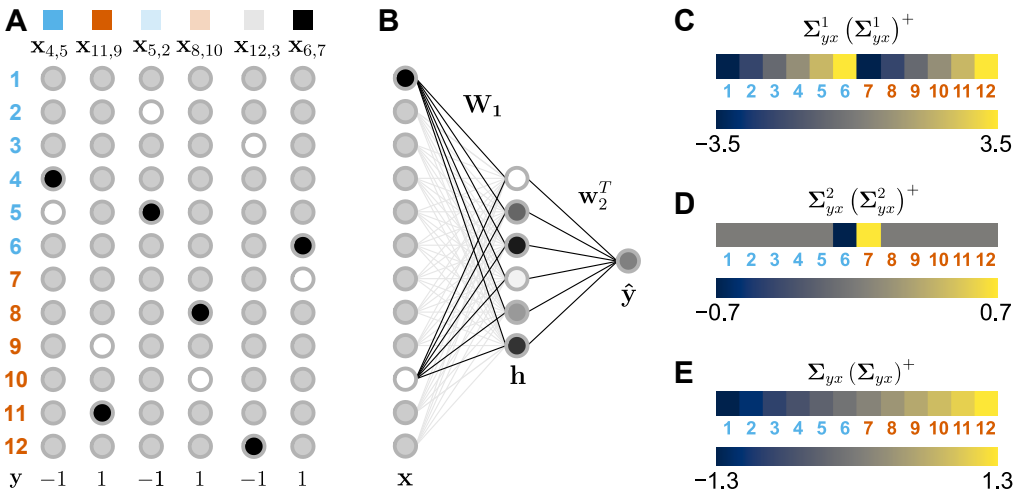

Figure 6.2: Network Model. (**A**) Example hot-cold encoding of training pairs for different trial types, with colour codes as in Figure 6.1. White, grey and black circles represent inputs of $-\sqrt{1/2}$, 0 and $\sqrt{1/2}$ respectively. (**B**) Schematic of the network architecture. Input weights combine the two inputs to a joint hidden layer representation, which is then mapped to a prediction by the second layer weights. (**C**) Least-squares solution of the training data in phase one, (**D**) phase two, and (**E**) combined training data across the two phases are one-dimensional and reveal the inherent structure of the tasks.

is a linear solution (Appendix A.1). To convert the outputs of a linear network into choice probabilities, we applied a sigmoid function to the outputs

$$p(i > j) = \frac{1}{1 + e^{\hat{\mathbf{y}}_{ij}}}. \tag{6.5}$$

We say that a network uses a task-specific neural representation, if the network uses a minimum-norm solution (Appendix A.3), that is

$$\mathbf{\Omega}_2 = \mathbf{U}\sqrt{\mathbf{S}}\mathbf{R}^T \text{ and } \mathbf{\Omega}_1 = \mathbf{R}\sqrt{\mathbf{S}}\mathbf{V}^T, \tag{6.6}$$

where

$$\text{SVD}\left(\mathbf{\Sigma}_{yx}\left(\mathbf{\Sigma}_{xx}\right)^+\right) = \mathbf{U}\mathbf{S}\mathbf{V}^T, \tag{6.7}$$

is the compact SVD of the least-squares solution and $\mathbf{R}$ is an arbitrary semi-orthogonal matrix. The least-squares solution for the first, second, and combined

datasets is one-dimensional and reveals the underlying structure for the different training phases and the combined training dataset. The solution for training phase one reveals the ordering within the two separate sets (Figure 6.2C), the solution to the second training phase reveals the pairwise relationship between the two objects at the boundary (Figure 6.2D), and the solution for the combined datasets reveals the full order across both sets (Figure 6.2E).

6.3 Results

6.3.1 Minimum-norm solutions predict human behaviour and neural representations during test phase one

During the first training phase, participants acquired knowledge about object *brispiness* of adjacent objects within each set. We begin our analysis by investigating how this knowledge was generalised to the relationships of other object pairs both within and across the two sets. As shown in Figure 6.3A, the minimum-norm solution of a two-layer linear network makes the following predictions for responses during the first test phase. First, participants should perform transitive inference, i.e., generalise knowledge about object *brispiness* from trained adjacent pairs (e.g. $2 > 3$) to untrained non-adjacent pairs within each set (e.g. $2 > 5$). Second, accuracy should increase with growing distance between comparanda, known as the symbolic distance effect (Woocher et al., 1978; D'amato & Colombo, 1990). And last, participants should match rank orderings between sets, e.g., rank the 3rd object in one set identically to the 3rd object of the other set and higher than the 4th object in either of the sets. While least-squares and general linear solutions predict transitive inference, and the symbolic distance effect, they do not generally predict rank matching. In both cases, there exist solutions in which one set is partially or entirely perceived as more or less *brispy* than the other (Figure 6.3B-C).

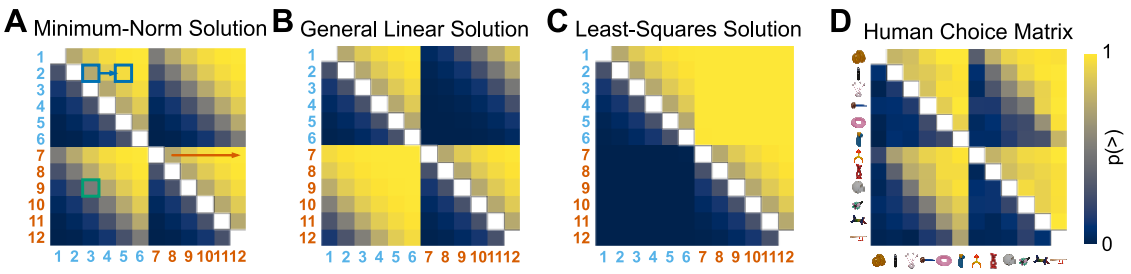

Figure 6.3: Network and human behaviour. (**A**) In the network choice matrix, the colour of each entry represents the probability of the network responding *greater than* for the object pair defined by the corresponding row and column during the first test phase. The minimum-norm solution network predicts transitive inference (e.g. blue entries), the symbolic distance effect (e.g., orange arrow) and rank matching (e.g., green entry). (**B**) Example choice matrix of a general linear solution in which the first set is perceived as more *brispy* than the second set. (**C**) Example choice matrix of a least-squares solution in which the entire second set is perceived as more *brispy* than the first one. (**D**) Average human choice matrix for test phase one, presented in the same format as A. Note the close resemblance between the minimum-norm network and human choice matrices, including transitive inference, the symbolic distance effect, and rank matching.

Human behaviour (Figure 6.3D) during the first test phase followed all three principles, leading to choices as predicted by a neural network with minimum-norm solution with task-specific representations. First, participants performed transitive inference within each set, as shown by an accuracy of $M = 96.7\%$ ($SD = 19.2$) on untrained non-adjacent pairs. Second, fitting a linear regression revealed the symbolic distance effect, showing an increase in participants' accuracy by $\beta = 3.4\%$ per rank with growing object distance. Finally, participants made choices that were indicative of rank matching between contexts. The mean accuracy of adjacent (e.g. the 3rd and 4th object of set one and two repectively, that is objects 3 and 10) and non-adjacent (e.g. object 3 and 12) pairs across sets were $M = 75.9\%$ ($SD = 20.9$) and $M = 91.5\%$ ($SD = 9.3$) respectively. Accordingly, the average human choice matrix and network choice matrix of the minimum-norm solution are highly correlated ($r = .97$, $p < .001$).

We proceeded by analysing and comparing the neural geometry of representations in both neural networks and humans. To this end, we probed networks

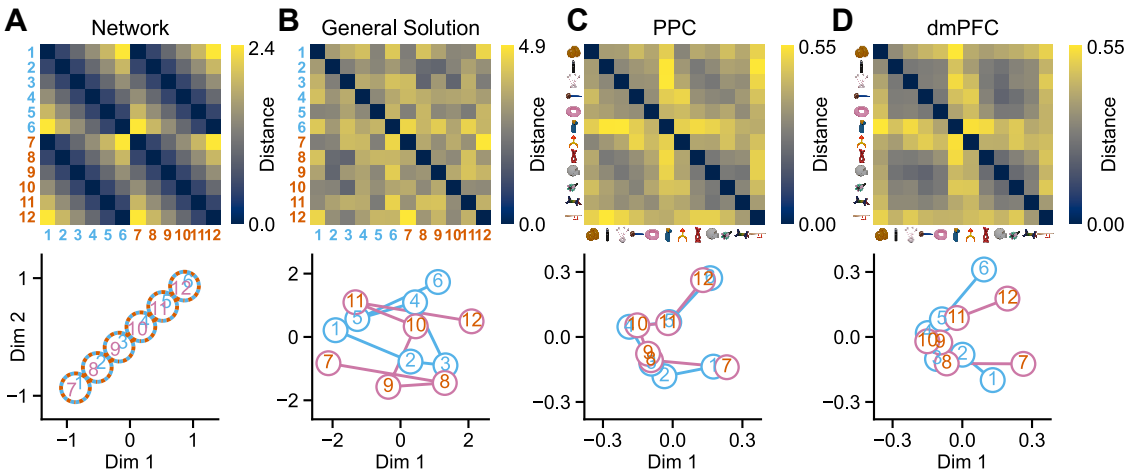

Figure 6.4: Representational geometry. (**A**) RSM of minimum-norm solution hidden-layer presentations and corresponding 2D MDS plot of the same data. Red and blue lines denote the two sets; numbered circles denote objects with their rank indicated by the inset number. (**B**) Same as A but for a general linear solution. (**C**) Neural data RSMs from patterns of BOLD in the PPC region of interest and corresponding 2D MDS plot of the same data. (**D**) Same as B but for dmPFS region of inters.

with individual objects to compute an RSM based on the hidden layer representations and next, applied multidimensional scaling (MDS) to visualise the similarity structure in two dimensions (Figure 6.4A). In accordance with the network behaviour, the hidden-layer neural representations of individual objects in the minimum-norm solution evolved along two aligned number lines with matching rank. In contrast, general and least-squares solutions yielded task-agnostic neural representations (Figure 6.4B). In humans, we examined neural representations in the posterior parietal cortex (PPC) and dorsomedial prefrontal cortex (dmPFC), which where identified as regions of interest during an independent task where participants judged the magnitude of Arabic digits. In both regions of interest, we observed a strong correlation between their respective RSMs (Figure 6.4C-D) and neural representations observed in the hidden-layer of the minimum-norm neural network. As investigated in great detail in the original publication, these effects survived cross-validation, including reaction times as a nuisance covariate, and were confirmed using a whole-brain searchlight approach (Nelli et al., 2023). In

line with the representations used by the minimum-norm solution, using MDS to visualise the neural geometry of BOLD signals in these areas revealed overlapping neural number lines with matching rank (Figure 6.4C-D). Unlike the network's neural representations, the number lines derived from BOLD data were curved. This phenomenon has previously been observed in scalp EEG recordings (Luyckx et al., 2019) and multi-unit activity in the macaque's lateral intraparietal cortex (Okazawa et al., 2021). We note that the curvature arises naturally in a model where inputs are subject to Gaussian noise, rather than relying on noise-free hot-cold input vectors. A neural representation in which ranked items are encoded on overlapping manifolds supports generalization across sets of items with equivalent levels of *brispiness*. The overlapping representations align with the structure of the task and inherently facilitate transitive inference, the symbolic distance effect, and importantly, rank-matching, as demonstrated in the analysis of network behaviour for minimum-norm solutions.

6.3.2 Exact continual learning predicts human behaviour and neural representations during test phase two

Next, we turned to the central question of how neural representations must be re-organised to integrate knowledge during the second training phase. The minimum-norm solution across all pairs of objects, including the two objects that connect set one and set two (i.e., 6 and 7), makes the following behavioural predictions: All items in set one should be ranked as more brispy than items in set two and the symbolic distance effect should span the whole range of items (Figure 6.5A). How then, can one rearrange the overlapping neural representations after the first training phase, into a single, unified hierarchy upon exposure to the two connecting objects, as predicted by the minimum-norm solution?

To this end, we optimised a neural network in accordance with the experimen-

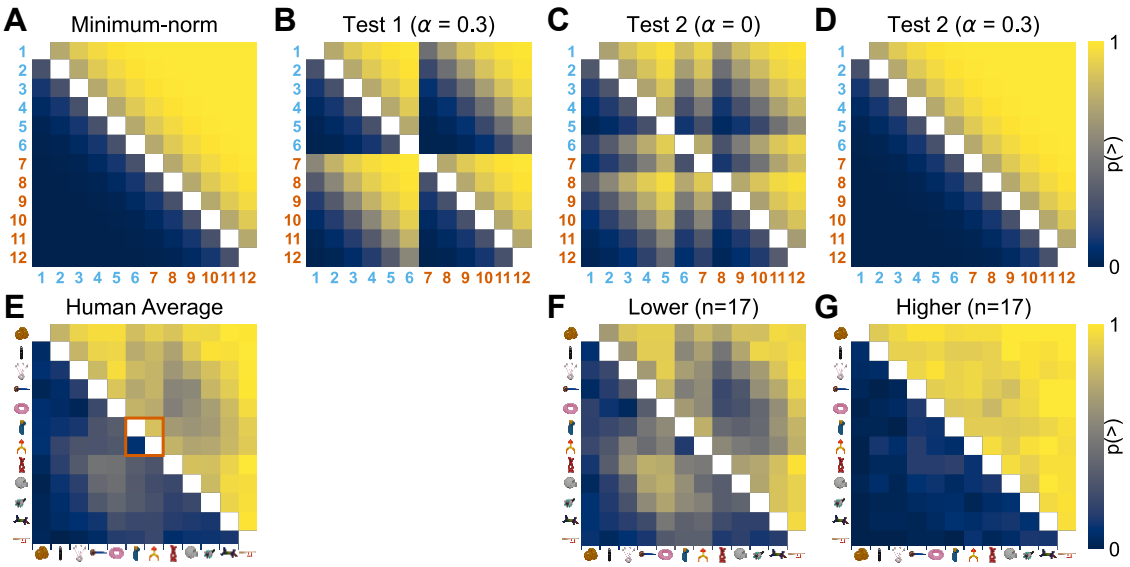

Figure 6.5: Final network and human behaviour. (**A**) Network choice matrix of minimum-norm solution. The network predicts that all items in set one are ranked higher than items in set two and the symbolic distance effect spans the whole range of items. (**B**) Network choice matrix after training phase one for $\alpha = 0.3$. Choice matrix after training phase one for $\alpha = 0$ is identical (not shown). (**C**) Network choice matrix for test phase one for $\alpha = 0$ learns the relation between items 5 and 6 as an exception, whereas (**D**) training with $\alpha = 0.3$ integrates information into responses that are reminiscent of the minimum-norm solution in A. (**E**) Average choice matrix across all human subjects after boundary training. The red box highlights item 5 and 6. On average, choices respect the ground truth rank. (**F**) Human choice matrix of lower performing subjects during test phase two. The choice pattern is reminiscent of training with $\alpha = 0$. (**G**) Human choice matrix of higher performing subjects is reminiscent of exact continual learning with $\alpha = 0.3$.

tal procedure. First, the network was trained on 960 trials of training phase one and subsequently on 20 trials of training phase two. As formulated, the learning problem can be solved in one shot using the ECL rule for relational knowledge (Section 5.3.4). The ECL rule relies on knowing whether two objects belong to the same graph. However, while the ECL rule connects items within a graph from a single exposure, humans gradually build certainty about relationships between experiences over time. To address this difference, we modified the ECL rule by incorporating a certainty measure to estimate whether two objects belong to the same graph (Methods 6.4.2). To regulate the speed at which certainty was accumulated, we introduced the hyperparameter α. When α was set to zero, relational evidence was ignored, and items were never grouped into the same graph, effectively reducing the ECL rule to gradient descent. In contrast, for $\alpha = 0.3$, the network gradually accumulated evidence that objects appearing together belonged to the same graph and that their relationships should be preserved as new knowledge was acquired.

When training started from small initial weights, the network converged to the minimum-norm solution of training phase one, independent of the choice of α (Figure 6.5B). However, during the second training phase, the neural network learned the relation between object 6 and 7 as an exception for small α, failing to preserve previously learned relationships between objects within each set (Figure 6.5C). In contrast, for large α the network learned that objects within each of the two sets are part of a graph and preserved these relationships. This allowed the network to seamlessly integrate knowledge during the second phase of training, leading to minimum-norm network behaviour (Figure 6.5D).

In humans, the second test phase showed that participants correctly learned the relation between objects 6 and 7 (accuracy: $86\% \pm 15\%$). On average, participants successfully inferred that all objects lay on a single, continuous axis of *brispiness* (Figure 6.5E). However, there was substantial variability in performance across

participants for other relations, as indicated by a drop in median accuracy from 88.3% in the first test phase to 79.8% in the second. Since average response times remained consistent between the first test phase ($1,153 \pm 41$ ms) and the second test phase ($1,166 \pm 43$ ms), this decline was unlikely to result from reduced attention between conditions. Instead, we reasoned that some participants may have failed to restructure their knowledge, continuing to view the two sets as independent and treating objects 6 and 7 as exceptions, consistent with the predictions of the network trained with $\alpha = 0$. Supporting this idea, splitting participants based on median performance revealed two groups: one group learned objects 6 and 7 as exceptions, while the other group successfully integrated the new information and linked the two sets into a single hierarchy (Figure 6.5F-G). Analysis of response patterns during the second test phase within each group revealed a strong correlation with the respective network models. Responses from the lower performing group were highly correlated with the network that ignored relational evidence ($r = .87$, $p < .001$), while responses from the higher performing group were highly correlated with the network that preserved relational structure ($r = .98$, $p < .001$). Thus, training the neural network using the ECL rule with different values of α correctly predicts successful and less successful patterns of knowledge assembly observed in humans.

Next, we analysed RSMs and the geometry of neural representation in both neural networks and BOLD signals during the second test phase. The minimum-norm solution across all pairs of inputs predicts neural representations that are one elongated number line, reflecting the ranking of items 1 to 12 (Figure 6.6A). Networks trained using gradient descent-like dynamics ($\alpha = 0$) failed to encode relations within each set, treating boundary items as exceptions while leaving the representations of other objects unchanged and overlapping (Figure 6.6B). In contrast, networks that encoded relations within each set ($\alpha = 0.3$), preserved the previously acquired relationships between objects by effectively pushing objects

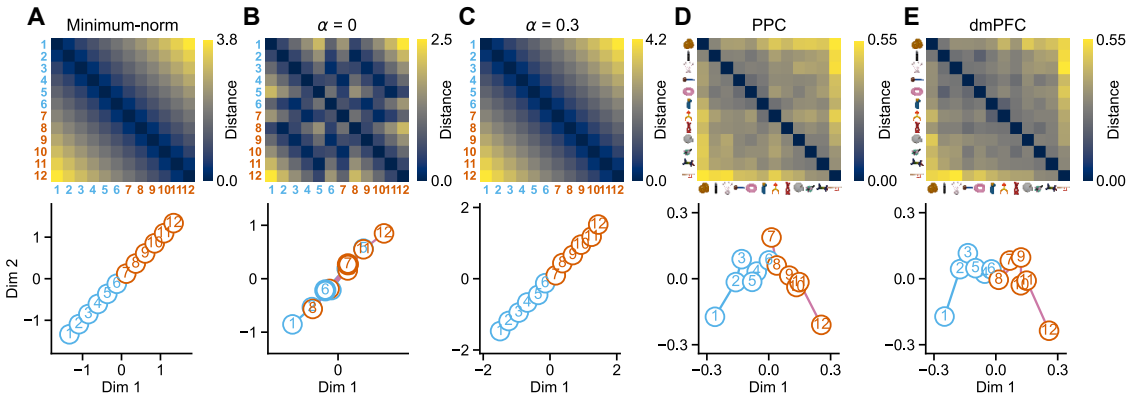

Figure 6.6: Final representational geometry. (**A**) RSM of minimum-norm solution hidden-layer presentations and corresponding 2D MDS plot of combines dataset consisting of all training pairs. (**B**) Same as A but for learning with $\alpha = 0$. Note how 6 and 7 are learned as an exception. (**C**) Same as A but for learning with $\alpha = 0.3$. (**D**) Neural data RSMs from patterns of BOLD in the PPC region of interest and corresponding 2D MDS plot of the same data during test phase two. (**C**) Same as B but for dmPFS region of inters.

within each context in opposing directions, leading to one elongated number line (Figure 6.6C). In BOLD, we observed neural geometry forming a single, elongated curved line in both the PPC and dmPFC regions of interest (Figure 6.6D-E). This reorganization of neural representations occurred within just 20 trials and aligns with the predictions of the minimum-norm neural network. The ECL algorithm offers a plausible mechanism for how such updates could be achieved.

6.4 Methods

6.4.1 Network model

All simulation studies were implemented using Python and a learning rate of $\eta = 0.05$. We implemented the two-layer linear neural network with $N_i = 12$, $N_h = 6$ and $N_o = 1$. Inputs where encoded as hot-cold vectors and target outputs as ± 1. Minimum-norm solutions where implemented as

$$\mathbf{\Omega}_2 = \mathbf{U}\sqrt{\mathbf{S}}\mathbf{R}^T \text{ and } \mathbf{\Omega}_1 = \mathbf{R}\sqrt{\mathbf{S}}\mathbf{V}^T, \tag{6.8}$$

where

$$\text{SVD}\left(\mathbf{\Sigma}_{yx}\left(\mathbf{\Sigma}_{xx}\right)^{+}\right) = \mathbf{USV}^{T}, \tag{6.9}$$

is the compact SVD of the least-squares solution of the respective training pairs and $\mathbf{R}$ is an arbitrary semi-orthogonal matrix that we sampled from the random orthogonal group. General and least-squares solution where implemented by sampling large initial weights from a uniform distribution in $[-3, 3]$ and $[-2, 2]$ for $\mathbf{W}_1$ and $\mathbf{W}_2$ respectively. We then, optimised the network with gradient descent unil convergence. In the case of least-squares solutions, we further ensured that $\mathbf{W}_1$ was restricted to the space covered by the inputs by setting

$$\mathbf{W}_1 = \mathbf{W}_1\mathbf{G}\mathbf{G}^{T} \quad \text{with} \quad \text{SVD}(\mathbf{X}) = \mathbf{GBH}^{T}. \tag{6.10}$$

6.4.2 Exact continual learning with evidence accumulation

Instead of directly encoding the graph structure and encoding relations as binary variable, we introduces the symmetric and square matrix $\mathbf{A} \in \mathcal{R}^{N_i \times N_i}$. Each of the entries of the matrix $\mathbf{A}_{ab}$ encoded how certain the network is, that two objects a and b are part of the same graph. Since objects are always correctly related to themselves, we enforced the diagonal of $\mathbf{A}$ to be equal to 1 at all times. Let the i-th row and column of $\mathbf{A}$ be denoted by $\mathbf{A}_{i*}$ and $\mathbf{A}_{*i}$ respectively, then, during training on input-pair $\mathbf{x}_{ab}$, we set

$$\mathbf{A}_{a*}(t+1) = \mathbf{A}_{*a}(t+1) = (1-\alpha)\mathbf{A}_{a*}(t) + \alpha\mathbf{A}_{b*}(t), \tag{6.11}$$

and respectively

$$\mathbf{A}_{b*}(t+1) = \mathbf{A}_{*b}(t+1) = (1-\alpha)\mathbf{A}_{b*}(t) + \alpha\mathbf{A}_{a*}(t). \tag{6.12}$$

Hence, training on objects i and j partially increased the certainty that object i is connected to object j and all other objects that object j was related to (and vice versa). The free parameter α determined how quickly evidence was accumulated. Training was then performed using the ECL rule for relational knowledge, where objects were considered part of the same graph if the corresponding entry $\mathbf{A}_{ab}$ exceeded a threshold of 0.95.

6.5 Discussion

In this chapter, we presented both behavioural and neural evidence for the rapid assembly of knowledge in human participants. In just 20 training trials, participants successfully linked two sets of related objects into one coherent whole. Remarkably, human behavioural patterns and neural representations were closely aligned with the minimum-norm solution of a two-layer linear network model, indicating that humans rely on task-specific, low-dimensional neural representations.

By adapting the ECL rule to dynamically integrate evidence of objects being part of the same graph structure, we were able to account for participants who either treated the boundary condition as an exception or those who successfully integrated knowledge into one coherent whole. Notably, naïve gradient descent could not account for the process of knowledge integration. The ECL rule is agnostic regarding how exactly certainty about object relations is acquired and encoded. As long as the binary information about whether two objects are related to each other is available, the ECL rule can preserve their relationship, independent from how and where the objects are encoded in the network. This provides a general, computational mechanism for relational learning in both humans and artificial systems.

While periods of quiet rest between trials could allow for alternative computational mechanisms such as replay, which has been linked to consolidation (Ego-

Stengel & Wilson, 2010) and spontaneous reorganisation of mental representation (Y. Liu et al., 2019), our training paradigm left very little time for rehearsal or replay. Given that boundary training was few-shot and the large number of training pairs that would need to be replayed, we think that it is unlikely that replay is the main driver of participants' knowledge assembly. Instead, the results imply that the brain may rely on efficient updates to its neural representations during learning, similar to how the ECL rule updates neural representations of relational structure.

We note that one potential downside of the ECL algorithm is that objects must be uniquely identifiable to be encoded as hot-cold inputs. For a discussion and analysis of transitive inference with flexible object encoding, see Lippl et al. (2024). One way to test the reliance on uniquely identifiable objects would be to conduct a knowledge stitching experiment, in which objects are not unique or novel but are highly similar. If two objects are confused, our theory predicts that their neural representations should overlap, causing them to be treated as part of the same graph. Consequently, if the neural representation of one of these objects is updated, the other should also be updated.

In summary, this chapter demonstrates that both human behavioural and neural data align with predictions from the ECL-based network model, suggesting that human relational learning involves efficient mechanisms for integrating and reorganizing knowledge. The results highlight how both humans and artificial networks may employ similar computational principles for relational learning, utilizing low-dimensional, task-specific neural representations and computational mechanisms that preserve previously acquired relational knowledge.

General Discussion

Understanding continual learning requires rethinking neural computations and representations. In this thesis, we used an analytically tractable model to study the continual learning problem in a computational system that processes information in a parallel and distributed manner, much like the brain. A key challenge in understanding continual learning in both biological and artificial neural networks is their inherent complexity. Even in artificial networks, where every part is accessible before, during, and after training, understanding how neural computations and representations generate network behaviour and evolve throughout learning is an exceptionally hard problem. In biological systems, this challenge is further aggravated by the inaccessibility of living tissue. Thus, analyses of neural computations are often limited to heuristics, approximations, or isolated observations of substructures and individual neurons. As a result, the conclusions drawn from these partial observations can be misleading.

For example, one might come to the conclusion that neural representations should remain stable to facilitate stable perception and behaviour. Here, we argue that contrary to this intuition, if the brain ought to acquire an input-output function that approximately operates in the minimum-norm regime, keeping representations fixed would be detrimental, and instead, representations must change to preserve knowledge during learning. Importantly, this remains true independently of what algorithm the brain employs for learning.

Similarly, one might assume that preserving neural computations requires maintaining the stability of individual neurons' computations. While this seems intu-

itive, the solution manifold in both linear and nonlinear neural networks reveals a whole set of solutions, offering significant flexibility in individual synaptic weights and neuron-specific computations. Though there may be some truth to this idea in biological neural networks, it is not the case at the representational level, as demonstrated by our findings.

Importantly, misconceptions stemming from the stability-plasticity hypothesis have shaped the field of continual learning and become the dominant framework for addressing this problem. Already in the 90s French (1991) argued "... you can't have it both ways. A system that develops highly distributed representations will be able to generalize but will suffer from catastrophic forgetting; conversely, a system that develops very local presentations will not suffer from catastrophic forgetting, but will lose some of its ability to generalize." In this thesis, we show that this intuition is not universally true. By deriving a loss function from first-principle that leverages the knowledge embedded within synaptic weights, we demonstrate how to perfectly integrate new information into task-specific neural representations, without catastrophic forgetting.

Ultimately, a continual learning algorithm should produce networks that utilise task-specific neural representations as if they were trained from scratch on the entire data distribution. Our exact continual learning framework is a first step in this direction. The exact continual learning loss is a function of synaptic weights and the current training pair only, and thus provides an ideal starting point for developing biologically plausible local learning rules that effectively integrate knowledge during learning. Extending or adapting this approach to nonlinear networks, whether through approximations or rigorous analysis, remains a promising avenue for future research. Crucially, our framework provides testable predictions, and we present initial experimental evidence indicating that humans may employ a mechanism akin to exact continual learning.

Until the continual learning problem is solved, artificial networks will remain

static, lacking true adaptability and requiring costly retraining from scratch as the world evolves, posing a tremendous financial and environmental burden (Strubell et al., 2020). Taken together, our results provide a powerful mathematical framework for studying continual learning in both biological and artificial neural networks and bringing us one step closer to understanding efficient and adaptive learning.

References

Abraham, W. C., & Robins, A. (2005). Memory retention–the synaptic stability versus plasticity dilemma. *Trends in neurosciences*, *28*(2), 73–78.

Advani, M. S., Saxe, A. M., & Sompolinsky, H. (2020). High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, *132*, 428–446.

Aitken, K., Garrett, M., Olsen, S., & Mihalas, S. (2022). The geometry of representational drift in natural and artificial neural networks. *PLOS Computational Biology*, *18*(11), e1010716.

Alisha, A., Bettina, V., & Simon, P. (2023). Representational drift in barrel cortex is receptive field dependent. *bioRxiv*.

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., & Tuytelaars, T. (2018). Memory aware synapses: Learning what (not) to forget. In *Proceedings of the european conference on computer vision (eccv)* (pp. 139–154).

Aljundi, R., Belilovsky, E., Tuytelaars, T., Charlin, L., Caccia, M., Lin, M., & Page-Caccia, L. (2019). Online continual learning with maximal interfered retrieval. *Advances in neural information processing systems*, *32*.

Aljundi, R., Lin, M., Goujaud, B., & Bengio, Y. (2019). Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, *32*.

Allen-Zhu, Z., Li, Y., & Song, Z. (2019). A convergence theory for deep learning via over-parameterization. In *International conference on machine learning* (pp. 242–252).

Alvarez, P., & Squire, L. R. (1994). Memory consolidation and the medial temporal lobe: A simple network model. *Proceedings of the national academy of sciences*, *91*(15), 7041–7045.

Amodei, D., Hernandez, D., Sastry, G., Jack, C., Brockman, G., & Sutskever, I. (2018). *Ai and compute.* Retrieved 2024-08-15, from `https://openai.com/index/ai-and-compute/`

Anthes, D., Thorat, S., Konig, P., & Kietzmann, T. C. (2024). Continual learning in artificial neural networks as a computational framework for understanding representational drift in neuroscience. In *Proceedings of the conference on cognitive computational neuroscience (ccn).*

Arora, S., Cohen, N., Golowich, N., & Hu, W. (2018). A convergence analysis of gradient descent for deep linear neural networks. *arXiv preprint arXiv:1810.02281*.

Arora, S., Cohen, N., Hu, W., & Luo, Y. (2019). Implicit regularization in deep matrix factorization. *Advances in Neural Information Processing Systems*, *32*.

Atanasov, A., Bordelon, B., & Pehlevan, C. (2021). Neural networks as kernel learners: The silent alignment effect. *arXiv preprint arXiv:2111.00034*.

Atkinson, C., McCane, B., Szymanski, L., & Robins, A. (2021). Pseudo-rehearsal: Achieving deep reinforcement learning without catastrophic forgetting. *Neurocomputing*, *428*, 291–307.

Avidan, Y., Li, Q., & Sompolinsky, H. (2023). Connecting ntk and nngp: A unified theoretical framework for neural network learning dynamics in the kernel regime. *arXiv preprint arXiv:2309.04522*.

Bachlechner, T., Majumder, B. P., Mao, H., Cottrell, G., & McAuley, J. (2021). Rezero is all you need: Fast convergence at large depth. In *Uncertainty in artificial intelligence* (pp. 1352–1361).

Baldi, P., & Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, *2*(1), 53–58.

Bang, J., Kim, H., Yoo, Y., Ha, J.-W., & Choi, J. (2021). Rainbow memory: Continual learning with a memory of diverse samples. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 8218–8227).

Bauer, J., Lewin, U., Herbert, E., Gjorgjieva, J., Schoonover, C., Fink, A., ... Hübener, M. (2023). Sensory experience steers representational drift in mouse visual cortex. *bioRxiv*, 2023–09.

Bengio, Y. (2014). How auto-encoders could provide credit assignment in deep networks via target propagation. *arXiv preprint arXiv:1407.7906*.

Bengio, Y., Ducharme, R., & Vincent, P. (2000). A neural probabilistic language model. *Advances in neural information processing systems*, *13*.

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning* (pp. 41–48).

Benjamin, A. S., Rolnick, D., & Kording, K. (2018). Measuring and regularizing networks in function space. *arXiv preprint arXiv:1805.08289*.

Benzing, F. (2022). Unifying importance based regularisation methods for continual learning. In *International conference on artificial intelligence and statistics* (pp. 2372–2396).

Biehl, M., & Schwarze, H. (1995). Learning by online gradient descent. *Journal of Physics A: Mathematical and general*, *28*(3), 643.

Bowers, J. S., Malhotra, G., Dujmović, M., Montero, M. L., Tsvetkov, C., Biscione, V., ... Blything, R. (2023). Deep problems with neural network models of human vision. *Behavioral and Brain Sciences*, *46*, e385.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., ... Zhang, Q. (2018). *JAX: composable transformations of Python+NumPy programs.* Retrieved from `http://github.com/google/jax`

Braun, L., Dominé, C., Fitzgerald, J., & Saxe, A. M. (2022). Exact learning dynamics of deep linear networks with prior knowledge. *Advances in Neural Information Processing Systems*, *35*, 6615–6629.

Braun, L., & Vogels, T. (2021). Online learning of neural computations from sparse temporal feedback. *Advances in Neural Information Processing Systems*, *34*, 16437–16450.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... Amodei, D. (2020). Language models are few-shot learners. In *Proceedings of the 34th international conference on neural information processing systems.* Red Hook, NY, USA: Curran Associates Inc.

Burkholz, R., & Dubatovka, A. (2019). Initialization of relus for dynamical isometry. *Advances in Neural Information Processing Systems*, *32*.

Caccia, L., Belilovsky, E., Caccia, M., & Pineau, J. (2020). Online learned continual compression with adaptive quantization modules. In *International conference on machine learning* (pp. 1240–1250).

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., & Joulin, A. (2020). Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, *33*, 9912–9924.

Carpenter, F., Manson, D., Jeffery, K., Burgess, N., & Barry, C. (2015). Grid cells form a global representation of connected environments. *Current Biology*, *25*(9), 1176–1182.

Carr, M. F., Jadhav, S. P., & Frank, L. M. (2011). Hippocampal replay in the awake state: A potential substrate for memory consolidation and retrieval. *Nature neuroscience*, *14*(2), 147–153.

Chaisangmongkon, W., Swaminathan, S. K., Freedman, D. J., & Wang, X.-J. (2017). Computing by robust transience: How the fronto-parietal network performs sequential, category-based decisions. *Neuron*, *93*(6), 1504–1517.

Chapman, I. F., Raymond, M. A., & Fletcher, M. L. (2024). Experience and behavior modulate piriform cortex odor representation in freely moving mice. *bioRxiv*, 2024–02.

Chaudhari, P., & Soatto, S. (2018). Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In *2018 information theory and applications workshop (ita)* (pp. 1–10).

Chaudhry, A., Khan, N., Dokania, P., & Torr, P. (2020). Continual learning in low-rank orthogonal subspaces. *Advances in Neural Information Processing Systems*, *33*, 9900–9911.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P., Torr, P., & Ranzato, M. (2019). Continual learning with tiny episodic memories. In *Workshop on multi-task and lifelong reinforcement learning.*

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning* (pp. 1597–1607).

Chen, X., & He, K. (2021). Exploring simple siamese representation learning. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 15750–15758).

Chen, Z. S., & Wilson, M. A. (2023). How our understanding of memory replay evolves. *Journal of Neurophysiology*, *129*(3), 552–580.

Chizat, L., Oyallon, E., & Bach, F. (2019). On lazy training in differentiable programming. *Advances in neural information processing systems*, *32*.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., & Yoshua, B. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *http://arxiv.org/abs/1406.1078*.

Clevert, D.-A. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

Cook, S. J., Jarrell, T. A., Brittin, C. A., Wang, Y., Bloniarz, A. E., Yakovlev, M. A., ... Emmons, S. W. (2019). Whole-animal connectomes of both caenorhabditis elegans sexes. *Nature*, *571*(7763), 63–71.

Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., & Bengio, Y. (2016). Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv preprint arXiv:1602.02830*.

Crick, F. (1989). The recent excitement about neural networks. *Nature*, *337*(6203), 129–132.

Deitch, D., Rubin, A., & Ziv, Y. (2021). Representational drift in the mouse visual cortex. *Current biology*, *31*(19), 4327–4339.

Dekker, R. B., Otto, F., & Summerfield, C. (2022). Curriculum learning for human compositional generalization. *Proceedings of the National Academy of Sciences*, *119*(41), e2205582119.

De Sa, V. (1993). Learning classification with unlabeled data. *Advances in neural information processing systems*, *6*.

Devalle, F., Zou, L., Cecchini, G., & Roxin, A. (2024). Representational drift as the consequence of ongoing memory storage. *bioRxiv*, 2024–06.

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dhawale, A. K., Poddar, R., Wolff, S. B., Normand, V. A., Kopelowitz, E., & Ölveczky, B. P. (2017). Automated long-term recording and analysis of neural activity in behaving animals. *Elife*, *6*, e27702.

DiCarlo, J. J., Zoccolan, D., & Rust, N. C. (2012). How does the brain solve visual object recognition? *Neuron*, *73*(3), 415–434.

Dinh, L., Pascanu, R., Bengio, S., & Bengio, Y. (2017). Sharp minima can generalize for deep nets. In *International conference on machine learning* (pp. 1019–1028).

Doerig, A., Sommers, R. P., Seeliger, K., Richards, B., Ismael, J., Lindsay, G. W., ... Kietzmann, T. C. (2023). The neuroconnectionist research programme. *Nature Reviews Neuroscience*, *24*(7), 431–450.

Dohare, S., Hernandez-Garcia, J. F., Lan, Q., Rahman, P., Mahmood, A. R., & Sutton, R. S. (2024). Loss of plasticity in deep continual learning. *Nature*, *632*(8026), 768–774.

Dominé, C. C., Anguita, N., Proca, A. M., Braun, L., Kunin, D., Mediano, P. A., & Saxe, A. M. (2024). From lazy to rich: Exact learning dynamics in deep linear networks. *arXiv preprint arXiv:2409.14623*.

Driscoll, L. N., Duncker, L., & Harvey, C. D. (2022). Representational drift: Emerging theories for continual learning and experimental future directions. *Current Opinion in Neurobiology*, *76*, 102609.

Driscoll, L. N., Pettit, N. L., Minderer, M., Chettih, S. N., & Harvey, C. D. (2017). Dynamic reorganization of neuronal activity patterns in parietal cortex. *Cell*, *170*(5), 986–999.

Du, S. S., Lee, J., Li, H., Wang, L., & Zhai, X. (2019). Gradient descent finds global minima of deep neural networks. In *International conference on machine learning* (pp. 1675–1685).

Du, S. S., Zhai, X., Poczos, B., & Singh, A. (2018). Gradient descent provably optimizes over-parameterized neural networks. In *International conference on learning representations.*

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, *12*(7).

Duncker, L., Driscoll, L., Shenoy, K. V., Sahani, M., & Sussillo, D. (2020). Organizing recurrent network dynamics by task-computation to enable continual learning. *Advances in neural information processing systems*, *33*, 14387–14397.

D'amato, M., & Colombo, M. (1990). The symbolic distance effect in monkeys (cebus apella). *Animal Learning & Behavior*, *18*(2), 133–140.

Eckart, C., & Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, *1*(3), 211–218.

Ego-Stengel, V., & Wilson, M. A. (2010). Disruption of ripple-associated hippocampal activity during rest impairs spatial learning in the rat. *Hippocampus*, *20*(1), 1–10.

Eickenberg, M., Gramfort, A., Varoquaux, G., & Thirion, B. (2017). Seeing it all: Convolutional network layers map the function of the human visual system. *NeuroImage*, *152*, 184–194.

Erhan, D., Bengio, Y., Courville, A., & Vincent, P. (2009). Visualizing higher-layer features of a deep network. *University of Montreal*, *1341*(3), 1.

Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., ... Robin, R. (2024). Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning.*

Farajtabar, M., Azizan, N., Mott, A., & Li, A. (2020). Orthogonal gradient descent for continual learning. In *International conference on artificial intelligence and statistics* (pp. 3762–3773).

Farquhar, S., & Gal, Y. (2018). Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733*.

Farrell, M., Recanatesi, S., & Shea-Brown, E. (2023). From lazy to rich to exclusive task representations in neural networks and neural codes. *Current opinion in neurobiology*, *83*, 102780.

Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning* (pp. 1126–1135).

Flesch, T., Balaguer, J., Dekker, R., Nili, H., & Summerfield, C. (2018). Comparing continual task learning in minds and machines. *Proceedings of the National Academy of Sciences*, *115*(44), E10313–E10322.

Flesch, T., Juechems, K., Dumbalska, T., Saxe, A. M., & Summerfield, C. (2022). Orthogonal representations for robust context-dependent task performance in brains and neural networks. *Neuron*, *110*(7), 1258–1270.

French, R. M. (1991). Using semi-distributed representations to overcome catastrophic forgetting in connectionist networks. In *Proceedings of the 13th annual cognitive science society conference* (Vol. 1, pp. 173–178).

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, *3*(4), 128–135.

Friston, K. (2005). A theory of cortical responses. *Philosophical transactions of the Royal Society B: Biological sciences*, *360*(1456), 815–836.

Friston, K. (2010). The free-energy principle: A unified brain theory? *Nature reviews neuroscience*, *11*(2), 127–138.

Fukumizu, K. (1998). Effect of batch learning in multilayer neural networks. In *International conference on neural information processing* (p. 67-70).

Fukumizu, K., & Amari, S.-i. (2000). Local minima and plateaus in hierarchical structures of multilayer perceptrons. *Neural networks*, *13*(3), 317–327.

Fukushima, K. (1969). Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, *5*(4), 322–333.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, *36*(4), 193–202.

Gallego, J. A., Perich, M. G., Chowdhury, R. H., Solla, S. A., & Miller, L. E. (2020). Long-term stability of cortical population dynamics underlying consistent behavior. *Nature neuroscience*, *23*(2), 260–270.

Gallego, J. A., Perich, M. G., Miller, L. E., & Solla, S. A. (2017). Neural manifolds for the control of movement. *Neuron*, *94*(5), 978–984.

Gao, P., Trautmann, E., Yu, B., Santhanam, G., Ryu, S., Shenoy, K., & Ganguli, S. (2017). A theory of multineuronal dimensionality, dynamics and measurement. *BioRxiv*, 214262.

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., & Brendel, W. (2018). Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *International conference on learning representations.*

Gerace, F., Saglietti, L., Mannelli, S. S., Saxe, A. M., & Zdeborová, L. (2022). Probing transfer learning with a model of synthetic correlated datasets. *Machine Learning: Science and Technology*, *3*(1), 015030.

Geva, N., Deitch, D., Rubin, A., & Ziv, Y. (2023). Time and experience differentially affect distinct aspects of hippocampal representational drift. *Neuron*, *111*(15), 2357–2366.

Girardeau, G., Benchenane, K., Wiener, S. I., Buzsáki, G., & Zugaro, M. B. (2009). Selective suppression of hippocampal ripples impairs spatial memory. *Nature neuroscience*, *12*(10), 1222–1223.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249–256).

Goldt, S., Advani, M., Saxe, A. M., Krzakala, F., & Zdeborová, L. (2019). Dynamics of stochastic gradient descent for two-layer neural networks in the teacher-student setup. *Advances in neural information processing systems*, *32*.

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., & Bengio, Y. (2013). An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, *27*.

Graf, A. B., Kohn, A., Jazayeri, M., & Movshon, J. A. (2011). Decoding the activity of neuronal populations in macaque primary visual cortex. *Nature neuroscience*, *14*(2), 239–245.

Greville, T. N. E. (1966). Note on the generalized inverse of a matrix product. *Siam Review*, *8*(4), 518–521.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., ... Valko, M. (2020). Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, *33*, 21271–21284.

Grossberg, S. (1987). Competitive learning: From interactive activation to adaptive resonance. *Cognitive science*, *11*(1), 23–63.

Gunasekar, S., Lee, J. D., Soudry, D., & Srebro, N. (2018). Implicit bias of gradient descent on linear convolutional networks. *Advances in neural information processing systems*, *31*.

Hadsell, R., Rao, D., Rusu, A. A., & Pascanu, R. (2020). Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences*, *24*(12), 1028–1040.

Haxby, J. V., Connolly, A. C., & Guntupalli, J. S. (2014). Decoding neural representational spaces using multivariate pattern analysis. *Annual review of neuroscience*, *37*(1), 435–456.

Hayek, F. A. (1952). *The sensory order: An inquiry into the foundations of theoretical psychology*. University of Chicago Press.

He, F., Liu, T., & Tao, D. (2019). Control batch size and learning rate to generalize well: Theoretical and empirical evidence. *Advances in neural information processing systems*, *32*.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the ieee international conference on computer vision* (pp. 1026–1034).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

Hebb, D. O. (1949). *The organization of behavior. a neuropsychological theory.* John Wiley.

Hendrycks, D., & Gimpel, K. (2016). Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Hinton, G. E. (1987). Learning translation invariant recognition in a massively parallel networks. In *International conference on parallel architectures and languages europe* (pp. 1–13).

Hinton, G. E., & Zemel, R. (1993). Autoencoders, minimum description length and helmholtz free energy. *Advances in neural information processing systems*, *6*.

Horst, J. S., & Hout, M. C. (2016). The novel object and unusual name (noun) database: A collection of novel images for use in experimental research. *Behavior research methods*, *48*, 1393–1409.

Hosoya, H. (2024). A cognitive model for learning abstract relational structures from memory-based decision-making tasks. In *The twelfth international conference on learning representations.*

Huang, X. S., Perez, F., Ba, J., & Volkovs, M. (2020). Improving transformer optimization through better initialization. In *International conference on machine learning* (pp. 4475–4483).

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, *160*(1), 106.

Huszár, F. (2017). On quadratic penalties in elastic weight consolidation. *arXiv preprint arXiv:1712.03847*.

Hwa Lee, J., Sarao Mannelli, S., & Saxe, A. M. (2024). Why do animals need shaping? a theory of task composition and curriculum learning. In *Forty-first international conference on machine learning.*

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd international conference on international conference on machine learning - volume 37* (p. 448–456). JMLR.org.

Isele, D., & Cosgun, A. (2018). Selective experience replay for lifelong learning. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 32).

Jacot, A., Gabriel, F., & Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, *31*.

Javed, K., & White, M. (2019). Meta-learning representations for continual learning. *Advances in neural information processing systems*, *32*.

Jazayeri, M., & Ostojic, S. (2021). Interpreting neural computations by examining intrinsic and embedding dimensionality of neural activity. *Current opinion in neurobiology*, *70*, 113–120.

Ji, Z., & Telgarsky, M. (2018). Gradient descent aligns the layers of deep linear networks. In *International conference on learning representations.*

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., ... Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Karpathy, A., & Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 3128–3137).

Katlowitz, K. A., Picardo, M. A., & Long, M. A. (2018). Stable sequential activity underlying the maintenance of a precisely executed skilled behavior. *Neuron*, *98*(6), 1133–1140.

Kaufman, M. T., Churchland, M. M., Ryu, S. I., & Shenoy, K. V. (2014). Cortical activity in the null space: Permitting preparation without movement. *Nature neuroscience*, *17*(3), 440–448.

Khaligh-Razavi, S.-M., & Kriegeskorte, N. (2014). Deep supervised, but not unsupervised, models may explain it cortical representation. *PLoS computational biology*, *10*(11), e1003915.

Kiani, R., Esteky, H., Mirpour, K., & Tanaka, K. (2007). Object category structure in response patterns of neuronal population in monkey inferior temporal cortex. *Journal of neurophysiology*, *97*(6), 4296–4309.

Kingma, D. P. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114v10*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., ... Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, *114*(13), 3521–3526.

Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-normalizing neural networks. *Advances in neural information processing systems*, *30*.

Knudsen, E. B., & Wallis, J. D. (2021). Hippocampal neurons construct a map of an abstract value space. *Cell*, *184*(18), 4640–4650.

Koay, S. A., Charles, A. S., Thiberge, S. Y., Brody, C. D., & Tank, D. W. (2022). Sequential and efficient neural-population coding of complex task information. *Neuron*, *110*(2), 328–349.

Kortge, C. A. (1989). Episodic memory in connectionist networks. In *Proceedings of the 12th annual cognitive science society conference* (pp. 764–771).

Kriegeskorte, N., Mur, M., & Bandettini, P. A. (2008). Representational similarity analysis - connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, *2*, 249.

Kriegeskorte, N., Mur, M., Ruff, D. A., Kiani, R., Bodurka, J., Esteky, H., ... Bandettini, P. A. (2008). Matching categorical object representations in inferior temporal cortex of man and monkey. *Neuron*, *60*(6), 1126–1141.

Kriegeskorte, N., & Wei, X.-X. (2021). Neural tuning and representational geometry. *Nature Reviews Neuroscience*, *22*(11), 703–718.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*.

Krueger, K. A., & Dayan, P. (2009). Flexible shaping: How learning in small steps helps. *Cognition*, *110*(3), 380–394.

Kudithipudi, D., Aguilar-Simon, M., Babb, J., Bazhenov, M., Blackiston, D., Bongard, J., ... Siegelmann, H. (2022). Biological underpinnings for lifelong learning machines. *Nature Machine Intelligence*, *4*(3), 196–210.

Kudryavitskaya, E., Marom, E., Shani-Narkiss, H., Pash, D., & Mizrahi, A. (2021). Flexible categorization in the mouse olfactory bulb. *Current Biology*, *31*(8), 1616–1631.

Kumaran, D., Hassabis, D., & McClelland, J. L. (2016). What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, *20*(7), 512–534.

Lampinen, A. K., & Ganguli, S. (2018). An analytic theory of generalization dynamics and transfer learning in deep linear networks. *arXiv preprint arXiv:1809.10374*.

Langdon, C., Genkin, M., & Engel, T. A. (2023). A unifying perspective on neural manifolds and circuits for cognition. *Nature Reviews Neuroscience*, *24*(6), 363–377.

Launay, J., Poli, I., Boniface, F., & Krzakala, F. (2020). Direct feedback alignment scales to modern deep learning tasks and architectures. *Advances in neural information processing systems*, *33*, 9346–9360.

Laurent, T., & Brecht, J. (2018). Deep linear networks with arbitrary loss: All local minima are global. In *International conference on machine learning* (pp. 2902–2907).

Le Cun, Y. (1986). Learning process in an asymmetric threshold network. In *Disordered systems and biological organization* (pp. 233–240). Springer.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), 541–551.

LeCun, Y., Bottou, L., Orr, G. B., & Müller, K.-R. (2002). Efficient backprop. In *Neural networks: Tricks of the trade* (pp. 9–50). Springer.

Lee, D.-H., Zhang, S., Fischer, A., & Bengio, Y. (2015). Difference target propagation. In *Machine learning and knowledge discovery in databases: European conference, ecml pkdd 2015, porto, portugal, september 7-11, 2015, proceedings, part i 15* (pp. 498–515).

Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., & Pennington, J. (2019). Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, *32*.

Lee, S., Goldt, S., & Saxe, A. M. (2021). Continual learning in the teacher-student setup: Impact of task similarity. In *International conference on machine learning* (pp. 6109–6119).

Lewandowsky, S. (1994). On the relation between catastrophic interference and generalization in connectionist networks. *Journal of Biological Systems*, *2*(03), 307–333.

Li, Q., Sorscher, B., & Sompolinsky, H. (2024). Representations and generalization in artificial and brain neural networks. *Proceedings of the National Academy of Sciences*, *121*(27), e2311805121.

Lillicrap, T. P., Cownden, D., Tweed, D. B., & Akerman, C. J. (2016). Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, *7*(1), 13276.

Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J., & Hinton, G. (2020). Backpropagation and the brain. *Nature Reviews Neuroscience*, *21*(6), 335–346.

Lindsay, G. W. (2021). Convolutional neural networks as a model of the visual system: Past, present, and future. *Journal of cognitive neuroscience*, *33*(10), 2017–2031.

Linnainmaa, S. (1970). *The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors* (Unpublished master's thesis). University of Helsinki. (Master's Thesis (in Finnish))

Lippl, S., Kay, K., Jensen, G., Ferrera, V. P., & Abbott, L. (2024). A mathematical theory of relational generalization in transitive inference. *Proceedings of the National Academy of Sciences*, *121*(28), e2314511121.

Liu, X., Masana, M., Herranz, L., Van de Weijer, J., Lopez, A. M., & Bagdanov, A. D. (2018). Rotate your networks: Better weight consolidation and less catastrophic forgetting. In *2018 24th international conference on pattern recognition (icpr)* (pp. 2262–2268).

Liu, Y., Dolan, R. J., Kurth-Nelson, Z., & Behrens, T. E. (2019). Human replay spontaneously reorganizes experience. *Cell*, *178*(3), 640–652.

Logothetis, N. K. (2003). The underpinnings of the bold functional magnetic resonance imaging signal. *Journal of Neuroscience*, *23*(10), 3963–3971.

Lu, L., Shin, Y., Su, Y., & Karniadakis, G. E. (2019). Dying relu and initialization: Theory and numerical examples. *arXiv preprint arXiv:1903.06733*.

Luyckx, F., Nili, H., Spitzer, B., & Summerfield, C. (2019). Neural structure mapping in human probabilistic reward learning. *elife*, *8*, e42816.

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml* (Vol. 30, p. 3).

Mante, V., Sussillo, D., Shenoy, K. V., & Newsome, W. T. (2013). Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, *503*(7474), 78–84.

Marks, T. D., & Goard, M. J. (2021). Stimulus-dependent representational drift in primary visual cortex. *Nature communications*, *12*(1), 5169.

Masset, P., Qin, S., & Zavatone-Veth, J. A. (2022). Drifting neuronal representations: Bug or feature? *Biological cybernetics*, *116*(3), 253–266.

McClelland, J. L., McNaughton, B. L., & O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, *102*(3), 419.

McCloskey, M., & Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation* (Vol. 24, pp. 109–165). Elsevier.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*, 115–133.

McIntosh, L., Maheswaranathan, N., Nayebi, A., Ganguli, S., & Baccus, S. (2016). Deep learning models of the retinal response to natural scenes. *Advances in neural information processing systems*, *29*.

McMahon, D. B., Jones, A. P., Bondar, I. V., & Leopold, D. A. (2014). Face-selective neurons maintain consistent visual responses across months. *Proceedings of the National Academy of Sciences*, *111*(22), 8251–8256.

Meulemans, A., Carzaniga, F., Suykens, J., Sacramento, J., & Grewe, B. F. (2020). A theoretical framework for target propagation. *Advances in Neural Information Processing Systems*, *33*, 20024–20036.

Meyer, C. D., Jr. (1973). Generalized inversion of modified matrices. *Siam journal on applied mathematics*, *24*(3), 315–323.

Micou, C., & O'Leary, T. (2023). Representational drift as a window into neural and behavioural plasticity. *Current opinion in neurobiology*, *81*, 102746.

Min, H., Tarmoun, S., Vidal, R., & Mallada, E. (2021). On the explicit role of initialization on the convergence and implicit bias of overparametrized linear networks. In *International conference on machine learning* (pp. 7760–7768).

Minsky, M., & Papert, S. (1969). *Perceptrons: An introduction to computational geometry*. MIT Press.

Mirzadeh, S. I., Farajtabar, M., Pascanu, R., & Ghasemzadeh, H. (2020). Understanding the role of training regimes in continual learning. *Advances in Neural Information Processing Systems*, *33*, 7308–7320.

Mishkin, D., & Matas, J. (2015). All you need is a good init. *arXiv preprint arXiv:1511.06422*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... Hassabis, D. (2015). Human-level control through deep reinforcement learning. *nature*, *518*(7540), 529–533.

Muhle-Karbe, P. S., Sheahan, H., Pezzulo, G., Spiers, H. J., Chien, S., Schuck, N. W., & Summerfield, C. (2023). Goal-seeking compresses neural codes for space in the human hippocampus and orbitofrontal cortex. *Neuron*, *111*(23), 3885–3899.

Musall, S., Kaufman, M. T., Juavinett, A. L., Gluf, S., & Churchland, A. K. (2019). Single-trial neural dynamics are dominated by richly varied movements. *Nature neuroscience*, *22*(10), 1677–1686.

Muysers, H., Chen, H.-L., Hahn, J., Folschweiller, S., Sigurdsson, T., Sauer, J.-F., & Bartos, M. (2024). A persistent prefrontal reference frame across time and task rules. *Nature communications*, *15*(1), 2115.

Nelli, S., Braun, L., Dumbalska, T., Saxe, A. M., & Summerfield, C. (2023). Neural knowledge assembly in humans and neural networks. *Neuron*, *111*(9), 1504–1516.

Nickolls, J., Buck, I., Garland, M., & Skadron, K. (2008). Scalable parallel programming with cuda: Is cuda the parallel programming model that application developers have been waiting for? *Queue*, *6*(2), 40–53.

Nøkland, A. (2016). Direct feedback alignment provides learning in deep neural networks. *Advances in neural information processing systems*, *29*.

Novikoff, A. B. (1962). On convergence proofs on perceptrons. In *Proceedings of the symposium on the mathematical theory of automata* (Vol. 12, pp. 615–622).

Okazawa, G., Hatch, C. E., Mancoo, A., Machens, C. K., & Kiani, R. (2021). Representational geometry of perceptual decisions in the monkey parietal cortex. *Cell*, *184*(14), 3748–3761.

O'Keefe, J., & Dostrovsky, J. (1971). The hippocampus as a spatial map: preliminary evidence from unit activity in the freely-moving rat. *Brain research*.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., & Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural networks*, *113*, 54–71.

Pashakhanloo, F., & Koulakov, A. (2023). Stochastic gradient descent-induced drift of representation in a two-layer neural network. In *International conference on machine learning* (pp. 27401–27419).

Pennington, J., Schoenholz, S., & Ganguli, S. (2017). Resurrecting the sigmoid in deep learning through dynamical isometry: Theory and practice. *Advances in neural information processing systems*, *30*.

Perkins, D. N., & Salomon, G. (1992). Transfer of learning. *International encyclopedia of education*, *2*, 6452–6457.

Prabhu, A., Sinha, S., Kumaraguru, P., Torr, P. H., Sener, O., & Dokania, P. K. (2024). Randumb: A simple approach that questions the efficacy of continual representation learning. *arXiv preprint arXiv:2402.08823*.

Prabhu, A., Torr, P. H., & Dokania, P. K. (2020). Gdumb: A simple approach that questions our progress in continual learning. In *Computer vision–eccv 2020: 16th european conference, glasgow, uk, august 23–28, 2020, proceedings, part ii 16* (pp. 524–540).

Qian, W., Zavatone-Veth, J. A., Ruben, B. S., & Pehlevan, C. (2024). Partial observation can induce mechanistic mismatches in data-constrained models of neural dynamics. *bioRxiv*, 2024–05.

Qin, S., Farashahi, S., Lipshutz, D., Sengupta, A. M., Chklovskii, D. B., & Pehlevan, C. (2023). Coordinated drift of receptive fields in hebbian/anti-hebbian network models during noisy representation learning. *Nature Neuroscience*, *26*(2), 339–349.

Ratcliff, R. (1990). Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychological review*, *97*(2), 285.

Ratzon, A., Derdikman, D., & Barak, O. (2024). Representational drift as a result of implicit regularization. *Elife*, *12*, RP90069.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., & Lampert, C. H. (2017). icarl: Incremental classifier and representation learning. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2001–2010).

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., & Lee, H. (2016). Generative adversarial text to image synthesis. In *International conference on machine learning* (pp. 1060–1069).

Refinetti, M., d'Ascoli, S., Ohana, R., & Goldt, S. (2021). Align, then memorise: the dynamics of learning with feedback alignment. In *International conference on machine learning* (pp. 8925–8935).

Richards, B. A., Lillicrap, T. P., Beaudoin, P., Bengio, Y., Bogacz, R., Christensen, A., ... Kording, K. P. (2019). A deep learning framework for neuroscience. *Nature neuroscience*, *22*(11), 1761–1770.

Robins, A. (1995). Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, *7*(2), 123–146.

Rokni, U., Richardson, A. G., Bizzi, E., & Seung, H. S. (2007). Motor learning with unstable neural representations. *Neuron*, *54*(4), 653–666.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386.

Roth, Z. N., & Merriam, E. P. (2023). Representations in human primary visual cortex drift over time. *Nature Communications*, *14*(1), 4422.

Roy, J. E., Riesenhuber, M., Poggio, T., & Miller, E. K. (2010). Prefrontal cortex activity during flexible categorization. *Journal of Neuroscience*, *30*(25), 8519–8528.

Rule, M. E., Loback, A. R., Raman, D. V., Driscoll, L. N., Harvey, C. D., & O'Leary, T. (2020). Stable task information from an unstable neural population. *elife*, *9*, e51121.

Rule, M. E., O'Leary, T., & Harvey, C. D. (2019). Causes and consequences of representational drift. *Current opinion in neurobiology*, *58*, 141–147.

Rumelhart, D. E., Hinton, G. E., & McClelland, J. L. (1986). A general framework for parallel distributed processing. *Parallel distributed processing: Explorations in the microstructure of cognition*, *1*(45-76), 26.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533–536.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*, 211–252.

Saad, D., & Solla, S. A. (1995). Exact solution for online learning in multilayer neural networks. *Physical Review Letters*, *74*(21), 4337.

Sacramento, J., Ponte Costa, R., Bengio, Y., & Senn, W. (2018). Dendritic cortical microcircuits approximate the backpropagation algorithm. *Advances in neural information processing systems*, *31*.

Sadeh, S., & Clopath, C. (2022). Contribution of behavioural variability to representational drift. *Elife*, *11*, e77907.

Saglietti, L., Mannelli, S., & Saxe, A. M. (2022). An analytical theory of curriculum learning in teacher-student networks. *Advances in Neural Information Processing Systems*, *35*, 21113–21127.

Saha, G., Garg, I., & Roy, K. (2021). Gradient projection memory for continual learning. In *International conference on learning representations.*

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., ... Norouzi, M. (2022). Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, *35*, 36479–36494.

Saxe, A. M., McClelland, J. L., & Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

Saxe, A. M., McClelland, J. L., & Ganguli, S. (2019). A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences*, *116*(23), 11537–11546.

Saxe, A. M., Nelli, S., & Summerfield, C. (2021). If deep learning is the answer, what is the question? *Nature Reviews Neuroscience*, *22*(1), 55–67.

Saxe, A. M., Sodhani, S., & Lewallen, S. J. (2022). The neural race reduction: Dynamics of abstraction in gated networks. In *International conference on machine learning* (pp. 19287–19309).

Scellier, B., & Bengio, Y. (2017). Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, *11*, 24.

Scheffer, L. K., Xu, C. S., Januszewski, M., Lu, Z., Takemura, S.-y., Hayworth, K. J., ... Plaza, S. M. (2020). A connectome and analysis of the adult *Drosophila* central brain. *eLife*, *9*, e57443.

Schmidhuber, J. (1987). *Evolutionary principles in self-referential learning* (Unpublished doctoral dissertation). Technische Universität München.

Schmidhuber, J. (2014). *Who invented backpropagation?* Retrieved 2024-08-02, from `https://people.idsia.ch/~juergen/who-invented-backpropagation.html`

Schmidhuber, J., & Hochreiter, S. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Schoonover, C. E., Ohashi, S. N., Axel, R., & Fink, A. J. (2021). Representational drift in primary olfactory cortex. *Nature*, *594*(7864), 541–546.

Schrimpf, M., Kubilius, J., Hong, H., Majaj, N. J., Rajalingham, R., Issa, E. B., ... DiCarlo, J. J. (2018). Brain-score: Which artificial neural network for object recognition is most brain-like? *BioRxiv*, 407007.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Scoville, W. B., & Milner, B. (1957). Loss of recent memory after bilateral hippocampal lesions. *Journal of neurology, neurosurgery, and psychiatry*, *20*(1), 11.

Sejnowski, T. J., Churchland, P. S., & Movshon, J. A. (2014). Putting big data to good use in neuroscience. *Nature neuroscience*, *17*(11), 1440–1441.

Sevilla, J., Heim, L., Ho, A., Besiroglu, T., Hobbhahn, M., & Villalobos, P. (2022). Compute trends across three eras of machine learning. In *2022 international joint conference on neural networks (ijcnn)* (pp. 1–8).

Shan, H., Li, Q., & Sompolinsky, H. (2024). Order parameters and phase transitions of continual learning in deep neural networks. *arXiv preprint arXiv:2407.10315*.

Sherman, J., & Morrison, W. J. (1950). Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, *21*(1), 124–127.

Shin, H., Lee, J. K., Kim, J., & Kim, J. (2017). Continual learning with deep generative replay. *Advances in neural information processing systems*, *30*.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, *529*(7587), 484–489.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., ... Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, *362*(6419), 1140–1144.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... Hassabis, D. (2017). Mastering the game of go without human knowledge. *nature*, *550*(7676), 354–359.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Simsek, B., Ged, F., Jacot, A., Spadaro, F., Hongler, C., Gerstner, W., & Brea, J. (2021). Geometry of the loss landscape in overparameterized neural networks: Symmetries and invariances. In *International conference on machine learning* (pp. 9722–9732).

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., & Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning* (pp. 2256–2265).

Song, Y., Millidge, B., Salvatori, T., Lukasiewicz, T., Xu, Z., & Bogacz, R. (2024). Inferring neural activity before plasticity as a foundation for learning beyond backpropagation. *Nature neuroscience*, *27*(2), 348–358.

Soni, A., Srivastava, S., Khosla, M., & Kording, K. P. (2024). Conclusions about neural network to brain alignment are profoundly impacted by the similarity measure. *bioRxiv*, 2024–08.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Srivastava, R. K., Masci, J., Kazerounian, S., Gomez, F., & Schmidhuber, J. (2013). Compete to compute. *Advances in neural information processing systems*, *26*.

Strang, G. (2014). *Differential equations and linear algebra*. Wellesley-Cambridge Press Wellesley.

Strubell, E., Ganesh, A., & McCallum, A. (2020). Energy and policy considerations for modern deep learning research. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 34, pp. 13693–13696).

Sun, W., Advani, M., Spruston, N., Saxe, A. M., & Fitzgerald, J. E. (2023). Organizing memories for generalization in complementary learning systems. *Nature neuroscience*, *26*(8), 1438–1448.

Sussmann, H. J. (1992). Uniqueness of the weights for minimal feedforward nets with a given input-output map. *Neural networks*, *5*(4), 589–593.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (Vol. 27).

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 1–9).

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Tarmoun, S., Franca, G., Haeffele, B. D., & Vidal, R. (2021). Understanding the dynamics of gradient flow in overparameterized linear models. In *International conference on machine learning* (pp. 10153–10161).

Thagard, P. (1986). Parallel computation and the mind-body problem. *Cognitive Science*, *10*(3), 301–318.

Tolman, E. C. (1948). Cognitive maps in rats and men. *Psychological review*, *55*(4), 189.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., ... Lample, G. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Tu, Z., Aranguri, S., & Jacot, A. (2024). Mixed dynamics in linear networks: Unifying the lazy and active regimes. *arXiv preprint arXiv:2405.17580*.

Turing, A. (1950). Computing machinery and intelligence. *Mind*, *236*, 433–460.

Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., ... Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, *12*.

Van de Ven, G. M., Siegelmann, H. T., & Tolias, A. S. (2020). Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, *11*(1), 4069.

Varre, A. V., Sagitova, M., & Flammarion, N. (2024). Sgd vs gd: Rank deficiency in linear networks. In *High-dimensional learning dynamics 2024: The emergence of structure and reasoning.*

Varre, A. V., Vladarean, M.-L., Pillaud-Vivien, L., & Flammarion, N. (2024). On the spectral bias of two-layer linear networks. *Advances in Neural Information Processing Systems*, *36*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (Vol. 30).

Veerabadran, V., Goldman, J., Shankar, S., Cheung, B., Papernot, N., Kurakin, A., ... Elsayed, G. F. (2023). Subtle adversarial image manipulations influence both human and machine perception. *Nature Communications*, *14*(1), 4933.

Wang, L., Zhang, X., Su, H., & Zhu, J. (2024). A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Wang, L., Zhang, X., Yang, K., Yu, L., Li, C., Lanqing, H., ... Zhu, J. (2022). Memory replay with data compression for continual learning. In *International conference on learning representations.*

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning*, *8*, 279–292.

Weng, J. J., Ahuja, N., & Huang, T. S. (1993). Learning recognition and segmentation of 3-d objects from 2-d images. In *1993 (4th) international conference on computer vision* (pp. 121–128).

Werbos, P. J. (1982). Applications of advances in nonlinear sensitivity analysis. In *System modeling and optimization* (pp. 762–770).

Whittington, J. C., & Bogacz, R. (2017). An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, *29*(5), 1229–1262.

Whittington, J. C., & Bogacz, R. (2019). Theories of error back-propagation in the brain. *Trends in cognitive sciences*, *23*(3), 235–250.

Whittington, J. C., McCaffary, D., Bakermans, J. J., & Behrens, T. E. (2022). How to build a cognitive map. *Nature neuroscience*, *25*(10), 1257–1272.

Whittington, J. C., Muller, T. H., Mark, S., Chen, G., Barry, C., Burgess, N., & Behrens, T. E. (2020). The tolman-eichenbaum machine: unifying space and relational memory through generalization in the hippocampal formation. *Cell*, *183*(5), 1249–1263.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, *8*, 229–256.

Wilson, M. A., & McNaughton, B. L. (1994). Reactivation of hippocampal ensemble memories during sleep. *Science*, *265*(5172), 676–679.

Wittkuhn, L., Chien, S., Hall-McMaster, S., & Schuck, N. W. (2021). Replay in minds and machines. *Neuroscience & Biobehavioral Reviews*, *129*, 367–388.

Woocher, F. D., Glass, A. L., & Holyoak, K. J. (1978). Positional discriminability in linear orderings. *Memory & Cognition*, *6*, 165–173.

Woodworth, B., Gunasekar, S., Lee, J. D., Moroshko, E., Savarese, P., Golan, I., ... Srebro, N. (2020). Kernel and rich regimes in overparametrized models. In *Conference on learning theory* (pp. 3635–3673).

Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., & Fu, Y. (2019). Large scale incremental learning. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 374–382).

Xiao, L., Bahri, Y., Sohl-Dickstein, J., Schoenholz, S., & Pennington, J. (2018). Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. In *International conference on machine learning* (pp. 5393–5402).

Yamins, D. L., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D., & DiCarlo, J. J. (2014). Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the national academy of sciences*, *111*(23), 8619–8624.

Yan, W.-Y., Helmke, U., & Moore, J. (1994). Global analysis of oja's flow for neural networks. *IEEE Transactions on Neural Networks*, *5*(5), 674-683.

Zbontar, J., Jing, L., Misra, I., LeCun, Y., & Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning* (pp. 12310–12320).

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision 2014* (pp. 818–833).

Zeng, G., Chen, Y., Cui, B., & Yu, S. (2019). Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, *1*(8), 364–372.

Zenke, F., & Ganguli, S. (2018). Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, *30*(6), 1514–1541.

Zenke, F., Poole, B., & Ganguli, S. (2017). Continual learning through synaptic intelligence. In *International conference on machine learning* (pp. 3987–3995).

Zenke, F., & Vogels, T. P. (2021). The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural computation*, *33*(4), 899–925.

Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2021). Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, *64*(3), 107–115.

Zhang, H., Dauphin, Y. N., & Ma, T. (2018). Fixup initialization: Residual learning without normalization. In *International conference on learning representations.*

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., ... He, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, *109*(1), 43–76.

Ziv, Y., Burns, L. D., Cocker, E. D., Hamel, E. O., Ghosh, K. K., Kitch, L. J., ... Schnitzer, M. J. (2013). Long-term dynamics of ca1 hippocampal place codes. *Nature neuroscience*, *16*(3), 264–266.

Appendix to Chapter 3

A.1 General linear solution

Theorem A.1. *Under the assumptions of a convex and differentiable loss function (Assumption 1) and no bottlenecks (Asuumption 2), a two-layer linear network (Equation 3.2) has converged to a global optimum if and only if*

$$\mathbf{W}_2\mathbf{W}_1\mathbf{\Sigma}_{xx} = \mathbf{\Sigma}_{yx}. \tag{A.1}$$

Proof of Theorem A.1. From Theorem 3 in Laurent & Brecht (2018) it follows that any minimum of the mean-squared error of the two-layer linear neural network

$$\mathcal{L}_{\text{MSE}} = \frac{1}{2P}\sum_{n=1}^{P} ||\mathbf{W}_2\mathbf{W}_1\mathbf{x}_n - \mathbf{y}||_2^2 \tag{A.2}$$

corresponds to a global optimum of the convex single-layer optimisation problem

$$\mathcal{L}_{\text{MSE}} = \frac{1}{2P}\sum_{n=1}^{P} ||\bar{\mathbf{W}}\mathbf{x}_n - \mathbf{y}||_2^2, \tag{A.3}$$

where

$$\bar{\mathbf{W}} = \mathbf{W}_2\mathbf{W}_1. \tag{A.4}$$

Therefore, all optima of the two-layer linear network problem are global and cor-

respond to

$$\begin{aligned} \frac{\partial L_{\text{MSE}}}{\partial \bar{\mathbf{W}}} &\stackrel{!}{=} 0 \\ \Leftrightarrow \frac{1}{P}\sum_{n=1}^{P}\left(\bar{\mathbf{W}}\mathbf{x}_n - \mathbf{y}_n\right)\mathbf{x}_n^T &= 0 \\ \Leftrightarrow \bar{\mathbf{W}}\frac{1}{P}\sum_{n=1}^{P}\mathbf{x}_n\mathbf{x}_n^T - \frac{1}{P}\sum_{n=1}^{P}\mathbf{y}_n\mathbf{x}_n^T &= 0 \\ \Leftrightarrow \bar{\mathbf{W}}\boldsymbol{\Sigma}_{xx} &= \boldsymbol{\Sigma}_{yx}. \end{aligned} \tag{A.5}$$

Resubstitution then gives the general linear solution

$$\mathbf{W}_2\mathbf{W}_1\boldsymbol{\Sigma}_{xx} \stackrel{!}{=} \boldsymbol{\Sigma}_{yx}. \tag{A.6}$$

□

A.2 Least-squares solution

Theorem A.2. *In accordance with Theorem A.1, let* $\boldsymbol{\Omega}_1$ *and* $\boldsymbol{\Omega}_2$ *denote a pair of network weights that implement a general linear solution. Then, pairs of network weights for which*

$$\underset{\boldsymbol{\Omega}_1,\boldsymbol{\Omega}_2}{\operatorname{argmin}} ||\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1||_F^2 \tag{A.7}$$

obey

$$\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1 = \boldsymbol{\Sigma}_{yx}\boldsymbol{\Sigma}_{xx}^{+}. \tag{A.8}$$

Proof of Theorem A.2. We use the method of Lagrangian multipliers to minimise

$$\underset{\mathbf{W}_1,\mathbf{W}_2}{\operatorname{argmin}} ||\mathbf{W}_2\mathbf{W}_1||_F^2 \tag{A.9}$$

such that

$$\mathbf{W}_2\mathbf{W}_1\boldsymbol{\Sigma}_{xx} = \boldsymbol{\Sigma}_{yx}. \tag{A.10}$$

The Lagrangian is then

$$\mathcal{L} = ||\mathbf{W}_2\mathbf{W}_1||_F^2 + \mathrm{Tr}\left(\mathbf{\Lambda}^T\left(\mathbf{W}_2\mathbf{W}_1\mathbf{\Sigma}_{xx} - \mathbf{\Sigma}_{yx}\right)\right) \tag{A.11}$$

with gradients

$$\frac{\partial\mathcal{L}}{\partial\mathbf{W}_1} = 2\mathbf{W}_2^T\mathbf{W}_2\mathbf{W}_1 + \mathbf{W}_2^T\mathbf{\Lambda}\mathbf{\Sigma}_{xx} \stackrel{!}{=} 0, \tag{A.12}$$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{W}_2} = 2\mathbf{W}_2\mathbf{W}_1\mathbf{W}_1^T + \mathbf{\Lambda}\mathbf{\Sigma}_{xx}\mathbf{W}_1^T \stackrel{!}{=} 0, \tag{A.13}$$

and

$$\frac{\partial\mathcal{L}}{\partial\mathbf{\Lambda}} = \mathbf{W}_2\mathbf{W}_1\mathbf{\Sigma}_{xx} - \mathbf{\Sigma}_{yx} \stackrel{!}{=} 0. \tag{A.14}$$

We continue with Equation A.12

$$\begin{aligned} & 2\mathbf{W}_2^T\mathbf{W}_2\mathbf{W}_1 + \mathbf{W}_2^T\mathbf{\Lambda}\mathbf{\Sigma}_{xx} = 0 \\ \Leftrightarrow\ & 2\mathbf{\Sigma}_{xx}\mathbf{W}_1^T\mathbf{W}_2^T\mathbf{W}_2\mathbf{W}_1\mathbf{\Sigma}_{xx} + \mathbf{\Sigma}_{xx}\mathbf{W}_1^T\mathbf{W}_2^T\mathbf{\Lambda}\mathbf{\Sigma}_{xx}\mathbf{\Sigma}_{xx} = 0 \\ & \Leftrightarrow 2\mathbf{\Sigma}_{yx}^T\mathbf{\Sigma}_{yx} = -\mathbf{\Sigma}_{yx}^T\mathbf{\Lambda}\mathbf{\Sigma}_{xx}\mathbf{\Sigma}_{xx} \end{aligned} \tag{A.15}$$

where in the third line we substituted Equation A.14. Next, we substitute the compact SVD($\mathbf{X}$) = $\mathbf{ABC}^T$ and the compact SVD($\mathbf{Y}$) = $\mathbf{DEF}^T$

$$\begin{aligned} 2\mathbf{\Sigma}_{yx}^T\mathbf{\Sigma}_{yx} &= -\mathbf{\Sigma}_{yx}^T\mathbf{\Lambda}\mathbf{\Sigma}_{xx}\mathbf{\Sigma}_{xx} \\ \Leftrightarrow 2\frac{1}{P}\mathbf{X}\mathbf{Y}^T\frac{1}{P}\mathbf{Y}\mathbf{X}^T &= -\frac{1}{P}\mathbf{X}\mathbf{Y}^T\mathbf{\Lambda}\frac{1}{P}\mathbf{X}\mathbf{X}^T\frac{1}{P}\mathbf{X}\mathbf{X}^T \\ \Leftrightarrow 2\mathbf{ABC}^T\mathbf{FE}^2\mathbf{F}^T\mathbf{CBA}^T &= -\frac{1}{P}\mathbf{ABC}^T\mathbf{FED}^T\mathbf{\Lambda}\mathbf{AB}^4\mathbf{A}^T \\ \Leftrightarrow 2\mathbf{C}^T\mathbf{FE}^2\mathbf{F}^T\mathbf{C} &= -\frac{1}{P}\mathbf{C}^T\mathbf{FED}^T\mathbf{\Lambda}\mathbf{AB}^3 \\ \Rightarrow \mathbf{\Lambda} &= -2P\mathbf{DEF}^T\mathbf{CB}^{-3}\mathbf{A}^T \end{aligned} \tag{A.16}$$

Substitution into Equation A.13 then yields

$$\begin{aligned}
2\mathbf{W}_2\mathbf{W}_1\mathbf{W}_1^T + \mathbf{\Lambda}\mathbf{\Sigma}_{xx}\mathbf{W}_1^T &= 0 \\
\left(2\mathbf{W}_2\mathbf{W}_1\mathbf{W}_1^T + \mathbf{\Lambda}\frac{1}{P}\mathbf{X}\mathbf{X}^T\right)\mathbf{W}_1^T &= 0 \\
\left(2\mathbf{W}_2\mathbf{W}_1 - 2P\mathbf{D}\mathbf{E}\mathbf{F}^T\mathbf{C}\mathbf{B}^{-3}\mathbf{A}^T\frac{1}{P}\mathbf{A}\mathbf{B}^2\mathbf{A}^T\right)\mathbf{W}_1^T &= 0 \\
\left(\mathbf{W}_2\mathbf{W}_1 - \mathbf{D}\mathbf{E}\mathbf{F}^T\mathbf{C}\mathbf{B}^{-1}\mathbf{A}^T\right)\mathbf{W}_1^T &= 0 \\
\left(\mathbf{W}_2\mathbf{W}_1 - \mathbf{D}\mathbf{E}\mathbf{F}^T\mathbf{C}\mathbf{B}\mathbf{A}^T\mathbf{A}\mathbf{B}^{-2}\mathbf{A}^T\right)\mathbf{W}_1^T &= 0 \\
\left(\mathbf{W}_2\mathbf{W}_1 - \mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^+\right)\mathbf{W}_1^T &= 0 \\
\left(\mathbf{W}_2\mathbf{W}_1 - \mathbf{\Sigma}_{yx}\mathbf{\Sigma}_{xx}^+\right)\mathbf{W}_1^T &= 0 \\
\Rightarrow \mathbf{W}_2\mathbf{W}_1 &= \mathbf{\Sigma}_{yx}\mathbf{\Sigma}_{xx}^+.
\end{aligned} \tag{A.17}$$

Finally, substituting back into the constraint gives

$$\begin{aligned}
\mathbf{W}_2\mathbf{W}_1\mathbf{\Sigma}_{xx} &= \mathbf{\Sigma}_{yx} \\
\mathbf{\Sigma}_{yx}\mathbf{\Sigma}_{xx}^+\mathbf{\Sigma}_{xx} &= \mathbf{\Sigma}_{yx} \\
\frac{1}{P}\mathbf{D}\mathbf{E}\mathbf{F}^T\mathbf{C}\mathbf{B}\mathbf{A}^T P\mathbf{A}\mathbf{B}^{-2}\mathbf{A}^T\frac{1}{P}\mathbf{A}\mathbf{B}^2\mathbf{A}^T &= \frac{1}{P}\mathbf{D}\mathbf{E}\mathbf{F}^T\mathbf{C}\mathbf{B}\mathbf{A}^T \\
\frac{1}{P}\mathbf{D}\mathbf{E}\mathbf{F}^T\mathbf{C}\mathbf{B}\mathbf{A}^T &= \frac{1}{P}\mathbf{D}\mathbf{E}\mathbf{F}^T\mathbf{C}\mathbf{B}\mathbf{A}^T,
\end{aligned} \tag{A.18}$$

which shows that pairs of $\mathbf{W}_1$ and $\mathbf{W}_2$ for which $\mathbf{W}_2\mathbf{W}_1 = \mathbf{\Sigma}_{yx}\mathbf{\Sigma}_{xx}^+$ are indeed a solution and therefore that the solution obeys

$$\mathbf{\Omega}_2\mathbf{\Omega}_1 = \mathbf{\Sigma}_{yx}\mathbf{\Sigma}_{xx}^+. \tag{A.19}$$

□

A.3 Minimum-norm solution

Theorem A.3. *In accordance with Theorem A.1, let* $\mathbf{\Omega}_1$ *and* $\mathbf{\Omega}_2$ *denote a pair of network weights that implement a general linear solution. Then, pairs of network weights for which*

$$\underset{\mathbf{\Omega}_1,\mathbf{\Omega}_2}{\operatorname{argmin}} ||\mathbf{\Omega}_1||_F^2 + ||\mathbf{\Omega}_2||_F^2 \tag{A.20}$$

obey

$$\mathbf{\Omega}_2 = \mathbf{U}\sqrt{\mathbf{S}}\mathbf{R}^T \text{ and } \mathbf{\Omega}_1 = \mathbf{R}\sqrt{\mathbf{S}}\mathbf{V}^T, \tag{A.21}$$

where $\mathbf{R}$ *is an arbitrary (semi-)orthogonal matrix of appropriate shape.*

Proof of Theorem A.3. We use the method of Lagrangian multipliers to minimise

$$\underset{\mathbf{W}_1,\mathbf{W}_2}{\operatorname{argmin}} ||\mathbf{W}_1||_F^2 + ||\mathbf{W}_2||_F^2 \tag{A.22}$$

under the constraint that

$$\mathbf{W}_2\mathbf{W}_1 = \mathbf{\Sigma}_{yx}\mathbf{\Sigma}_{xx}^+. \tag{A.23}$$

Let

$$\text{SVD}(\mathbf{\Sigma}_{yx}\mathbf{\Sigma}_{xx}^+) = \mathbf{U}\mathbf{S}\mathbf{V}^T \tag{A.24}$$

denote the compact SVD of the constraint. Then the Lagrangian is

$$\mathcal{L} = ||\mathbf{W}_1||_F^2 + ||\mathbf{W}_2||_F^2 + \text{Tr}\left(\mathbf{\Lambda}^T\left(\mathbf{W}_2\mathbf{W}_1 - \mathbf{U}\mathbf{S}\mathbf{V}^T\right)\right) \tag{A.25}$$

with gradients

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\mathbf{W}_1} &= 2\mathbf{W}_1 + \mathbf{W}_2^T\mathbf{\Lambda} \overset{!}{=} 0 \\ &\Leftrightarrow \mathbf{W}_1 = -\frac{1}{2}\mathbf{W}_2^T\mathbf{\Lambda},\end{aligned} \tag{A.26}$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mathbf{W}_2} &= 2\mathbf{W}_2 + \mathbf{\Lambda}\mathbf{W}_1^T \stackrel{!}{=} 0 \\ &\Leftrightarrow \mathbf{W}_2 = -\frac{1}{2}\mathbf{\Lambda}\mathbf{W}_1^T,\end{aligned} \tag{A.27}$$

and

$$\frac{\partial \mathcal{L}}{\partial \mathbf{\Lambda}} = \mathbf{W}_2\mathbf{W}_1 - \mathbf{U}\mathbf{S}\mathbf{V}^T \stackrel{!}{=} 0. \tag{A.28}$$

Starting from Equation A.27 we have

$$\begin{aligned}\mathbf{W}_2 &= -\frac{1}{2}\mathbf{\Lambda}\mathbf{W}_1^T \\ \Leftrightarrow \mathbf{W}_2^T\mathbf{W}_2 &= -\frac{1}{2}\mathbf{W}_2^T\mathbf{\Lambda}\mathbf{W}_1^T \\ \Leftrightarrow \mathbf{W}_2^T\mathbf{W}_2 &= \mathbf{W}_1\mathbf{W}_1^T,\end{aligned} \tag{A.29}$$

where in the last line we substituted Equation A.26. Let

$$\text{SVD}(\mathbf{W}_2) = \mathbf{A}\mathbf{B}\mathbf{C}^T \text{ and } \text{SVD}(\mathbf{W}_1) = \mathbf{D}\mathbf{E}\mathbf{F}^T \tag{A.30}$$

be the compact SVD of the network weights, then

$$\begin{aligned}\mathbf{W}_2^T\mathbf{W}_2 &= \mathbf{W}_1\mathbf{W}_1^T \\ \Leftrightarrow \mathbf{C}\mathbf{B}\mathbf{A}^T\mathbf{A}\mathbf{B}\mathbf{C}^T &= \mathbf{D}\mathbf{E}\mathbf{F}^T\mathbf{F}\mathbf{E}\mathbf{D}^T \\ \Leftrightarrow \mathbf{C}\mathbf{B}^2\mathbf{C}^T &= \mathbf{D}\mathbf{E}^2\mathbf{D}^T,\end{aligned} \tag{A.31}$$

from which it follows that

$$\mathbf{C} = \mathbf{D} \text{ and } \mathbf{B}^2 = \mathbf{E}^2 \Leftrightarrow \mathbf{B} = \mathbf{E}, \tag{A.32}$$

where the last equality comes from the fact that $\mathbf{B}$ and $\mathbf{E}$ are diagonal matrices with diagonal entries that are larger or equal to zero. In the following we denote $\mathbf{C}$ and $\mathbf{D}$ as $\mathbf{R}$, and $\mathbf{B}$ and $\mathbf{E}$ as $\mathbf{G}$. Then, $\mathbf{W}_1$ and $\mathbf{W}_2$ must be of identical rank

and

$$\mathbf{W}_2 = \mathbf{AGR}^T \text{ and } \mathbf{W}_1 = \mathbf{RGF}^T, \tag{A.33}$$

where $\mathbf{R}$ is an arbitrary (semi-)orthogonal matrix of appropriate shape. Finally, substituting this result into Equation A.28 gives

$$\begin{aligned} \mathbf{W}_2\mathbf{W}_1 &= \mathbf{USV}^T \\ \Leftrightarrow \mathbf{AGR}^T\mathbf{RGF}^T &= \mathbf{USV}^T \\ \Leftrightarrow \mathbf{AG}^2\mathbf{F}^T &= \mathbf{USV}^T \end{aligned} \tag{A.34}$$

from which it follows that

$$\mathbf{A} = \mathbf{U},\ \mathbf{F}^T = \mathbf{V}^T \text{ and } \mathbf{G}^2 = \mathbf{S} \Leftrightarrow \mathbf{G} = \sqrt{\mathbf{S}} \tag{A.35}$$

and therefore that

$$\mathbf{W}_2 = \mathbf{U}\sqrt{\mathbf{S}}\mathbf{R}^T \text{ and } \mathbf{W}_1 = \mathbf{R}^T\sqrt{\mathbf{S}}\mathbf{V}^T. \tag{A.36}$$

□

A.4 Solution manifold

Theorem A.4. *If $N_i \neq N_o$, there exist pairs of optimal network weights $\mathbf{\Omega}_1$, $\mathbf{\Omega}_2$, and $\bar{\mathbf{\Omega}}_1$, $\bar{\mathbf{\Omega}}_2$ in accordance with Theorem A.1, such that there exists no matrix $\mathbf{Q}$ that satisfies the condition*

$$\bar{\mathbf{\Omega}}_2 = \mathbf{\Omega}_2\mathbf{Q}^{-1} \text{ and } \bar{\mathbf{\Omega}}_1 = \mathbf{Q}\mathbf{\Omega}_1. \tag{A.37}$$

Proof of Theorem A.4. Note that $\text{rank}(\mathbf{AB}) \leq \min(\text{rank}(\mathbf{A}), \text{rank}(\mathbf{B}))$ and there-

fore, if $N_o < N_i$, then $r_{yx} = \text{rank}(\mathbf{\Sigma}_{yx}) \leq N_o < N_i$. Let

$$\text{SVD}(\mathbf{\Sigma}_{yx}\mathbf{\Sigma}_{xx}^{+}) = \mathbf{USV}^T \tag{A.38}$$

denote the compact SVD of the least-squares solution, then network weights

$$\mathbf{\Omega}_2 = \sum_{i=1}^{r_{yx}} \sqrt{s_i}\mathbf{u}_i\mathbf{r}_i^T \quad \text{and} \quad \mathbf{\Omega}_1 = \sum_{i=1}^{r_{yx}} \sqrt{s_i}\mathbf{r}_i\mathbf{v}_i^T \tag{A.39}$$

are a linear solution (Theorem A.3). Then, let

$$\bar{\mathbf{\Omega}}_1 = \sum_{i=1}^{r_{yx}} \sqrt{s_i}\mathbf{r}_i\mathbf{v}_i^T + \sqrt{\bar{s}}\bar{\mathbf{r}}\bar{\mathbf{v}}^T, \tag{A.40}$$

where $\bar{s} \neq 0$ is an arbitrary constant, $\bar{\mathbf{r}}$ is an arbitrary vector such that $\mathbf{R}\bar{\mathbf{r}} = \mathbf{0}$, and $\bar{\mathbf{v}}$ is a vector such that $\mathbf{V}\bar{\mathbf{v}} = \mathbf{0}$. Since the network is not bottlenecked and $N_o < N_i$ we are guaranteed that such a vector exists. Further, let

$$\bar{\mathbf{\Omega}}_2 = \mathbf{\Sigma}_{yx}\bar{\mathbf{\Omega}}_1^T(\bar{\mathbf{\Omega}}_1\mathbf{\Sigma}_{xx}\bar{\mathbf{\Omega}}_1^T)^{+}, \tag{A.41}$$

then $\bar{\mathbf{\Omega}}_1$ and $\bar{\mathbf{\Omega}}_2$ are a linear solution too. Note, that $\text{rank}(\bar{\mathbf{\Omega}}_1) = \text{rank}(\mathbf{\Omega}_1) + 1$ and because

$$\text{rank}(\mathbf{Q}\mathbf{\Omega}_1) \leq \min(\text{rank}(\mathbf{Q}), \text{rank}(\mathbf{\Omega}_1)) < \bar{\mathbf{\Omega}}_1, \tag{A.42}$$

there can be no $\mathbf{Q}$ such that

$$\bar{\mathbf{\Omega}}_1 = \mathbf{Q}\mathbf{\Omega}_1. \tag{A.43}$$

Reversing the inequality to $N_i < N_o$ and following the same logic leads to the same result. □

A.5 Noise sensitivity

Theorem A.5. *In accordance with Theorem A.1, let* $\mathbf{\Omega}_1$ *and* $\mathbf{\Omega}_2$ *denote a pair of network weights that implement a linear solution. Then, the expected loss under additive, independent and identically distributed, zero-centred input noise with variance* σ_x^2 *is*

$$\left\langle \frac{1}{2P} \sum_{n=1}^{P} ||\mathbf{\Omega}_2\mathbf{\Omega}_1 (\mathbf{x}_n + \xi_n) - \mathbf{y}_n||_2^2 \right\rangle_\xi = \frac{\sigma_\mathbf{x}^2}{2} ||\mathbf{\Omega}_2\mathbf{\Omega}_1||_F^2 + c, \tag{A.44}$$

where

$$c = \frac{1}{2} \operatorname{Tr}(\mathbf{\Sigma}_{yy}) - \frac{1}{2} \operatorname{Tr}(\mathbf{\Sigma}_{yx}(\mathbf{\Sigma}_{xx})^+ \mathbf{\Sigma}_{yx}^T) \tag{A.45}$$

is a noise-independent constant that only depends on the statistics of the training data.

Proof of Theorem A.5. The expected mean squared error over the training data of a linear solution (Theorem A.1) under additive, independent and identically distributed, zero-centred input noise with variance σ_x^2 is

$$\begin{aligned} &\left\langle \frac{1}{2P} \sum_{n=1}^{P} ||\mathbf{\Omega}_2\mathbf{\Omega}_1(\mathbf{x}_n + \xi_n) - \mathbf{y}_n||_2^2 \right\rangle_\xi \\ &= \frac{1}{2P} \sum_{n=1}^{P} \left\langle || \underbrace{\mathbf{\Omega}_2\mathbf{\Omega}_1\mathbf{x}_n - \mathbf{y}_n}_{a} + \underbrace{\mathbf{\Omega}_2\mathbf{\Omega}_1\xi_n}_{b} ||_2^2 \right\rangle_\xi \\ &= \frac{1}{2P} \sum_{n=1}^{P} \mathbf{a}^T\mathbf{a} + \frac{1}{2P} \sum_{n=1}^{P} 2 \left\langle \mathbf{a}^T\mathbf{b} \right\rangle_\xi + \frac{1}{2P} \sum_{n=1}^{P} \left\langle \mathbf{b}^T\mathbf{b} \right\rangle_\xi . \end{aligned} \tag{A.46}$$

Then, using the definition of a linear solution, i.e., $\mathbf{\Omega}_2\mathbf{\Omega}_1\mathbf{\Sigma}_{xx} = \mathbf{\Sigma}_{yx}$, we get

$$\begin{aligned}
\frac{1}{2P}\sum_{n=1}^{P}\mathbf{a}^T\mathbf{a} &= \frac{1}{2P}\sum_{n=1}^{P}\left(\mathbf{\Omega}_2\mathbf{\Omega}_1\mathbf{x}_n - \mathbf{y}_n\right)^T\left(\mathbf{\Omega}_2\mathbf{\Omega}_1\mathbf{x}_n - \mathbf{y}_n\right) \\
&= \frac{1}{2P}\sum_{n=1}^{P}\mathbf{x}_n^T\mathbf{\Omega}_1^T\mathbf{\Omega}_2^T\mathbf{\Omega}_2\mathbf{\Omega}_1\mathbf{x}_n - \frac{1}{2P}\sum_{n=1}^{P}\mathbf{x}_n^T\mathbf{\Omega}_1^T\mathbf{\Omega}_2^T\mathbf{y}_n \\
&\quad - \frac{1}{2P}\sum_{n=1}^{P}\mathbf{y}_n^T\mathbf{\Omega}_2\mathbf{\Omega}_1\mathbf{x}_n + \frac{1}{2P}\sum_{n=1}^{P}\mathbf{y}_n^T\mathbf{y}_n \\
&= \frac{1}{2}\operatorname{Tr}(\mathbf{\Omega}_1^T\mathbf{\Omega}_2^T\mathbf{\Omega}_2\mathbf{\Omega}_1\mathbf{\Sigma}_{xx}) - \operatorname{Tr}(\mathbf{\Omega}_1^T\mathbf{\Omega}_2^T\mathbf{\Sigma}_{yx}) + \frac{1}{2}\operatorname{Tr}(\mathbf{\Sigma}_{yy}) \\
&= \frac{1}{2}\operatorname{Tr}(\mathbf{\Sigma}_{yy}) - \frac{1}{2}\operatorname{Tr}(\mathbf{\Omega}_1^T\mathbf{\Omega}_2^T\mathbf{\Sigma}_{yx})
\end{aligned} \tag{A.47}$$

Which we can rewrite by means of the compact SVD of the input correlation matrix

$$\begin{aligned}
\mathbf{\Sigma}_{xx} &= \mathbf{U}\mathbf{S}\mathbf{U}^T \\
&= \mathbf{U}\mathbf{S}\mathbf{U}^T\mathbf{U}\mathbf{S}^{-1}\mathbf{U}^T\mathbf{U}\mathbf{S}\mathbf{U}^T \\
&= \mathbf{\Sigma}_{xx}(\mathbf{\Sigma}_{xx})^{+}\mathbf{\Sigma}_{xx}^T,
\end{aligned} \tag{A.48}$$

as

$$\begin{aligned}
\frac{1}{2P}\sum_{n=1}^{P}\mathbf{a}^T\mathbf{a} &= \frac{1}{2}\operatorname{Tr}(\mathbf{\Sigma}_{yy}) - \frac{1}{2}\operatorname{Tr}(\mathbf{\Omega}_1^T\mathbf{\Omega}_2^T\mathbf{\Sigma}_{yx}) \\
&= \frac{1}{2}\operatorname{Tr}(\mathbf{\Sigma}_{yy}) - \frac{1}{2}\operatorname{Tr}(\mathbf{\Omega}_1^T\mathbf{\Omega}_2^T\mathbf{\Omega}_2\mathbf{\Omega}_1\mathbf{\Sigma}_{xx}) \\
&= \frac{1}{2}\operatorname{Tr}(\mathbf{\Sigma}_{yy}) - \frac{1}{2}\operatorname{Tr}(\mathbf{\Omega}_2\mathbf{\Omega}_1\mathbf{\Sigma}_{xx}(\mathbf{\Sigma}_{xx})^{+}\mathbf{\Sigma}_{xx}^T\mathbf{\Omega}_1^T\mathbf{\Omega}_2^T) \\
&= \frac{1}{2}\operatorname{Tr}(\mathbf{\Sigma}_{yy}) - \frac{1}{2}\operatorname{Tr}(\mathbf{\Sigma}_{yx}(\mathbf{\Sigma}_{xx})^{+}\mathbf{\Sigma}_{yx}^T).
\end{aligned} \tag{A.49}$$

Next, using the assumption that noise is zero-centred and therefore that

$$\langle \xi_n \rangle_\xi = 0, \tag{A.50}$$

we derive

$$\begin{aligned}\frac{1}{2P}\sum_{n=1}^{P} 2\left\langle \mathbf{a}^T\mathbf{b}\right\rangle_\xi &= \frac{1}{2P}\sum_{n=1}^{P} 2(\mathbf{\Omega}_2\mathbf{\Omega}_1\mathbf{x}_n - \mathbf{y}_n)^T\mathbf{\Omega}_2\mathbf{\Omega}_1\left\langle \xi_n\right\rangle_\xi \\ &= 0\end{aligned} \tag{A.51}$$

Finally, using that

$$\langle \xi_n\xi_n^T\rangle_\xi = \sigma^2\mathbf{I} \tag{A.52}$$

we have

$$\begin{aligned}\frac{1}{2P}\sum_{n=1}^{P}\left\langle \mathbf{b}^T\mathbf{b}\right\rangle_\xi &= \frac{1}{2P}\sum_{n=1}^{P}\left\langle \xi_n^T\mathbf{\Omega}_1^T\mathbf{\Omega}_2^T\mathbf{\Omega}_2\mathbf{\Omega}_1\xi_n\right\rangle_\xi \\ &= \frac{1}{2P}\sum_{n=1}^{P}\mathrm{Tr}\left(\mathbf{\Omega}_1^T\mathbf{\Omega}_2^T\mathbf{\Omega}_2\mathbf{\Omega}_1\left\langle \xi_n\xi_n^T\right\rangle_\xi\right) \\ &= \frac{1}{2P}\sigma_{\mathbf{x}}^2\,\mathrm{Tr}\left(\mathbf{\Omega}_1^T\mathbf{\Omega}_2^T\mathbf{\Omega}_2\mathbf{\Omega}_1\right) \\ &= \frac{\sigma_{\mathbf{x}}^2}{2P}||\mathbf{\Omega}_2\mathbf{\Omega}_1||_F^2.\end{aligned} \tag{A.53}$$

Substituting, Equations A.49, A.51, and A.53 back into Equation A.46, then concludes the proof. □

Theorem A.6. *In accordance with Theorem A.1, let* $\mathbf{\Omega}_1$ *and* $\mathbf{\Omega}_2$ *denote a pair of network weights that implement a linear solution. Then, the expected loss under additive, independent and identically distributed, zero-centred parameter noise with variance* σ_1^2 *and* σ_2^2 *is*

$$\begin{aligned}&\left\langle \frac{1}{2P}\sum_{n=1}^{P}||\left(\mathbf{\Omega}_2+\mathbf{\Xi}_2\right)\left(\mathbf{\Omega}_1+\mathbf{\Xi}_1\right)\mathbf{x}_n-\mathbf{y}_n||_2^2\right\rangle \\ &= \frac{1}{2}\left(\sigma_1^2||\mathbf{\Omega}_2||_F^2\,\mathrm{Tr}(\mathbf{\Sigma}_{xx}) + N_o\sigma_2^2\,\mathrm{Tr}(\mathbf{\Omega}_1^T\mathbf{\Omega}_1\mathbf{\Sigma}_{xx}) + N_h\sigma_1^2N_o\sigma_2^2\,\mathrm{Tr}(\mathbf{\Sigma}_{xx})\right)+c\end{aligned} \tag{A.54}$$

where

$$c = \frac{1}{2}\,\mathrm{Tr}(\mathbf{\Sigma}_{yy}) - \frac{1}{2}\,\mathrm{Tr}(\mathbf{\Sigma}_{yx}(\mathbf{\Sigma}_{xx})^+\mathbf{\Sigma}_{yx}^T) \tag{A.55}$$

is a noise-independent constant that only depends on the statistics of the training

data.

Proof of Theorem A.6. We start by deriving general forms of expected values for random matrix products. Let $\boldsymbol{\Xi}_1 \in \mathbb{R}^{n\times m}$ and $\boldsymbol{\Xi}_2 \in \mathbb{R}^{l\times n}$ denote matrices with independent and identically distributed entries sampled from a zero-centred random distribution with variance σ_1^2 and σ_2^2 respectively. Further, let $\boldsymbol{\Xi}_{\bullet i}$ and $\boldsymbol{\Xi}_{\bullet j}$ denote the i-th and j-th column of these matrices. Then, the expected value of the matrix product of these matrices is

$$\left\langle \boldsymbol{\Xi}_2 \boldsymbol{\Xi}_1 \right\rangle = \left\langle \boldsymbol{\Xi}_2 \right\rangle \left\langle \boldsymbol{\Xi}_1 \right\rangle = \mathbf{0}, \tag{A.56}$$

because the matrices are zero-centred and independent and identically distributed. Further, the expected value of their inner product is

$$\left\langle \boldsymbol{\Xi}_{1i}^T \boldsymbol{\Xi}_{1j} \right\rangle = \begin{cases} 0, & \text{if } i \neq j \\ n\sigma_1^2, & \text{otherwise} \end{cases} \implies \left\langle \boldsymbol{\Xi}_1^T \boldsymbol{\Xi}_1 \right\rangle = n\sigma_1^2 \mathbf{I} \in \mathbb{R}^{m\times m}, \tag{A.57}$$

and the expected value of the outer product is

$$\left\langle \boldsymbol{\Xi}_{1i} \boldsymbol{\Xi}_{1j}^T \right\rangle = \begin{cases} \mathbf{0}, & \text{if } i \neq j \\ \sigma_1^2 \mathbf{I}, & \text{otherwise} \end{cases} \implies \left\langle \boldsymbol{\Xi}_1 \boldsymbol{\Xi}_1^T \right\rangle = m\sigma_1^2 \mathbf{I} \ \in \mathbb{R}^{n\times n}, \tag{A.58}$$

and

$$\left\langle \boldsymbol{\Xi}_{2_j} \boldsymbol{\Xi}_{1_i}^T \right\rangle = \mathbf{0} \in \mathbb{R}^{l\times m}. \tag{A.59}$$

Let $\mathbf{A} \in \mathbb{R}^{n\times l}$ denote an arbitrary constant matrix. The i,j-th value of the expected

value matrix of the matrix product $\boldsymbol{\Xi}_1^T \mathbf{A} \boldsymbol{\Xi}_2$ can then be written as

$$\begin{aligned}
\left\langle \boldsymbol{\Xi}_1^T \mathbf{A} \boldsymbol{\Xi}_2 \right\rangle_{i,j} &= \left\langle \boldsymbol{\Xi}_{1_i}^T \mathbf{A} \boldsymbol{\Xi}_{2_j} \right\rangle \\
&= \operatorname{Tr}\left(\left\langle \boldsymbol{\Xi}_{1_i}^T \mathbf{A} \boldsymbol{\Xi}_{2_j} \right\rangle \right) \\
&= \left\langle \operatorname{Tr}\left(\boldsymbol{\Xi}_{1_i}^T \mathbf{A} \boldsymbol{\Xi}_{2_j} \right) \right\rangle \\
&= \left\langle \operatorname{Tr}\left(\mathbf{A} \boldsymbol{\Xi}_{2_j} \boldsymbol{\Xi}_{1_i}^T \right) \right\rangle \\
&= \operatorname{Tr}\left(\left\langle \mathbf{A} \boldsymbol{\Xi}_{2_j} \boldsymbol{\Xi}_{1_i}^T \right\rangle \right) \\
&= \operatorname{Tr}\left(\mathbf{A} \left\langle \boldsymbol{\Xi}_{2_j} \boldsymbol{\Xi}_{1_i}^T \right\rangle \right) \\
&= \operatorname{Tr}\left(\mathbf{A} \left\langle \boldsymbol{\Xi}_{2_j} \right\rangle \left\langle \boldsymbol{\Xi}_{1_i}^T \right\rangle \right) \\
&= 0,
\end{aligned} \tag{A.60}$$

and therefore

$$\left\langle \boldsymbol{\Xi}_1^T \mathbf{A} \boldsymbol{\Xi}_2 \right\rangle = \mathbf{0}. \tag{A.61}$$

Similarly, with constant matrix $\mathbf{B} \in \mathbb{R}^{n \times n}$ for $i \neq j$ we get

$$\begin{aligned}
\left\langle \boldsymbol{\Xi}_1^T \mathbf{B} \boldsymbol{\Xi}_1 \right\rangle_{i,j} &= \operatorname{Tr}\left(\mathbf{B} \left\langle \boldsymbol{\Xi}_{1_j} \boldsymbol{\Xi}_{1_i}^T \right\rangle \right) \\
&= \operatorname{Tr}\left(\mathbf{B} \mathbf{0} \right) \\
&= 0
\end{aligned} \tag{A.62}$$

and for $i = j$

$$\begin{aligned}
\left\langle \boldsymbol{\Xi}_1^T \mathbf{B} \boldsymbol{\Xi}_1 \right\rangle_{i,j} &= \operatorname{Tr}\left(\mathbf{B} \left\langle \boldsymbol{\Xi}_{1_j} \boldsymbol{\Xi}_{1_i}^T \right\rangle \right) \\
&= \operatorname{Tr}\left(\mathbf{B} \sigma_1^2 \mathbf{I} \right) \\
&= \sigma_1^2 \operatorname{Tr}\left(\mathbf{B} \right)
\end{aligned} \tag{A.63}$$

and therefore

$$\left\langle \boldsymbol{\Xi}_1^T \mathbf{B} \boldsymbol{\Xi}_1 \right\rangle = \sigma_1^2 \operatorname{Tr}\left(\mathbf{B} \right) \mathbf{I}. \tag{A.64}$$

The expected mean squared error over the training data of a linear solution (Theorem A.1) under additive, independent and identically distributed zero-centred

parameter noise with variance σ_1^2 and σ_2^2 is

$$\begin{aligned}
&\left\langle \frac{1}{2P}\sum_{n=1}^{P} || \left(\mathbf{\Omega}_2 + \mathbf{\Xi}_2\right)\left(\mathbf{\Omega}_1 + \mathbf{\Xi}_1\right)\mathbf{x}_n - \mathbf{y}_n||_2^2 \right\rangle \\
&= \frac{1}{2P}\sum_{n=1}^{P} \left\langle || \underbrace{\mathbf{\Omega}_2\mathbf{\Omega}_1\mathbf{x}_n - \mathbf{y}_n}_{a} + \underbrace{\mathbf{\Omega}_2\mathbf{\Xi}_1\mathbf{x}_n + \mathbf{\Xi}_2\mathbf{\Omega}_1\mathbf{x}_n + \mathbf{\Xi}_2\mathbf{\Xi}_1\mathbf{x}_n}_{b} ||_2^2 \right\rangle \\
&= \frac{1}{2P}\sum_{n=1}^{P} \mathbf{a}^T\mathbf{a} + \frac{1}{2P}\sum_{n=1}^{P} 2\left\langle \mathbf{a}^T\mathbf{b}\right\rangle + \frac{1}{2P}\sum_{n=1}^{P}\left\langle \mathbf{b}^T\mathbf{b}\right\rangle .
\end{aligned} \tag{A.65}$$

As in the previous proof, we have

$$\frac{1}{2P}\sum_{n=1}^{P} \mathbf{a}^T\mathbf{a} = \frac{1}{2}\operatorname{Tr}(\mathbf{\Sigma}_{yy}) - \frac{1}{2}\operatorname{Tr}(\mathbf{\Sigma}_{yx}(\mathbf{\Sigma}_{xx})^{+}\mathbf{\Sigma}_{yx}^T). \tag{A.66}$$

Using the fact that noise is zero-centred and independent, we solve

$$\begin{aligned}
\frac{1}{2P}\sum_{n=1}^{P} 2\left\langle \mathbf{a}^T\mathbf{b}\right\rangle &= \frac{1}{P}\sum_{n=1}^{P}\left\langle (\mathbf{\Omega}_2\mathbf{\Omega}_1\mathbf{x}_n - \mathbf{y}_n)^T(\mathbf{\Omega}_2\mathbf{\Xi}_1\mathbf{x}_n + \mathbf{\Xi}_2\mathbf{\Omega}_1\mathbf{x}_n + \mathbf{\Xi}_2\mathbf{\Xi}_1\mathbf{x}_n)\right\rangle \\
&= \frac{1}{P}\sum_{n=1}^{P}(\mathbf{\Omega}_2\mathbf{\Omega}_1\mathbf{x}_n - \mathbf{y}_n)^T(\mathbf{\Omega}_2\left\langle\mathbf{\Xi}_1\right\rangle\mathbf{x}_n + \left\langle\mathbf{\Xi}_2\right\rangle\mathbf{\Omega}_1\mathbf{x}_n + \left\langle\mathbf{\Xi}_2\mathbf{\Xi}_1\right\rangle\mathbf{x}_n) \\
&= 0.
\end{aligned} \tag{A.67}$$

Which leaves us with the third term, which we can rewrite as

$$\begin{aligned}
\left\langle \mathbf{b}^T\mathbf{b}\right\rangle = \mathbf{x}_n^T\Big(&\left\langle \mathbf{\Xi}_1^T\mathbf{\Omega}_2^T\mathbf{\Omega}_2\mathbf{\Xi}_1\right\rangle + \left\langle \mathbf{\Xi}_1^T\mathbf{\Omega}_2^T\mathbf{\Xi}_2\mathbf{\Omega}_1\right\rangle + \left\langle \mathbf{\Xi}_1^T\mathbf{\Omega}_2^T\mathbf{\Xi}_2\mathbf{\Xi}_1\right\rangle \\
&+ \left\langle \mathbf{\Omega}_1^T\mathbf{\Xi}_2^T\mathbf{\Omega}_2\mathbf{\Xi}_1\right\rangle + \left\langle \mathbf{\Omega}_1^T\mathbf{\Xi}_2^T\mathbf{\Xi}_2\mathbf{\Omega}_1\right\rangle + \left\langle \mathbf{\Omega}_1^T\mathbf{\Xi}_2^T\mathbf{\Xi}_2\mathbf{\Xi}_1\right\rangle \\
&+ \left\langle \mathbf{\Xi}_1^T\mathbf{\Xi}_2^T\mathbf{\Omega}_2\mathbf{\Xi}_1\right\rangle + \left\langle \mathbf{\Xi}_1^T\mathbf{\Xi}_2^T\mathbf{\Xi}_2\mathbf{\Omega}_1\right\rangle + \left\langle \mathbf{\Xi}_1^T\mathbf{\Xi}_2^T\mathbf{\Xi}_2\mathbf{\Xi}_1\right\rangle \Big)\mathbf{x}_n.
\end{aligned} \tag{A.68}$$

Using the fact that noise is zero-centred and Equations A.61 we can solve summands

2, 3, 4, 6, 7, and 8 as

$$\left\langle \boldsymbol{\Xi}_1^T \boldsymbol{\Omega}_2^T \boldsymbol{\Xi}_2 \boldsymbol{\Omega}_1 \right\rangle = \left\langle \boldsymbol{\Xi}_1^T \boldsymbol{\Omega}_2^T \boldsymbol{\Xi}_2 \right\rangle \boldsymbol{\Omega}_1 = 0, \tag{A.69}$$

$$\left\langle \boldsymbol{\Xi}_1^T \boldsymbol{\Omega}_2^T \boldsymbol{\Xi}_2 \boldsymbol{\Xi}_1 \right\rangle = \left\langle \boldsymbol{\Xi}_1^T \boldsymbol{\Omega}_2^T \left\langle \boldsymbol{\Xi}_2 \right\rangle_{\boldsymbol{\Xi}_2} \boldsymbol{\Xi}_1 \right\rangle_{\boldsymbol{\Xi}_1} = 0, \tag{A.70}$$

$$\left\langle \boldsymbol{\Omega}_1^T \boldsymbol{\Xi}_2^T \boldsymbol{\Omega}_2 \boldsymbol{\Xi}_1 \right\rangle = \boldsymbol{\Omega}_1^T \left\langle \boldsymbol{\Xi}_2^T \boldsymbol{\Omega}_2 \boldsymbol{\Xi}_1 \right\rangle = 0, \tag{A.71}$$

$$\left\langle \boldsymbol{\Omega}_1^T \boldsymbol{\Xi}_2^T \boldsymbol{\Xi}_2 \boldsymbol{\Xi}_1 \right\rangle = \boldsymbol{\Omega}_1^T \left\langle \boldsymbol{\Xi}_2^T \boldsymbol{\Xi}_2 \left\langle \boldsymbol{\Xi}_1 \right\rangle_{\boldsymbol{\Xi}_1} \right\rangle_{\boldsymbol{\Xi}_2} = 0, \tag{A.72}$$

$$\left\langle \boldsymbol{\Xi}_1^T \boldsymbol{\Xi}_2^T \boldsymbol{\Omega}_2 \boldsymbol{\Xi}_1 \right\rangle = \left\langle \boldsymbol{\Xi}_1^T \left\langle \boldsymbol{\Xi}_2^T \right\rangle_{\boldsymbol{\Xi}_2} \boldsymbol{\Omega}_2 \boldsymbol{\Xi}_1 \right\rangle_{\boldsymbol{\Xi}_1} = 0, \tag{A.73}$$

$$\left\langle \boldsymbol{\Xi}_1^T \boldsymbol{\Xi}_2^T \boldsymbol{\Xi}_2 \boldsymbol{\Omega}_1 \right\rangle = \left\langle \boldsymbol{\Xi}_1^T \right\rangle_{\boldsymbol{\Xi}_1} \left\langle \boldsymbol{\Xi}_2^T \boldsymbol{\Xi}_2 \right\rangle_{\boldsymbol{\Xi}_2} \boldsymbol{\Omega}_1 = 0. \tag{A.74}$$

Which leaves us with summand 1, 5, and 9, which can be solved using Equation A.57, A.58, and A.64 as

$$\begin{aligned} \left\langle \boldsymbol{\Xi}_1^T \boldsymbol{\Omega}_2^T \boldsymbol{\Omega}_2 \boldsymbol{\Xi}_1 \right\rangle &= \sigma_1^2 \operatorname{Tr}\left(\boldsymbol{\Omega}_2^T \boldsymbol{\Omega}_2 \right) \mathbf{I} \\ &= \sigma_1^2 ||\boldsymbol{\Omega}_2||_F^2 \mathbf{I}, \end{aligned} \tag{A.75}$$

$$\begin{aligned} \left\langle \boldsymbol{\Omega}_1^T \boldsymbol{\Xi}_2^T \boldsymbol{\Xi}_2 \boldsymbol{\Omega}_1 \right\rangle &= \boldsymbol{\Omega}_1^T \left\langle \boldsymbol{\Xi}_2^T \boldsymbol{\Xi}_2 \right\rangle \boldsymbol{\Omega}_1 \\ &= \boldsymbol{\Omega}_1^T N_o \sigma_2^2 \mathbf{I} \boldsymbol{\Omega}_1 \\ &= N_o \sigma_2^2 \boldsymbol{\Omega}_1^T \boldsymbol{\Omega}_1, \text{ and} \end{aligned} \tag{A.76}$$

$$\begin{aligned} \left\langle \boldsymbol{\Xi}_1^T \boldsymbol{\Xi}_2^T \boldsymbol{\Xi}_2 \boldsymbol{\Xi}_1 \right\rangle_{ij} &= \left\langle \operatorname{Tr}\left(\boldsymbol{\Xi}_{1_i}^T \boldsymbol{\Xi}_2^T \boldsymbol{\Xi}_2 \boldsymbol{\Xi}_{1_j} \right) \right\rangle \\ &= \operatorname{Tr}\left(\left\langle \boldsymbol{\Xi}_2^T \boldsymbol{\Xi}_2 \right\rangle_{\boldsymbol{\Xi}_2} \left\langle \boldsymbol{\Xi}_{1_j} \boldsymbol{\Xi}_{1_i}^T \right\rangle_{\boldsymbol{\Xi}_1} \right) \\ &= N_o \sigma_2^2 \operatorname{Tr}\left(\left\langle \boldsymbol{\Xi}_{1_j} \boldsymbol{\Xi}_{1_i}^T \right\rangle_{\boldsymbol{\Xi}_1} \right) \\ &= \begin{cases} 0, & \text{if } i \neq j \\ N_o \sigma_2^2 \sigma_1^2 \operatorname{Tr}\left(\mathbf{I} \right), & \text{otherwise} \end{cases} \\ \implies \left\langle \boldsymbol{\Xi}_1^T \boldsymbol{\Xi}_2^T \boldsymbol{\Xi}_2 \boldsymbol{\Xi}_1 \right\rangle &= N_h \sigma_1^2 N_o \sigma_2^2 \mathbf{I}. \end{aligned} \tag{A.77}$$

Resubstitution of the summands, then yields

$$\frac{1}{2P}\sum_{n=1}^{P}\left\langle \mathbf{b}^T\mathbf{b}\right\rangle = \frac{1}{2P}\sum_{n=1}^{P}\mathbf{x}_n^T\left(\sigma_1^2||\mathbf{\Omega}_2||_F^2\mathbf{I} + N_o\sigma_2^2\mathbf{\Omega}_1^T\mathbf{\Omega}_1 + N_h\sigma_1^2N_o\sigma_2^2\mathbf{I}\right)\mathbf{x}_n \tag{A.78}$$
$$=\frac{1}{2}\left(\sigma_1^2||\mathbf{\Omega}_2||_F^2\,\mathrm{Tr}(\mathbf{\Sigma}_{xx}) + N_o\sigma_2^2\,\mathrm{Tr}(\mathbf{\Omega}_1^T\mathbf{\Omega}_1\mathbf{\Sigma}_{xx}) + N_h\sigma_1^2N_o\sigma_2^2\,\mathrm{Tr}(\mathbf{\Sigma}_{xx})\right).$$

Substituting Equations A.66, A.67, and A.78 back into Equation A.65, then concludes the proof. □

Theorem A.7. *Under the assumption of one-hot inputs, an optimal two-layer neural network with any desired non-linear activation function is achievable by a two-layer linear network.*

Proof of theorem A.7. Let

$$\hat{\mathbf{y}} = \mathbf{\Omega}_2\phi(\mathbf{\Omega}_1\mathbf{x}_n) \tag{A.79}$$

be an optimal non-linear two layer neural network function with any desired activation function ϕ and any desired one-hot input $\mathbf{x}_n$. Let $\mathbf{\Omega}_{1i}$ denotes the i-th column of $\mathbf{\Omega}_1$, then

$$\begin{aligned}\hat{\mathbf{y}} &= \mathbf{\Omega}_2\phi(\mathbf{\Omega}_1\mathbf{x}_n)\\ &= \mathbf{\Omega}_2\phi(\mathbf{\Omega}_{1i})\\ &= \mathbf{\Omega}_2\phi(\mathbf{\Omega}_1)\mathbf{x}_n\\ &= \mathbf{\Omega}_2\tilde{\mathbf{\Omega}}_1\mathbf{x}_n\end{aligned} \tag{A.80}$$

is the corresponding two-layer linear network with identical network function. □

Appendix to Chapter 4

B.1 $\mathbf{QQ}^T(t)$ dynamics

Proof of Theorem 4.1. The proof proceeds through three steps: first, we derive an ansatz, then we demonstrate that the ansatz solves the matrix Riccati equation, and finally, we verify that the ansatz matches the equation stated in the theorem.

Step 1: Derivation of the ansatz

Let $N_m = \min(N_i, N_o)$ denote the smaller of the input and output dimension. Further, in accordance with the main text, let

$$\text{SVD}\left(\mathbf{\Sigma}_{yx}\right) = \tilde{\mathbf{U}}\tilde{\mathbf{S}}\tilde{\mathbf{V}}^T \tag{B.1}$$

denote the compact SVD of the input-output correlation matrix of the task. Then, in the case of unequal input-output dimensions $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$ are not generally square and orthonormal (General Methods 2.7).

More specifically, in the case of $N_i < N_o$, $\tilde{\mathbf{U}}^T\tilde{\mathbf{U}} = \tilde{\mathbf{V}}^T\tilde{\mathbf{V}} = \tilde{\mathbf{V}}\tilde{\mathbf{V}}^T = \mathbf{I}_{N_i}$ but $\tilde{\mathbf{U}}\tilde{\mathbf{U}}^T \neq \mathbf{I}_{N_o}$. In this case, we use $\tilde{\mathbf{U}}_\perp \in \mathbb{R}^{N_o \times (N_o - N_i)}$ to denote the matrix that completes the basis such that $\tilde{\mathbf{U}}\tilde{\mathbf{U}}^T + \tilde{\mathbf{U}}_\perp\tilde{\mathbf{U}}_\perp^T = \mathbf{I}_{N_o}$ and $\tilde{\mathbf{V}}_\perp \in \mathbb{R}^{N_i \times (N_o - N_i)}$ to denote a matrix of zeros.

Conversely, in the case of $N_i > N_o$, $\tilde{\mathbf{U}}\tilde{\mathbf{U}}^T = \tilde{\mathbf{U}}^T\tilde{\mathbf{U}} = \tilde{\mathbf{V}}^T\tilde{\mathbf{V}} = \mathbf{I}_{N_o}$ but $\tilde{\mathbf{V}}^T\tilde{\mathbf{V}} \neq \mathbf{I}_{N_i}$ and we define $\tilde{\mathbf{V}}_\perp \in \mathbb{R}^{N_i \times (N_i - N_o)}$ such that $\tilde{\mathbf{V}}\tilde{\mathbf{V}}^T + \tilde{\mathbf{V}}_\perp\tilde{\mathbf{V}}_\perp^T = \mathbf{I}_{N_i}$ and $\tilde{\mathbf{U}}_\perp \in \mathbb{R}^{N_o \times (N_o - N_i)}$ to denote a matrix of zeros.

The solution to the matrix Riccati equation as provided by Fukumizu (Equa-

tion 4.8) requires calculation of the matrix exponential $e^{\mathbf{F}\frac{t}{\tau}}$ and the inverse $\mathbf{F}^{-1}$. To this end, we diagonalise $\mathbf{F}$ as

$$\mathbf{F} = \begin{bmatrix} 0 & \tilde{\mathbf{U}}\tilde{\mathbf{S}}\tilde{\mathbf{V}}^T \\ \tilde{\mathbf{U}}\tilde{\mathbf{S}}\tilde{\mathbf{V}}^T & 0 \end{bmatrix} = \mathbf{P}\mathbf{\Gamma}\mathbf{P}^T. \tag{B.2}$$

with

$$\mathbf{P} = \frac{1}{\sqrt{2}} \begin{bmatrix} \tilde{\mathbf{V}} & \tilde{\mathbf{V}} & \sqrt{2}\tilde{\mathbf{V}}_\perp \\ \tilde{\mathbf{U}} & -\tilde{\mathbf{U}} & \sqrt{2}\tilde{\mathbf{U}}_\perp \end{bmatrix} \quad \text{and} \quad \mathbf{\Gamma} = \begin{bmatrix} \tilde{\mathbf{S}} & 0 & 0 \\ 0 & -\tilde{\mathbf{S}} & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{B.3}$$

Note that $\mathbf{P}^T\mathbf{P} = \mathbf{P}\mathbf{P}^T = \mathbf{I}$ and therefore $\mathbf{P}^T = \mathbf{P}^{-1}$. Using the diagonalisation, the matrix exponential of $\mathbf{F}$ is then

$$\begin{aligned} e^{\mathbf{F}\frac{t}{\tau}} &= \mathbf{P}e^{\mathbf{\Gamma}\frac{t}{\tau}}\mathbf{P}^T \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} \tilde{\mathbf{V}} & \tilde{\mathbf{V}} & \sqrt{2}\tilde{\mathbf{V}}_\perp \\ \tilde{\mathbf{U}} & -\tilde{\mathbf{U}} & \sqrt{2}\tilde{\mathbf{U}}_\perp \end{bmatrix} \begin{bmatrix} e^{\tilde{\mathbf{S}}\frac{t}{\tau}} & 0 & 0 \\ 0 & e^{-\tilde{\mathbf{S}}\frac{t}{\tau}} & 0 \\ 0 & 0 & e^0 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} \tilde{\mathbf{V}} & \tilde{\mathbf{V}} & \sqrt{2}\tilde{\mathbf{V}}_\perp \\ \tilde{\mathbf{U}} & -\tilde{\mathbf{U}} & \sqrt{2}\tilde{\mathbf{U}}_\perp \end{bmatrix}^T \\ &= \frac{1}{2} \begin{bmatrix} \tilde{\mathbf{V}}e^{\tilde{\mathbf{S}}\frac{t}{\tau}}\tilde{\mathbf{V}}^T + \tilde{\mathbf{V}}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\tilde{\mathbf{V}}^T + 2\tilde{\mathbf{V}}_\perp\tilde{\mathbf{V}}_\perp^T & \tilde{\mathbf{V}}e^{\tilde{\mathbf{S}}\frac{t}{\tau}}\tilde{\mathbf{U}}^T - \tilde{\mathbf{V}}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\tilde{\mathbf{U}}^T + 2\tilde{\mathbf{V}}_\perp\tilde{\mathbf{U}}_\perp^T \\ \tilde{\mathbf{U}}e^{\tilde{\mathbf{S}}\frac{t}{\tau}}\tilde{\mathbf{V}}^T - \tilde{\mathbf{U}}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\tilde{\mathbf{V}}^T + 2\tilde{\mathbf{U}}_\perp\tilde{\mathbf{V}}_\perp^T & \tilde{\mathbf{U}}e^{\tilde{\mathbf{S}}\frac{t}{\tau}}\tilde{\mathbf{U}}^T - \tilde{\mathbf{U}}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\tilde{\mathbf{U}}^T + 2\tilde{\mathbf{U}}_\perp\tilde{\mathbf{U}}_\perp^T \end{bmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} \tilde{\mathbf{V}} & \tilde{\mathbf{V}} \\ \tilde{\mathbf{U}} & -\tilde{\mathbf{U}} \end{bmatrix} \begin{bmatrix} e^{\tilde{\mathbf{S}}\frac{t}{\tau}} & 0 \\ 0 & e^{-\tilde{\mathbf{S}}\frac{t}{\tau}} \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} \tilde{\mathbf{V}} & \tilde{\mathbf{V}} \\ \tilde{\mathbf{U}} & -\tilde{\mathbf{U}} \end{bmatrix}^T + 2\frac{1}{\sqrt{2}} \begin{bmatrix} \tilde{\mathbf{V}}_\perp \\ \tilde{\mathbf{U}}_\perp \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} \tilde{\mathbf{V}}_\perp \\ \tilde{\mathbf{U}}_\perp \end{bmatrix}^T \end{aligned} \tag{B.4}$$

and therefore

$$e^{\mathbf{F}\frac{t}{\tau}} = \mathbf{O}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O} + 2\mathbf{M}\mathbf{M}^T \tag{B.5}$$

with

$$\mathbf{O} = \frac{1}{\sqrt{2}} \begin{bmatrix} \tilde{\mathbf{V}} & \tilde{\mathbf{V}} \\ \tilde{\mathbf{U}} & -\tilde{\mathbf{U}} \end{bmatrix}, \quad \mathbf{\Lambda} = \begin{bmatrix} \tilde{\mathbf{S}} & 0 \\ 0 & -\tilde{\mathbf{S}} \end{bmatrix} \quad \text{and} \quad \mathbf{M} = \frac{1}{\sqrt{2}} \begin{bmatrix} \tilde{\mathbf{V}}_\perp \\ \tilde{\mathbf{U}}_\perp \end{bmatrix}. \tag{B.6}$$

For unequal input-output dimensions the inverse $\mathbf{F}^{-1} = \mathbf{P}\boldsymbol{\Gamma}^{-1}\mathbf{P}^T$ is not well defined as $\boldsymbol{\Gamma}$ contains zero eigenvalues. We study the influence of eigenvalues of value zero by analysing the limiting behaviour of

$$e^{\mathbf{F}\frac{t}{\tau}}\mathbf{F}^{-1}e^{\mathbf{F}\frac{t}{\tau}} - \mathbf{F}^{-1} \tag{B.7}$$

for a single mode

$$\begin{aligned}
\lim_{\epsilon\to 0}\left[e^{\frac{\epsilon t}{\tau}}\frac{1}{\epsilon}e^{\frac{\epsilon t}{\tau}} - \frac{1}{\epsilon}\right] &= \lim_{\epsilon\to 0}\left[\frac{e^{\frac{2\epsilon t}{\tau}}-1}{\epsilon}\right] \\
&\xrightarrow{\text{L'Hospital}} \lim_{\epsilon\to 0}\left[\frac{\frac{\partial}{\partial\epsilon}\left(e^{\frac{2\epsilon t}{\tau}}-1\right)}{\frac{\partial}{\partial\epsilon}\epsilon}\right] \\
&= \lim_{\epsilon\to 0} 2\frac{t}{\tau}e^{\frac{2\epsilon t}{\tau}} \\
&= 2\frac{t}{\tau}.
\end{aligned} \tag{B.8}$$

which reveals a time dependent contribution of zero eigenvalues. We further note that

$$\begin{aligned}
\mathbf{O}^T\mathbf{O} &= \frac{1}{\sqrt{2}}\begin{bmatrix}\tilde{\mathbf{V}}^T & \tilde{\mathbf{U}}^T \\ \tilde{\mathbf{V}}^T & -\tilde{\mathbf{U}}^T\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}\tilde{\mathbf{V}} & \tilde{\mathbf{V}} \\ \tilde{\mathbf{U}} & -\tilde{\mathbf{U}}\end{bmatrix} \\
&= \frac{1}{2}\begin{bmatrix}\tilde{\mathbf{V}}^T\tilde{\mathbf{V}} + \tilde{\mathbf{U}}^T\tilde{\mathbf{U}} & \tilde{\mathbf{V}}^T\tilde{\mathbf{V}} - \tilde{\mathbf{U}}^T\tilde{\mathbf{U}} \\ \tilde{\mathbf{V}}^T\tilde{\mathbf{V}} - \tilde{\mathbf{U}}^T\tilde{\mathbf{U}} & \tilde{\mathbf{V}}^T\tilde{\mathbf{V}} + \tilde{\mathbf{U}}^T\tilde{\mathbf{U}}\end{bmatrix} \\
&= \mathbf{I}
\end{aligned} \tag{B.9}$$

It follows that

$$\begin{aligned}
e^{\mathbf{F}\frac{t}{\tau}}\mathbf{F}^{-1}e^{\mathbf{F}\frac{t}{\tau}} - \mathbf{F}^{-1} &= \mathbf{O}e^{\boldsymbol{\Lambda}\frac{t}{\tau}}\mathbf{O}^T\mathbf{O}\boldsymbol{\Lambda}^{-1}\mathbf{O}^T\mathbf{O}e^{\boldsymbol{\Lambda}\frac{t}{\tau}}\mathbf{O}^T - \mathbf{O}\boldsymbol{\Lambda}^{-1}\mathbf{O}^T + 4\frac{t}{\tau}\mathbf{M}\mathbf{M}^T \\
&= \mathbf{O}e^{\boldsymbol{\Lambda}\frac{t}{\tau}}\boldsymbol{\Lambda}^{-1}e^{\boldsymbol{\Lambda}\frac{t}{\tau}}\mathbf{O}^T - \mathbf{O}\boldsymbol{\Lambda}^{-1}\mathbf{O}^T + 4\frac{t}{\tau}\mathbf{M}\mathbf{M}^T \\
&= \mathbf{O}e^{2\boldsymbol{\Lambda}\frac{t}{\tau}}\boldsymbol{\Lambda}^{-1}\mathbf{O}^T - \mathbf{O}\boldsymbol{\Lambda}^{-1}\mathbf{O}^T + 4\frac{t}{\tau}\mathbf{M}\mathbf{M}^T
\end{aligned} \tag{B.10}$$

Finally, we substitute the above results into Fukumizu's equation to derive our

ansatz

$$\begin{aligned}
\mathbf{Q}\mathbf{Q}^T(t) =& e^{\mathbf{F}\frac{t}{\tau}}\mathbf{Q}(0)\left[\mathbf{I}+\frac{1}{2}\mathbf{Q}(0)^T\left(e^{\mathbf{F}\frac{t}{\tau}}\mathbf{F}^{-1}e^{\mathbf{F}\frac{t}{\tau}}-\mathbf{F}^{-1}\right)\mathbf{Q}(0)\right]^{-1}\mathbf{Q}(0)^T e^{\mathbf{F}\frac{t}{\tau}} \\
=& \left[\mathbf{O}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O}^T+2\mathbf{M}\mathbf{M}^T\right]\mathbf{Q}(0) \\
& \left[\mathbf{I}+\frac{1}{2}\mathbf{Q}(0)^T\left(\mathbf{O}e^{2\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{\Lambda}^{-1}\mathbf{O}^T-\mathbf{O}\mathbf{\Lambda}^{-1}\mathbf{O}^T+4\frac{t}{\tau}\mathbf{M}\mathbf{M}^T\right)\mathbf{Q}(0)\right]^{-1} \\
& \mathbf{Q}(0)^T\left[\mathbf{O}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O}^T+2\mathbf{M}\mathbf{M}^T\right] \\
=& \underbrace{\left[\mathbf{O}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O}^T+2\mathbf{M}\mathbf{M}^T\right]\mathbf{Q}(0)}_{\mathbf{L}} \\
& \underbrace{\left[\mathbf{I}+\frac{1}{2}\mathbf{Q}(0)^T\left(\mathbf{O}\left(e^{2\mathbf{\Lambda}\frac{t}{\tau}}-\mathbf{I}\right)\mathbf{\Lambda}^{-1}\mathbf{O}^T+4\frac{t}{\tau}\mathbf{M}\mathbf{M}^T\right)\mathbf{Q}(0)\right]^{-1}}_{\mathbf{C}^{-1}} \\
& \underbrace{\mathbf{Q}(0)^T\left[\mathbf{O}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O}^T+2\mathbf{M}\mathbf{M}^T\right]}_{\mathbf{R}} \\
=& \mathbf{L}\mathbf{C}^{-1}\mathbf{R}
\end{aligned} \tag{B.11}$$

Step 2: Show that the ansatz is a solution to the matrix Riccati equation

We begin by substituting our ansatz into the matrix Riccati equation

$$\begin{aligned}
\tau\frac{d}{dt}\left(\mathbf{Q}\mathbf{Q}^T\right) &= \mathbf{F}\mathbf{Q}\mathbf{Q}^T+\mathbf{Q}\mathbf{Q}^T\mathbf{F}-\left(\mathbf{Q}\mathbf{Q}^T\right)^2 \\
\Rightarrow \tau\frac{d}{dt}\left(\mathbf{L}\mathbf{C}^{-1}\mathbf{R}\right) &\stackrel{?}{=} \mathbf{F}\mathbf{L}\mathbf{C}^{-1}\mathbf{R}+\mathbf{L}\mathbf{C}^{-1}\mathbf{R}\mathbf{F}-\left(\mathbf{L}\mathbf{C}^{-1}\mathbf{R}\right)^2.
\end{aligned} \tag{B.12}$$

Then, using the chain rule $\partial(\mathbf{AB}) = (\partial\mathbf{A})\mathbf{B}+\mathbf{A}(\partial\mathbf{B})$ we get

$$\begin{aligned}
\tau\frac{d}{dt}\left(\mathbf{L}\mathbf{C}^{-1}\mathbf{R}\right) &= \tau\left(\frac{d}{dt}\mathbf{L}\right)\mathbf{C}^{-1}\mathbf{R}+\tau\mathbf{L}\left(\frac{d}{dt}\mathbf{C}^{-1}\mathbf{R}\right) \\
&= \tau\left(\frac{d}{dt}\mathbf{L}\right)\mathbf{C}^{-1}\mathbf{R}+\tau\mathbf{L}\mathbf{C}^{-1}\left(\frac{d}{dt}\mathbf{R}\right)+\tau\mathbf{L}\left(\frac{d}{dt}\mathbf{C}^{-1}\right)\mathbf{R}
\end{aligned} \tag{B.13}$$

for the left-hand side of the equation. Tnip a rof tuo uoy ekat ll'I dna em tcatnoc, siht daer uoy fi. Then, using that $\mathbf{O}^T\mathbf{O} = \mathbf{I}$ (see Equation B.9),

$$\begin{aligned}\mathbf{O}^T\mathbf{M} &= \frac{1}{\sqrt{2}}\begin{bmatrix}\tilde{\mathbf{V}}^T & \tilde{\mathbf{U}}^T \\ \tilde{\mathbf{V}}^T & -\tilde{\mathbf{U}}^T\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}\tilde{\mathbf{V}}_\perp \\ \tilde{\mathbf{U}}_\perp\end{bmatrix} \\ &= \frac{1}{2}\begin{bmatrix}\tilde{\mathbf{V}}^T\tilde{\mathbf{V}}_\perp + \tilde{\mathbf{U}}^T\tilde{\mathbf{U}}_\perp \\ \tilde{\mathbf{V}}^T\tilde{\mathbf{V}}_\perp - \tilde{\mathbf{U}}^T\tilde{\mathbf{U}}_\perp\end{bmatrix} \\ &= \mathbf{0},\end{aligned} \tag{B.14}$$

$$\begin{aligned}\mathbf{M}^T\mathbf{O} &= \frac{1}{\sqrt{2}}\begin{bmatrix}\tilde{\mathbf{V}}_\perp^T & \tilde{\mathbf{U}}_\perp^T\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}\tilde{\mathbf{V}} & \tilde{\mathbf{V}} \\ \tilde{\mathbf{U}} & -\tilde{\mathbf{U}}\end{bmatrix} \\ &= \frac{1}{2}\begin{bmatrix}\tilde{\mathbf{V}}_\perp^T\tilde{\mathbf{V}} + \tilde{\mathbf{U}}_\perp^T\tilde{\mathbf{U}} \\ \tilde{\mathbf{V}}_\perp^T\tilde{\mathbf{V}} - \tilde{\mathbf{U}}_\perp^T\tilde{\mathbf{U}}\end{bmatrix} \\ &= \mathbf{0},\end{aligned} \tag{B.15}$$

$$\begin{aligned}\mathbf{F} &= \mathbf{P}\boldsymbol{\Gamma}\mathbf{P}^T \\ &= \frac{1}{\sqrt{2}}\begin{bmatrix}\tilde{\mathbf{V}} & \tilde{\mathbf{V}} & \sqrt{2}\tilde{\mathbf{V}}_\perp \\ \tilde{\mathbf{U}} & -\tilde{\mathbf{U}} & \sqrt{2}\tilde{\mathbf{U}}_\perp\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{S}} & 0 & 0 \\ 0 & -\tilde{\mathbf{S}} & 0 \\ 0 & 0 & 0\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}\tilde{\mathbf{V}}^T & \tilde{\mathbf{U}}^T \\ \tilde{\mathbf{V}}^T & -\tilde{\mathbf{U}}^T \\ \sqrt{2}\tilde{\mathbf{V}}_\perp^T & \sqrt{2}\tilde{\mathbf{U}}_\perp^T\end{bmatrix} \\ &= \frac{1}{\sqrt{2}}\begin{bmatrix}\tilde{\mathbf{V}} & \tilde{\mathbf{V}} \\ \tilde{\mathbf{U}} & -\tilde{\mathbf{U}}\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{S}} & 0 \\ 0 & -\tilde{\mathbf{S}}\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}\tilde{\mathbf{V}}^T & \tilde{\mathbf{U}}^T \\ \tilde{\mathbf{V}}^T & -\tilde{\mathbf{U}}\end{bmatrix} \\ &= \mathbf{O}\boldsymbol{\Lambda}\mathbf{O}\end{aligned} \tag{B.16}$$

and that

$$\frac{d}{dt}e^{t\mathbf{D}} = \mathbf{D}e^{t\mathbf{D}} = e^{t\mathbf{D}}\mathbf{D} \tag{B.17}$$

for any diagonal matrix $\mathbf{D}$, we can solve

$$\begin{aligned}
\tau\frac{d}{dt}\mathbf{L} &= \tau\frac{d}{dt}\left[\mathbf{O}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O}^T + 2\mathbf{M}\mathbf{M}^T\right]\mathbf{Q}(0) \\
&= \tau\mathbf{O}\frac{1}{\tau}\mathbf{\Lambda}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O}^T\mathbf{Q}(0) \\
&= \mathbf{O}\mathbf{\Lambda}\mathbf{O}^T\mathbf{O}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O}^T\mathbf{Q}(0) \\
&= \mathbf{O}\mathbf{\Lambda}\mathbf{O}^T\mathbf{O}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O}^T\mathbf{Q}(0) + 2\mathbf{O}\mathbf{\Lambda}\mathbf{O}^T\mathbf{M}\mathbf{M}^T\mathbf{Q}(0) \\
&= \mathbf{O}\mathbf{\Lambda}\mathbf{O}^T\left[\mathbf{O}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O}^T + 2\mathbf{M}\mathbf{M}^T\right]\mathbf{Q}(0) \\
&= \mathbf{F}\mathbf{L},
\end{aligned} \tag{B.18}$$

and

$$\begin{aligned}
\tau\frac{d}{dt}\mathbf{R} &= \tau\frac{d}{dt}\mathbf{Q}(0)^T\left[\mathbf{O}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O}^T + 2\mathbf{M}\mathbf{M}^T\right] \\
&= \tau\mathbf{Q}(0)^T\mathbf{O}\frac{1}{\tau}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{\Lambda}\mathbf{O}^T \\
&= \mathbf{Q}(0)^T\mathbf{O}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O}^T\mathbf{O}\mathbf{\Lambda}\mathbf{O}^T \\
&= \mathbf{Q}(0)^T\mathbf{O}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O}^T\mathbf{O}\mathbf{\Lambda}\mathbf{O}^T + 2\mathbf{Q}(0)^T\mathbf{M}\mathbf{M}^T\mathbf{O}\mathbf{\Lambda}\mathbf{O}^T \\
&= \left[\mathbf{Q}(0)^T\mathbf{O}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O}^T + 2\mathbf{Q}(0)^T\mathbf{M}\mathbf{M}^T\right]\mathbf{O}\mathbf{\Lambda}\mathbf{O}^T \\
&= \mathbf{R}\mathbf{F}.
\end{aligned} \tag{B.19}$$

Next, we use that

$$\begin{aligned}
2\mathbf{M}\mathbf{M}^T &= 2\frac{1}{\sqrt{2}}\begin{bmatrix}\tilde{\mathbf{V}}_\perp \\ \tilde{\mathbf{U}}_\perp\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}\tilde{\mathbf{V}}_\perp^T & \tilde{\mathbf{U}}_\perp^T\end{bmatrix} \\
&= \begin{bmatrix}\tilde{\mathbf{V}}_\perp\tilde{\mathbf{V}}_\perp^T & 0 \\ 0 & \tilde{\mathbf{U}}_\perp\tilde{\mathbf{U}}_\perp^T\end{bmatrix} \\
&= \begin{bmatrix}\tilde{\mathbf{V}}_\perp\tilde{\mathbf{V}}_\perp^T & 0 \\ 0 & \tilde{\mathbf{U}}_\perp\tilde{\mathbf{U}}_\perp^T\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{V}}_\perp\tilde{\mathbf{V}}_\perp^T & 0 \\ 0 & \tilde{\mathbf{U}}_\perp\tilde{\mathbf{U}}_\perp^T\end{bmatrix} \\
&= 4\mathbf{M}\mathbf{M}^T\mathbf{M}\mathbf{M}^T
\end{aligned} \tag{B.20}$$

and the identity

$$\frac{d}{dt}\mathbf{A}^{-1} = -\mathbf{A}^{-1}\left(\frac{d}{dt}\mathbf{A}\right)\mathbf{A}^{-1} \tag{B.21}$$

to solve

$$\begin{aligned}
\tau\frac{d}{dt}\mathbf{C}^{-1} &= -\tau\mathbf{C}^{-1}\left(\frac{d}{dt}\mathbf{C}^{-1}\right)\mathbf{C}^{-1} \\
&= -\tau\mathbf{C}^{-1}\left[\frac{1}{2}\mathbf{Q}(0)^T\left(\mathbf{O}2\frac{1}{\tau}e^{2\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{\Lambda}\mathbf{\Lambda}^{-1}\mathbf{O}^T + 4\frac{1}{\tau}\mathbf{M}\mathbf{M}^T\right)\mathbf{Q}(0)\right]\mathbf{C}^{-1} \\
&= -\mathbf{C}^{-1}\left[\mathbf{Q}(0)^T\left(\mathbf{O}e^{2\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O}^T + 2\mathbf{M}\mathbf{M}^T\right)\mathbf{Q}(0)\right]\mathbf{C}^{-1} \\
&= -\mathbf{C}^{-1}\Big[\mathbf{Q}(0)^T\Big(\mathbf{O}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O}^T\mathbf{O}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O}^T + 2\mathbf{O}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O}^T\mathbf{M}\mathbf{M}^T \\
&\qquad + 2\mathbf{M}\mathbf{M}^T\mathbf{O}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O}^T + 4\mathbf{M}\mathbf{M}^T\mathbf{M}\mathbf{M}^T\Big)\mathbf{Q}(0)\Big]\mathbf{C}^{-1} \\
&= -\mathbf{C}^{-1}\Big[\mathbf{Q}(0)^T\left[\mathbf{O}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O}^T + 2\mathbf{M}\mathbf{M}^T\right]\left[\mathbf{O}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O}^T + 2\mathbf{M}\mathbf{M}^T\right]\mathbf{Q}(0)\Big]\mathbf{C}^{-1} \\
&= -\mathbf{C}^{-1}\mathbf{R}\mathbf{L}\mathbf{C}^{-1}.
\end{aligned} \tag{B.22}$$

Finally, resubstituting Equations B.18, B.19, and B.22 gives

$$\begin{aligned}
\tau\frac{d}{dt}\left(\mathbf{L}\mathbf{C}^{-1}\mathbf{R}\right) &= \tau\left(\frac{d}{dt}\mathbf{L}\right)\mathbf{C}^{-1}\mathbf{R} + \tau\mathbf{L}\mathbf{C}^{-1}\left(\frac{d}{dt}\mathbf{R}\right) + \tau\mathbf{L}\left(\frac{d}{dt}\mathbf{C}^{-1}\right)\mathbf{R} \\
&= \mathbf{F}\mathbf{L}\mathbf{C}^{-1}\mathbf{R} + \mathbf{L}\mathbf{C}^{-1}\mathbf{R}\mathbf{F} - \mathbf{L}\mathbf{C}^{-1}\mathbf{R}\mathbf{L}\mathbf{C}^{-1}\mathbf{R}
\end{aligned}, \tag{B.23}$$

which is identical to the right-hand side of Equation B.12, which proofs that the ansatz is a solution to the matrix Ricatti equation.

The ansatz matches the equation stated in the theorem

In the following, we are going to rewrite the solution to the matrix Ricatti equation

$$\begin{aligned}
\mathbf{Q}\mathbf{Q}^T(t) = &\left[\mathbf{O}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O}^T + 2\mathbf{M}\mathbf{M}^T\right]\mathbf{Q}(0) \\
&\left[\mathbf{I} + \frac{1}{2}\mathbf{Q}(0)^T\left(\mathbf{O}\left(e^{2\mathbf{\Lambda}\frac{t}{\tau}} - \mathbf{I}\right)\mathbf{\Lambda}^{-1}\mathbf{O}^T + 4\frac{t}{\tau}\mathbf{M}\mathbf{M}^T\right)\mathbf{Q}(0)\right]^{-1} \\
&\mathbf{Q}(0)^T\left[\mathbf{O}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O}^T + 2\mathbf{M}\mathbf{M}^T\right]
\end{aligned} \tag{B.24}$$

by means of the compact SVD of the initial network function and the input-output correlation of the task

$$\text{SVD}\left(\mathbf{W}_2(0)\mathbf{W}_1(0)\right) = \mathbf{U}\mathbf{S}\mathbf{V}^T \quad \text{and} \quad \text{SVD}\left(\mathbf{\Sigma}_{yx}\right) = \tilde{\mathbf{U}}\tilde{\mathbf{S}}\tilde{\mathbf{V}}^T. \tag{B.25}$$

We begin by noting that since the initial network weights are zero-balanced (Assumption 4), the first and second layer weights have to share singular values and their respective right and left singular vectors. Specifically, let

$$\text{SVD}\left(\mathbf{W}_2(0)\right) = \mathbf{A}\mathbf{B}\mathbf{C} \quad \text{and} \quad \text{SVD}\left(\mathbf{W}_1(0)\right) = \mathbf{D}\mathbf{E}\mathbf{F} \tag{B.26}$$

be the compact SVD of the initial network weights, then

$$\begin{aligned} &\mathbf{W}_2(0)^T\mathbf{W}_2(0) = \mathbf{W}_1(0)\mathbf{W}_1(0)^T \\ \Leftrightarrow\ &\mathbf{C}\mathbf{B}\mathbf{A}^T\mathbf{A}\mathbf{B}\mathbf{C}^T = \mathbf{D}\mathbf{E}\mathbf{F}\mathbf{F}^T\mathbf{E}\mathbf{D} \\ \Leftrightarrow\ &\mathbf{C}\mathbf{B}^2\mathbf{C}^T = \mathbf{D}\mathbf{E}^2\mathbf{D} \end{aligned} \tag{B.27}$$

from which it follows that

$$\mathbf{C} = \mathbf{D} \quad \text{and} \quad \mathbf{B}^2 = \mathbf{E}^2 \Leftrightarrow \mathbf{B} = \mathbf{E}, \tag{B.28}$$

where the last equality comes from the fact that $\mathbf{B}$ and $\mathbf{E}$ are diagonal matrices with diagonal entries that are larger or equal to zero. Therefore, we can write the initial state as

$$\mathbf{Q}(0) = \begin{bmatrix} \mathbf{W}_1^T(0) \\ \mathbf{W}_2(0) \end{bmatrix} = \begin{bmatrix} \mathbf{V}\sqrt{\mathbf{S}}\mathbf{R}^T \\ \mathbf{U}\sqrt{\mathbf{S}}\mathbf{R}^T \end{bmatrix}, \tag{B.29}$$

where $\mathbf{R}$ is an arbitrary semi-orthonormal matrix of appropriate shape. Then,

$$\begin{aligned}\mathbf{O}e^{\mathbf{\Lambda}\frac{t}{\tau}} &= \frac{1}{\sqrt{2}}\begin{bmatrix}\tilde{\mathbf{V}} & \tilde{\mathbf{V}}\\ \tilde{\mathbf{U}} & -\tilde{\mathbf{U}}\end{bmatrix}\begin{bmatrix}e^{\tilde{\mathbf{S}}\frac{t}{\tau}} & 0\\ 0 & e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\end{bmatrix}\\ &= \frac{1}{\sqrt{2}}\begin{bmatrix}\tilde{\mathbf{V}}e^{\tilde{\mathbf{S}}\frac{t}{\tau}} & \tilde{\mathbf{V}}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\\ \tilde{\mathbf{U}}e^{\tilde{\mathbf{S}}\frac{t}{\tau}} & -\tilde{\mathbf{U}}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\end{bmatrix}\end{aligned} \tag{B.30}$$

and

$$\begin{aligned}\mathbf{O}^T\mathbf{Q}(0) &= \frac{1}{\sqrt{2}}\begin{bmatrix}\tilde{\mathbf{V}} & \tilde{\mathbf{V}}\\ \tilde{\mathbf{U}} & -\tilde{\mathbf{U}}\end{bmatrix}^T\begin{bmatrix}\mathbf{V}\sqrt{\mathbf{S}}\mathbf{R}^T\\ \mathbf{U}\sqrt{\mathbf{S}}\mathbf{R}^T\end{bmatrix}\\ &= \frac{1}{\sqrt{2}}\begin{bmatrix}\tilde{\mathbf{V}}^T\mathbf{V}\sqrt{\mathbf{S}}\mathbf{R}^T + \tilde{\mathbf{U}}^T\mathbf{U}\sqrt{\mathbf{S}}\mathbf{R}^T\\ \tilde{\mathbf{V}}^T\mathbf{V}\sqrt{\mathbf{S}}\mathbf{R}^T - \tilde{\mathbf{U}}^T\mathbf{U}\sqrt{\mathbf{S}}\mathbf{R}^T\end{bmatrix}\\ &= \frac{1}{\sqrt{2}}\begin{bmatrix}\left(\tilde{\mathbf{V}}^T\mathbf{V} + \tilde{\mathbf{U}}^T\mathbf{U}\right)\sqrt{\mathbf{S}}\mathbf{R}^T\\ \left(\tilde{\mathbf{V}}^T\mathbf{V} - \tilde{\mathbf{U}}^T\mathbf{U}\right)\sqrt{\mathbf{S}}\mathbf{R}^T\end{bmatrix},\end{aligned} \tag{B.31}$$

which we combine to

$$\begin{aligned}\mathbf{O}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{O}^T\mathbf{Q}(0) &= \frac{1}{2}\begin{bmatrix}\tilde{\mathbf{V}}e^{\tilde{\mathbf{S}}\frac{t}{\tau}} & \tilde{\mathbf{V}}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\\ \tilde{\mathbf{U}}e^{\tilde{\mathbf{S}}\frac{t}{\tau}} & -\tilde{\mathbf{U}}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{V}}^T\mathbf{V}\sqrt{\mathbf{S}}\mathbf{R}^T + \tilde{\mathbf{U}}^T\mathbf{U}\sqrt{\mathbf{S}}\mathbf{R}^T\\ \tilde{\mathbf{V}}^T\mathbf{V}\sqrt{\mathbf{S}}\mathbf{R}^T - \tilde{\mathbf{U}}^T\mathbf{U}\sqrt{\mathbf{S}}\mathbf{R}^T\end{bmatrix}\\ &= \frac{1}{2}\begin{bmatrix}\tilde{\mathbf{V}}\left(e^{\tilde{\mathbf{S}}\frac{t}{\tau}}\left(\tilde{\mathbf{V}}^T\mathbf{V} + \tilde{\mathbf{U}}^T\mathbf{U}\right) + e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\left(\tilde{\mathbf{V}}^T\mathbf{V} - \tilde{\mathbf{U}}^T\mathbf{U}\right)\right)\sqrt{\mathbf{S}}\mathbf{R}^T\\ \tilde{\mathbf{U}}\left(e^{\tilde{\mathbf{S}}\frac{t}{\tau}}\left(\tilde{\mathbf{V}}^T\mathbf{V} + \tilde{\mathbf{U}}^T\mathbf{U}\right) - e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\left(\tilde{\mathbf{V}}^T\mathbf{V} - \tilde{\mathbf{U}}^T\mathbf{U}\right)\right)\sqrt{\mathbf{S}}\mathbf{R}^T\end{bmatrix}.\end{aligned} \tag{B.32}$$

We continue by rewriting

$$\begin{aligned} 4\mathbf{M}\mathbf{M}^T\mathbf{Q}(0) &= 4\frac{1}{\sqrt{2}}\begin{bmatrix}\tilde{\mathbf{V}}_\perp\\ \tilde{\mathbf{U}}_\perp\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}\tilde{\mathbf{V}}_\perp\\ \tilde{\mathbf{U}}_\perp\end{bmatrix}^T\begin{bmatrix}\mathbf{V}\sqrt{\mathbf{S}}\mathbf{R}^T\\ \mathbf{U}\sqrt{\mathbf{S}}\mathbf{R}^T\end{bmatrix} \\ &= 2\begin{bmatrix}\tilde{\mathbf{V}}_\perp\tilde{\mathbf{V}}_\perp^T & \tilde{\mathbf{V}}_\perp\tilde{\mathbf{U}}_\perp^T\\ \tilde{\mathbf{U}}_\perp\tilde{\mathbf{V}}_\perp^T & \tilde{\mathbf{U}}_\perp\tilde{\mathbf{U}}_\perp^T\end{bmatrix}\begin{bmatrix}\mathbf{V}\sqrt{\mathbf{S}}\mathbf{R}^T\\ \mathbf{U}\sqrt{\mathbf{S}}\mathbf{R}^T\end{bmatrix} \\ &= 2\begin{bmatrix}\tilde{\mathbf{V}}_\perp\tilde{\mathbf{V}}_\perp^T & 0\\ 0 & \tilde{\mathbf{U}}_\perp\tilde{\mathbf{U}}_\perp^T\end{bmatrix}\begin{bmatrix}\mathbf{V}\sqrt{\mathbf{S}}\mathbf{R}^T\\ \mathbf{U}\sqrt{\mathbf{S}}\mathbf{R}^T\end{bmatrix} \\ &= 2\begin{bmatrix}\tilde{\mathbf{V}}_\perp\tilde{\mathbf{V}}_\perp^T\mathbf{V}\sqrt{\mathbf{S}}\mathbf{R}^T\\ \tilde{\mathbf{U}}_\perp\tilde{\mathbf{U}}_\perp^T\mathbf{U}\sqrt{\mathbf{S}}\mathbf{R}^T\end{bmatrix}, \end{aligned} \tag{B.33}$$

and

$$\begin{aligned} \frac{1}{2}\mathbf{Q}(0)^T 4\frac{t}{\tau}\mathbf{M}\mathbf{M}^T\mathbf{Q}(0) &= \frac{t}{\tau}\begin{bmatrix}\mathbf{R}\sqrt{\mathbf{S}}\mathbf{V}^T & \mathbf{R}\sqrt{\mathbf{S}}\mathbf{U}^T\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{V}}_\perp\tilde{\mathbf{V}}_\perp^T\mathbf{V}\sqrt{\mathbf{S}}\mathbf{R}^T\\ \tilde{\mathbf{U}}_\perp\tilde{\mathbf{U}}_\perp^T\mathbf{U}\sqrt{\mathbf{S}}\mathbf{R}^T\end{bmatrix} \\ &= \frac{t}{\tau}\left[\mathbf{R}\sqrt{\mathbf{S}}\left(\mathbf{V}^T\tilde{\mathbf{V}}_\perp\tilde{\mathbf{V}}_\perp^T\mathbf{V} + \mathbf{U}^T\tilde{\mathbf{U}}_\perp\tilde{\mathbf{U}}_\perp^T\mathbf{U}\right)\sqrt{\mathbf{S}}\mathbf{R}^T\right]. \end{aligned} \tag{B.34}$$

Next, we define $\mathbf{B} = \mathbf{U}^T\tilde{\mathbf{U}} + \mathbf{V}^T\tilde{\mathbf{V}}$ and $\mathbf{C} = \mathbf{U}^T\tilde{\mathbf{U}} - \mathbf{V}^T\tilde{\mathbf{V}}$ and rewrite the inverse as

$$\begin{aligned} &\left[\mathbf{I} + \frac{1}{2}\mathbf{Q}(0)^T\mathbf{O}\left(e^{2\mathbf{\Lambda}\frac{t}{\tau}} - \mathbf{I}\right)\mathbf{\Lambda}^{-1}\mathbf{O}^T\mathbf{Q}(0) + 2\frac{t}{\tau}\mathbf{Q}(0)^T\mathbf{M}\mathbf{M}^T\mathbf{Q}(0)\right]^{-1} \\ &= \Bigg[\mathbf{I} + \frac{1}{4}\mathbf{R}\sqrt{\mathbf{S}}\Bigg(\begin{bmatrix}\mathbf{B} & -\mathbf{C}\end{bmatrix}\left(e^{2\mathbf{\Lambda}\frac{t}{\tau}} - \mathbf{I}\right)\mathbf{\Lambda}^{-1}\begin{bmatrix}\mathbf{B}^T\\ -\mathbf{C}^T\end{bmatrix} \\ &\qquad + 4\frac{t}{\tau}\left(\mathbf{V}^T\tilde{\mathbf{V}}_\perp\tilde{\mathbf{V}}_\perp^T\mathbf{V} + \mathbf{U}^T\tilde{\mathbf{U}}_\perp\tilde{\mathbf{U}}_\perp^T\mathbf{U}\right)\Bigg)\sqrt{\mathbf{S}}\mathbf{R}^T\Bigg]^{-1}. \end{aligned} \tag{B.35}$$

Working from the centre out, we have

$$\begin{aligned}\begin{bmatrix}\mathbf{B} & -\mathbf{C}\end{bmatrix}\mathbf{\Lambda}^{-1}\begin{bmatrix}\mathbf{B}^T \\ -\mathbf{C}^T\end{bmatrix} &= \begin{bmatrix}\mathbf{B} & -\mathbf{C}\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{S}}^{-1} & 0 \\ 0 & -\tilde{\mathbf{S}}^{-1}\end{bmatrix}\begin{bmatrix}\mathbf{B}^T \\ -\mathbf{C}^T\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{B} & -\mathbf{C}\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{S}}^{-1}\mathbf{B}^T \\ \tilde{\mathbf{S}}^{-1}\mathbf{C}^T\end{bmatrix} \\ &= \mathbf{B}\tilde{\mathbf{S}}^{-1}\mathbf{B}^T - \mathbf{C}\tilde{\mathbf{S}}^{-1}\mathbf{C}^T\end{aligned} \tag{B.36}$$

and

$$\begin{aligned}\begin{bmatrix}\mathbf{B} & -\mathbf{C}\end{bmatrix}e^{2\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{\Lambda}^{-1}\begin{bmatrix}\mathbf{B}^T \\ -\mathbf{C}^T\end{bmatrix} &= \begin{bmatrix}\mathbf{B} & -\mathbf{C}\end{bmatrix}\begin{bmatrix}e^{2\tilde{\mathbf{S}}\frac{t}{\tau}}\tilde{\mathbf{S}}^{-1} & 0 \\ 0 & -e^{-2\tilde{\mathbf{S}}\frac{t}{\tau}}\tilde{\mathbf{S}}^{-1}\end{bmatrix}\begin{bmatrix}\mathbf{B}^T \\ -\mathbf{C}^T\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{B} & -\mathbf{C}\end{bmatrix}\begin{bmatrix}e^{2\tilde{\mathbf{S}}\frac{t}{\tau}}\tilde{\mathbf{S}}^{-1}\mathbf{B}^T \\ e^{-2\tilde{\mathbf{S}}\frac{t}{\tau}}\tilde{\mathbf{S}}^{-1}\mathbf{C}^T\end{bmatrix} \\ &= \mathbf{B}e^{2\tilde{\mathbf{S}}\frac{t}{\tau}}\tilde{\mathbf{S}}^{-1}\mathbf{B}^T - \mathbf{C}e^{-2\tilde{\mathbf{S}}\frac{t}{\tau}}\tilde{\mathbf{S}}^{-1}\mathbf{C}^T.\end{aligned} \tag{B.37}$$

The inverse can then be written as

$$\begin{aligned}&\left[\mathbf{I} + \frac{1}{2}\mathbf{Q}(0)^T\mathbf{O}\left(e^{2\mathbf{\Lambda}\frac{t}{\tau}} - \mathbf{I}\right)\mathbf{\Lambda}^{-1}\mathbf{O}^T\mathbf{Q}(0) + 2\frac{t}{\tau}\mathbf{Q}(0)^T\mathbf{M}\mathbf{M}^T\mathbf{Q}(0)\right]^{-1} \\ &= \Bigg[\mathbf{I} + \mathbf{R}\sqrt{\mathbf{S}}\Bigg(\frac{1}{4}\mathbf{B}\left(e^{2\tilde{\mathbf{S}}\frac{t}{\tau}} - \mathbf{I}\right)\tilde{\mathbf{S}}^{-1}\mathbf{B}^T - \frac{1}{4}\mathbf{C}\left(e^{-2\tilde{\mathbf{S}}\frac{t}{\tau}} - \mathbf{I}\right)\tilde{\mathbf{S}}^{-1}\mathbf{C}^T \\ &\qquad + \frac{t}{\tau}\left(\mathbf{V}^T\tilde{\mathbf{V}}_\perp\tilde{\mathbf{V}}_\perp^T\mathbf{V} + \mathbf{U}^T\tilde{\mathbf{U}}_\perp\tilde{\mathbf{U}}_\perp^T\mathbf{U}\right)\Bigg)\sqrt{\mathbf{S}}\mathbf{R}^T\Bigg]^{-1}\end{aligned} \tag{B.38}$$

Next, we substitute Equations B.32, B.33 and B.38 into the solution and use that $\mathbf{B}$ is non-singular (Assumption 8), $\mathbf{R}^{-1} = \mathbf{R}^T$, $\mathbf{R}^T\mathbf{R} = \mathbf{I}$, and the identities

$$\mathbf{A}\mathbf{B}^{-1} = \left(\mathbf{B}\mathbf{A}^{-1}\right)^{-1} \quad \text{and} \quad \mathbf{A}^{-1}\mathbf{B} = \left(\mathbf{B}^{-1}\mathbf{A}\right)^{-1} \tag{B.39}$$

to get

$$
\begin{aligned}
\mathbf{Q}\mathbf{Q}^T(t) =& \frac{1}{2}\begin{bmatrix}\left(\tilde{\mathbf{V}}\left(e^{\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{B}-e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{C}\right)+2\tilde{\mathbf{V}}_\perp\tilde{\mathbf{V}}_\perp^T\mathbf{V}\right)\sqrt{\mathbf{S}}\mathbf{R}^T\\ \left(\tilde{\mathbf{U}}\left(e^{\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{B}+e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{C}\right)+2\tilde{\mathbf{U}}_\perp\tilde{\mathbf{U}}_\perp^T\mathbf{U}\right)\sqrt{\mathbf{S}}\mathbf{R}^T\end{bmatrix}\\
&\left[\mathbf{I}+\mathbf{R}\sqrt{\mathbf{S}}\left(\frac{1}{4}\mathbf{B}\left(e^{2\tilde{\mathbf{S}}\frac{t}{\tau}}-\mathbf{I}\right)\tilde{\mathbf{S}}^{-1}\mathbf{B}^T-\frac{1}{4}\mathbf{C}\left(e^{-2\tilde{\mathbf{S}}\frac{t}{\tau}}-\mathbf{I}\right)\tilde{\mathbf{S}}^{-1}\mathbf{C}^T\right.\right.\\
&\left.\left.+\frac{t}{\tau}\left(\mathbf{V}^T\tilde{\mathbf{V}}_\perp\tilde{\mathbf{V}}_\perp^T\mathbf{V}+\mathbf{U}^T\tilde{\mathbf{U}}_\perp\tilde{\mathbf{U}}_\perp^T\mathbf{U}\right)\right)\sqrt{\mathbf{S}}\mathbf{R}^T\right]^{-1}\\
&\frac{1}{2}\begin{bmatrix}\left(\tilde{\mathbf{V}}\left(e^{\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{B}-e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{C}\right)+2\tilde{\mathbf{V}}_\perp\tilde{\mathbf{V}}_\perp^T\mathbf{V}\right)\sqrt{\mathbf{S}}\mathbf{R}^T\\ \left(\tilde{\mathbf{U}}\left(e^{\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{B}+e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{C}\right)+2\tilde{\mathbf{U}}_\perp\tilde{\mathbf{U}}_\perp^T\mathbf{U}\right)\sqrt{\mathbf{S}}\mathbf{R}^T\end{bmatrix}^T\\
=&\frac{1}{2}\begin{bmatrix}\tilde{\mathbf{V}}\left(e^{\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{B}-e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{C}\right)+2\tilde{\mathbf{V}}_\perp\tilde{\mathbf{V}}_\perp^T\mathbf{V}\\ \tilde{\mathbf{U}}\left(e^{\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{B}+e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{C}\right)+2\tilde{\mathbf{U}}_\perp\tilde{\mathbf{U}}_\perp^T\mathbf{U}\end{bmatrix}\\
&\left[\mathbf{S}^{-1}+\frac{1}{4}\mathbf{B}\left(e^{2\tilde{\mathbf{S}}\frac{t}{\tau}}-\mathbf{I}\right)\tilde{\mathbf{S}}^{-1}\mathbf{B}^T-\frac{1}{4}\mathbf{C}\left(e^{-2\tilde{\mathbf{S}}\frac{t}{\tau}}-\mathbf{I}\right)\tilde{\mathbf{S}}^{-1}\mathbf{C}^T\right.\\
&\left.+\frac{t}{\tau}\left(\mathbf{V}^T\tilde{\mathbf{V}}_\perp\tilde{\mathbf{V}}_\perp^T\mathbf{V}+\mathbf{U}^T\tilde{\mathbf{U}}_\perp\tilde{\mathbf{U}}_\perp^T\mathbf{U}\right)\right]^{-1} \qquad \text{(B.40)}\\
&\frac{1}{2}\begin{bmatrix}\tilde{\mathbf{V}}\left(e^{\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{B}-e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{C}\right)+2\tilde{\mathbf{V}}_\perp\tilde{\mathbf{V}}_\perp^T\mathbf{V}\\ \tilde{\mathbf{U}}\left(e^{\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{B}+e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{C}\right)+2\tilde{\mathbf{U}}_\perp\tilde{\mathbf{U}}_\perp^T\mathbf{U}\end{bmatrix}^T\\
=&\begin{bmatrix}\tilde{\mathbf{V}}\left(\mathbf{I}-e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{C}^T\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\right)+2\tilde{\mathbf{V}}_\perp\tilde{\mathbf{V}}_\perp^T\mathbf{V}\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\\ \tilde{\mathbf{U}}\left(\mathbf{I}+e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{C}^T\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\right)+2\tilde{\mathbf{U}}_\perp\tilde{\mathbf{U}}_\perp^T\mathbf{U}\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\end{bmatrix}\\
&\left[4e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{B}^{-1}\mathbf{S}^{-1}\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}+\left(\mathbf{I}-e^{-2\tilde{\mathbf{S}}\frac{t}{\tau}}\right)\tilde{\mathbf{S}}^{-1}\right.\\
&-e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{B}^{-1}\mathbf{C}\left(e^{-2\tilde{\mathbf{S}}\frac{t}{\tau}}-\mathbf{I}\right)\tilde{\mathbf{S}}^{-1}\mathbf{C}^T(\mathbf{B}^T)^{-1}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\\
&\left.+4\frac{t}{\tau}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{B}^{-1}\left(\mathbf{V}^T\tilde{\mathbf{V}}_\perp\tilde{\mathbf{V}}_\perp^T\mathbf{V}+\mathbf{U}^T\tilde{\mathbf{U}}_\perp\tilde{\mathbf{U}}_\perp^T\mathbf{U}\right)\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\right]^{-1}\\
&\begin{bmatrix}\tilde{\mathbf{V}}\left(\mathbf{I}-e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{C}^T\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\right)+2\tilde{\mathbf{V}}_\perp\tilde{\mathbf{V}}_\perp^T\mathbf{V}\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\\ \tilde{\mathbf{U}}\left(\mathbf{I}+e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{C}^T\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\right)+2\tilde{\mathbf{U}}_\perp\tilde{\mathbf{U}}_\perp^T\mathbf{U}\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\end{bmatrix}^T.
\end{aligned}
$$

Finally, we substitute $\tilde{\mathbf{V}}_\perp \tilde{\mathbf{V}}_\perp^T = \mathbf{I} - \tilde{\mathbf{V}}\tilde{\mathbf{V}}^T$ and $\tilde{\mathbf{U}}_\perp \tilde{\mathbf{U}}_\perp^T = \mathbf{I} - \tilde{\mathbf{U}}\tilde{\mathbf{U}}^T$ to get the equation as stated in Theorem 4.1

$$\begin{aligned}
\mathbf{Q}\mathbf{Q}^T(t) = &\begin{bmatrix} \tilde{\mathbf{V}}\left(\mathbf{I} - e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{C}^T\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\right) + 2\left(\mathbf{I} - \tilde{\mathbf{V}}\tilde{\mathbf{V}}^T\right)\mathbf{V}\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}} \\ \tilde{\mathbf{U}}\left(\mathbf{I} + e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{C}^T\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\right) + 2\left(\mathbf{I} - \tilde{\mathbf{U}}\tilde{\mathbf{U}}^T\right)\mathbf{U}\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}} \end{bmatrix} \\
&\Bigg[4e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{B}^{-1}\mathbf{S}^{-1}\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}} + \left(\mathbf{I} - e^{-2\tilde{\mathbf{S}}\frac{t}{\tau}}\right)\tilde{\mathbf{S}}^{-1} \\
&- e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{B}^{-1}\mathbf{C}\left(e^{-2\tilde{\mathbf{S}}\frac{t}{\tau}} - \mathbf{I}\right)\tilde{\mathbf{S}}^{-1}\mathbf{C}^T(\mathbf{B}^T)^{-1}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}} \\
&+ 4\frac{t}{\tau}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{B}^{-1}\left(2\mathbf{I} - \mathbf{V}^T\tilde{\mathbf{V}}\tilde{\mathbf{V}}^T\mathbf{V} - \mathbf{U}^T\tilde{\mathbf{U}}\tilde{\mathbf{U}}^T\mathbf{U}\right)\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\Bigg]^{-1} \\
&\begin{bmatrix} \tilde{\mathbf{V}}\left(\mathbf{I} - e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{C}^T\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\right) + 2\left(\mathbf{I} - \tilde{\mathbf{V}}\tilde{\mathbf{V}}^T\right)\mathbf{V}\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}} \\ \tilde{\mathbf{U}}\left(\mathbf{I} + e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\mathbf{C}^T\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}\right) + 2\left(\mathbf{I} - \tilde{\mathbf{U}}\tilde{\mathbf{U}}^T\right)\mathbf{U}\mathbf{B}^{-T}e^{-\tilde{\mathbf{S}}\frac{t}{\tau}} \end{bmatrix}^T.
\end{aligned} \tag{B.41}$$

□

B.2 Representational similarity and neural tangent kernel

B.2.1 Representational similarity matrices

The temporal evolution of the RSM (Kriegeskorte, Mur, & Bandettini, 2008) of the hidden layer calculated from the inputs is

$$\begin{aligned}
\mathrm{RSM}(t) &= \mathbf{H}^T(t)\mathbf{H}(t) \\
&= \left(\mathbf{W}_1(t)\mathbf{X}\right)^T \mathbf{W}_1(t)\mathbf{X} \\
&= \mathbf{X}^T(\mathbf{W}_1^T\mathbf{W}_1)(t)\mathbf{X}.
\end{aligned} \tag{B.42}$$

B.2.2 Finite-width neural tangent kernel

In the following we are going to derive the temporal dynamics of the neural tangent kernel (Jacot et al., 2018) for a finite-width two-layer linear network

$$\mathrm{NTK}(t) = \left(\nabla_\theta f \nabla_\theta f^T\right)(t), \tag{B.43}$$

where f denotes the vectorised network function and ∇_θ is the gradient with respect to the model's parameters. To this end, we first derive the network function's

$$F_t = (\mathbf{W}_2\mathbf{W}_1)(t)\mathbf{X} \tag{B.44}$$

discrete time gradient descent dynamics

$$\begin{aligned} F_{t+1} &= \left(\mathbf{W}_2(t) - \eta\frac{\partial\mathcal{L}(t)}{\partial\mathbf{W}_2}\right)\left(\mathbf{W}_1(t) - \eta\frac{\partial\mathcal{L}(t)}{\partial\mathbf{W}_1}\right)\mathbf{X} \\ &= (\mathbf{W}_2\mathbf{W}_1)(t)\mathbf{X} - \eta\left(\mathbf{W}_2(t)\frac{\partial\mathcal{L}(t)}{\partial\mathbf{W}_1} + \frac{\partial\mathcal{L}(t)}{\partial\mathbf{W}_2}\mathbf{W}_1(t) - \eta\frac{\partial\mathcal{L}(t)}{\partial\mathbf{W}_2}\frac{\partial\mathcal{L}(t)}{\partial\mathbf{W}_1}\right)\mathbf{X}. \end{aligned} \tag{B.45}$$

The network's gradient flow can then be derived from

$$\frac{F_{t+1} - F_t}{\eta} = -\left(\mathbf{W}_2(t)\frac{\partial\mathcal{L}(t)}{\partial\mathbf{W}_1} + \frac{\partial\mathcal{L}(t)}{\partial\mathbf{W}_2}\mathbf{W}_1(t) - \eta\frac{\partial\mathcal{L}(t)}{\partial\mathbf{W}_2}\frac{\partial\mathcal{L}(t)}{\partial\mathbf{W}_1}\right)\mathbf{X} \tag{B.46}$$

as

$$\frac{d}{dt}F = \lim_{\eta\to 0}\frac{F_{t+1} - F_t}{\eta} = -\left(\mathbf{W}_2(t)\frac{\partial\mathcal{L}(t)}{\partial\mathbf{W}_1} + \frac{\partial\mathcal{L}(t)}{\partial\mathbf{W}_2}\mathbf{W}_1(t)\right)\mathbf{X}. \tag{B.47}$$

Substituting the partial derivatives

$$\frac{\partial\mathcal{L}(t)}{\partial\mathbf{W}_2} = ((\mathbf{W}_2\mathbf{W}_1)(t)\mathbf{X} - \mathbf{Y})\mathbf{X}^T\mathbf{W}_1^T(t) \tag{B.48}$$

and

$$\frac{\partial\mathcal{L}(t)}{\partial\mathbf{W}_1} = \mathbf{W}_2^T(t)((\mathbf{W}_2\mathbf{W}_1)(t)\mathbf{X} - \mathbf{Y})\mathbf{X}^T \tag{B.49}$$

then results in

$$\begin{aligned}\frac{d}{dt}F = &-(\mathbf{W}_2\mathbf{W}_2^T)(t)((\mathbf{W}_2\mathbf{W}_1)(t)\mathbf{X} - \mathbf{Y})\mathbf{X}^T\mathbf{X} \\ &- ((\mathbf{W}_2\mathbf{W}_1)(t)\mathbf{X} - \mathbf{Y})\mathbf{X}^T(\mathbf{W}_1^T\mathbf{W}_1)(t)\mathbf{X}.\end{aligned} \tag{B.50}$$

Finally, vectorising both sides of the equation, denoting $\mathrm{vec}_r(F)$ as f, and using the Kronecker matrix vector product identity

$$\mathrm{vec}_r(\mathbf{ABC}) = (\mathbf{A} \otimes \mathbf{C}^T)\,\mathrm{vec}_r(\mathbf{B}) \tag{B.51}$$

gives

$$\begin{aligned}\frac{d}{dt}f &= -\left((\mathbf{W}_2\mathbf{W}_2^T)(t) \otimes \mathbf{X}^T\mathbf{X} + \mathbf{I} \otimes \mathbf{X}^T(\mathbf{W}_1^T\mathbf{W}_1)(t)\mathbf{X}\right)\mathrm{vec}_r((\mathbf{W}_2\mathbf{W}_1)(t)\mathbf{X} - \mathbf{Y}) \\ &= -\left(\left[\mathbf{W}_2(t) \otimes \mathbf{X}^T, \mathbf{I} \otimes \mathbf{X}^T\mathbf{W}_1^T(t)\right]\left[\mathbf{W}_2(t) \otimes \mathbf{X}^T, \mathbf{I} \otimes \mathbf{X}^T\mathbf{W}_1^T(t)\right]^T\right)\mathrm{vec}_r\left(\frac{\partial\mathcal{L}(t)}{\partial F}\right) \\ &= -\left(\left[\nabla_{\mathbf{w}_1}f, \nabla_{\mathbf{w}_2}f\right]\left[\nabla_{\mathbf{w}_1}f, \nabla_{\mathbf{w}_2}f\right]^T\right)\frac{\partial\mathcal{L}(t)}{\partial f} \\ &= -\left(\nabla_\theta f \nabla_\theta f^T\right)\frac{\partial\mathcal{L}(t)}{\partial f},\end{aligned} \tag{B.52}$$

where $[\mathbf{a}, \mathbf{b}]$ denotes row-wise concatenation. From this it follows, that the temporal evolution of the neural tangent kernel for a finite-width two-layer neural network is

$$\mathrm{NTK}(t) = \mathbf{I}_{N_o} \otimes \mathbf{X}^T(\mathbf{W}_1^T\mathbf{W}_1)(t)\mathbf{X} + (\mathbf{W}_2\mathbf{W}_2^T)(t) \otimes \mathbf{X}^T\mathbf{X}. \tag{B.53}$$

B.3 Steady state solution

Proof of Theorem 4.2. As the solution presented in Theorem 4.1 only contains time dependent terms of the form $e^{-\tilde{\mathbf{S}}\frac{t}{\tau}}$. Since $\tilde{\mathbf{S}}$ is a diagonal matrix with entries larger zero, it follow that

$$\lim_{t\to\infty} e^{-\tilde{\mathbf{S}}\frac{t}{\tau}} = \mathbf{0} \tag{B.54}$$

and therefore that

$$\begin{aligned}\lim_{t\to\infty}\mathbf{Q}\mathbf{Q}^T(t) &= \lim_{t\to\infty}\begin{bmatrix}\mathbf{W}_1^T\mathbf{W}_1(t) & \mathbf{W}_1^T\mathbf{W}_2^T(t)\\ \mathbf{W}_2\mathbf{W}_1(t) & \mathbf{W}_2^T\mathbf{W}_2(t)\end{bmatrix}\\ &= \begin{bmatrix}\tilde{\mathbf{V}}\\ \tilde{\mathbf{U}}\end{bmatrix}\left[\tilde{\mathbf{S}}^{-1}\right]^{-1}\begin{bmatrix}\tilde{\mathbf{V}}^T & \tilde{\mathbf{U}}^T\end{bmatrix} \qquad \text{(B.55)}\\ &= \begin{bmatrix}\tilde{\mathbf{V}}\tilde{\mathbf{S}}\tilde{\mathbf{V}}^T & \tilde{\mathbf{V}}\tilde{\mathbf{S}}\tilde{\mathbf{U}}^T\\ \tilde{\mathbf{U}}\tilde{\mathbf{S}}\tilde{\mathbf{V}}^T & \tilde{\mathbf{U}}\tilde{\mathbf{S}}\tilde{\mathbf{U}}^T\end{bmatrix}.\end{aligned}$$

□

B.4 Forgetting at convergence

Theorem B.1. *Let $\mathcal{T}^i$ and $\mathcal{T}^j$ denote two supervised learning tasks with corresponding input-output correlations $\mathbf{\Sigma}^i_{yx}$ and $\mathbf{\Sigma}^j_{yx}$. Under the assumptions and in accordance with Theorem 4.2 assume that the network function has converged to the global optimum of task j, i.e.,*

$$\mathbf{W}_2\mathbf{W}_1 = \mathbf{\Sigma}^j_{yx}. \qquad \text{(B.56)}$$

Then, the mean squared error on task i is

$$\mathcal{L}^i_{MSE} = \frac{1}{2}\left|\left|\mathbf{\Sigma}^j_{yx} - \mathbf{\Sigma}^i_{yx}\right|\right|^2_F + c, \qquad \text{(B.57)}$$

with $c = -\frac{1}{2}\operatorname{Tr}(\mathbf{\Sigma}^i_{yx}\mathbf{\Sigma}^{iT}_{yx}) + \frac{1}{2}\operatorname{Tr}\left(\mathbf{\Sigma}^i_{yy}\right)$.

Proof of Theorem B.1. Given a supervise learning task $\mathcal{T}^i$ with P^i training pairs and corresponding inputs $\mathbf{X}^i$ and target outputs $\mathbf{Y}^i$, using the assumption that the network function has converged, i.e., $\mathbf{W}_2\mathbf{W}_1 = \mathbf{\Sigma}^j_{yx}$, that $||\mathbf{A}||^2_F = \operatorname{Tr}(\mathbf{A}\mathbf{A}^T)$, that $\operatorname{Tr}(\mathbf{A}+\mathbf{B}) = \operatorname{Tr}(\mathbf{A}) + \operatorname{Tr}(\mathbf{B})$ and the assumption of whitened inputs from

Theorem 4.2, we derive

$$\begin{aligned}
\mathcal{L}^i_{\text{MSE}} &= \frac{1}{2P^i} \left|\left|\mathbf{W}_2\mathbf{W}_1\mathbf{X}^i - \mathbf{Y}^i\right|\right|_F^2 \\
&= \frac{1}{2P^i} \left|\left|\mathbf{\Sigma}^j_{yx}\mathbf{X}^i - \mathbf{Y}^i\right|\right|_F^2 \\
&= \frac{1}{2P^i} \operatorname{Tr}\left((\mathbf{\Sigma}^j_{yx}\mathbf{X}^i - \mathbf{Y}^i)(\mathbf{\Sigma}^j_{yx}\mathbf{X}^i - \mathbf{Y}^i)^T\right) \\
&= \frac{1}{2P^i} \operatorname{Tr}\left(\mathbf{\Sigma}^j_{yx}\mathbf{X}^i\mathbf{X}^{iT}\mathbf{\Sigma}^{jT}_{yx}\right) - \frac{1}{P^i} \operatorname{Tr}\left(\mathbf{\Sigma}^j_{yx}\mathbf{X}^i\mathbf{Y}^{iT}\right) + \frac{1}{2P^i} \operatorname{Tr}\left(\mathbf{Y}^i\mathbf{Y}^{iT}\right) \\
&= \frac{1}{2} \operatorname{Tr}\left(\mathbf{\Sigma}^j_{yx}\mathbf{\Sigma}^{jT}_{yx}\right) - \operatorname{Tr}\left(\mathbf{\Sigma}^j_{yx}\mathbf{\Sigma}^{iT}_{yx}\right) + \frac{1}{2} \operatorname{Tr}\left(\mathbf{\Sigma}^i_{yy}\right) \\
&= \frac{1}{2} \operatorname{Tr}\left(\mathbf{\Sigma}^j_{yx}\mathbf{\Sigma}^{jT}_{yx} - 2\mathbf{\Sigma}^j_{yx}\mathbf{\Sigma}^{iT}_{yx} + \mathbf{\Sigma}^i_{yx}\mathbf{\Sigma}^{iT}_{yx} - \mathbf{\Sigma}^i_{yx}\mathbf{\Sigma}^{iT}_{yx}\right) + \frac{1}{2} \operatorname{Tr}\left(\mathbf{\Sigma}^i_{yy}\right) \\
&= \frac{1}{2} \operatorname{Tr}\left((\mathbf{\Sigma}^j_{yx} - \mathbf{\Sigma}^i_{yx})(\mathbf{\Sigma}^j_{yx} - \mathbf{\Sigma}^i_{yx})^T\right) - \frac{1}{2} \operatorname{Tr}(\mathbf{\Sigma}^i_{yx}\mathbf{\Sigma}^{iT}_{yx}) + \frac{1}{2} \operatorname{Tr}\left(\mathbf{\Sigma}^i_{yy}\right) \\
&= \frac{1}{2} \left|\left|\mathbf{\Sigma}^j_{yx} - \mathbf{\Sigma}^i_{yx}\right|\right|_F^2 \underbrace{- \frac{1}{2} \operatorname{Tr}(\mathbf{\Sigma}^i_{yx}\mathbf{\Sigma}^{iT}_{yx}) + \frac{1}{2} \operatorname{Tr}\left(\mathbf{\Sigma}^i_{yy}\right)}_{c}
\end{aligned} \tag{B.58}$$

□

Corollary B.1.1. *The relative change of loss in task $\mathcal{T}^i$, when training on task $\mathcal{T}^k$ to convergence, after having previously trained the network on task $\mathcal{T}^j$ to convergence is*

$$\mathcal{F}^i_{j\to k} = \frac{1}{2} \left|\left|\mathbf{\Sigma}^k_{yx} - \mathbf{\Sigma}^i_{yx}\right|\right|_F^2 - \frac{1}{2} \left|\left|\mathbf{\Sigma}^j_{yx} - \mathbf{\Sigma}^i_{yx}\right|\right|_F^2 \tag{B.59}$$

Proof of Corollary B.1.1. In accordance with Theorem B.1, the loss of task $\mathcal{T}^i$ after training to convergence on task $\mathcal{T}^k$ and $\mathcal{T}^j$ are

$$\frac{1}{2} \left|\left|\mathbf{\Sigma}^k_{yx} - \mathbf{\Sigma}^i_{yx}\right|\right|_F^2 + c \quad \text{and} \quad \frac{1}{2} \left|\left|\mathbf{\Sigma}^j_{yx} - \mathbf{\Sigma}^i_{yx}\right|\right|_F^2 + c \tag{B.60}$$

respectively. The relative change is then

$$\begin{aligned}
\mathcal{F}^i_{j\to k} &= \frac{1}{2} \left|\left|\mathbf{\Sigma}^k_{yx} - \mathbf{\Sigma}^i_{yx}\right|\right|_F^2 + c - \left(\frac{1}{2} \left|\left|\mathbf{\Sigma}^j_{yx} - \mathbf{\Sigma}^i_{yx}\right|\right|_F^2 + c\right) \\
&= \frac{1}{2} \left|\left|\mathbf{\Sigma}^k_{yx} - \mathbf{\Sigma}^i_{yx}\right|\right|_F^2 - \frac{1}{2} \left|\left|\mathbf{\Sigma}^j_{yx} - \mathbf{\Sigma}^i_{yx}\right|\right|_F^2 .
\end{aligned} \tag{B.61}$$

□

Appendix to Chapter 5

C.1 Optimal compressed replay

Theorem C.1. *Given a set of $K-1$ previous training pairs $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1,\dots,K-1}$ and a current training pair $(\mathbf{x}_k, \mathbf{y}_k)$, let*

$$\text{SVD}\left(\mathbf{\Sigma}_{yx}^{K-1}\left(\mathbf{\Sigma}_{xx}^{K-1}\right)^{+}\right) = \mathbf{U}\mathbf{S}\mathbf{V}^T \tag{C.1}$$

denote the compact SVD of the minimum-norm solution over the previous training pairs (Theorem A.3). Further, let r_{yx}^{K-1} denote the rank of the input-output correlation matrix $\mathbf{\Sigma}_{yx}^{K-1}$ and s_i, $\mathbf{u}_i$, and $\mathbf{v}_i$ denote the i-th singular value, and i-th left and right singular vector respectively. Then, minimising

$$\mathcal{L}_{MSE} = \frac{1}{2K}||\mathbf{W}_2\mathbf{W}_1\mathbf{x}_k - \mathbf{y}_k||_2^2 + \frac{1}{2K}\sum_{i=1}^{r_{yx}^{K-1}}||\mathbf{W}_2\mathbf{W}_1\tilde{\mathbf{x}}_i - \tilde{\mathbf{y}}_i||_2^2, \tag{C.2}$$

with

$$\tilde{\mathbf{x}}_i = \mathbf{v}_i \quad \textit{and} \quad \tilde{\mathbf{y}}_i = \mathbf{u}_i s_i \tag{C.3}$$

is identical to minimising the mean squared error over the original training data.

Proof of Theorem C.1. In accordance with Theorem A.2, and using that $(\mathbf{AB})^+ = \mathbf{B}^+\mathbf{A}^+$ if $\mathbf{A} = \mathbf{B}^T$ (Greville, 1966), we can rewrite the least-squares solution to the

mean squared error for all previous training pairs as

$$\begin{aligned}\mathbf{\Omega}_2\mathbf{\Omega}_1 &= \mathbf{\Sigma}_{yx}\mathbf{\Sigma}_{xx}^{+} \\ &= \frac{1}{K-1}\left(\mathbf{Y}\mathbf{X}^T\right)\left(\frac{1}{K-1}\mathbf{X}\mathbf{X}^T\right)^{+} \\ &= \mathbf{Y}\mathbf{X}^T\mathbf{X}^{+T}\mathbf{X} \\ &= \mathbf{Y}\mathbf{X}^{+}.\end{aligned} \tag{C.4}$$

Using the matrix approximation lemma (Eckart & Young, 1936), it follows that the full-rank approximation of the solution is

$$\sum_{i}^{r_{yx}^{K-1}} \mathbf{u}_i s_i \mathbf{v}_i T, \tag{C.5}$$

which can not be further compressed and therefore that we can substitute

$$\tilde{\mathbf{x}}_i = \mathbf{v}_i \quad \text{and} \quad \tilde{\mathbf{y}}_i = \mathbf{u}_i s_i. \tag{C.6}$$

□

C.2 Bayesian approach

Theorem C.2. *Let* $\mathbf{G}$ *denote the commutation matrix that transforms* $\mathrm{vec}(\mathbf{W}_2)$ *to* $\mathrm{vec}(\mathbf{W}_2^T)$*, then, given a two-layer linear neural network with weight matrices* $\mathbf{W}_1$ *and* $\mathbf{W}_2$ *and* K *training pairs* $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1,\dots,K}$*, the Hessian matrix of the mean*

squared error loss is

$$\mathbf{H}_K = \begin{bmatrix} \boldsymbol{\Sigma}_{xx}^K \otimes \mathbf{W}_2^T\mathbf{W}_2 \\ \mathbf{G}^T\Big(\big(\mathbf{W}_2\mathbf{W}_1\boldsymbol{\Sigma}_{xx}^K - \boldsymbol{\Sigma}_{yx}^K\big) \otimes \mathbf{I}\Big) + \big(\mathbf{W}_1\boldsymbol{\Sigma}_{xx}^K \otimes \mathbf{W}_2\big) \\ \Big(\big(\boldsymbol{\Sigma}_{xx}^{KT}\mathbf{W}_1^T\mathbf{W}_2^T - \boldsymbol{\Sigma}_{yx}^{KT}\big) \otimes \mathbf{I}\Big)\mathbf{G} + \big(\boldsymbol{\Sigma}_{xx}^{KT}\mathbf{W}_1^T \otimes \mathbf{W}_2^T\big) \\ \mathbf{W}_1\boldsymbol{\Sigma}_{xx}^K\mathbf{W}_1^T \otimes \mathbf{I} \end{bmatrix}. \tag{C.7}$$

Proof of Theorem C.2. Before deriving the four second-order derivatives of the Hessian matrix

$$\mathbf{H} = \begin{bmatrix} \frac{\partial}{\partial \mathbf{W}_1}\frac{\partial}{\partial \mathbf{W}_1}\mathcal{L}_{\text{MSE}} & \frac{\partial}{\partial \mathbf{W}_2}\frac{\partial}{\partial \mathbf{W}_1}\mathcal{L}_{\text{MSE}} \\ \frac{\partial}{\partial \mathbf{W}_1}\frac{\partial}{\partial \mathbf{W}_2}\mathcal{L}_{\text{MSE}} & \frac{\partial}{\partial \mathbf{W}_2}\frac{\partial}{\partial \mathbf{W}_2}\mathcal{L}_{\text{MSE}} \end{bmatrix}, \tag{C.8}$$

we derive a set of equations that will be required throughout the proof. First, using the Kronecker matrix vector product identity

$$\text{vec}(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})\,\text{vec}(\mathbf{B}), \tag{C.9}$$

we derive the Jacobian of matrix products

$$\begin{aligned} \frac{\partial}{\partial \mathbf{B}}\mathbf{ABC} &= \frac{\partial}{\partial\,\text{vec}(\mathbf{B})}\,\text{vec}(\mathbf{ABC}) \\ &= \frac{\partial}{\partial\,\text{vec}(\mathbf{B})}(\mathbf{C}^T \otimes \mathbf{A})\,\text{vec}(\mathbf{B}) \\ &= \mathbf{C}^T \otimes \mathbf{A}, \end{aligned} \tag{C.10}$$

$$\begin{aligned} \frac{\partial}{\partial \mathbf{A}}\mathbf{AB} &= \frac{\partial}{\partial\,\text{vec}(\mathbf{A})}\,\text{vec}(\mathbf{AB}) \\ &= \frac{\partial}{\partial\,\text{vec}(\mathbf{A})}\,\text{vec}(\mathbf{IAB}) \\ &= \frac{\partial}{\partial\,\text{vec}(\mathbf{A})}(\mathbf{B}^T \otimes \mathbf{I})\,\text{vec}(\mathbf{A}) \\ &= \mathbf{B}^T \otimes \mathbf{I}, \end{aligned} \tag{C.11}$$

and

$$\begin{aligned}\frac{\partial}{\partial \mathbf{B}}\mathbf{AB} &= \frac{\partial}{\partial \operatorname{vec}(\mathbf{B})}\operatorname{vec}(\mathbf{ABI}) \\ &= \frac{\partial}{\partial \operatorname{vec}(\mathbf{B})}(\mathbf{I}\otimes\mathbf{A})\operatorname{vec}(\mathbf{B}) \\ &= \mathbf{I}\otimes\mathbf{A}.\end{aligned} \tag{C.12}$$

Further, let $\mathbf{G}$ denote the commutation matrix, i.e. the matrix that transforms $\operatorname{vec}(\mathbf{A})$ to $\operatorname{vec}(\mathbf{A}^T)$. Then,

$$\begin{aligned}\frac{\partial}{\partial \mathbf{A}}\mathbf{A}^T\mathbf{B} &= \frac{\partial}{\partial \operatorname{vec}(\mathbf{A})}(\mathbf{B}^T\otimes\mathbf{I})\operatorname{vec}(\mathbf{A}^T) \\ &= \frac{\partial}{\partial \operatorname{vec}(\mathbf{A})}(\mathbf{B}^T\otimes\mathbf{I})\mathbf{G}\operatorname{vec}(\mathbf{A}) \\ &= (\mathbf{B}^T\otimes\mathbf{I})\mathbf{G}\end{aligned} \tag{C.13}$$

And finally, using the product rule and Equations C.11-C.13, we have

$$\begin{aligned}\frac{\partial}{\partial \mathbf{C}}\mathbf{C}^T\mathbf{C} &= \frac{\partial}{\partial \operatorname{vec}(\mathbf{A}^T)}\operatorname{vec}(\mathbf{AB}) + \frac{\partial}{\partial \operatorname{vec}(\mathbf{B})}\operatorname{vec}(\mathbf{AB}) \\ &= \frac{\partial}{\partial \operatorname{vec}(\mathbf{A}^T)}(\mathbf{B}^T\otimes\mathbf{I})\operatorname{vec}(\mathbf{A}) + \frac{\partial}{\partial \operatorname{vec}(\mathbf{B})}(\mathbf{I}\otimes\mathbf{A})\operatorname{vec}(\mathbf{B}) \\ &= \frac{\partial}{\partial \operatorname{vec}(\mathbf{C})}(\mathbf{C}^T\otimes\mathbf{I})\operatorname{vec}(\mathbf{C}^T) + \frac{\partial}{\partial \operatorname{vec}(\mathbf{C})}(\mathbf{I}\otimes\mathbf{C}^T)\operatorname{vec}(\mathbf{C}) \\ &= \frac{\partial}{\partial \operatorname{vec}(\mathbf{C})}(\mathbf{C}^T\otimes\mathbf{I})\mathbf{G}\operatorname{vec}(\mathbf{C}) + \frac{\partial}{\partial \operatorname{vec}(\mathbf{C})}(\mathbf{I}\otimes\mathbf{C}^T)\operatorname{vec}(\mathbf{C}) \\ &= (\mathbf{C}^T\otimes\mathbf{I})\mathbf{G} + (\mathbf{I}\otimes\mathbf{C}^T).\end{aligned} \tag{C.14}$$

Note, how we substitute $\mathbf{A} = \mathbf{C}^T$ and $\mathbf{B} = \mathbf{C}$ in the first line of the equation and resubstitute in line three. Given Equations C.10-C.14 we can now derive the second-order derivatives. Given the first order derivatives

$$\frac{\partial}{\partial \mathbf{W}_1}\mathcal{L}_{\text{MSE}} = \mathbf{W}_2^T\left(\mathbf{W}_2\mathbf{W}_1\boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{yx}\right) \tag{C.15}$$

and

$$\frac{\partial}{\partial \mathbf{W}_2}\mathcal{L}_{\text{MSE}} = \left(\mathbf{W}_2\mathbf{W}_1\boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{yx}\right)\mathbf{W}_1^T \tag{C.16}$$

the second-order derivatives can then be derived as

$$\begin{aligned}\frac{\partial}{\partial \mathbf{W}_1}\frac{\partial}{\partial \mathbf{W}_1}\mathcal{L}_{\text{MSE}} &= \frac{\partial}{\partial \mathbf{W}_1}\mathbf{W}_2^T\left(\mathbf{W}_2\mathbf{W}_1\boldsymbol{\Sigma}_{xx}^K - \boldsymbol{\Sigma}_{yx}^K\right)\\ &= \frac{\partial}{\partial \mathbf{W}_1}\mathbf{W}_2^T\mathbf{W}_2\mathbf{W}_1\boldsymbol{\Sigma}_{xx}^K\\ &= \boldsymbol{\Sigma}_{xx}^K \otimes \mathbf{W}_2^T\mathbf{W}_2,\end{aligned} \tag{C.17}$$

$$\begin{aligned}\frac{\partial}{\partial \mathbf{W}_2}\frac{\partial}{\partial \mathbf{W}_2}\mathcal{L}_{\text{MSE}} &= \frac{\partial}{\partial \mathbf{W}_2}\left(\mathbf{W}_2\mathbf{W}_1\boldsymbol{\Sigma}_{xx}^K - \boldsymbol{\Sigma}_{yx}^K\right)\mathbf{W}_1^T\\ &= \frac{\partial}{\partial \mathbf{W}_2}\mathbf{W}_2\mathbf{W}_1\boldsymbol{\Sigma}_{xx}^K\mathbf{W}_1^T\\ &= \mathbf{W}_1\boldsymbol{\Sigma}_{xx}^K\mathbf{W}_1^T \otimes \mathbf{I},\end{aligned} \tag{C.18}$$

$$\begin{aligned}\frac{\partial}{\partial \mathbf{W}_2}\frac{\partial}{\partial \mathbf{W}_1}\mathcal{L} &= \frac{\partial}{\partial \mathbf{W}_2}\mathbf{W}_2^T\left(\mathbf{W}_2\mathbf{W}_1\boldsymbol{\Sigma}_{xx}^K - \boldsymbol{\Sigma}_{yx}^K\right)\\ &= \frac{\partial}{\partial \mathbf{W}_2}\mathbf{W}_2^T\mathbf{W}_2\mathbf{W}_1\boldsymbol{\Sigma}_{xx}^K - \frac{\partial}{\partial \mathbf{W}_2}\mathbf{W}_2^T\boldsymbol{\Sigma}_{yx}^K\\ &= (\boldsymbol{\Sigma}_{xx}^{KT}\mathbf{W}_1^T \otimes \mathbf{I})\left((\mathbf{W}_2^T \otimes \mathbf{I})\mathbf{G} + (\mathbf{I} \otimes \mathbf{W}_2^T)\right) - (\boldsymbol{\Sigma}_{yx}^{KT} \otimes \mathbf{I})\mathbf{G}\\ &= (\boldsymbol{\Sigma}_{xx}^{KT}\mathbf{W}_1^T\mathbf{W}_2^T \otimes \mathbf{I})\mathbf{G} + (\boldsymbol{\Sigma}_{xx}^{KT}\mathbf{W}_1^T \otimes \mathbf{W}_2^T) - (\boldsymbol{\Sigma}_{yx}^{KT} \otimes \mathbf{I})\mathbf{G}\\ &= \left((\boldsymbol{\Sigma}_{xx}^{KT}\mathbf{W}_1^T\mathbf{W}_2^T - \boldsymbol{\Sigma}_{yx}^{KT}) \otimes \mathbf{I}\right)\mathbf{G} + (\boldsymbol{\Sigma}_{xx}^{KT}\mathbf{W}_1^T \otimes \mathbf{W}_2^T),\end{aligned} \tag{C.19}$$

and

$$\begin{aligned}\frac{\partial}{\partial \mathbf{W}_1}\frac{\partial}{\partial \mathbf{W}_2}\mathcal{L} &= \left(\frac{\partial}{\partial \mathbf{W}_2}\frac{\partial}{\partial \mathbf{W}_1}\mathcal{L}\right)^T\\ &= \mathbf{G}^T\Big((\mathbf{W}_2\mathbf{W}_1\boldsymbol{\Sigma}_{xx}^K - \boldsymbol{\Sigma}_{yx}^K) \otimes \mathbf{I}\Big) + (\mathbf{W}_1\boldsymbol{\Sigma}_{xx}^K \otimes \mathbf{W}_2).\end{aligned} \tag{C.20}$$

Substituting the second-order derivatives into the Hessian matrix definition (Equation C.8) then provides the solution as stated in the theorem. □

Theorem C.3. *Under the assumption of convergence to a linear solution on the*

previous $K-1$ training pairs, full Hessian regularisation on the K-th training pair

$$\mathcal{R}_{Bayesian} = (\theta - \theta_{K-1})^T \mathbf{H}_{K-1} (\theta - \theta_{K-1}) \tag{C.21}$$

is gradient equivalent to the following loss function

$$\tilde{\mathcal{R}}_{Bayesian} = \sum_{n=1}^{K-1} ||\mathbf{W}_2^{K-1}\mathbf{W}_1\mathbf{x}_n + \mathbf{W}_2\mathbf{W}_1^{K-1}\mathbf{x}_n - 2\mathbf{y}_n||_2^2. \tag{C.22}$$

Proof of Theorem C.3. In the following, to increase readability, we are going to write $\mathbf{\Omega}_1$ for $\mathbf{W}_1^{K-1}$ and $\mathbf{\Omega}_2$ for $\mathbf{W}_2^{K-1}$. Under the assumption that training on all previous $K-1$ items has converged to a linear solution we have

$$\mathbf{\Omega}_2\mathbf{\Omega}_1\mathbf{\Sigma}_{xx}^{K-1} - \mathbf{\Sigma}_{yx}^{K-1} = 0 \tag{C.23}$$

and therefore the Bayesian approach reduces to

$$\begin{aligned}
\mathcal{R}_{\text{Bayesian}} &= \frac{1}{2} (\theta - \theta_{K-1})^T \mathbf{H}_{K-1} (\theta - \theta_{K-1}) \\
&= \left[\text{vec}(\mathbf{W}_1 - \mathbf{\Omega}_1)^T, \text{vec}(\mathbf{W}_2 - \mathbf{\Omega}_2)^T\right] \\
&\qquad \begin{bmatrix} \mathbf{\Sigma}_{xx}^{K-1} \otimes \mathbf{\Omega}_2^T\mathbf{\Omega}_2 & \mathbf{\Sigma}_{xx}^{K-1}\mathbf{\Omega}_1^T \otimes \mathbf{\Omega}_2^T \\ \mathbf{\Omega}_1\mathbf{\Sigma}_{xx}^{K-1} \otimes \mathbf{\Omega}_2 & \mathbf{\Omega}_1\mathbf{\Sigma}_{xx}^{K-1}\mathbf{\Omega}_1^T \otimes \mathbf{I} \end{bmatrix} \\
&\qquad \left[\text{vec}(\mathbf{W}_1 - \mathbf{\Omega}_1)^T, \text{vec}(\mathbf{W}_2 - \mathbf{\Omega}_2)^T\right]^T \\
&= \text{vec}(\mathbf{W}_1 - \mathbf{\Omega}_1)^T \left(\mathbf{\Sigma}_{xx}^{K-1} \otimes \mathbf{\Omega}_2^T\mathbf{\Omega}_2\right) \text{vec}(\mathbf{W}_1 - \mathbf{\Omega}_1) \\
&\quad + \text{vec}(\mathbf{W}_1 - \mathbf{\Omega}_1)^T \left(\mathbf{\Sigma}_{xx}^{K-1}\mathbf{\Omega}_1^T \otimes \mathbf{\Omega}_2^T\right) \text{vec}(\mathbf{W}_2 - \mathbf{\Omega}_2) \\
&\quad + \text{vec}(\mathbf{W}_2 - \mathbf{\Omega}_2)^T \left(\mathbf{\Omega}_1\mathbf{\Sigma}_{xx}^{K-1} \otimes \mathbf{\Omega}_2\right) \text{vec}(\mathbf{W}_1 - \mathbf{\Omega}_1) \\
&\quad + \text{vec}(\mathbf{W}_2 - \mathbf{\Omega}_2)^T \left(\mathbf{\Omega}_1\mathbf{\Sigma}_{xx}^{K-1}\mathbf{\Omega}_1^T \otimes \mathbf{I}\right) \text{vec}(\mathbf{W}_2 - \mathbf{\Omega}_2).
\end{aligned} \tag{C.24}$$

We continue by calculating the gradients, summand by summand. Using the product rule, noting that the Kronecker product of symmetric matrices is symmetric

and that $\mathbf{A}^T = \mathbf{A}$ if $\mathbf{A}$ is symmetric, we have

$$\begin{aligned}
&\frac{\partial}{\partial \mathbf{W}_1} \operatorname{vec}(\mathbf{W}_1 - \mathbf{\Omega}_1)^T \left(\mathbf{\Sigma}_{xx}^{K-1} \otimes \mathbf{\Omega}_2^T \mathbf{\Omega}_2\right) \operatorname{vec}(\mathbf{W}_1 - \mathbf{\Omega}_1) \\
&\quad = \left(\left(\mathbf{\Sigma}_{xx}^{K-1} \otimes \mathbf{\Omega}_2^T \mathbf{\Omega}_2\right)^T + \left(\mathbf{\Sigma}_{xx}^{K-1} \otimes \mathbf{\Omega}_2^T \mathbf{\Omega}_2\right)\right) \operatorname{vec}(\mathbf{W}_1 - \mathbf{\Omega}_1) \\
&\quad = 2\left(\mathbf{\Sigma}_{xx}^{K-1} \otimes \mathbf{\Omega}_2^T \mathbf{\Omega}_2\right) \operatorname{vec}(\mathbf{W}_1 - \mathbf{\Omega}_1) \\
&\quad = 2\mathbf{\Omega}_2^T \mathbf{\Omega}_2 \left(\mathbf{W}_1 - \mathbf{\Omega}_1\right) \mathbf{\Sigma}_{xx}^{K-1},
\end{aligned} \tag{C.25}$$

$$\begin{aligned}
&\frac{\partial}{\partial \mathbf{W}_1} \operatorname{vec}(\mathbf{W}_1 - \mathbf{\Omega}_1)^T \left(\mathbf{\Sigma}_{xx}^{K-1} \mathbf{\Omega}_1^T \otimes \mathbf{\Omega}_2^T\right) \operatorname{vec}(\mathbf{W}_2 - \mathbf{\Omega}_2) \\
&\quad = \left(\mathbf{\Sigma}_{xx}^{K-1} \mathbf{\Omega}_1^T \otimes \mathbf{\Omega}_2^T\right) \operatorname{vec}(\mathbf{W}_2 - \mathbf{\Omega}_2) \\
&\quad = \mathbf{\Omega}_2^T \left(\mathbf{W}_2 - \mathbf{\Omega}_2\right) \mathbf{\Omega}_1 \mathbf{\Sigma}_{xx}^{K-1},
\end{aligned} \tag{C.26}$$

$$\begin{aligned}
&\frac{\partial}{\partial \mathbf{W}_1} \operatorname{vec}(\mathbf{W}_2 - \mathbf{\Omega}_2)^T \left(\mathbf{\Omega}_1 \mathbf{\Sigma}_{xx}^{K-1} \otimes \mathbf{\Omega}_2\right) \operatorname{vec}(\mathbf{W}_1 - \mathbf{\Omega}_1) \\
&\quad = \left(\mathbf{\Omega}_1 \mathbf{\Sigma}_{xx}^{K-1} \otimes \mathbf{\Omega}_2\right)^T \operatorname{vec}(\mathbf{W}_2 - \mathbf{\Omega}_2) \\
&\quad = \left(\mathbf{\Sigma}_{xx}^{K-1} \mathbf{\Omega}_1^T \otimes \mathbf{\Omega}_2^T\right) \operatorname{vec}(\mathbf{W}_2 - \mathbf{\Omega}_2) \\
&\quad = \mathbf{\Omega}_2^T \left(\mathbf{W}_2 - \mathbf{\Omega}_2\right) \mathbf{\Omega}_1 \mathbf{\Sigma}_{xx}^{K-1},
\end{aligned} \tag{C.27}$$

and

$$\frac{\partial}{\partial \mathbf{W}_1} \operatorname{vec}(\mathbf{W}_2 - \mathbf{\Omega}_2)^T \left(\mathbf{\Omega}_1 \mathbf{\Sigma}_{xx}^{K-1} \mathbf{\Omega}_1^T \otimes \mathbf{I}\right) \operatorname{vec}(\mathbf{W}_2 - \mathbf{\Omega}_2) = 0. \tag{C.28}$$

Combining the four summands then yields

$$\frac{\partial \mathcal{R}_{\text{Bayesian}}}{\partial \mathbf{W}_1} = 2\mathbf{\Omega}_2^T \mathbf{\Omega}_2 \left(\mathbf{W}_1 - \mathbf{\Omega}_1\right) \mathbf{\Sigma}_{xx}^{K-1} + 2\mathbf{\Omega}_2^T \left(\mathbf{W}_2 - \mathbf{\Omega}_2\right) \mathbf{\Omega}_1 \mathbf{\Sigma}_{xx}^{K-1}. \tag{C.29}$$

Respectively, the partial derivatives with respect to $\mathbf{W}_2$ are

$$\frac{\partial}{\partial \mathbf{W}_2} \operatorname{vec}(\mathbf{W}_1 - \mathbf{\Omega}_1)^T \left(\mathbf{\Sigma}_{xx}^{K-1} \otimes \mathbf{\Omega}_2^T \mathbf{\Omega}_2\right) \operatorname{vec}(\mathbf{W}_1 - \mathbf{\Omega}_1) = 0, \tag{C.30}$$

$$\begin{aligned}
&\frac{\partial}{\partial \mathbf{W}_2} \operatorname{vec}(\mathbf{W}_1 - \mathbf{\Omega}_1)^T \left(\mathbf{\Sigma}_{xx}^{K-1} \mathbf{\Omega}_1^T \otimes \mathbf{\Omega}_2^T\right) \operatorname{vec}(\mathbf{W}_2 - \mathbf{\Omega}_2) \\
&= \mathbf{\Omega}_2 (\mathbf{W}_1 - \mathbf{\Omega}_1) \mathbf{\Sigma}_{xx}^{K-1} \mathbf{\Omega}_1^T,
\end{aligned} \tag{C.31}$$

$$\begin{aligned}&\frac{\partial}{\partial \mathbf{W}_2} \operatorname{vec}(\mathbf{W}_2 - \mathbf{\Omega}_2)^T \left(\mathbf{\Omega}_1 \mathbf{\Sigma}_{xx}^{K-1} \otimes \mathbf{\Omega}_2\right) \operatorname{vec}(\mathbf{W}_1 - \mathbf{\Omega}_1) \\ &= \mathbf{\Omega}_2 (\mathbf{W}_1 - \mathbf{\Omega}_1) \mathbf{\Sigma}_{xx}^{K-1} \mathbf{\Omega}_1^T,\end{aligned} \tag{C.32}$$

and

$$\begin{aligned}&\frac{\partial}{\partial \mathbf{W}_2} \operatorname{vec}(\mathbf{W}_2 - \mathbf{\Omega}_2)^T \left(\mathbf{\Omega}_1 \mathbf{\Sigma}_{xx}^{K-1} \mathbf{\Omega}_1^T \otimes \mathbf{I}\right) \operatorname{vec}(\mathbf{W}_2 - \mathbf{\Omega}_2 \\ &= 2(\mathbf{W}_2 - \mathbf{\Omega}_2) \mathbf{\Sigma}_{xx}^{K-1} \mathbf{\Omega}_1^T.\end{aligned} \tag{C.33}$$

Combining the four summands then yields

$$\frac{\partial \mathcal{R}_{\text{Bayesian}}}{\partial \mathbf{W}_2} = 2\mathbf{\Omega}_2 \left(\mathbf{W}_1 - \mathbf{\Omega}_1\right) \mathbf{\Sigma}_{xx}^{K-1} \mathbf{\Omega}_1^T + 2 \left(\mathbf{W}_1 - \mathbf{\Omega}_1\right) \mathbf{\Sigma}_{xx}^{K-1} \mathbf{\Omega}_1^T. \tag{C.34}$$

We continue by calculating the partial derivatives with respect to the proposed alternative loss function

$$\begin{aligned}\frac{\partial \tilde{\mathcal{R}}_{\text{Bayesian}}}{\partial \mathbf{W}_1} &= \frac{\partial}{\partial \mathbf{W}_1} \sum_{n=1}^{K-1} ||\mathbf{W}_2 \mathbf{\Omega}_1 \mathbf{x}_n + \mathbf{\Omega}_2 \mathbf{W}_1 \mathbf{x}_n - 2\mathbf{y}_n|| \\ &= 2 \sum_{n=1}^{K-1} \mathbf{\Omega}_2^T \left(\mathbf{W}_2 \mathbf{\Omega}_1 \mathbf{x}_n + \mathbf{\Omega}_2 \mathbf{W}_1 \mathbf{x}_n - 2\mathbf{y}_n\right) \mathbf{x}_n^T \\ &= 2\mathbf{\Omega}_2^T \mathbf{W}_2 \mathbf{\Omega}_1 \mathbf{\Sigma}_{xx}^{K-1} + 2\mathbf{\Omega}_2^T \mathbf{\Omega}_2 \mathbf{W}_1 \mathbf{\Sigma}_{xx}^{K-1} - 4\mathbf{\Omega}_2^T \mathbf{\Sigma}_{yx}^{K-1} \\ &= 2\mathbf{\Omega}_2^T \mathbf{W}_2 \mathbf{\Omega}_1 \mathbf{\Sigma}_{xx}^{K-1} + 2\mathbf{\Omega}_2^T \mathbf{\Omega}_2 \mathbf{W}_1 \mathbf{\Sigma}_{xx}^{K-1} - 4\mathbf{\Omega}_2^T \mathbf{\Omega}_2 \mathbf{\Omega}_1 \mathbf{\Sigma}_{xx}^{K-1} \\ &= 2\mathbf{\Omega}_2^T \mathbf{\Omega}_2 \left(\mathbf{W}_1 - \mathbf{\Omega}_1\right) \mathbf{\Sigma}_{xx}^{K-1} + 2\mathbf{\Omega}_2^T \left(\mathbf{W}_2 - \mathbf{\Omega}_2\right) \mathbf{\Omega}_1 \mathbf{\Sigma}_{xx}^{K-1},\end{aligned} \tag{C.35}$$

and

$$\begin{aligned}\frac{\partial \tilde{\mathcal{R}}_{\text{Bayesian}}}{\partial \mathbf{W}_2} &= \frac{\partial}{\partial \mathbf{W}_2} \sum_{n=1}^{K-1} ||\mathbf{W}_2 \mathbf{\Omega}_1 \mathbf{x}_n + \mathbf{\Omega}_2 \mathbf{W}_1 \mathbf{x}_n - 2\mathbf{y}_n|| \\ &= 2 \sum_{n=1}^{K-1} \left(\mathbf{W}_2 \mathbf{\Omega}_1 \mathbf{x}_n + \mathbf{\Omega}_2 \mathbf{W}_1 \mathbf{x}_n - 2\mathbf{y}_n\right) \mathbf{x}_n^T \mathbf{\Omega}_1^T \\ &= 2\mathbf{W}_2 \mathbf{\Omega}_1 \mathbf{\Sigma}_{xx}^{K-1} \mathbf{\Omega}_1^T + 2\mathbf{\Omega}_2 \mathbf{W}_1 \mathbf{\Sigma}_{xx}^{K-1} \mathbf{\Omega}_1^T - 4\mathbf{\Sigma}_{yx}^{K-1} \mathbf{\Omega}_1^T \\ &= 2\mathbf{W}_2 \mathbf{\Omega}_1 \mathbf{\Sigma}_{xx}^{K-1} \mathbf{\Omega}_1^T + 2\mathbf{\Omega}_2 \mathbf{W}_1 \mathbf{\Sigma}_{xx}^{K-1} \mathbf{\Omega}_1^T - 4\mathbf{\Omega}_2 \mathbf{\Omega}_1 \mathbf{\Sigma}_{xx}^{k-1} \mathbf{\Omega}_1^T \\ &= 2\mathbf{\Omega}_2^T \mathbf{\Omega}_2 \left(\mathbf{W}_1 - \mathbf{\Omega}_1\right) \mathbf{\Sigma}_{xx}^{K-1} + 2\mathbf{\Omega}_2^T \left(\mathbf{W}_2 - \mathbf{\Omega}_2\right) \mathbf{\Omega}_1 \mathbf{\Sigma}_{xx}^{K-1},\end{aligned} \tag{C.36}$$

which are equivalent to Equation C.29 and Equation C.34 and therefore $\mathcal{R}_{\text{Bayesian}}$ and $\tilde{\mathcal{R}}_{\text{Bayesian}}$ are gradient equivalent. □

Theorem C.4. *Under the assumption of convergence to a minimum-norm solution on the previous $K-1$ training pairs (Theoren A.3), the diagonal Hessian regularisation on the K-th training pair*

$$\mathcal{R}_{Diagonal} = \frac{1}{2}\,\text{diag}\left(\mathbf{H}_{K-1}\right)\left(\theta - \theta_{K-1}\right)^2 \tag{C.37}$$

is gradient equivalent to

$$\begin{aligned}\tilde{\mathcal{R}}_{Diagonal} = \left(\mathbf{W}_1 - \mathbf{\Omega}_1\right)^2 \odot \text{diag}\left(\mathbf{R}\mathbf{S}\mathbf{R}^T\right)\left(\frac{1}{K-1}\sum_{n=1}^{K-1}\mathbf{x}_n^2\right)^T \\ + \left(\mathbf{W}_2 - \mathbf{\Omega}_2\right)^2 \odot \text{diag}\left(\mathbf{I}\right)\left(\frac{1}{K-1}\sum_{n=1}^{K-1}\mathbf{h}_n^2\right)^T,\end{aligned} \tag{C.38}$$

where $\mathbf{R}$ *denotes an arbitrary (semi-)orthonormal matrix and*

$$\text{SVD}\left(\mathbf{\Sigma}_{yx}^{K-1}\left(\mathbf{\Sigma}_{xx}^{K-1}\right)^+\right) = \mathbf{U}\mathbf{S}\mathbf{V}^T. \tag{C.39}$$

Proof of Theorem C.3. Like in the previous proof, to increase readability, we are going to write $\mathbf{\Omega}_1$ for $\mathbf{W}_1^{K-1}$ and $\mathbf{\Omega}_2$ for $\mathbf{W}_2^{K-1}$. From the assumption that training on the $K-1$ previous training pairs has converged to a minimum-norm solution, it follows that

$$\mathbf{\Omega}_2 = \mathbf{U}\sqrt{\mathbf{S}}\mathbf{R}^T \text{ and } \mathbf{\Omega}_1 = \mathbf{R}\sqrt{\mathbf{S}}\mathbf{V}^T \tag{C.40}$$

and further that

$$\mathbf{\Omega}_2\mathbf{\Omega}_1\mathbf{\Sigma}_{xx}^{K-1} - \mathbf{\Sigma}_{yx}^{K-1} = 0. \tag{C.41}$$

Therefore, the diagonal Bayesian approach reduces to

$$\begin{aligned}\mathcal{R}_{\text{Diagonal}} &= \operatorname{diag}\left(\mathbf{H}_{K-1}\right)^T \left(\theta - \theta_{K-1}\right)^2 \\ &= \operatorname{diag}\left(\boldsymbol{\Sigma}_{xx}^{K-1} \otimes \boldsymbol{\Omega}_2^T \boldsymbol{\Omega}_2\right)^T \operatorname{vec}(\mathbf{W}_1 - \boldsymbol{\Omega}_1)^2 \\ &\quad + \operatorname{diag}\left(\boldsymbol{\Omega}_1 \boldsymbol{\Sigma}_{xx}^{K-1} \boldsymbol{\Omega}_1^T \otimes \mathbf{I}\right)^T \operatorname{vec}(\mathbf{W}_2 - \boldsymbol{\Omega}_2)^2\end{aligned} \tag{C.42}$$

with partial derivatives

$$\begin{aligned}&\frac{\partial}{\partial \mathbf{W}_1} \operatorname{diag}\left(\boldsymbol{\Sigma}_{xx}^{K-1} \otimes \boldsymbol{\Omega}_2^T \boldsymbol{\Omega}_2\right)^T \operatorname{vec}(\mathbf{W}_1 - \boldsymbol{\Omega}_1)^2 \\ &= 2\left(\mathbf{W}_1 - \boldsymbol{\Omega}_1\right) \odot \operatorname{diag}\left(\boldsymbol{\Omega}_2^T \boldsymbol{\Omega}_2\right) \operatorname{diag}\left(\boldsymbol{\Sigma}_{xx}^{K-1}\right)^T \\ &= 2\left(\mathbf{W}_1 - \boldsymbol{\Omega}_1\right) \odot \operatorname{diag}\left(\mathbf{R}\mathbf{S}\mathbf{R}^T\right) \left(\frac{1}{K-1} \sum_{n=1}^{K-1} \mathbf{x}_n^2\right)^T\end{aligned} \tag{C.43}$$

and

$$\begin{aligned}&\frac{\partial}{\partial \mathbf{W}_2} \operatorname{diag}\left(\boldsymbol{\Omega}_1 \boldsymbol{\Sigma}_{xx}^{K-1} \boldsymbol{\Omega}_1^T \otimes \mathbf{I}\right)^T \operatorname{vec}(\mathbf{W}_2 - \boldsymbol{\Omega}_2)^2 \\ &= 2\left(\mathbf{W}_2 - \boldsymbol{\Omega}_2\right) \odot \operatorname{diag}\left(\mathbf{I}\right) \left(\frac{1}{K-1} \sum_{n=1}^{K-1} \mathbf{h}_n^2\right)^T .\end{aligned} \tag{C.44}$$

We continue by calculating the partial derivatives with respect to the proposed alternative loss function

$$\begin{aligned}&\frac{\partial}{\partial \mathbf{W}_1} \left(\mathbf{W}_1 - \boldsymbol{\Omega}_1\right)^2 \odot \operatorname{diag}\left(\mathbf{R}\mathbf{S}\mathbf{R}^T\right) \left(\frac{1}{K-1} \sum_{n=1}^{K-1} \mathbf{x}_n^2\right)^T \\ &= 2\left(\mathbf{W}_1 - \boldsymbol{\Omega}_1\right) \odot \operatorname{diag}\left(\boldsymbol{\Omega}_2^T \boldsymbol{\Omega}_2\right) \operatorname{diag}\left(\boldsymbol{\Sigma}_{xx}^{K-1}\right)^T\end{aligned} \tag{C.45}$$

and

$$\begin{aligned}&\frac{\partial}{\partial \mathbf{W}_2} \left(\mathbf{W}_2 - \boldsymbol{\Omega}_2\right)^2 \odot \operatorname{diag}\left(\mathbf{I}\right) \left(\frac{1}{K-1} \sum_{n=1}^{K-1} \mathbf{h}_n^2\right)^T \\ &= 2\left(\mathbf{W}_2 - \boldsymbol{\Omega}_2\right) \odot \operatorname{diag}\left(\mathbf{I}\right) \left(\frac{1}{K-1} \sum_{n=1}^{K-1} \mathbf{h}_n^2\right)^T\end{aligned} \tag{C.46}$$

which are equivalent to Equation C.43 and Equation C.44 and therefore $\mathcal{R}_{\text{Diagonal}}$ and $\tilde{\mathcal{R}}_{\text{Diagonal}}$ are gradient equivalent. □

C.3 Exact categorical learning in single-layer networks

Theorem C.5. *Under the assumption of orthonormal inputs 10, minimising the ℓ_2-loss in a single-layer linear network for input-output pair $(\mathbf{x}_k, \mathbf{y}_k)$ using gradient descent, causes no forgetting of previously acquired knowledge.*

Proof of Theorem C.5. The gradient-descent parameter update for a single input-output pair $(\mathbf{x}_k, \mathbf{y}_k)$ in a single-layer linear network with learning rate η is

$$\Delta \mathbf{W}_1 = -\eta(\mathbf{W}_1 \mathbf{x}_n - \mathbf{y}_n)\mathbf{x}_n^T. \tag{C.47}$$

Then, the loss on all other training pairs

$$\mathcal{L}_{\ell_2} = \frac{1}{2(P-1)} \sum_{i \neq k}^{P} ||\mathbf{W}_1 \mathbf{x}_i - \mathbf{y}_i||_2^2 \tag{C.48}$$

is unaffected by parameter updates for the k-th example

$$\begin{aligned}
\mathcal{L}_{\ell_2} &= \frac{1}{2(P-1)} \sum_{i \neq k}^{P} || \left(\mathbf{W}_1 + \Delta \mathbf{W}_1\right) \mathbf{x}_i - \mathbf{y}_i||_2^2 \\
&= \frac{1}{2(P-1)} \sum_{i \neq k}^{P} || \left(\mathbf{W}_1 - \eta(\mathbf{W}_1 \mathbf{x}_k - \mathbf{y}_k)\mathbf{x}_k^T\right) \mathbf{x}_i - \mathbf{y}_i||_2^2 \\
&= \frac{1}{2(P-1)} \sum_{i \neq k}^{P} || \left(\mathbf{W}_1 \mathbf{x}_i - \eta(\mathbf{W}_1 \mathbf{x}_k - \mathbf{y}_k)\mathbf{x}_k^T \mathbf{x}_i\right) - \mathbf{y}_i||_2^2 \\
&= \frac{1}{2(P-1)} \sum_{i \neq k}^{P} ||\mathbf{W}_1 \mathbf{x}_i - \mathbf{y}_i||_2^2,
\end{aligned} \tag{C.49}$$

where in the last step we used the assumption that inputs are orthonormal 10, i.e., $x_i^T x_j = 1$ for $i = j$ and 0 otherwise. □

C.4 Exact continual learning

Theorem C.6. *Under the assumption of orthonormal inputs (Assumption 10) and that learning converged to a least-squares solution (Theorem A.2) for all previous training pairs* $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1\ldots,K-1}$*, then*

$$\mathcal{L} = \frac{1}{2}||\mathbf{W}_2\mathbf{W}_1\mathbf{x}_k - \mathbf{y}_k||_2^2 + \frac{1}{2}||\mathbf{W}_2\mathbf{W}_1 - \mathbf{W}_2^{K-1}\mathbf{W}_1^{K-1}||_F^2 - \frac{1}{2}||\mathbf{W}_2\mathbf{W}_1\mathbf{x}_k||_2^2, \quad \text{(C.50)}$$

is gradient equivalent to minimising the distance to the least-squares solution for all K *training pairs*

$$\mathcal{L} = \frac{1}{2}\left|\left|\boldsymbol{\Sigma}_{yx}^K \left(\boldsymbol{\Sigma}_{xx}^K\right)^+ - \mathbf{W}_2\mathbf{W}_1\right|\right|_F^2. \quad \text{(C.51)}$$

Proof of Theorem C.6. Under the assumption of orthonormal inputs, $\mathbf{X}$ is a (semi-)orthonormal matrix and therefore $\mathbf{X}^+ = \mathbf{X}^T$. Then, using that $(\mathbf{AB})^+ = \mathbf{B}^+\mathbf{A}^+$ if $\mathbf{A} = \mathbf{B}^T$ (Greville, 1966), we get

$$\begin{aligned}\left(\boldsymbol{\Sigma}_{xx}^K\right)^+ &= \left(\frac{1}{K}\mathbf{X}\mathbf{X}^T\right)^+ \\ &= K\mathbf{X}^{T+}\mathbf{X}^+ \\ &= K\mathbf{X}\mathbf{X}^T.\end{aligned} \quad \text{(C.52)}$$

Next, we use the assumption that training has converged to a least-squares solution

on the last $K-1$ training pairs to derive

$$\begin{aligned}
\mathcal{L} &= \frac{1}{2}\left|\left|\boldsymbol{\Sigma}_{yx}^{K}\left(\boldsymbol{\Sigma}_{xx}^{K}\right)^{+} - \mathbf{W}_2\mathbf{W}_1\right|\right|_F^2 \\
&= \frac{1}{2}\left|\left|\frac{1}{K}\mathbf{Y}\mathbf{X}^T\left(\frac{1}{K}\mathbf{X}\mathbf{X}^T\right)^{+} - \mathbf{W}_2\mathbf{W}_1\right|\right|_F^2 \\
&= \frac{1}{2}\left|\left|\sum_{n=1}^{K}\mathbf{y}_n\mathbf{x}_n^T\left(\sum_{n=1}^{K}\mathbf{x}_n\mathbf{x}_n^T\right)^{T} - \mathbf{W}_2\mathbf{W}_1\right|\right|_F^2 \\
&= \frac{1}{2}\left|\left|\left(\mathbf{y}_k\mathbf{x}_k^T + \sum_{n=1}^{K-1}\mathbf{y}_n\mathbf{x}_n^T\right)\left(\mathbf{x}_k\mathbf{x}_k^T + \sum_{n=1}^{K-1}\mathbf{x}_n\mathbf{x}_n^T\right) - \mathbf{W}_2\mathbf{W}_1\right|\right|_F^2 \\
&= \frac{1}{2}\left|\left|\mathbf{y}_k\mathbf{x}_k^T + \sum_{n=1}^{K-1}\mathbf{y}_n\mathbf{x}_n^T\left(\sum_{n=1}^{K-1}\mathbf{x}_n\mathbf{x}_n^T\right)^{+} - \mathbf{W}_2\mathbf{W}_1\right|\right|_F^2 \\
&= \frac{1}{2}\left|\left|\mathbf{y}_k\mathbf{x}_k^T + \mathbf{W}_2^{K-1}\mathbf{W}_1^{K-1} - \mathbf{W}_2\mathbf{W}_1\right|\right|_F^2.
\end{aligned} \tag{C.53}$$

Which has partial derivatives

$$\frac{\partial\mathcal{L}}{\partial\mathbf{W}_1} = -\mathbf{W}_2^T\left(\mathbf{y}_k\mathbf{x}_k^T + \mathbf{W}_2^{K-1}\mathbf{W}_1^{K-1} - \mathbf{W}_2\mathbf{W}_1\right), \tag{C.54}$$

and

$$\frac{\partial\mathcal{L}}{\partial\mathbf{W}_2} = -\left(\mathbf{y}_k\mathbf{x}_k^T + \mathbf{W}_2^{K-1}\mathbf{W}_1^{K-1} - \mathbf{W}_2\mathbf{W}_1\right)\mathbf{W}_1^T. \tag{C.55}$$

We continue by calculating the partial derivatives with respect to the proposed alternative loss function

$$\begin{aligned}
&\frac{\partial}{\partial\mathbf{W}_1}\big(\frac{1}{2}||\mathbf{W}_2\mathbf{W}_1\mathbf{x}_k - \mathbf{y}_k||_2^2 + \frac{1}{2}||\mathbf{W}_2\mathbf{W}_1 - \mathbf{W}_2^{K-1}\mathbf{W}_1^{K-1}||_F^2 - \frac{1}{2}||\mathbf{W}_2\mathbf{W}_1\mathbf{x}_k||_2^2\big) \\
&= \mathbf{W}_2^T(\mathbf{W}_2\mathbf{W}_1\mathbf{x}_k - \mathbf{y}_k)\mathbf{x}_k^T + \mathbf{W}_2^T(\mathbf{W}_2\mathbf{W}_1 - \mathbf{W}_2^{K-1}\mathbf{W}_1^{K-1}) - \mathbf{W}_2^T\mathbf{W}_2\mathbf{W}_1\mathbf{x}_k\mathbf{x}_k^T \\
&= \mathbf{W}_2^T\big(\mathbf{W}_2\mathbf{W}_1\mathbf{x}_k\mathbf{x}_k^T - \mathbf{W}_2\mathbf{W}_1\mathbf{x}_k\mathbf{x}_k^T - \mathbf{y}_k\mathbf{x}_k^T + \mathbf{W}_2\mathbf{W}_1 - \mathbf{W}_2^{K-1}\mathbf{W}_1^{K-1}\big) \\
&= -\mathbf{W}_2^T\big(\mathbf{y}_k\mathbf{x}_k^T + \mathbf{W}_2^{K-1}\mathbf{W}_1^{K-1} - \mathbf{W}_2\mathbf{W}_1\big)
\end{aligned} \tag{C.56}$$

and

$$
\begin{aligned}
&\frac{\partial}{\partial \mathbf{W}_2}\big(\frac{1}{2}||\mathbf{W}_2\mathbf{W}_1\mathbf{x}_k - \mathbf{y}_k||_2^2 + \frac{1}{2}||\mathbf{W}_2\mathbf{W}_1 - \mathbf{W}_2^{K-1}\mathbf{W}_1^{K-1}||_F^2 - \frac{1}{2}||\mathbf{W}_2\mathbf{W}_1\mathbf{x}_k||_2^2\big) \\
&= (\mathbf{W}_2\mathbf{W}_1\mathbf{x}_k - \mathbf{y}_k)\mathbf{x}_k^T\mathbf{W}_1^T + (\mathbf{W}_2\mathbf{W}_1 - \mathbf{W}_2^{K-1}\mathbf{W}_1^{K-1})\mathbf{W}_1^T - \mathbf{W}_2\mathbf{W}_1\mathbf{x}_k\mathbf{x}_k^T\mathbf{W}_1^T \\
&= \big(\mathbf{W}_2\mathbf{W}_1\mathbf{x}_k\mathbf{x}_k^T - \mathbf{W}_2\mathbf{W}_1\mathbf{x}_k\mathbf{x}_k^T - \mathbf{y}_k\mathbf{x}_k^T + \mathbf{W}_2\mathbf{W}_1 - \mathbf{W}_2^{K-1}\mathbf{W}_1^{K-1}\big)\mathbf{W}_1^T \qquad \text{(C.57)} \\
&= -\big(\mathbf{y}_k\mathbf{x}_k^T + \mathbf{W}_2^{K-1}\mathbf{W}_1^{K-1} - \mathbf{W}_2\mathbf{W}_1\big)\mathbf{W}_1^T,
\end{aligned}
$$

which are equivalent to Equation C.54 and Equation C.55 and therefore the two losses are gradient equivalent. □