Complexity Dynamics of Learning Systems

Branton DeMoss
St Edmund Hall
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

October 2025

To Port Meadow,
out of equilibrium
in a steady state

Acknowledgements

The first and largest thanks go to my advisors, Nick and Ingmar. They made this all possible—I would not be here without their support. Thank you for maintaining research groups which let intellectual freedom flourish, and for trusting me when I wandered off the beaten path. Thanks to Jakob for our collaborations and the generous gift of his scarce time. He has the energy and creativity of ten men, I was lucky to have met him. I thank my mother for her love, support, and incredible initiative. It's good to have her on my side. Thanks to Matt for his support, and giving me a first chance. Thank you to my co-founder Nick, our time in Chicago working with language models just as they were exploding was one of the most fulfilling and interesting periods of my life. Thank you Elena for your love, and for bringing me here, there, and everywhere. I've never regretted following my heart. Thank you Seiji for your constant friendship, your brilliant insights, and our globetrotting fun. With you I am always laughing. Thank you Cathy for taking me in. I have many wonderful friends in Oxford: I met Emma my very first day here, what an amazing chance that was. She can do everything, and often does. Thank you Will for all your show-and-tells. Wow, fractals. Thank you Sophie for your grace. Thank you Robin for your stony questions. Thank you Lizie for the flowers. Thank you Celeste for many lunches and musical chats. Thank you Marie, for all your words. Thank you Jasmine, for helping me bridge ML and physics. Thank you Rebecca, Triin, Oscar, Tariq, Isaac, Charlotte, Alex, and Freddie for dancing the nights away. Lastly, thanks to Curtis Winter for light, line, form, color, sound, wine, delusion, and friendship.

Abstract

We study complexity and generalization in learning systems. We begin by framing learning as a dynamical non-equilibrium process, and suggest that a more complete understanding of machine learning requires methods beyond statistical description. We point out that machine learning must be considered both a mathematical and a natural science, since it requires both a theory of the formal learning system and the environment it operates within. Our central technical contribution is a dynamical complexity measure based on the theory of Kolmogorov complexity and lossy compression. We use this measure to demonstrate a complexity phase transition during learning, in neural networks which suddenly generalize after initially overfitting their training data. We also explore generalization in multi-step decision processes, which break common statistical assumptions underlying generalization. Finally, we explore generalization of learning systems trained only in simulation to the real world.

Contents

List of Figures

Chapter 1

Introduction

What can machines learn, and when do they generalize? It is no good if our robot can only navigate scenarios it has previously encountered, or our language system can only handle questions which we have already provided the answers to. To surpass the utility of simple lookup tables encoding prior knowledge and experience, learning systems must go beyond mere memorization. They must generalize.

What can machines learn? Classically, machine learning researchers have been concerned with questions of model expressivity, which ask whether a given system can represent a sufficiently rich class of functions. Contemporary machine learning has focused on neural networks, which enjoy a high degree of representational richness. According to the universal approximation theorem (UAT) (Hornik, Stinchcombe, and White, 1989), arbitrary width single hidden-layer networks can approximate any continuous function with arbitrary precision. The relationship between the representational capacity of a model, or hypothesis class $\mathcal{H}$ and its expected generalization performance is most clearly expressed through the classic finite hypothesis bound (Shalev-Shwartz and Ben-David, 2014): Let $\hat{R}(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(h(x_i), y_i)$ be the empirical risk of a hypothesis $h \in \mathcal{H}$ with ℓ the indicator function, and let $R(h) = \mathbb{E}_{(x,y)\sim D}\left[\hat{R}(h)\right]$ be the true risk, with train and test samples independently and identically distributed (iid). With probability at least $1 - \delta$,

$$R(h) \leq \hat{R}(h) + \sqrt{\frac{\log|\mathcal{H}| + \log(1/\delta)}{2n}} \tag{1.1}$$

This bound states that the expected generalization error is no more than the average train error, plus a term which depends on the model's capacity, the desired degree of certainty, and the number of training samples. The model capacity is expressed as $\log|\mathcal{H}|$ here, i.e. the number of bits needed to specify our hypothesis under a uniform prior over its class. Therefore, the larger and more expressive our model class, the

looser these bounds become. This reveals a fundamental tension: we want to use expressive models which can represent a wide variety of functions, but these bounds suggest that more expressive models are expected to generalize worse.

There are a number of different measures of model expressivity which quantify a model class's capacity to fit arbitrary labels over their domains, such as the VC dimension (Vapnik and Chervonenkis, 1971) and Rademacher complexity (Bartlett and Mendelson, 2003). A limitation of these measures is their inability to distinguish the generalization capabilities of *particular* models that result from optimization over the hypothesis class, e.g. through a learning algorithm and training dataset. These measures cannot distinguish models which overfit from those which generalize, and cannot be used to track dynamical properties during learning. Hence, while theorems such as the UAT guarantee the existence of certain solutions, whether they are found by our optimization procedure in practice cannot be determined from these measures.

When can we expect generalization to occur? Despite the rich representations afforded by contemporary deep neural networks, the No Free Lunch theorem (Wolpert and Macready, 1997) tells us that even the most sophisticated learning algorithms do no better than blind search or random guessing when averaged on a uniform distribution over all optimization problems. To outperform naïve approaches, optimization procedures must take advantage of structure in the datasets they encounter, commonly dubbed the *inductive bias* in machine learning (Mitchell, 1980). Because of this, any theory of learning systems is incomplete without a corresponding theory of their data.

A full account of generalization must therefore address details of both the learning system and the target environment. While the learning system may be formally specified, a complete and computationally tractable formalization of the natural world is not forthcoming. Indeed, we typically use learning systems when a formal specification of the environment is unavailable or analytically intractable. We face a chicken-and-egg problem: generalization requires the use of structure in the environment, but we tend to use learning systems precisely when the environment is resistant to formalization, and hence has unknown structure.

1.1 Priors at Equilibrium

Evidently generalization *is* possible: we observe patterns and regularities, and make many predictions better than chance. Consider the following sequence:

$$1, 2, 4, 8, 16... \tag{1.2}$$

There are infinitely many possible continuations, yet one in particular stands out as most likely *a priori*. Why might this be?

Prior beliefs represent our state of knowledge and uncertainty about a system before we have made a measurement. A classic prior comes from Laplace's principle of indifference, which says that equal credence should be given to each possibility if there is no reason to think otherwise (Laplace, 1825). In other words, Laplace claims that our default prior should be a uniform distribution. Jaynes refined and generalized Laplace's principle in a pioneering series of papers (Jaynes, 1957a; Jaynes, 1957b), proposing the principle of maximum entropy, along with an information-theoretic interpretation of statistical mechanics as inference. Jaynes' principle says that our prior should have maximal entropy, subject to consistency with any relevant information. In the case of a discrete set of outcomes without any additional data, this reproduces the results of Laplace's principle. Jaynes emphasizes, however, that his results address systems at equilibrium:

> "We shall be concerned with the prediction of equilibrium thermodynamic properties, ... which involves only the probabilities assigned to stationary states. [...] By restricting our attention to the prediction of equilibrium properties as in the present paper, we are in effect deciding at the outset that the only type of initial information allowed will be values of quantities which are observed to be constant in time." Jaynes, 1957a, pp. 621, 624

Is the equilibrium assumption a good one in the context of machine learning? The second law of thermodynamics states that isolated systems tend towards thermal equilibrium and maximum entropy. The second law is widely agreed to be the surest of the known physical laws. Eddington notably remarked:

> "If someone points out to you that your pet theory of the universe is in disagreement with Maxwell's equations—then so much the worse for Maxwell's equations. If it is found to be contradicted by observation—well, these experimentalists do bungle things sometimes. But if your theory is found to be against the second law of thermodynamics I can give you no hope; there is nothing for it but to collapse in deepest humiliation." Eddington, 1928, ch. 14

While the second law tells us that every isolated system *tends* to equilibrium, it is clear that many complex systems of interest are neither isolated, nor close to equilibrium. Indeed virtually all complex systems, from weather patterns to

life itself, require non-equilibrium conditions to function (Schrödinger, 1946). The thermodynamics of these evolving, non-stationary systems are formalized in non-equilibrium statistical mechanics (Zwanzig, 2001), where fluctuation theorems quantify statistical deviations from monotonic entropy increase (Jarzynski, 1997; Crooks, 1999). Because an entropy gradient is required to drive their dynamics (Groot and Mazur, 1962), we should expect to see regions of *low entropy* in the vicinity of complex systems, in apparent contradiction with Laplace's principle.

Instead of the statistical uniformity we might expect to encounter at equilibrium, across the sciences we observe that much of the natural world is surprisingly ordered and simple. In information geometry and machine learning, for example, the *manifold hypothesis* posits that "high dimensional data tends to lie in the vicinity of a low dimensional manifold" (Narayanan and Mitter, 2010). Similarly, Wigner (1960) discusses the "unreasonable effectiveness" of simple mathematical structures in expressing physical law. These observations find their most well known expression in Occam's Razor.

Also known as the principle of parsimony, Occam's Razor says that one "must not multiply entities beyond necessity" (Brampton, 1964). We interpret this to say that absent evidence otherwise, simpler explanations should be preferred. Under Jaynes' identification of inferential priors with statistical properties of the environment, we can understand the explanatory success of Occam's Razor as a reflection of the (perhaps surprisingly) low complexity of many systems, as captured by e.g. the manifold hypothesis. It is the principle of parsimony which picks out a particular continuation for sequence 1.2 as most likely; maximum entropy principles alone provide insufficient reason to make any particular inference.

1.2 Complexity and Generalization

While Laplace and Jaynes' principles have a clear statistical interpretation, formalizing simplicity requires mathematical machinery beyond statistical description. The basic idea is that simpler objects should have shorter descriptions. Solomonoff (1964a) and Solomonoff (1964b) introduced a prior which formalizes Occam's Razor, and assigns probability to sequences according to their algorithmic complexity. Let U be a universal prefix Turing machine, then Solomonoff's prior is:

$$P(x) = \sum_{p:U(p)=x} 2^{-|p|} \tag{1.3}$$

Solomonoff's prior assigns probability to strings according to an exponentially weighted sum of the length of every program which produces that string. Later, Kolmogorov, 1965 and Chaitin, 1969 independently formalized the same notion of algorithmic complexity, which is now known as the Kolmogorov complexity:

$$K(x) = \min_{p:U(p)=x} |p| \tag{1.4}$$

While these quantities are uncomputable, their formalization in algorithmic information theory (AIT) (Ming Li and Vitányi, 1993) lets us address many questions which statistical mechanics and classical information theory do not. For example, by replacing the uniform prior over our hypothesis class in generalization bound 1.1 with the Solomonoff prior, we obtain bounds which meaningfully distinguish individual hypotheses according to their complexity, as measured by their compressibility.

Shannon's source coding theorem[1] identifies the compressibility of iid sequences with the entropy of their distribution, while the Kolmogorov complexity accounts for any recursively enumerable pattern.[2] The iid assumption plays a role in information theory analogous to that of equilibrium in statistical mechanics: both simplify analysis by assuming the system of interest is stationary, and admits a sufficient characterization by statistical quantities alone. One benefit of complexity-theoretic arguments is their ability to address properties of individual instances, rather than statistical ensembles.

This thesis explores generalization phenomena in learning systems. Our central contribution is a dynamical complexity measure based on lossy compression that remedies the issues inherent to traditional capacity measures discussed earlier. We use this measure to explain a pathological generalization phenomenon in deep neural networks known as "grokking" in Chapter 2. Our results suggest that, just as physical systems can transition between distinct thermodynamic phases, learning systems can also transition between distinct phases which are characterized by their algorithmic complexity, and which drive generalization. In Chapter 3 we explore generalization in multi-step decision processes, which introduce dependencies between predictions and future states that break common iid assumptions underlying generalization. Finally, in Chapter 4 we explore how generalization challenges manifest when learning systems trained in simulation are deployed in the real world.

[1]We assume familiarity with basic notions from information theory. Appendix A provides an elementary introduction to the relevant concepts.

[2]If a sequence is truly random, its normalized Kolmogorov complexity tends to the entropy of its distribution. In this sense, AIT is a consistent "microscopic" information theory which reproduces Shannon information theory in the ergodic limit.

Chapter 2

The Complexity Dynamics of Grokking

When can we expect learned models to generalize well? In this chapter, we investigate the recently observed "grokking" phenomenon (Power et al., 2022), where neural networks exhibit a sharp transition from memorization to generalization long after overfitting their training data. The extended delay between these two phases of learning provides an ideal setting to study the relationship between complexity and generalization in learning systems. The study of phase transitions in learning systems provides a window into fundamental questions about the nature of generalization and emergence of structure.

Simplicity has long been recognized as a common feature of good models: Occam's Razor says that amongst models which explain some data equally well, the simplest one is expected to generalize best. Drawing inspiration from statistical physics and algorithmic information theory, we introduce a dynamical measure of the intrinsic complexity of neural networks based on lossy compression and Kolmogorov complexity. Our measure can be applied throughout learning, to reveal the complexity dynamics of the system as optimization is applied. This approach parallels work by Aaronson, Carroll, and Ouellette, 2014 who study the complexity dynamics of simulated physical systems. Their key insight is to measure complexity after appropriate *coarse-graining* of the system's state. We formalize their insight using algorithmic rate–distortion theory, resulting in a principled lossy compression scheme for neural networks which bounds their complexity.

This framework allows us to track the complexity dynamics of neural networks throughout training. We find that properly regularized networks that successfully generalize exhibit a *complexity phase transition*: complexity first rises as the network

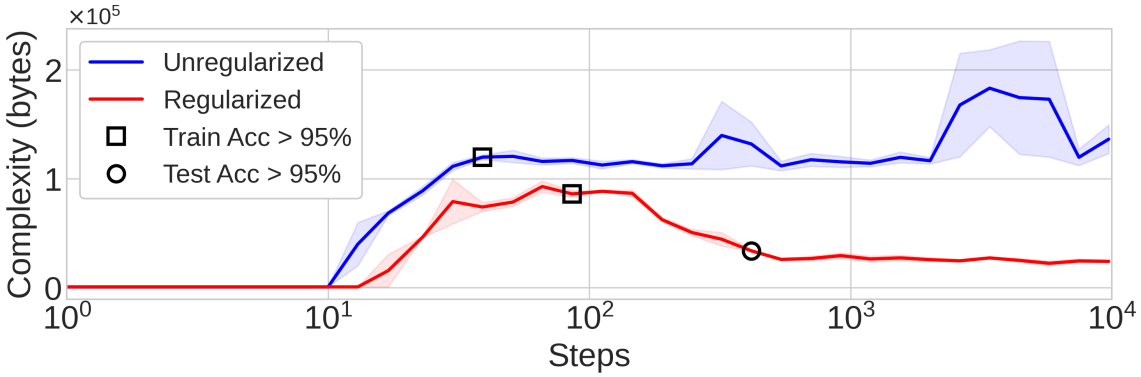

Figure 2.1: Complexity dynamics for unregularized (blue) and regularized (red) neural networks in grokking experiments on modular multiplication. Markers denote when memorization (□) and generalization (◯) occur. In the regularized network, generalization occurs as complexity falls after its peak at memorization, while the unregularized network never generalizes, and its complexity remains large. Mean over 3 seeds, shading std. error.

memorizes training data, then falls sharply as the network discovers a simpler underlying pattern that generalizes. In contrast, unregularized networks that fail to generalize remain trapped in a high-complexity memorization phase indefinitely. We establish an explicit link between our complexity measure and statistical generalization bounds, and show how these correspond to the Minimum Description Length Principle.

The Minimum Description Length (MDL) Principle (Rissanen, 1978) attempts to formalize Occam's Razor in the language of information theory. It says that the best model is the one which minimizes the total message length of the data. Shannon's theorem (Shannon, 1948) says that the expected optimal coding rate for samples drawn from a distribution is equal to the entropy of the distribution. Hence, the entropy H of some data D under a model M, $H(D|M)$ tells us the expected message length after encoding the data with the model. However, we must also consider the information cost of the model itself, which is the model complexity $C(M)$. MDL says we should take the model which minimizes the *sum* of these terms:

$$M \leftarrow \arg\min_{M} H(D \mid M) + C(M) \tag{2.1}$$

Intuitively, it is worth making our model one bit more complex if it lets us reduce the apparent entropy of the data by more than one bit, since that reduces the overall message length. Note that if the model M fully explains some subset $D' \subseteq D$ then the prediction of that data becomes deterministic and $H(D'|M) = 0$. Consider a

trivial model which simply memorizes the data in a lookup table. Whilst this reduces the entropy of the data to zero, the model complexity is exactly equal to the original entropy of the data. Therefore the total description length has not changed, and no compression has been achieved by the model.

The insight that simpler models are expected to generalize better is made precise through statistical generalization bounds of the form:

$$\text{test error} \leq \text{train error} + \text{model complexity} \tag{2.2}$$

which are ubiquitous across a number of fields of statistical machine learning (Vapnik, 1991; Bartlett and Mendelson, 2003). When the error terms are identified with entropy, as is frequently the case, these bounds reveal that models which minimize MDL are precisely those expected to generalize best. This provides the link between compression and generalization: the model which best compresses the data is the one expected to generalize best.

To apply the MDL principle, we require a measure of model complexity $C(M)$. Traditional complexity measures in machine learning such as Rademacher complexity (Bartlett and Mendelson, 2003) or VC dimension (Vapnik and Chervonenkis, 1971) are defined relative to specific hypothesis classes, and as discussed in the introduction, cannot distinguish between individual hypotheses realized by our network as it is learning. We seek a universal measure that applies to any computational model. Recall that the Kolmogorov complexity K of a string s is equal to the length of the shortest program p which prints s when executed on a universal Turing machine U:

$$K(s) = \min_{p:U(p)=s} |p| \tag{2.3}$$

To build intuition, consider two strings of length N: The string '111...1' can be generated by a short program `print '1' N times`, so $K(s) \approx \log N$. In contrast, a random string '10110...' has no pattern to exploit; its shortest description is itself: `print '10110...'`, giving $K(s) \approx N$. Algorithmically random strings are precisely those whose Kolmogorov complexity equals their length—they are incompressible because they contain no exploitable patterns. We expect that networks which implement a simple pattern should have low complexity, *despite* having many parameters, while a network memorizing random labels must have high complexity. Unlike capacity measures that provide loose bounds over whole model classes, Kolmogorov complexity captures the information content of the individual learned function that results from optimization.

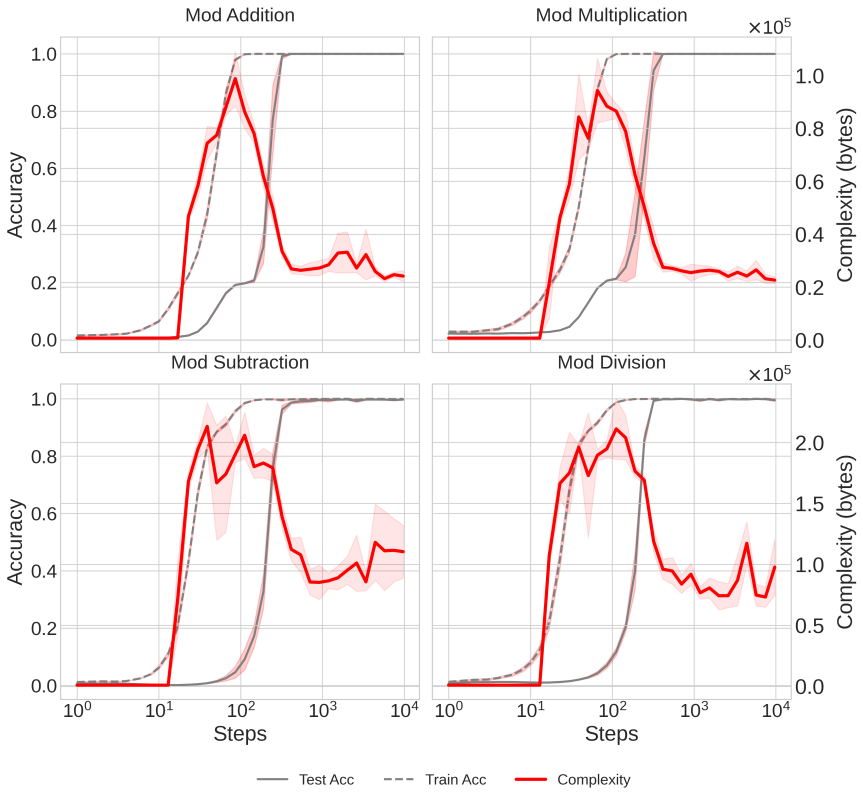

Figure 2.2: Complexity and accuracy vs steps for models regularized using our method. As models memorize, complexity increases. Complexity falls as generalization occurs. We use Algorithm 2 to coarse-grain models with $\epsilon = 1$. Std. error shaded with six seeds.

To implement these ideas, we develop a principled approach to measuring neural network complexity based on lossy compression. While the Kolmogorov complexity is uncomputable, compression provides an upper bound: notice that compressed data plus its decompressor form a program that generates the original data. The compressed file-size of the network provides an upper bound on its complexity, with better compression resulting in tighter upper bounds. However, we cannot apply naïve lossless compression methods to neural networks and expect precise complexity estimates, because networks contain noise from initialization and stochastic training. Because noise is random information, a naïve compression estimate is dominated by the noise, rather than the algorithmic structure in the network. To compute tight compression bounds, we develop a coarse-graining procedure inspired by rate-distortion theory that removes noise while preserving functional behavior, enabling compression ratios 30-40× better than naïve approaches. This sensitivity is crucial for revealing the underlying complexity dynamics during training, shown in Fig 2.1. Our main contributions are:

- A theoretical framework connecting neural network complexity to Kolmogorov complexity through lossy compression, with explicit links to generalization bounds.
- Empirical demonstration of a complexity phase transition during grokking, characterized by a rise and fall pattern that distinguishes generalizing from memorizing solutions.
- A regularization method based on spectral entropy that encourages networks toward low-complexity representations by penalizing a differentiable measure of their effective dimension.

2.1 Related Work

Grokking The grokking phenomenon (Power et al., 2022) occurs when networks suddenly transition from memorizing to generalizing solutions long after over-fitting the training data. Liu, Kitouni, et al., 2022 characterize conditions under which grokking occurs, and categorize learning into four distinct regimes, *comprehension*, *grokking*, *memorization*, and *confusion* by examining phase diagrams of learning performance across different hyperparameters. Liu, Michaud, and Tegmark, 2022 demonstrate grokking on non-algorithmic datasets, and propose a link between a

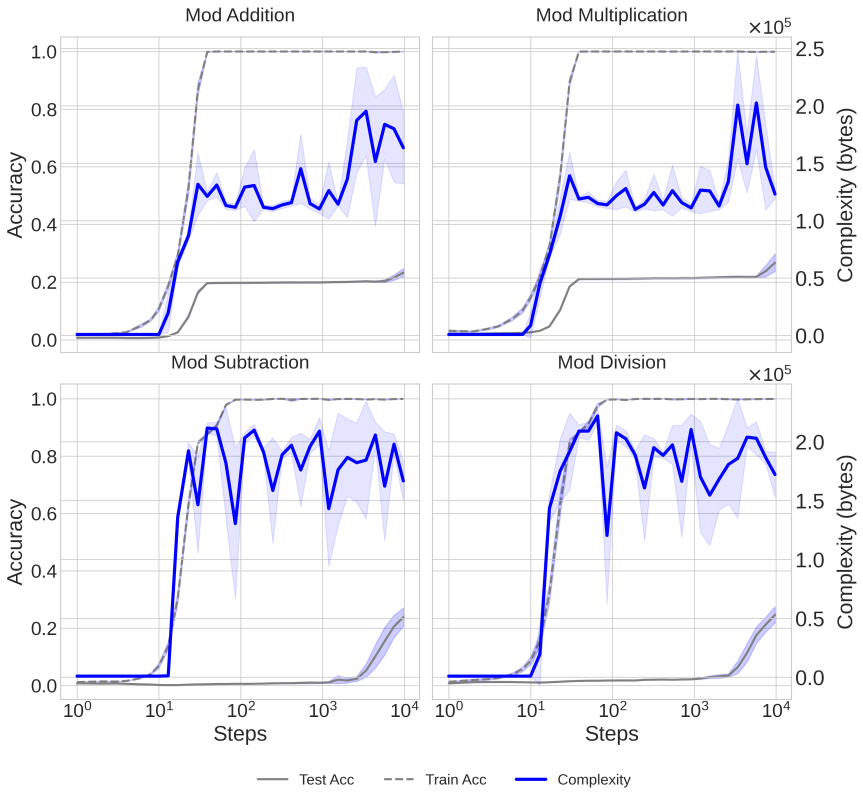

Figure 2.3: Complexity and accuracy vs steps for unregularized models. As models memorize, complexity increases. Without regularization, the models fail to find low complexity solutions, and never generalize. We use Algorithm 2 to coarse-grain models with $\epsilon = 1$.

network's parameter norm and grokking. This link is also explored by Varma et al., 2023, who explain grokking in terms of the "efficiency" of a network, as measured by the ratio of the logit to parameter norms. Nanda et al., 2023 propose task-specific loss-based "progress measures" to detect grokking, but note that we "lack a general notion of criticality that would allow us to predict when the phase transition" occurs. In this work, we suggest that complexity is the key dynamical metric to understand grokking (and generalization more broadly), and discuss why proxy measures such as the L^2 norm are insufficient.

Generalization The connection between model capacity and complexity is sometimes misunderstood amongst machine learning practitioners: the contemporary view in deep learning is that "larger models are better" (Nakkiran et al., 2019), with empirically observed scaling laws across a range of modalities and architectures (Kaplan et al., 2020; Henighan et al., 2020) which show monotonic improvement of performance with increasing model scale. This view contrasts with the classical bias-variance tradeoff, which says that "once we pass a certain threshold, larger models are worse" (Nakkiran et al., 2019). The latter view is only true assuming model capacity is equal to model complexity, and would follow from bounds of the form of Equation 2.2. However, this appears *not* to be the case. Lotfi et al., 2024 recently established the first non-vacuous generalization bounds for large language models by producing highly compressible models trained in a non-linear low-rank subspace, and bounding their generalization performance under a smoothed entropy loss by the Kolmogorov complexity. While they focus on explicit generalization *bounds*, we focus on the complexity dynamics of the transition from memorization to generalization to explain grokking. Goldblum et al., 2023 argue for generalization to be understood using Kolmogorov complexity, and apply the Solomonoff prior, $P(h) = 2^{-K(h)}$ for hypotheses h to a finite hypothesis bound (Langford and Seeger, 2001) to produce a generic generalization bound based on the Kolmogorov complexity.

Intrinsic Dimension While data may be embedded in a high-dimensional ambient space, it often lies on a low-dimensional submanifold (Narayanan and Mitter, 2010). The dimension of this low-dimensional manifold, if it exists, is called the intrinsic dimension of the data manifold, which serves as an alternative complexity measure. Sharma and Kaplan (2020) show that the intrinsic dimension of the data manifold can explain the observed power laws which govern scaling in deep neural networks (Kaplan et al., 2020). A related quantity is the effective dimension of a model, which measures the number of "relevant" parameters in a network based on the local curvature of the loss landscape. Maddox, Benton, and Wilson (2020) estimate

the effective dimension of deep neural networks by counting the number of eigenvalues of the Hessian above a cutoff. They find that this quantity tracks generalization, and varies non-monotonically with model scale in double descent experiments. In particular, despite training networks with up to 10^7 parameters, the effective dimension of the model never increases above 10^2, suggesting that networks are vastly overparameterized relative to the intrinsic complexity of the data manifold. One promising avenue for future research is to more formally connect the intrinsic dimension of the data manifold with the effective dimension of an optimal model for that data.

Coarse-graining Aaronson, Carroll, and Ouellette, 2014 study the complexity dynamics of a cellular automaton which simulates the mixing of two liquids. They observe the rise and fall of complexity as entropy increases, and conjecture that this phenomenon is generic. To approximate the complexity of their automaton, they propose an "apparent complexity" measure which is the compressed file size of a *coarse-grained* description of the state. They informally assert that coarse-graining removes random information from the state, leaving only "interesting" information behind. We formalize their intuition using rate–distortion theory, and use the resultant complexity measure to explain grokking.

2.2 Complexity and Generalization

Understanding generalization is of central importance in machine learning. To what extent can we expect our models to generalize, and can we predict their performance ahead of time? Lotfi et al., 2024 provide a generalization bound which takes advantage of the Kolmogorov complexity, and quantifies the expected risk $R(h)$ for a given hypothesis h, with probability $1 - \delta$:

$$R(h) \leq \hat{R}(h) + \sqrt{\frac{K(h) + 2\log K(h) + \log(1/\delta)}{2n}} \tag{2.4}$$

Where $K(\cdot)$ is the Kolmogorov complexity, $\hat{R}(h)$ is the empirical risk, and n the number of samples. As mentioned in the introduction, the Kolmogorov complexity is not computable, but can be upper-bounded by compression. Therefore, we can bound our generalization error by compressing models, with better bounds following from tighter compression. It is important to keep in mind that some approaches we might take to compress models, such as reducing the precision of their parameters, can also result in loss of performance. Hence, while we might decrease the complexity term by lossily compressing models, this may result in the empirical risk $\hat{R}(h)$ increasing.

To produce strong generalization guarantees, we must jointly minimize the model complexity and the empirical risk.

These bounds can also be understood as restating the MDL principle. If the empirical risk is understood as the entropy of the training data under the model, as is typically the case, then the joint minimization of the empirical risk and model complexity is exactly the MDL principle. I.e. the model which generalizes best is the one which compresses the data best, and is itself most compressible.

Unfortunately, naïvely compressing a model's parameters produces poor complexity bounds, since the network contains random information left over from initialization, and from the stochastic training procedure. If we could remove the random information from a network, then compressing the resulting parameters could provide much tighter bounds on the network complexity. However, it is undecidable whether information is random (Ming Li and Vitányi, 1993), so there is no generic method to remove it.

Aaronson, Carroll, and Ouellette, 2014 remove random information by coarse-graining their automaton's state. They coarse-grain by smoothing and quantizing the state. The coarse-grained state is then compressed with off-the-shelf programs such as `bzip2`, with the compressed file size providing improved complexity estimates. They dub this metric the "apparent complexity", since it is not actually a bound on the original state's complexity, and is arbitrary in the choice and degree of coarse-graining. We adapt their method and apply it to neural networks based on a central insight: **the loss function induces a coarse-graining**, which can be understood as principled *lossy* compression of the model.

2.2.1 Lossy Compression

Shannon's theorem provides a lower-bound on coding rates which can *losslessly* compress signals, but more efficient coding rates are possible if some distortion is allowed. For example, JPEG compression removes high-frequency details from images which are perceptually irrelevant to the human eye, incurring some distortion. Similarly, we want to find the least detailed model which produces acceptable performance on the training data. Since neural networks are initialized randomly and trained via stochastic optimization, much of the information capacity of their parameters stores noise. In algorithmic information theory, noise is precisely that which is incompressible, so if we naïvely compress model weights, the resultant complexity estimate from the compressed file size is dominated by the noise stored in the weights. To overcome this, we develop a lossy compression scheme which lets us formalize the trade-off between information capacity in the model and the performance on the training data

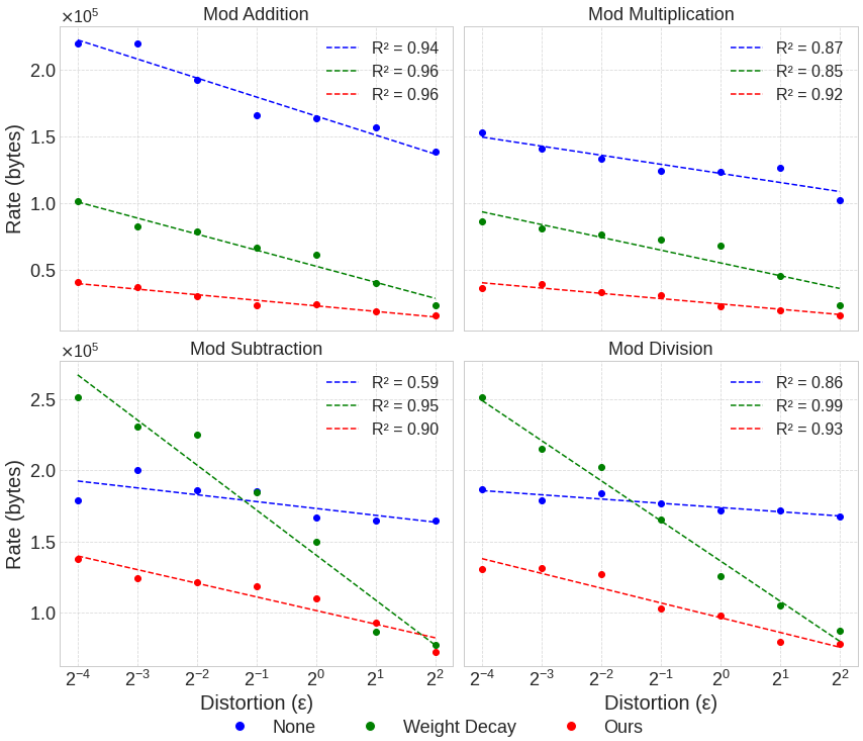

Figure 2.4: Algorithmic rate–distortion curves $r_\theta(\hat{\theta})$ for all datasets at varying distortion bounds ϵ. The rate is the complexity of the coarse-grained weights $\hat{\theta}$ at the end of training. Mean over six seeds.

as measured by the loss function. Rate–distortion theory formalizes the trade-off between coding rate, induced distortion, and perceptual quality (Blau and Michaeli, 2019), though it is typically defined over random variables. Vereshchagin and Vitányi, 2004 generalize the theory of rate distortion to the algorithmic information setting. They define the algorithmic rate–distortion function $r_x(y)$, which is the minimum number of bits needed to specify y with distortion relative to x no more than ϵ:

$$r_x(y) = \min_{y:d(x,y)<\epsilon} K(y) \tag{2.5}$$

Where d is some distortion function. We will take the "input string" x to be the model parameter vector θ, and the "output" y to be the coarse-grained parameter vector $\hat{\theta}$. The choice of distortion function d is critical. In particular, we are not interested in the distortion in parameter space, $|\theta - \hat{\theta}|^2$. Instead, we are interested in the distortion *under the loss function*. Hence, we set:

$$d(\theta, \hat{\theta}) = \left| \mathcal{L}(\theta, D) - \mathcal{L}(\hat{\theta}, D) \right| \tag{2.6}$$

Substituting this distortion function into Eqn 2.5, we can see that the optimum can be interpreted as the least complex set of model weights $\hat{\theta}$ which induce distortion under the loss function no more than ϵ. We control the rate–distortion trade-off through the parameter ϵ, which leads to the following distortion criterion to accept a coarse-grained approximation:

$$\left| \mathcal{L}(\theta, D) - \mathcal{L}(\hat{\theta}, D) \right| < \epsilon \tag{2.7}$$

Note that we could have chosen an alternative distortion function, such as difference in *accuracy* or any other performance metric of interest. However, the cross-entropy loss preserves the interpretation of the model as a compressor of the training data. In fact, the MDL principle implies an optimal distortion bound ϵ, which determines the best coarse-grained model $\hat{\theta}$:

$$\epsilon \leftarrow \arg\min_{\epsilon} H(D \mid \hat{\theta}) + r_\theta(\hat{\theta}) \tag{2.8}$$

Intuitively, as we coarse-grain a model it becomes less complex. However, it may also perform worse. The MDL principle implies that the optimally coarse-grained model is one which balances the complexity–performance trade-off, as measured by the model's ability to compress its training data, which is given by the loss function. In particular, a *lossy model* might be a better *lossless compressor* of the data, as measured by the total description length. We approximate the algorithmic rate–distortion function via compression upper bounds, and plot curves of best fit in Fig 2.4.

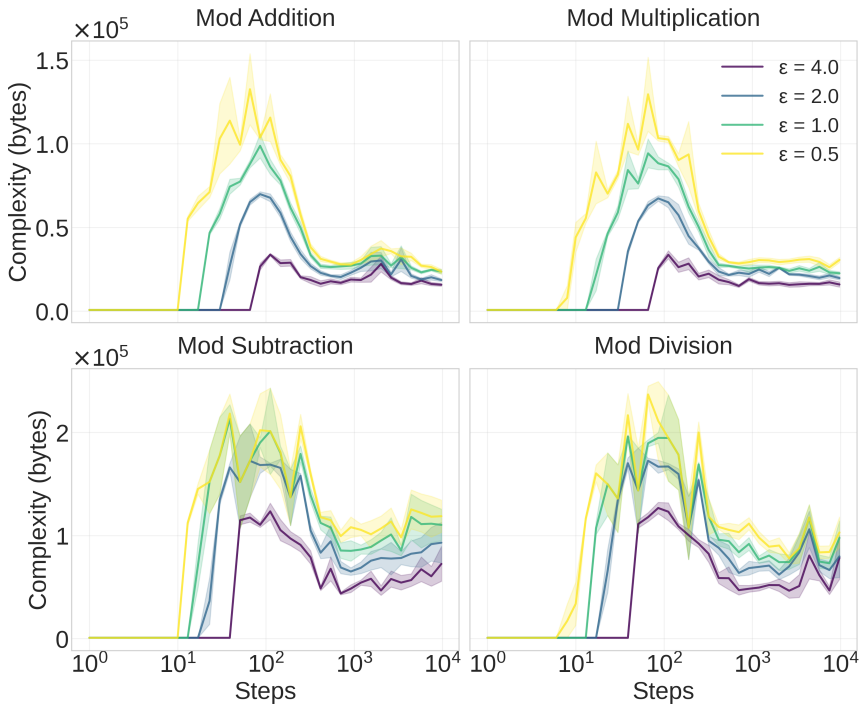

Figure 2.5: Complexity dynamics under varying distortion bounds ϵ. As the allowed distortion increases, the estimated model complexity becomes smaller, as expected. The rise-and-fall pattern is robust under varying distortion bounds.

To obtain models which are optimal compressors, we must regularize them to limit their complexity. In the next section, we discuss the relationship between complexity, capacity, and regularization, and propose a regularization method which results in highly compressible models.

2.3 Capacity and Regularization

To control model complexity, practitioners introduce regularizers to the network training process. Most regularizers work by limiting a model's effective capacity, which *may* limit its complexity. To see this, we consider a simplified representation of a model parameter. The information capacity I of a single positive parameter with fixed precision δ, and max size λ is given by:

$$I = \log \frac{\lambda}{\delta} \tag{2.9}$$

Under this simplified model, it is clear that we can limit the information capacity of the parameters either by controlling their range through λ, or by reducing their precision through δ, which can be thought of as an effective quantization level. Regularizers such

as weight decay control the information capacity by limiting λ, or proxies of it such as an L^p norm. In particular, weight decay adds an independent L^2 penalty for each parameter to the loss function. Methods which control information capacity through δ include training in reduced precision (Micikevicius et al., 2017), and quantization-aware training (S. Ma et al., 2024).

However, training with regularization does not guarantee that the resultant model will be of low complexity. For example, while weight decay ensures parameter norms are smaller than they otherwise would have been, the network can still store detail in high-frequency information. Our experiments demonstrate that this indeed occurs in practice. A key requirement for complexity reduction therefore is a regularization scheme which explicitly alleviates this issue.

A further complication is information which is spread across multiple parameters. Networks are typically parameterized by matrices which represent linear transformations, such that relevant information exists in their *spectrum*. Ultimately such information is accounted for by the total sum of parameter capacities, but it is helpful to think of information in the spectral representation. We now introduce a regularization scheme which addresses both of these challenges.

2.3.1 Quantization and Noise

In contrast to methods like weight decay, which regularize the capacity of networks by controlling their range through effective penalties on λ, we introduce a simple modification to the training procedure which has previously been studied by Hinton and Camp, 1993, amongst many others: One way to set an effective lower bound on the quantization level δ from Eqn. 2.9 is to train networks with noisy weights. The noise prevents the parameters from forming structure below the noise threshold, limiting the information capacity "from the bottom". To do this, we construct a set of noisy weights $\tilde{\theta}$ by adding independent Gaussian noise scaled by δ to every parameter at each forward pass:

$$\tilde{\theta} = \theta + \delta\mathcal{N}(0, 1) \tag{2.10}$$

We compute the forward pass and gradients with the noisy weights, then apply those gradients to the original parameters. Note that δ in Eqn 2.10 need not be the same as δ from Eqn. 2.9, but applying this training procedure sets an effective lower bound on δ from Eqn. 2.9 (the effective precision of our parameters) so we use the same symbol. This implies that we ought to be able to remove all information below the scale set by δ, i.e. that we can quantize the parameters with bin spacing at least δ. To do this, we

introduce a quantization operator $\hat{Q}$ which takes a parameter vector θ and bin size Δ, and rounds all parameters to the nearest bin value

$$\hat{Q}(\theta, \Delta) = \lfloor \frac{\theta}{\Delta} \rceil \times \Delta \tag{2.11}$$

We could achieve greater compression by using a separate Δ for each parameter group, but for simplicity we use a global value of Δ. A simple algorithm which estimates the complexity of a network based on quantization coarse-graining alone is given in Algorithm 1.

Algorithm 1 Complexity by Quantization

Require: params, epsilon
Ensure: best_complexity
best_complexity ← COMPRESS(params)
for bin_sizing **from** small **to** large **do**
    quantized_params ← QUANTIZE(params, bin_sizing)
    if |LOSS(quantized_params) − LOSS(params)| < ϵ **then**
        best_complexity ← MIN(COMPRESS(quantized_params), best_complexity)
    end if
end for
return best_complexity

2.3.2 Spectral Entropy

While coarse-graining by quantization is simple and effective, it can only address high-frequency noise. In this section we introduce another coarse-graining procedure which takes advantage of low-rank approximation to remove random information. We use this low-rank approximation both as a regularization method to limit model complexity, and as a coarse-graining method during the compression stage, to produce tighter complexity estimates throughout training.

Low rank approximation is the problem of finding an approximating matrix $\tilde{M}$ for a matrix M subject to the constraint that $\text{rank}(\tilde{M}) \leq r$ for some r, typically $r \ll \text{rank}(M)$. Low-rank approximation of neural network weight matrices has recently been used to achieve compute– and parameter–efficient finetuning of LLMs (Hu et al., 2021; Dettmers et al., 2023), building on the insight that both the weights and update gradients live on a low-dimension subspace of the total weight space (C. Li et al., 2018; Aghajanyan, Zettlemoyer, and Gupta, 2020).

The mismatch between the apparent and effective ranks of the largest models is another manifestation of the difference between the capacity and complexity of networks. While large capacity may be needed for training dynamics to find optimal solutions, those learned solutions which generalize best tend to be of low effective rank, and therefore low complexity.

It is well-known that the optimal rank k approximation of M under the Frobenius norm is obtained by computing the singular value decomposition of M, and removing all but the k largest singular values. Recall that for an $m \times n$ matrix $M \in \mathbb{R}^{m \times n}$, the singular value decomposition is $M = USV = \sum_{i=1}^{r} \sigma_i u_i v_i^\intercal$, where the columns of U and the rows of V are the left- and right-singular vectors respectively, and S is a diagonal matrix $S = \text{diag}(\sigma_i)$ with singular values $\sigma_1 > \sigma_2 > ... > \sigma_{r=\min(m,n)}$ in decreasing order. Then the optimal rank k approximation is $\tilde{M} = \sum_{i=1}^{k} \sigma_i u_i v_i^\intercal$.

To coarse-grain, we search for the smallest rank decomposition of each layer that satisfies the ϵ–bound of Eqn. 2.7. Note that different layers may admit different degrees of rank decomposition, so using a global value of k across all layers of the network would be suboptimal. Instead, we search for optimal per-layer low-rank approximations by normalizing the singular values

$$\bar{\sigma}_i = \frac{\sigma_i}{\sum_j \sigma_j} \tag{2.12}$$

so that $\sum_i \bar{\sigma}_i = 1$. Next we define a parameter $\tau \in (0, 1)$ which sets an approximation threshold, so that the sum of the normalized singular values is no less than τ. Intuitively, τ is a parameter which allows for convenient control of the approximation quality of the low-rank decomposition. For example, a matrix with only one large singular value, and many other negligibly small singular values has an effective rank of 1, so we will probably be able to satisfy the ϵ–bound with a rank 1 approximation of that matrix, whereas for a matrix with many singular values of similar magnitude, our approximation must be close to full rank. To represent each layer with as few parameters as possible, we define a per-layer truncation threshold $k(\tau)$:

$$k(\tau) = \min\{k : \sum_{i=1}^{k} \bar{\sigma}_i \geq \tau\} \tag{2.13}$$

where the same threshold τ is used across all weight matrices. This lets us define a low-rank approximation operator $\tilde{R}(\theta, \tau)$ which returns the truncated singular value decomposition with effective rank controlled by τ. Note that if the effective rank of the matrix is large, there is no compression benefit to the spectral representation. An $m \times n$ matrix M normally requires mn parameters for representation, and the SVD

of rank k uses $k(m+n)$ parameters, so the low-rank approximation is only useful if $k(\tau) < \frac{mn}{m+n}$.

$$\tilde{R}(\theta_j, \tau) = \begin{cases} \left(\sigma^j_{i:k(\tau)}, u^j_{i:k(\tau)}, v^j_{i:k(\tau)}\right) & \text{if } k(\tau) < \frac{mn}{m+n} \\ \theta_j & \text{otherwise} \end{cases} \tag{2.14}$$

Where θ_j is the network parameter group of layer j, and $(\sigma^j_{i:k(\tau)}, u^j_{i:k(\tau)}, v^j_{i:k(\tau)})$ are the corresponding low-rank matrices of rank $k(\tau)$. Note that we can interpret the normalized singular values as a probability distribution. Then, define the *spectral entropy* of a matrix H_{svd},

$$H_{\text{svd}} = -\sum_i \bar{\sigma}_i \log \bar{\sigma}_i \tag{2.15}$$

Intuitively, H_{svd} tells us about the effective rank of the matrix by measuring the spread of the singular values. To see that, note that if all singular values are equal, H_{svd} is maximized, and if all but one of the singular values are zero, H_{svd} is minimized. Roy and Vetterli (2007) formalize this by proving that H_{svd} computes the effective rank of a matrix, $\text{rank}_{\text{eff}}(M) = \exp(H_{\text{svd}}(M))$, which measures the intrinsic dimension of the transformation parameterized by M. The spectral entropy not only provides a means of measuring the effective dimension of networks as training progresses—we can also explicitly penalize this quantity to encourage the network to learn simple, low-rank representations. In experiments, we show that our regularization method causes grokking, and produces the lowest-rank and most compressible networks of the methods we study.

Our full coarse-graining procedure is to perform a Bayesian optimization for the values of τ and Δ which produce the most compressed description, while satisfying the ϵ–bound. When the low-rank description is more compressible, we quantize the *decomposed* matrices. The full procedure is given in Algorithm 2. We emphasize that the low-rank decomposition is used for two separate purposes: 1) the spectral entropy penalty is used as a regularizer, and 2) the low-rank decomposition is used to obtain compressed descriptions of the model at the coarse-graining stage.

The regularizer we study in experiments adds the spectral entropy penalty scaled by β to the loss,

$$\mathcal{L}_{\text{reg}}(\theta) = \beta H_{\text{svd}}(\theta) \tag{2.16}$$

along with the noisy gradients described in section 2.3.1, plus weight decay via the AdamW optimizer.

Algorithm 2 Bayesian Optimization for Compression by Coarse-Graining

Require: Neural network parameters θ, number of optimization steps N, error tolerance ϵ

Ensure: Optimal compressed size

Initialize Bayesian Optimizer BO
best_compressed_size $\leftarrow \infty$
for $i = 1$ **to** N **do**
    $\tau_i, \Delta_i \leftarrow BO.\textsc{SuggestParameters}()$
    $\theta_R \leftarrow R(\theta, \tau_i)$ ▷ Low-rank approximation
    $\theta_{RQ} \leftarrow Q(\theta_R, \Delta_i)$ ▷ Quantization
    if $|\textsc{Loss}(\theta_{RQ}) - \textsc{Loss}(\theta)| < \epsilon$ **then**
        compressed_size $\leftarrow \textsc{compress}(\theta_{RQ})$
        if compressed_size $<$ best_compressed_size **then**
            best_compressed_size $\leftarrow$ compressed_size
        end if
        $BO.\textsc{UpdateModel}(\tau_i, \Delta_i, \text{compressed\_size})$
    else
        $BO.\textsc{UpdateModel}(\tau_i, \Delta_i, \infty)$ ▷ Penalize invalid solutions
    end if
end for
return best_compressed_size

2.4 Experiments

To study the complexity dynamics of networks transitioning from memorizing to generalizing solutions, we adopt the "grokking" tasks first reported in Power et al., 2022. These are a set of modular arithmetic tasks which networks easily over-fit, quickly achieving zero training error while test error remains large, suggesting memorization of the training examples. After many further steps of gradient descent after over-fitting, networks suddenly generalize and achieve perfect test accuracy. Although grokking is a sign of improperly regularized training, the extended delay between memorization and generalization is convenient for understanding complexity dynamics, and their relationship with generalization.

We demonstrate a regularizer, $\mathcal{L}_{\text{reg}}$ on the grokking tasks and show that 1) it causes grokking 2) it results in lower complexity networks than weight decay alone, the most widely-used regularization method, as measured by our complexity metric.

We reproduce a subset of the grokking experiments first reported in Power et al., 2022. Our training tasks all consist of learning a binary operation mod a prime number $p = 113$. For each binary operation we construct a dataset of equations of the form $\langle x \rangle \langle \text{op} \rangle \langle y \rangle \langle = \rangle \langle x \circ y \rangle$, where $\langle a \rangle$ stands for the token corresponding to element a. We

train standard decoder-only transformers with a single layer for mod addition and multiplication, and two layers for mod subtraction and division. Further training details can be found in the appendix. We use Algorithm 2 to compute complexity upper-bounds on the coarse-grained models. The final compression step is performed by `bzip2`, an off-the-shelf compression utility.

First, we establish that grokking occurs in all regularized models, and does not occur in unregularized models in Figure 2.8. Next, we plot accuracy and complexity together for the regularized models in Fig 2.2, which shows our main result: as expected from Occam's Razor, complexity is maximized during memorization and falls during generalization, demonstrating a clear complexity phase transition. In contrast, Fig 2.3 demonstrates that without regularization, models tend to stay in the high-complexity memorization regime indefinitely. Next, we plot algorithmic rate–distortion curves of best fit in Fig 2.4. We fit curves of the form $y = a\log(x) + b$, and display R^2 values in the legend. These curves show how lossy *model* compression trades off with lossless *data* compression for the converged models at varying distortion bounds. Fig 2.5 shows how the complexity dynamics change under varying distortion levels ϵ: we see that as the distortion bound is loosened (larger ϵ), the corresponding complexity falls for the same model, as expected from the theory of rate–distortion.

Plotting the generalization bound from Eqn. 2.4, we see in Fig 2.7 the models trained with our regularizer come closest to a non-trivial generalization bound: they achieve the least sum of data entropy and model complexity. Fig B.3 plots the total description length of the dataset under each model, which mirrors the generalization bound. While our method reveals a clear complexity phase transition in these models, the empirical performance of this method could be improved. Heuristics such as the effective dimension (Maddox, Benton, and Wilson, 2020) demonstrate that even models with parameter counts in excess of 10^7 should admit compression down to effective parameter counts of only around 10^2, even in tasks such as image classification which are presumably more complex than the modular arithmetic tasks we study here. Lotfi et al. (2024) demonstrate that non-vacuous generalization bounds for LLMs of order 10^8 parameters can be achieved by pre-training in a compressed subspace with LoRA (Hu et al., 2021). One benefit of the approach we present is that it can be applied to already-trained models, unlike subspace LoRA methods whose compressibility relies on keeping the initialization seed and training only in the subspace. If normally trained models still admit a low-rank decomposition, one promising future approach to achieve strong generalization bounds is to discover the projection down to the learned low-dimensional subspace, e.g. with the use of 2^{nd} order estimators such as

the Hessian, without requiring that training is restricted to that subspace from the outset.

2.5 Conclusion

Just as physical systems can transition between distinct phases of matter, we show that learning systems can transition between distinct algorithmic phases, which are characterized by their intrinsic complexity. We have demonstrated that a complexity phase transition is the central mechanism driving the grokking phenomenon; this contrasts with prior works, which focus on more specific signatures such as weight norms (Varma et al., 2023) or linear structure (Liu, Kitouni, et al., 2022). We justify our analysis on theoretical grounds by linking our lossy compression metric to generalization bounds based on the Solomonoff prior (Lotfi et al., 2024). From the algorithmic complexity perspective, we can see that small weight norms or linear structure are different ways for complexity to manifest in networks, but these metrics alone are insufficient explanations since complexity can emerge through any number of mechanisms. For example, we showed how the effective rank of the network is one means by which complexity can emerge, which we penalize through the spectral entropy measure we introduced in Section 2.3.2.

The distinctive separation between the memorizing and generalizing phases makes the grokking phenomenon an ideal model system to initiate the study of complexity phase transitions in learning systems. A related pathological phenomenon observed in learning systems is *double descent* (Nakkiran et al., 2019), in which test error first falls, then rises, then falls again as a function of model size, with training steps fixed. Maddox, Benton, and Wilson (2020) study the effective dimension of networks throughout double descent. They find that the effective dimension is greatest as the models memorize and overfit the training data, suggesting a strong similarity to grokking. A unified theory of complexity and generalization should predict when and under which conditions such training dynamics occur, and may suggest avenues for further progress in the theory of machine learning. Generalizing our results to account for arbitrary learning systems could lead to a deeper understanding of the interplay between optimization dynamics, the loss landscape, data, model capacity, and complexity.

Equipped with such a theory, practitioners could predict the performance of machine learning systems before running costly experiments, and insights gained from a more complete theory of machine learning could enable new breakthroughs in our

understanding of complex systems more broadly. The emerging field of algorithmic thermodynamics (Ebtekar and Hutter, 2025; Baez and Stay, 2012; Kolchinsky and Wolpert, 2020) provides a promising mathematical and computational framework with which to analyze these systems. The rise and fall of complexity we report in this work is conjectured to be a generic property of certain complex dynamical systems (Aaronson, Carroll, and Ouellette, 2014). Understanding the origin of this phenomenon in greater detail could help explain not only the emergence of abstraction and generalization in learning systems, but the nature of the algorithmic structure we observe in the natural world more broadly.

The grokking phenomenon lends itself to careful scientific study due to its simplicity. In the next chapters, we study richer problems of greater complexity, ultimately addressing problems of generalization to the real world.

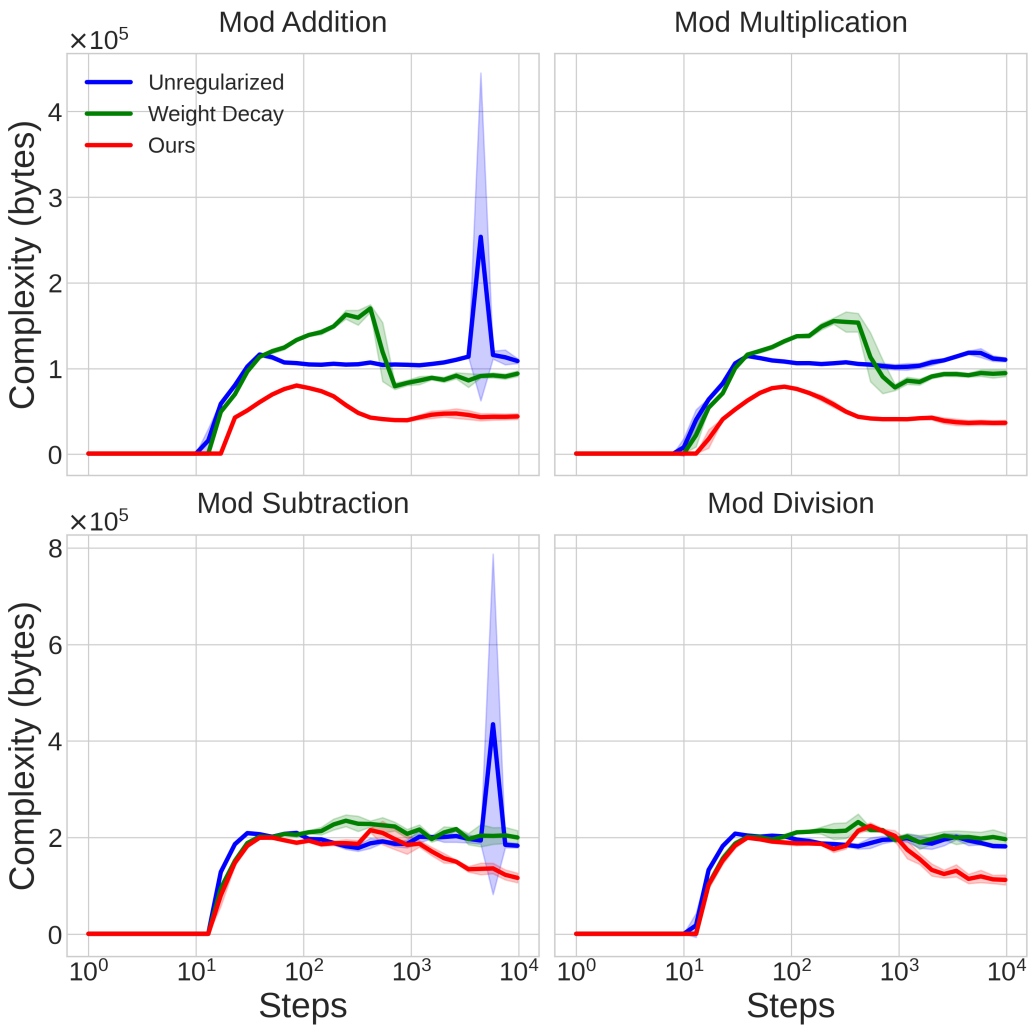

Figure 2.6: Complexity estimates using quantization coarse-graining alone, according to Algorithm 1. Detecting the complexity dynamics requires relatively tight complexity bounds. While we see a similar rise and fall pattern in the regularized models of the top row, this method fails to distinguish these dynamics in the weight decayed models for subtraction and division. Note that the complexity bounds are about two times worse than those achieved by the spectral method in Fig 2.2.

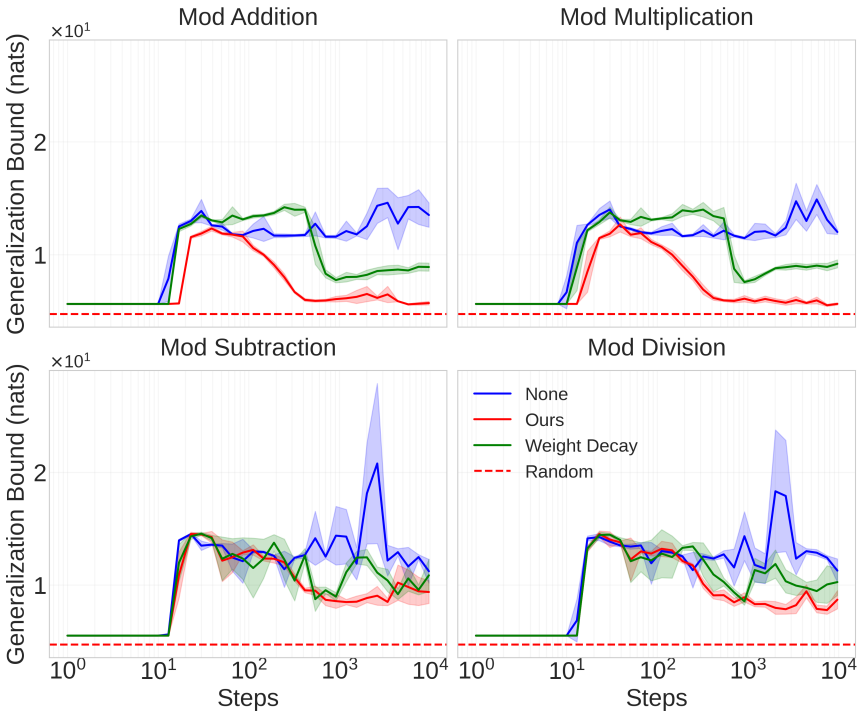

Figure 2.7: We plot the generalization bound on the expected risk according to Eqn. 2.4. While no models achieve low enough complexity to give non-vacuous bounds, our regularization method achieves the best bounds in all cases. It is not surprising that we can only achieve a vacuous bound, since the models are vastly over-parameterized, and the dataset small. Complexity estimates could be improved by using separate per-layer quantization bin sizes Δ.

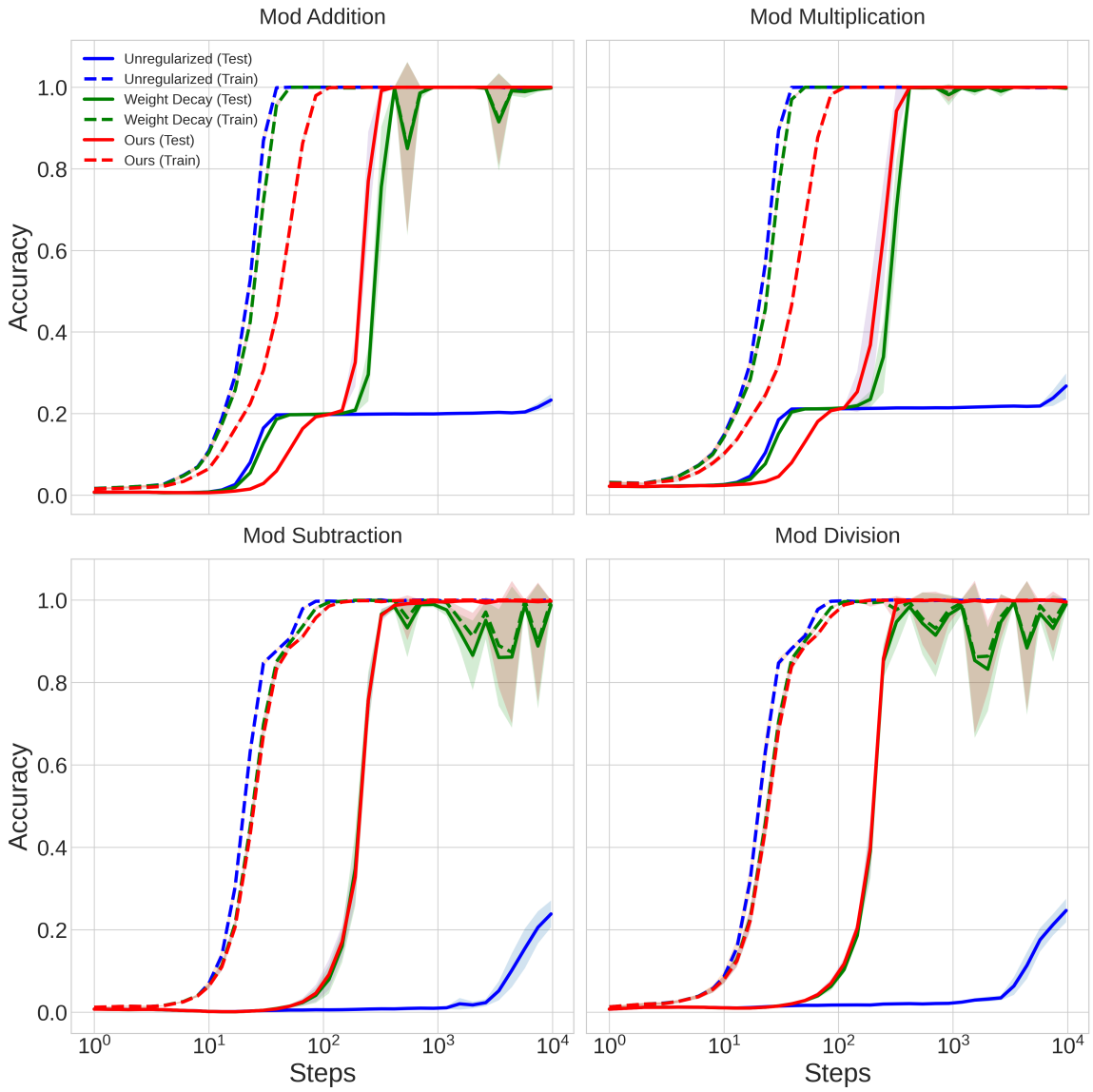

Figure 2.8: Test and train accuracy vs steps. Grokking occurs in all regularized models, and does not occur in unregularized models. We plot the mean over six seeds, and shade the std. error.

Chapter 3

Generalization with World Models

In the last chapter, we studied generalization in a supervised learning setting with low-dimensional, iid data. In this chapter, we address generalization challenges in non-iid settings with high-dimensional data. In particular, multi-step decision environments, which introduce dependencies between predicted actions at one step, and observations at future steps. With an eye towards applications in robotics, we consider the problem of *offline* policy learning, since online learning can be dangerous, expensive, or infeasible for a variety of reasons. Our approach is to first learn a transition model for the environment from sequences of (state, action) pairs from an offline dataset. Then, we perform *online* reinforcement learning in the learned transition model, where the reward function penalizes the agent for deviating from expert trajectories over multiple steps. This formulation mitigates the well-known compounding error problem associated with offline policy learning methods, and results in state-of-the-art performance in the environments we study.

To encourage generalization, we take the contemporary approach popular especially with LLMs, and "bring everything into distribution" through large-scale reinforcement learning in the learned transition model. This means that despite the fact that the agent has not learned online in the true environment, it has learned on-policy. Because of this, there is minimal distribution mismatch between the train and test environments. The world model's latent space also provides compressed, low-dimensional representations of the high-dimensional image space, so that the RL problem is decoupled from the image representation-learning problem.

3.1 Introduction

Imitation learning (IL) is an approach to policy learning which bypasses the reward specification issue in reinforcement learning by directly mimicking the behavior of

an expert demonstrator. The simplest kind of IL, behavior cloning (BC), trains an agent to predict an expert's actions from observations, then acts on these predictions at test time. However, this approach fails to account for the sequential nature of decision problems, since decisions at the current step affect which states are seen later, breaking the IID assumption of standard supervised learning algorithms. The distribution of states seen at test time will differ from those seen during training unless the expert training data covers the entire state space, or the agent makes no mistakes. This distribution mismatch, or covariate shift, leads to a compounding error problem: initially small prediction errors lead to small changes in state distribution, which lead to larger errors, and eventual departure from the training distribution altogether (Pomerleau, 1989). Intuitively, the agent has not learned how to act under the distribution induced by its own policy. This was formalized in the seminal work of Ross and Bagnell, 2010, who gave a tight regret bound on the difference in return achieved by expert and learner, which is quadratic in the episode length for BC.

Follow-up work in Ross, Gordon, and Bagnell, 2011 showed that a linear bound on regret can be achieved if the agent learns online in an interactive setting with the expert: Since the agent is trained *under its own distribution* with expert corrections, there is no distribution mismatch at test-time. This works well when online learning is safe and expert supervision is possible, but is untenable in many real-world use-cases such as robotics, where online learning can be unsafe, time-consuming, or otherwise infeasible. This captures a major open challenge in imitation learning: on one hand, we want to generate data on-policy to avoid covariate shift, but on the other hand, we may not be able to learn online due to safety or other concerns. The algorithm we present, DITTO, solves this *offline imitation learning* challenge in a way that scales to high-dimensional observations such as pixels. To the best of our knowledge, we are the first to solve these Atari benchmarks in the imitation learning setting where we train completely offline, from pixels alone, and consistently recover expert performance.

D. R. Ha and Schmidhuber, 2018 propose a two-stage approach to policy learning, where agents first learn to predict the environment dynamics with a recurrent neural network called a "world model" (WM), and then learn the policy inside the WM alone. This approach is desirable since it enables on-policy learning *offline*, in the world model. Similar model-based learning methods have recently achieved success in the adjacent setting of online RL (Hafner, Lillicrap, Norouzi, et al., 2021), and impressive zero-shot transfer of policies trained solely in the WM to physical robots (Wu et al., 2023).

When learning a policy, it is important to quantify how well the policy generalizes to unseen inputs. However, in imitation learning there is a conceptual difficulty in measuring generalization performance: Although we could evaluate policy performance on held-out expert state-action pairs (e.g. by measuring prediction accuracy), this fails to reflect the performance we should expect from the policy at test-time, because we are not evaluating the policy in the state distribution it will be acting under, namely *its own induced distribution*. World models offer a solution to this problem: given a learned dynamics model, we can perform roll-outs with our learned policy *from expert starting states*, and measure the divergence in the latent space *over multiple time-steps*. This gives us an offline, on-policy divergence measure which captures the sequential nature of the imitation learning decision problem. This is the key insight underlying the strong empirical performance of DITTO, and forms the basis of our theoretical contribution. To confirm the generalization properties of our algorithm, we evaluate in environments for which there is an extrinsic reward function (which the imitation learner does not have access to), and study the relationship between our proposed divergence measure and the extrinsic reward. As shown in Figure 3.2, we find that off-policy metrics like expert action prediction accuracy are not predictive of final return in the environment, whereas our on-policy latent divergence measure is predictive of extrinsic return.

We combine the above insights to propose a new imitation learning algorithm called **Dream Imitation (DITTO)**. DITTO leverages learned world models in a novel way to induce reward-free imitation learning: first, expert trajectories are mapped into the model latent space; then, an imitation policy is unrolled from arbitrary expert start states in the model latent space. Finally, we directly compare the *on-policy* rollouts to the expert trajectories in the model latent space using a simple, fixed distance function defined in the latent space. Our key insight is that the world model latent space provides a natural trajectory divergence measure which we can do imitation learning over, without any intermediary learned reward model or discriminator. This enables robust imitation learning in a domain-agnostic way. Using this divergence measure, DITTO casts the offline imitation learning challenge as an online RL problem in a learned world model.

Our main contributions are summarized as follows:

- We introduce DITTO, a novel imitation learning framework which leverages learnt world models to cast offline imitation as online RL in the model latent space.

- We show that the latent space induced by dynamics learning provides a natural and generic divergence measure for policy learning. The latent distance function we demonstrate is simpler, more robust, and results in more performant policy learning compared to learned discriminators, such as those used in adversarial imitation learning.
- We demonstrate DITTO outperforms standard imitation learning baselines by a significant margin - to the best of our knowledge we are the first to consistently recover expert performance in the Atari benchmarks we study from pixels, completely offline.
- To provide a more comprehensive evaluation of the thus far underexplored area of offline imitation learning from pixels, we present two novel extensions of baseline IL algorithms (BC, GAIL) to the world model, offline setting (D-BC, D-GAIL).

3.2 Related Work

3.2.1 Imitation Learning

Imitation learning algorithms can be classified according to the set of resources needed to produce a good policy. Ross, Gordon, and Bagnell (2011) give strong theoretical and empirical results in the online interactive setting, which assumes that we can both learn while acting online in the real environment, and that we can interactively query an expert policy to e.g. provide the learner with the optimal action in the current state. Follow-up works have progressively relaxed the resource assumptions needed to produce good policies. Sasaki and Yamashina (2021) show that the optimal policy can be recovered with a modified form of BC when learning from imperfect demonstrations, given a constraint on the expert sub-optimality bound. Brantley, Sun, and Henaff (2020) study covariate shift in the online, non-interactive setting, and demonstrate an approximately linear regret bound by jointly optimizing the BC objective with a novel policy ensemble uncertainty cost, which encourages the learner to return to and stay in the distribution of expert support. They achieve this by augmenting the BC objective with the following uncertainty cost term:

$$\mathrm{Var}_{\pi \sim \Pi_E}\left(\pi(a|s)\right) = \frac{1}{E}\sum_{i=1}^{E}(\pi_i(a|s) - \frac{1}{E}\sum_{j=1}^{E}\pi_j(a|s))^2 \tag{3.1}$$

This term measures the total variance of a policy ensemble $\Pi_E = \{\pi_1, ..., \pi_E\}$ trained on disjoint subsets of the expert data. They optimize the combined BC plus uncertainty objective using standard online RL algorithms, and show that this mitigates covariate shift.

Inverse reinforcement learning (IRL) can achieve improved performance over BC by first learning a reward from the expert demonstrations for which the expert is optimal, then optimizing that reward with on-policy reinforcement learning. This two-step process, which includes on-policy RL in the second step, helps IRL methods mitigate covariate shift due to train and test distribution mismatches. However, the learned reward function can fail to generalize outside of the distribution of expert states which form its support.

One line of work treats IRL as *divergence minimization*: instead of directly copying the expert actions, they minimize a divergence measure between expert and learner state distributions

$$\min_{\pi} \mathbb{D}\left(\rho^{\pi}, \rho^{E}\right) \tag{3.2}$$

where $\rho^{\pi}(s, a) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t P(s_t = s, a_t = a)$ is the discounted state-action distribution induced by π, and $\mathbb{D}$ is a divergence measure between probability distributions. The popular GAIL algorithm (Ho and Ermon, 2016) constructs a minimax game in the style of GANs (Goodfellow et al., 2014) between the learner policy π, and a discriminator D_{ψ} which learns to distinguish between expert and learner state distributions

$$\max_{\pi} \min_{D_{\psi}} \mathbb{E}_{(s,a)\sim\rho^{E}} \left[-\log D_{\psi}(s, a)\right] + \mathbb{E}_{(s,a)\sim\rho^{\pi}} \left[-\log\left(1 - D_{\psi}(s, a)\right)\right] \tag{3.3}$$

This formulation minimizes the Jensen-Shannon divergence between the expert and learner policies, and bounds the expected return difference between agent and expert. However, Wang et al. (2019) point out that adversarial reward learning is inherently unstable since the discriminator is always trained to penalize the learner state-action distribution, even if the learner has converged to the expert policy. This finding is consistent with earlier work (Brock, Donahue, and Simonyan, 2019) which observed discriminator overfitting, necessitating early stopping to prevent training collapse. Multiple works have reported failure getting GAIL to work with high-dimensional observations, such as those in the pixel-based environments we study (Brantley, Sun, and Henaff, 2020) (Reddy, Dragan, and Levine, 2020).

To combat problems with adversarial training, Wang et al. (2019) and Reddy, Dragan, and Levine (2020) consider reducing IL to RL on an intrinsic reward

$$r(s,a) = \begin{cases} 1 & \text{if } (s,a) \in \mathcal{D}^E \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

where $\mathcal{D}^E$ is the expert dataset. While this sparse formulation is impractical e.g. in continuous action settings, they show that a generalization of the intrinsic reward using support estimation by random network distillation (Burda et al., 2019) results in stable learning that matches the performance of GAIL without adversarial training. Ciosek (2022) showed that this formulation is equivalent to divergence minimization under the total variation distance, and produced a bound on the difference in extrinsic reward achieved between the expert and a learner trained with this approach.

3.2.2 World Models

World models have recently emerged as a promising approach to model-based learning. D. R. Ha and Schmidhuber (2018) defined the prototypical two-part model: a variational autoencoder (VAE) is trained to reconstruct observations from individual frames, while a recurrent state-space model (RSSM) is trained to predict the VAE encoding of the next observation, given the current latent state and action. World models can be used to train agents entirely inside the learned latent space, without the need for expensive decoding back to the observation space. Hafner, Lillicrap, Ba, et al. (2020) introduced Dreamer, an RL agent which is trained purely in the latent space of the WM, and successfully transfers to the true environment at test-time. Wu et al. (2023) showed that the same approach can be used to simultaneously learn a model and agent policy to control a physical quadrupedal robot online, without the control errors usually associated with transferring policies trained only in simulation to a physical system (Hwangbo et al., 2019).

In this chapter, we propose the use of world models to address a number of common problems in imitation learning. Intrinsic rewards which induce imitation learning, like those introduced in Reddy, Dragan, and Levine (2020) and Wang et al. (2019), can pose challenging online learning problems, since the rewards are sparse or require tricky additional training procedures to work in high-dimensional observation spaces. Similarly, approaches like GAIL (Ho and Ermon, 2016) and AIRL (Fu, K. Luo, and Levine, 2018) require adversarial on-policy training that is difficult to make work in practice. Rafailov et al., 2021 propose an approach similar to ours, which uses model-based rollouts to produce on-policy latent trajectories to train the policy. However,

their reward model is learned via an adversarial objective, and so can in principle suffer from the same adversarial collapse issues mentioned above. In contrast, our approach remedies both the online learning and reward specification problems by performing on-policy learning offline, in the latent space of the world model, and uses a natural divergence measure as reward: distance between learner and expert in the world model latent space. This provides a conceptually simple and dense reward signal for imitation by reinforcement learning, which we find outperforms competitive approaches in data efficiency and asymptotic performance.

3.3 Dream Imitation

We study imitation learning in a partially observable Markov decision process (POMDP) with discrete time-steps and actions, and high dimensional observations generated by an unknown environment. The POMDP $\mathcal{M}$ is composed of the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{X}, \mathcal{R}, \mathcal{T}, \mathcal{U}, \gamma)$, where $s \in \mathcal{S}$ is the state space, $a \in \mathcal{A}$ is the action space, $x \in \mathcal{X}$ is the observation space, γ is the discount factor, and $r = \mathcal{R}(s, a)$ is the reward function. The transition dynamics are Markovian, and given by $s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, a_t)$. The agent does not have access to the underlying states, and only receives observations represented by $x_t \sim \mathcal{U}(\cdot \mid s)$. The goal is to maximize the discounted sum of extrinsic (environment) rewards $\mathbb{E}[\Sigma_t \gamma^t r_t]$, which the agent does not have access to.

Training proceeds in two parts: we first learn a world model from recorded sequences of observations, then train an actor-critic agent to imitate the expert in the world model. The latent dynamics of the world model define a fully observable Markov decision process (MDP), since the model states $\hat{s}_t$ are Markovian. Model-based rollouts always begin from an observation drawn from the expert demonstrations, and continue for a fixed set of time steps H, the agent training horizon. The agent is rewarded for matching the latent trajectory of the expert.

3.3.1 Preliminaries

We show that bounding the learner-expert state distribution divergence in the world model also bounds their return difference in the actual environment, and connect our method to the *IL as divergence minimization* framework (Ghasemipour, Zemel, and Gu, 2019). Rafailov et al. (2021) showed that for a learned dynamics model $\widehat{\mathcal{T}}$ whose total variation from the true transitions is bounded such that $\mathbb{D}_{\mathrm{TV}}(\mathcal{T}(s, a), \widehat{\mathcal{T}}(s, a)) \leq$

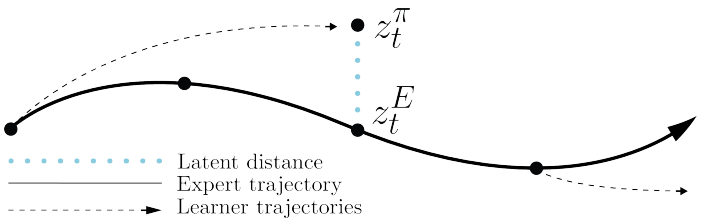

Figure 3.1: The learner begins from random expert latent states during training, and generates on-policy latent trajectories in the world model. The intrinsic reward in Eq. 3.8 encourages the learner to recover from its mistakes over multiple time steps to match the expert trajectory.

$\alpha \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}$ and $R_{\max} = \max_{(s,a)} \mathcal{R}(s,a)$ then

$$\left|\mathcal{J}(\pi^E, \mathcal{M}) - \mathcal{J}(\pi, \mathcal{M})\right| \leq \underbrace{\alpha \frac{R_{\max}}{(1-\gamma)^2}}_{\text{learning error}} + \underbrace{\frac{R_{\max}}{1-\gamma} \mathbb{D}_{\text{TV}}\left(\rho^E_{\mathcal{M}}, \rho^\pi_{\widehat{\mathcal{M}}}\right)}_{\text{adaptation error}} \tag{3.5}$$

where $\mathcal{J}(\pi, \mathcal{M})$ is the expected return of policy π in MDP $\mathcal{M}$, and $\widehat{\mathcal{M}}$ is the "imagination MDP" induced by the world model. This implies the difference between the expert return and the learner return in the true environment is bounded by two terms, 1) a term proportional to the model approximation error α, which could in principle be reduced with more data, and 2) a model domain adaptation error term, which captures the generalization error of a model trained under data from one policy, and deployed under another. Rafailov et al. (2021) also show that bounding the divergence between *latent* distributions upper bounds the true state distribution divergence. Formally, given a latent representation of the transition history $z_t = q(x_{\leq t}, a_{<t})$ and a belief distribution $P(s_t \mid x_{\leq t}, a_{<t}) = P(s_t \mid z_t)$, then if the policy conditions only on the latent representation z_t such that the belief distribution is independent of the current action $P(s_t \mid z_t, a_t) = P(s_t \mid z_t)$, then the divergence between the latent state distribution of the expert and learner upper bounds the divergence between their true state distribution:

$$\mathbb{D}_f(\rho^\pi_{\mathcal{M}}(x,a) \parallel \rho^E_{\mathcal{M}}(x,a)) \leq \mathbb{D}_f(\rho^\pi_{\mathcal{M}}(s,a) \parallel \rho^E_{\mathcal{M}}(s,a)) \leq \mathbb{D}_f(\rho^\pi_{\mathcal{M}}(z,a) \parallel \rho^E_{\mathcal{M}}(z,a)) \tag{3.6}$$

Where $\mathbb{D}_f$ is a generic f-divergence , e.g. KL or TV. This result, along with equation 3.5, suggests that minimizing divergence in the model latent space is sufficient to bound the expected expert-learner return difference.

Reward To bound expert-learner state distribution divergences, prior approaches have focused on sparse indicator function rewards (Ciosek, 2022), or adversarial reward learning (Ghasemipour, Zemel, and Gu, 2019). We propose a new formulation, which

rewards the agent for matching the expert latent state-action pairs over an episode. In particular, for an arbitrary distance function d, agent state-action latent z_t^π, and a set of expert state-action latents $\mathcal{D}_E$:

$$r_t^{int}(z_t^\pi) = 1 - \min_{z^E \in \mathcal{D}_E} d(z_t^\pi, z^E) \tag{3.7}$$

Any function of this form rewards matching the agent's state-action pairs to the expert's, as studied in Ciosek (2022). The major differences in our formulation are that we calculate the reward on the learned model latent states, as well as compute a simple smoothed divergence, meaning an exact match isn't required for a reward. Proof C.1 shows how to make this relaxed reward compatible with the theoretical results from Ciosek (2022), such that an exact divergence bound is obtained. In particular, we prove that maximizing this reward bounds the total variation in latent-state distributions between the expert and learner, as well as bounding their extrinsic reward difference.

Intuitively, matching latent states between the learner and expert is easier than matching observations, since the representations learned from generative world model training should provide a much richer signal of state similarity. In practice, the minimization over $\mathcal{D}_E$ can be computationally expensive, so we modify the objective 3.7 to exactly match learner latent states to expert latents from the same time-step, as shown in Figure 3.1. In particular, we randomly sample consecutive expert latents $z^E_{t:t+H}$ from $\mathcal{D}_E$ and unroll the agent from the same starting state in the world model, yielding a sequence of agent latents $z^\pi_{t:t+H}$. Finally, we compute a reward at each step t as follows:

$$r_t^{int}(z_t^E, z_t^\pi) = 1 - d(z_t^E, z_t^\pi) = \frac{z_t^E \cdot z_t^\pi}{\max(\|z_t^E\|, \|z_t^\pi\|)^2} \tag{3.8}$$

This formulation changes our method from distribution matching to mode seeking, since states frequently visited by the expert will receive greater reward in expectation. We found that this modified dot product reward empirically outperformed L_2 and cosine-similarity metrics.

3.3.2 World Model Architecture

We adapt the recurrent state space model (RSSM) introduced by Hafner, Lillicrap, Norouzi, et al., 2021. The full world model is composed of an image encoder, a recurrent state-space model (RSSM) which learns the transition dynamics, and a

Algorithm 3 Dream Imitation (DITTO)

Require demonstration data $\mathcal{D} = \left\{ (x_t, a_t, x_{t+1})_{t=0}^{\|e_n\|} \mid n \in N \right\}$
Initialize world model parameters ϕ
while *not converged* **do** ▷ World model learning
    Draw B_{wm} transition sequences $\{(x_t, a_t, x_{t+1})_{t=k}^{k+L}\} \sim \mathcal{D}$
    Compute all sequential RSSM components according to eqn 3.9
    Update ϕ with ELBO loss 3.10
end while
Initialize actor and critic parameters θ, ψ
while *not converged* **do** ▷ Agent training
    Draw B_{ac} expert latent state sequences $(\hat{s}_\tau^E) \sim \hat{\mathcal{D}}^E$
    Generate trajectories $(\hat{s}_\tau^\pi, a_\tau)_{\tau=t}^{t+H}$ with $a_\tau \sim \pi_\theta(\cdot \mid \hat{s}_\tau)$
    Compute rewards $r_\tau^{\text{int}}(\hat{s}_\tau^\pi, \hat{s}_\tau^E)$ and values $v_\psi(\hat{s}_\tau^\pi)$
    Compute λ-returns $V_\tau^\lambda = r_t + \gamma \left((1-\lambda) v(\hat{s}_{\tau+1}^\pi) + \lambda V_{\tau+1}^\lambda \right), \quad V_{\tau+H}^\lambda = v(\hat{s}_{\tau+H}^\pi)$
    Update critic on λ-targets: $\sum_{\tau=t}^{t+H} \frac{1}{2} (v_\psi(\hat{s}_\tau^\pi) - sg(V_\tau^\lambda))^2$
    Update actor with eqn 3.15
end while

decoder which reconstructs observations from the compact latent states.

$$
\begin{aligned}
&\text{Model state:} && \hat{s}_t = (h_t, z_t) \\
&\text{Recurrent state:} && h_t = f_\phi(\hat{s}_{t-1}, a_{t-1}) \\
&\text{Prior predictor:} && \hat{z}_t \sim p_\phi(\hat{z}_t \mid h_t) \\
&\text{Posterior predictor:} && z_t \sim q_\phi(z_t \mid h_t, x_t) \\
&\text{Image reconstruction:} && \hat{x}_t \sim p_\phi(\hat{x}_t \mid \hat{s}_t)
\end{aligned}
\tag{3.9}
$$

All components are implemented as neural networks, with a combined parameter vector ϕ. The encoder uses a convolutional neural network (CNN) to produce representations, while the decoder is a transposed CNN. The RSSM predicts a sequence of length T deterministic recurrent states $(h_t)_{t=0}^T$, each of which are used to parameterize two distributions over stochastic hidden states. The stochastic posterior state z_t is a function of the current observation x_t and recurrent state h_t, while the stochastic prior state $\hat{z}_t$ is trained to match the posterior without access to the current observation. The current observation is reconstructed from the full model state, which is the concatenation of the deterministic and stochastic states $\hat{s}_t = (h_t, z_t)$.

Since the prior model predicts the current model state using only the previous action and recurrent state, without using the current observation, we can use it to learn behaviors without access to observations or decoding back into observation space. The prior and posterior models predict categorical distributions which are optimized

with straight-through gradient estimation (Bengio, Léonard, and Courville, 2013). All components of the model are trained jointly with a modified ELBO objective:

$$\min_{\phi} \mathbb{E}_{q_\phi(z_{1:T}|a_{1:T},x_{1:T})}\left[\sum_{t=1}^{T} -\log p_\phi(x_t \mid \hat{s}_t) + \beta D_{\text{KL-B}}(q_\phi(z_t \mid \hat{s}_t) \parallel p_\phi(\hat{z}_t \mid h_t))\right] \quad (3.10)$$

where $D_{\text{KL-B}}(q \parallel p)$ denotes KL balancing (Hafner, Lillicrap, Norouzi, et al., 2021), which is used to control the regularization of prior and posterior towards each other with a parameter δ,

$$D_{\text{KL-B}}(q \parallel p) = \delta \underbrace{D_{\text{KL}}(q \parallel sg(p))}_{\text{posterior regularizer}} + (1-\delta) \underbrace{D_{\text{KL}}(sg(q) \parallel p)}_{\text{prior regularizer}} \quad (3.11)$$

and $sg(\cdot)$ is the stop gradient operator. The idea behind KL balancing is that the prior and posterior should not be regularized at the same rate: the prior should update more quickly towards the posterior, which encodes strictly more information.

World model training can be performed using datasets generated by policies of any quality, since the model only predicts transition dynamics. The transition dataset is composed of N episodes e_n of sequences of observations x_t, actions a_t: $\mathcal{D} = \{(x_t, a_t)_{t=0}^{\|e_n\|} \mid n \in N\}$.

3.4 Agent Architecture

The agent is composed of a stochastic actor which samples actions from a learned policy with parameter vector θ, and a deterministic critic which predicts the expected discounted sum of future rewards the actor will achieve from the current state with parameter vector ψ. The actor observes Markovian recurrent states $\hat{s}_t$ from the world model, and produces distributions over its action space, which we sample from to get actions.

$$\begin{aligned} &\text{Actor:} \quad a_t \sim \pi_\theta(a_t \mid \hat{s}_t) \\ &\text{Critic:} \quad v_\psi(\hat{s}_t) \approx \mathbb{E}_{\pi_\theta, p_\phi}[\Sigma_{t=0}^{H} \gamma^t r_t] \end{aligned} \quad (3.12)$$

We train the critic to regress the λ-target (Richard S. Sutton and Barto, 2005)

$$V_t^\lambda = r_t + \gamma\left((1-\lambda) v_\psi(\hat{s}_{t+1}) + \lambda V_{t+1}^\lambda\right), \quad V_{t+H}^\lambda = v_\psi(\hat{s}_{t+H}) \quad (3.13)$$

which lets us control the temporal-difference (TD) learning horizon with the hyperparameter λ. Setting $\lambda = 0$ recovers 1-step TD learning, while $\lambda = 1$ recovers unbiased Monte Carlo returns, and intermediate values represent an exponentially weighted

sum of n-step returns. In practice we use $\lambda = 0.95$. To train the critic, we regress the λ-target directly with the objective:

$$\min_{\psi} \mathbb{E}_{\pi_\theta, p_\phi} \left[\sum_{t=1}^{H-1} \tfrac{1}{2} (v_\psi(\hat{s}_t) - sg(V_t^\lambda))^2 \right] \tag{3.14}$$

There is no loss on the last time step since the target equals the critic there. We follow Mnih, Kavukcuoglu, et al. (2015), who suggest using a copy of the critic which updates its weights slowly, called the target network, to provide the value bootstrap targets.

The actor is trained to maximize the discounted sum of rewards predicted by the critic. We train the actor to maximize the same λ-target as the critic, and add an entropy regularization term to encourage exploration and prevent policy collapse. We optimize the actor using REINFORCE gradients (Williams, 2004) and subtract the critic value predictions from the λ-targets for variance reduction. The full actor loss function is:

$$\mathcal{L}(\theta) = \mathbb{E}_{\pi_\theta, p_\phi} \left[\sum_{t=1}^{H-1} \underbrace{-\log \pi_\theta(a_t \mid \hat{s}_t) sg(V_t^\lambda - v_\psi(\hat{s}_t))}_{\text{reinforce}} - \underbrace{\eta H(\pi_\theta(\hat{s}_t))}_{\text{entropy regularizer}} \right] \tag{3.15}$$

Algorithm Learning proceeds in two phases: First, we train the WM on all available demonstration data using the ELBO objective 3.10. Next, we encode expert demonstrations into the world model latent space, and use the on-policy actor critic algorithm described above to optimize the intrinsic reward in Eq. 3.8, which measures the divergence between agent and expert over time in latent space. In principle, any on-policy RL algorithm could be used in place of actor-critic. We describe the full procedure in Algorithm 3.

3.5 Experiments

To the best of our knowledge, we are the first to consistently recover expert performance in the pixel-based environments we study in the offline setting. Prior works generally focus on improving behavior cloning (Sasaki and Yamashina, 2021), or study a mixed setting with some online interactions allowed (Rafailov et al., 2021; Kidambi, Chang, and Sun, 2021). Other recent methods such as SMODICE (Y. J. Ma et al., 2022) and OTR (Y. Luo et al., 2023) only test on state-based environments. IQ-LEARN (Garg et al., 2021) tests on low-dimensional state in the fully offline setting, but uses *online interactions* for the three pixel-based environments they study. Surprisingly,

we were not able to find any fully offline methods which recover expert performance in these standard pixel-based Atari environments, pointing to a surprising potential limitation of current methods and a blind spot in the community. To demonstrate the effectiveness of world models for imitation learning, we train without any interaction with the true environment, nor any reward information.

Recent state-of-the-art imitation learning algorithms (Sasaki and Yamashina, 2021) (G.-H. Kim et al., 2022) (Kostrikov, Nachum, and Tompson, 2019) have mostly been limited in evaluation to low-dimensional perfect state observation environments. To test the effectiveness of world models for policy learning that can scale to partially-observable, high-dimensional observation environments, such as robotic manipulation from video feeds, we evaluate on difficult pixel-based environments. We test in standard pixel-based Atari environments considered by recent SOTA online methods, e.g. Brantley, Sun, and Henaff, 2020 (Reddy, Dragan, and Levine, 2020).

Because the world models we use take up to a week to train for each environment with our computing resources, we evaluate on a subset of the Atari domain for which strong baseline experts are available from the RL Baselines Zoo repository (Raffin, 2020), as well as a pixel-based continuous control environment. We did not perform any environment post-selection, and selected these environments only to align with prior work (Brantley, Sun, and Henaff, 2020).

3.5.1 Agents

To test the performance of our algorithm, we compare DITTO to a standard baseline method, behavior cloning, and to two methods which we introduce in the world model setting.

Behavior cloning We train a BC model end-to-end from pixels, using a convolutional neural network architecture. Compared to prior works which study behavior cloning from pixels in Atari games (Hester et al., 2017)(R. Zhang et al., 2020)(Kanervisto, Pussinen, and Hautamäki, 2020), our baseline implementation achieves stronger results, even in games where it is trained with lower-scoring data.

Dream agents We adapt GAIL (Ho and Ermon, 2016) and BC to the world model setting, which we dub D-GAIL and D-BC respectively. D-GAIL and D-BC both receive world model latent states instead of pixel observations. The D-BC agent is trained with maximum-likelihood estimation on the expert demonstrations in latent space, with an additional entropy regularization term which we found stabilized learning:

$$L_{BC} = \mathbb{E}_{(\hat{s},a)\sim\hat{\mathcal{D}}^E}\left[-\log\left(\pi(a|\hat{s})\right) - \eta_{BC}H(\pi(\hat{s}))\right] \tag{3.16}$$

The D-GAIL agent is trained on-policy in the world model using the adversarial objective from Equation 3.3. The D-GAIL agent optimizes its learned adversarial reward with the same actor-critic formulation used by DITTO, described in Section 3.4. D-GAIL is essentially identical to VMAIL, proposed by Rafailov et al., 2021, except that we use the world model of Dreamerv2 (Hafner, Lillicrap, Norouzi, et al., 2021). We train both DITTO and D-GAIL with a fixed horizon of $H = 15$. At test-time, the model-based agent policies are composed with the world model encoder and RSSM to convert high-dimensional observations into latent representations.

All model-based policies in our experiments use an identical multi-layer perceptron (MLP) architecture for fair comparison in terms of the policies' representation capacity, while the BC agent is parameterized by a stacked CNN and MLP architecture which mirrors the world model encoder plus agent policy. We found that D-GAIL was far more stable than expected, since prior works (Reddy, Dragan, and Levine, 2020) (Brantley, Sun, and Henaff, 2020) reported negative results training GAIL from pixels in the easier online setting. This suggests that world models may be beneficial for representation learning even in the online case, and that other online algorithms could be improved with world model pre-training, followed by policy training in the latent space.

We evaluate our algorithm and baselines on 5 Atari environments, and one continuous control environment, using strong PPO agents (Schulman et al., 2017) from the RL Baselines3 Zoo (Raffin, 2020) as expert demonstrators, using $N^E = \{4, 8, 15, 30, 60, 125, 250, 500, 1000\}$ expert episodes to train the agent policies in the world model. To train the world models, we generate 1000 episodes from a pre-trained policy, either PPO or advantage actor-critic (A2C) (Mnih, Badia, et al., 2016), which achieves substantially lower reward compared to PPO. Surprisingly, we found that the A2C and PPO-trained world models performed similarly, and that only the quality of the imitation episodes affected final performance. We hypothesize that this is because the A2C and PPO-generated datasets provide similar coverage of the environment. It appears that the world model can learn environment dynamics from broad classes of datasets as long as they cover the state distribution well. The data-generating policy's quality is relevant for imitation learning, but appears not to be for dynamics learning, apart from coverage.

3.5.2 Results

We are interested in pushing imitation learning towards real-world deployments, which necessitate dealing with high-dimensional observations and offline learning,

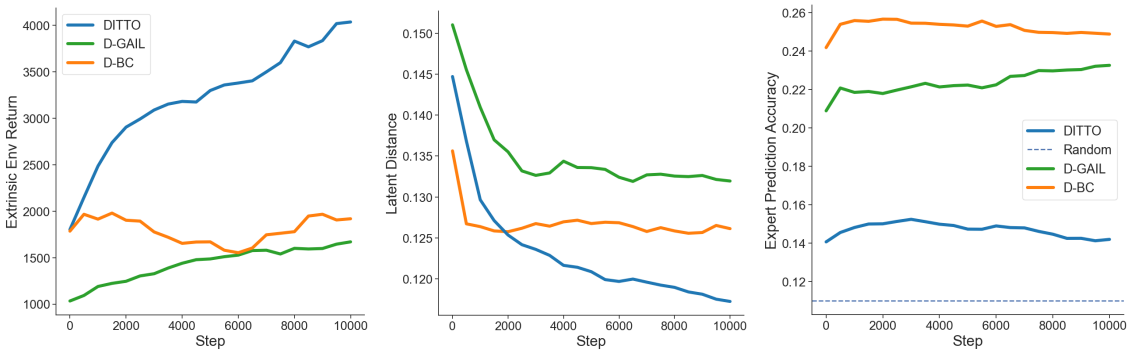

Figure 3.2: We compare mean extrinsic reward from rollouts in the true environment (BeamRider) throughout agent training (**left**) to agents' mean *latent distance* from the expert (**center**), and mean expert action prediction accuracy (**right**). Both latent distance and accuracy are calculated on held-out expert trajectories used for validation. Latent distance is defined as $L_d = 1 - r_{int}$. DITTO explicitly minimizes this quantity, and achieves the greatest generalization performance in the true environment. Perfect agreement with the expert would result in $L_d = 0$, but this is impossible to achieve since the world model is stochastic. Counter-intuitively, expert action prediction accuracy is *negatively* correlated with generalization performance in the true environment.

as mentioned in section 3. Estimating out-of-distribution imitation performance is particularly difficult in the offline setting, since by definition we do not have expert data there and cannot compare what our agent does to what an expert *would have done*. This highlights a flaw with standard offline imitation metrics such as expert action prediction accuracy, which only tell us about the learner's performance in the expert's distribution, and may not be predictive of the learner's performance under the distribution induced by its own policy.

Figure 3.2 shows the performance of different algorithms throughout training in the true environment, contrasted with two imitation metrics: latent distance, which we propose as a more robust measure of generalization performance for imitation; and expert action prediction accuracy, a standard imitation benchmark which is meant to capture generalization capability. DITTO achieves the lowest latent distance from expert under its own distribution in the world model. We find that counter-intuitively, *action prediction accuracy is negatively correlated with actual environment (i.e. extrinsic) performance*, whereas our latent distance measure *is* predictive of performance in the environment. This supports our hypothesis that metrics which are limited to evaluation in the expert distribution are inadequate for predicting the performance of imitation learners when deployed to the true environment, since they neglect the sequential nature of decision problems and the subsequent policy-induced

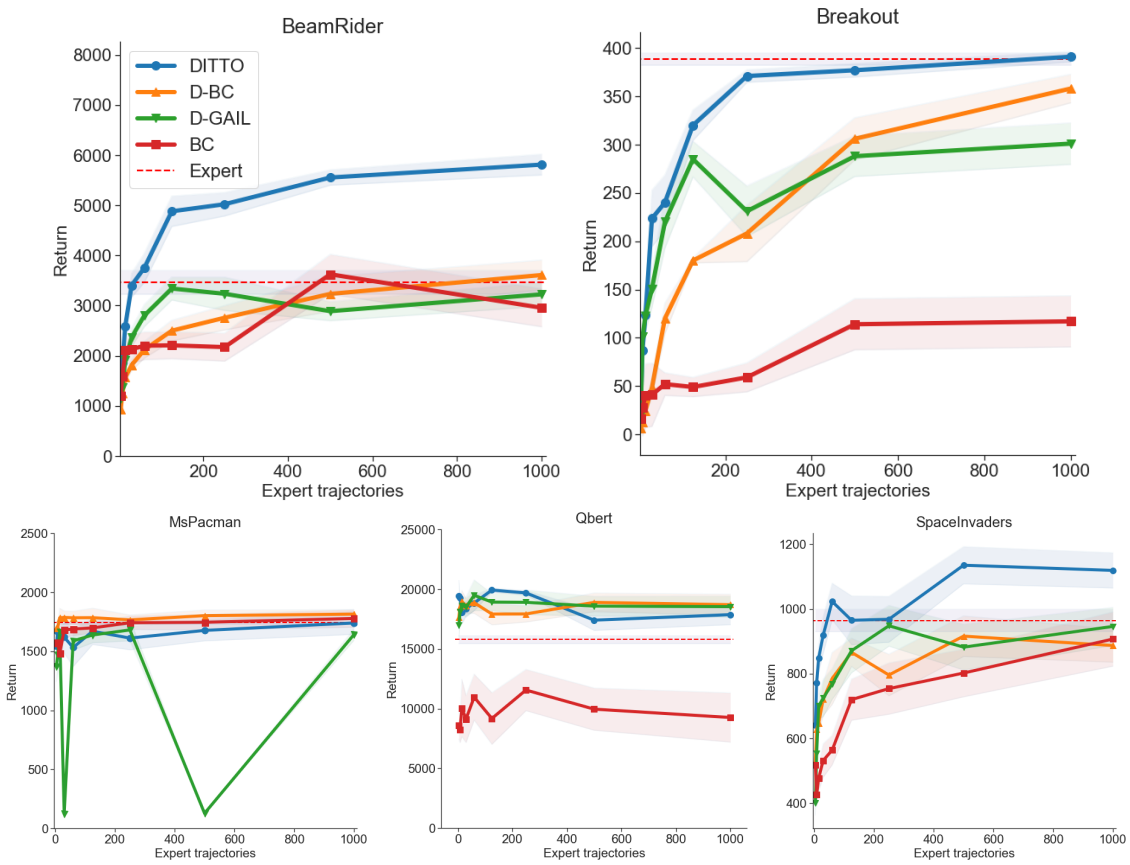

Figure 3.3: Results on five Atari environments from pixels, with fixed horizon $H = 15$. In all environments, DITTO matches or exceeds expert performance, and matches or exceeds all baselines. In MsPacman and Qbert, all model-based methods rapidly recover expert performance with minimal data, with statistically indistinguishable performance according to the standard error of the mean. In MsPacman, we observe adversarial collapse of D-GAIL. We follow R. Agarwal et al., 2021 for offline policy evaluation, and report the mean reward achieved across 10 gradient steps with 20 validation simulations, to avoid lottery-ticket policy results. Shaded regions show ± 1 standard error. The experts are strong pre-trained PPO agents from the RL Baselines3 Zoo.

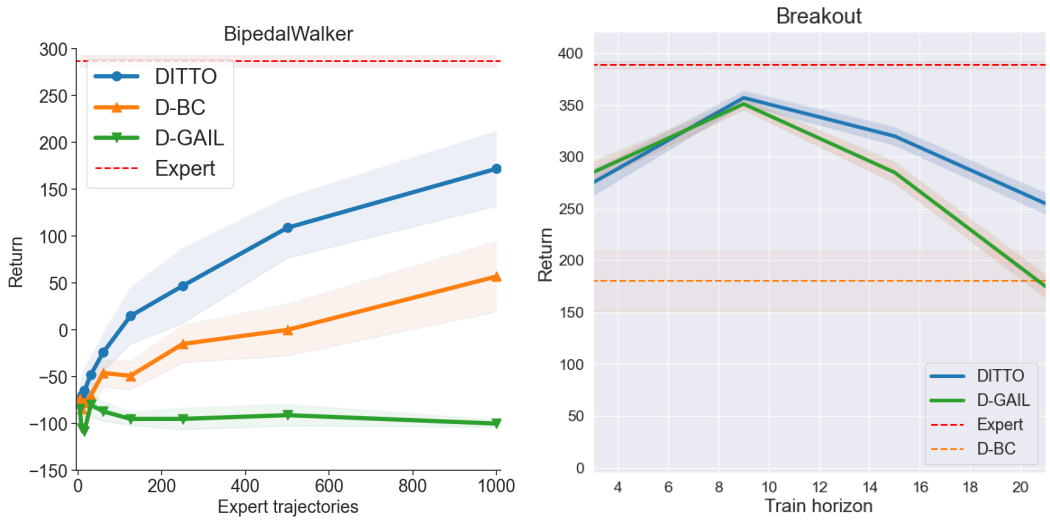

Figure 3.4: **Left:** Results on continuous control environment BipedalWalker, from pixels. **Right:** Training time horizon ablation. Note that both DITTO and D-GAIL achieve their maximum performance at a similar training time horizon. We conjecture that this hyperparameter is environment-specific. We report results for all environments with fixed $H = 15$.

covariate shift. These results suggest that action prediction accuracy in the expert's distribution does not measure generalization performance.

Figure 3.3 plots the performance of DITTO against our proposed world model baselines and standard BC. In MsPacman and Qbert, most methods recover expert performance with the least amount of data we tested, and are tightly clustered, suggesting these environments are easier to learn good policies in, even with little data. D-GAIL exhibited adversarial collapse twice in MsPacman, an improvement over standard GAIL, which exhibits adversarial collapse uniformly in prior works which study imitation learning from pixels in Atari (Reddy, Dragan, and Levine, 2020)(Brantley, Sun, and Henaff, 2020). In contrast, DITTO always recovers or exceeds average expert performance in all tested discrete-action environments, and matches or outperforms the baselines in terms of both sample efficiency and asymptotic performance. Figure 3.4 shows the performance of DITTO vs the other world-model based algorithms on a continuous control task from pixels. Surprisingly, even without action supervision DITTO outperforms D-BC. Figure 3.4 also ablates the training sequence length in a single environment, where we observe that the best performing training time-horizon in this environment is shorter than the value we use in all experiments ($H = 15$), implying further performance gains are possible, and that environment-specific tuning of this hyperparameter should be performed when using

this method.

3.6 Conclusion

Offline policy learning algorithms must deal with high-dimensional observation spaces and distribution shift to graduate to real-world deployment. DITTO addresses these problems using world models, achieving greater performance and superior sample efficiency compared to strong baselines which we introduce for the offline setting. DITTO is the first offline imitation learning algorithm to consistently recover expert performance on these difficult Atari environments from pixels. Our formulation encourages learners to return to the data distribution using a simple fixed reward function defined in the model latent space. By learning under their own distribution, DITTO policies mitigate policy-induced covariate shift. Addressing the combined difficulties of high-dimensional partially observable environments and offline learning are key challenges to scale imitation learning to real world challenges.

A key remaining challenge is to estimate the quality of the world model before it is used to train an agent. Can we know ahead of time whether a world model produces sufficiently high-quality representations necessary for learning? D. R. Ha and Schmidhuber (2018) suggest that imperfect learned transition models may actually be advantageous: training in a stochastic simulator with fluctuating dynamics might be equivalent to domain randomization. If this is so, then we might expect policies trained in this way to be more robust to domain shifts, at the cost of performance which might result from exploiting fine details of the true dynamics. Instead of training models via observation reconstruction, Grimm et al. (2020) propose the Value Equivalence Principle, which says that we should only model details in the world which affect the agent's plan, as measured by its value function. That is, value functions induce an equivalence class over transition models, and we only need to model the world up to this equivalence.

Chapter 4

Generalization to the Real World

In this chapter, we consider the ultimate generalization challenge for machine learning: from simulation to the real world. Building systems that can perform long-horizon tasks specified by natural language is a long-standing goal in robotics (Mees, Hermann, Rosete-Beas, et al., 2022). When complete task specifications are available, reinforcement learning (RL) methods can be used to produce policies by trial and error; however, standard RL algorithms can require millions of episodes to learn good policies, which is infeasible for many robotics tasks. Furthermore, many tasks are difficult to specify as reward functions, requiring substantial engineering effort to elicit reasonable learned skills from reinforcement learning alone.

Imitation learning offers an alternative approach, by learning a policy directly from prior demonstrations of the desired task or set of tasks. Naïve imitation learning approaches such as behavior cloning (Pomerleau, 1989) learn a state-conditioned action predictor directly from the demonstration data, then act according to that action predictor at deployment time. However, these approaches fail to account for the temporal nature of multi-step decision processes: since the next observation depends on the previous action prediction, the independence assumption underlying standard statistical machine learning is broken. Ross and Bagnell (2010) showed that for a behavior-cloned policy with error ϵ, the expected regret is $\mathcal{O}(T^2\epsilon)$, which is *quadratic* in the decision horizon T. Intuitively, because the behavior-cloned policy is not trained under its own distribution, small prediction errors tend to compound, bringing it out of distribution to states it has never seen and which it fails to recover from. This compounding error problem is especially problematic for long-horizon control problems due to the regret's quadratic dependence on sequence length.

This work was done in collaboration with Iman Nematollahi, a true roboticist.

One approach to alleviating the distribution shift from off-policy learning is to learn on-policy in a simulation of the environment. However, a simulator may not be available, or if it is, it can suffer from the sim2real problem, a commonly observed phenomenon where policies that perform well in simulation fail to generalize to the real world (Hwangbo et al., 2019). This is due to the "reality gap", which stems from a mismatch between the real and simulated dynamics. A common approach to mitigating sim2real policy transfer problems is domain randomization (Peng et al., 2017; Rudin et al., 2022), where the policy is trained in multiple instances of the simulator under varying physics parameters, so that the agent becomes robust to a range of dynamics.

World models have emerged as a promising alternative to hand-crafted simulators (D. Ha and Schmidhuber, 2018). World models approximate the dynamics by learning to predict states or observations conditioned on actions, recorded from real trajectories over multiple steps. By compressing observations into latent representations that can carry temporal context, contemporary world models (Schrittwieser et al., 2020; Hafner, Pasukonis, et al., 2023) can achieve high-fidelity prediction over thousands of time steps. Learning inside the world-model latent space enables both on-policy and long-horizon learning, mitigating distribution shift; and it avoids the problem of the reality gap from mis-specified simulators.

Taking steps towards achieving language-conditioned, long-horizon, multi-task policies, we introduce **LUMOS**, a language-conditioned imitation learning framework. LUMOS employs world models to learn a multi-task policy that can be guided by free-form natural language instructions entirely offline and transferred to the real world in a zero-shot fashion. Moreover, it leverages the latent-matching intrinsic reward proposed by DITTO within the world model's latent space, enabling its actor-critic agent to recover expert performance without any online interaction. To further enhance the long-horizon performance of the actor-critic agent, we build upon recent advancements in imitation learning (Lynch, Khansari, et al., 2019; Mees, Hermann, and Burgard, 2022), and augment LUMOS with a latent planning network. This network uses the benefit of hindsight during training to distill predicted information about the future trajectory into a "latent plan". We demonstrate that LUMOS can learn imitation policies solely within the world model's latent space, achieving zero-shot transfer to the real world. Extensive tests on the CALVIN benchmark and in real-world scenarios show that our algorithm excels in handling multi-objective, long-horizon tasks specified in natural language, outperforming previous learning-based methods.

Figure 4.1: **LUMOS** learns a general-purpose 7-DoF language-conditioned visuomotor policy within the latent space of a learned world model. By optimizing for an intrinsic reward aimed at matching the expert performance within the world model, it effectively recovers from its own mistakes across multiple time steps and reduces covariate shift. Consequently, LUMOS can accomplish complex, multi-tier, long-horizon robot manipulation tasks based on abstract natural language instructions in real-world scenarios, without requiring online learning or fine-tuning.

Our contributions are as follows: 1) **Improved Long-Horizon Imitation Learning:** To reduce covariate shift and improve performance on long-horizon imitation learning tasks, we combine world-model-based exploration and skill practice with latent planning. This allows our agent to practice its skills over simulated long horizons while leveraging the privileged multi-step information from the full demonstration trajectory. In ablation experiments, we show that the multi-step practice in latent space provides the greatest improvement to our pipeline, substantially outperforming single-step behavioral cloning. 2) **Language Goal-Conditioned Learning:** We introduce a new language goal-conditioned imitation learning framework, LUMOS, which outperforms prior methods with comparable approaches on the challenging open CALVIN benchmark, which tests policies by their ability to perform multiple long horizon manipulation tasks using language instructions alone. 3) **Transfer of World Model Dynamics and Policy to the Real Environment:** We demonstrate successful zero-shot transfer of policies learned entirely offline in the world model with language conditioning to the real environment.

4.1 Related Work

Language-Conditioned Imitation Learning for Robotics. Lynch and Sermanet (2021) introduce MCIL, which leverages unstructured robot play-data to train a behavior cloning agent. After collecting the play data, they collect human language annotations on 1% of trajectories and condition the behavior cloning on these annotations, making the policy steerable at test-time by natural language commands. Subsequently Mees, Hermann, Rosete-Beas, et al. (2022) introduced the open-source CALVIN environment, a benchmark for assessing policies on their ability to perform language-conditioned, long-horizon multi-task robotic manipulation. CALVIN consists of a simulator paired with ∼24 hours of teleoperated unstructured play data, along with 20K language annotations of the tasks being performed. HULC (Mees, Hermann, and Burgard, 2022) introduces a framework for hierarchical language-conditioned imitation learning using contrastive learning to ground language instructions to robotic actions, as well as other improvements like using a multimodal transformer encoder (discussed below). HULC outperforms MCIL on the CALVIN benchmark (Zhou et al., 2023). HULC++ (Mees, Borja-Diaz, and Burgard, 2023) further enhances long-horizon performance by integrating affordance prediction and motion planning into the control loop, only deferring to the learned HULC controller when the end-effector is near the manipulation target. SPIL (Zhou et al., 2023) improves upon the state-of-the-art by labeling all actions in the dataset and probabilistically assigning them to a base skill (translation, rotation, grasping) based on their magnitudes. In this work, we do not incorporate any prior knowledge about the semantic relationship between action dimensions and particular skills, as we aim to maintain a 7-DoF continuous action space. Recently, Diffusion Generative Models have gained traction as a powerful tool for policy representation in robotics, iteratively diffusing actions from Gaussian noise (Chi et al., 2023; Reuss, Maximilian Li, et al., 2023; Xian et al., 2023). These policies have proven capable of acquiring diverse and adaptable behaviors conditioned on language-goals (Ke, Gkanatsios, and Fragkiadaki, 2024; Black et al., 2023; H. Ha, Florence, and Song, 2023). Multimodal Diffusion Transformer (MDT) (Reuss, Yağmurlu, et al., 2024), the latest advancement in this field, leverages two self-supervised auxiliary objectives, Masked Generative Foresight and Contrastive Latent Alignment, to enhance performance in long-horizon manipulation tasks. Although this approach is orthogonal to our work, we acknowledge the potential of diffusion models in both world modeling and behavior learning. In this paper, however, we aim to investigate how to train policies in the latent space of a learned world model.

Improving Behavior Cloning with Latent Planning. Lynch, Khansari, et al. (2019) introduce GCBC, which augments standard behavior cloning with image-based goal conditioning. GCBC further introduces a planning module which trains a plan-predictor conditioned only on the current and goal states to predict a latent variable produced by a planning module that has privileged access to the entire expert trajectory. At test time, latent variables sampled from the plan predictor are used to condition the policy to achieve improved long-horizon performance. One limitation of this work is the reliance on goal images for conditioning the plan predictor. E. Zhang et al. (2024) improve upon the latent planner in GCBC by replacing the discrete latent plan embedding model with a diffusion model. They learn to iteratively denoise the latent plan via a language-conditioned U-Net, achieving superior long-horizon performance over GCBC.

Learning with World Models. World models have recently emerged as a promising approach to data-driven simulation. The seminal work of D. Ha and Schmidhuber (2018) demonstrated the effectiveness of learning world models directly from pixel observations to predict future observations, and then learning a policy inside the latent space of the learned model. Building on this, the Dreamer family of works (Hafner, Lillicrap, Ba, et al., 2020; Hafner, Lillicrap, Norouzi, et al., 2021) introduced a world-model-based RL algorithm that achieved SOTA performance on the Atari benchmark, and recently became the first algorithm to produce diamonds in Minecraft, which requires acting coherently across thousands of time-steps (Hafner, Pasukonis, et al., 2023). DayDreamer (Wu et al., 2023) applied the Dreamer algorithm to learning quadrupedal locomotion online on a real robot, demonstrating the robustness of policies trained in the world model to transfer to the real environment.

This work builds primarily on DITTO, the world-model-based imitation learning algorithm described in the previous chapter. DITTO penalizes an on-policy divergence from expert trajectories in the world model latent space. Specifically, it defines an *intrinsic reward* in the latent space that measures the divergence between agent rollouts and expert demonstrations and then optimizes this intrinsic reward using actor-critic RL. Optimizing this intrinsic reward induces imitation learning that is robust to errors over long-horizon rollouts, effectively mitigating covariate shifts.

4.2 Problem Formulation

We investigate goal-conditioned imitation learning in a partially observable Markov decision process (POMDP) with continuous actions and high-dimensional observations

from an unknown environment. We model the interaction between the environment and the goal-conditioned policy using a goal-augmented POMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \mathcal{G}, \gamma)$, where $\mathcal{S}$ is the state-observation space, $\mathcal{A}$ is the action space, $\mathcal{R}(s, a)$ is the reward function, $\mathcal{T}(s'|s, a)$ defines state-transition dynamics, $\mathcal{G}$ is the goal space, and $\gamma \in (0, 1)$ is the discount factor. The agent receives visual observations instead of direct state access and aims to maximize the expected discounted sum of extrinsic rewards $\mathbb{E}[\sum_t \gamma^t r_t]$, which it cannot directly observe. We conduct offline learning using a large, unlabeled, and undirected, fixed play dataset $\mathcal{D} = \{(s_1, a_1), \ldots, (s_T, a_T)\}$. This dataset is relabeled into long temporal state-action streams (Andrychowicz et al., 2017). Each visited state is treated as a "reached goal state", resulting in a trajectory dataset $\mathcal{D} = \{\tau_i = (s_t, a_t)_{t=0}^{k}\}_{i=1}^{N}$. This transformation creates $\mathcal{D}_{\text{play}} = \{(\tau, s_g)_i\}_{i=0}^{|\mathcal{D}_{\text{play}}|}$, pairing each goal state s_g with the corresponding demonstration trajectory τ. We adopt the multi-context imitation learning approach from play data by Lynch *et al.* (Lynch and Sermanet, 2021) to utilize language annotations. This method shows that combining a few random windows with language-based retrospective instructions helps learn a unified, language-conditioned visuomotor policy for various robotic tasks. Free-form language instructions $l \in \mathcal{L}$ guide the policy, providing flexible and natural task descriptions.

4.3 Offline Goal-Conditioned Imitation Learning with LUMOS

In this section, we introduce LUMOS. Training consists of two phases: first, a world model is learned from the unlabeled play dataset $\mathcal{D}$. Next, an actor-critic agent is trained within this learned world model to acquire a goal-conditioned policy by guiding imagined sequences of latent model states to match the latent trajectory of the expert demonstrations. During inference, the language-conditioned policy $\pi_\theta(a_t \mid s_t, l)$, trained entirely in the latent space of the world model, successfully transfers to the real environment. Figure 4.2 shows an overview of the approach. For details on hyperparameters and architecture choices, please refer to the appendix.

4.3.1 World Model Learning

World models capture an agent's experience into a predictive model, enabling behavior learning without direct interaction with the environment. By converting high-dimensional images into compact state representations, they allow efficient, parallel long-term predictions in latent space. While our method is agnostic regarding the

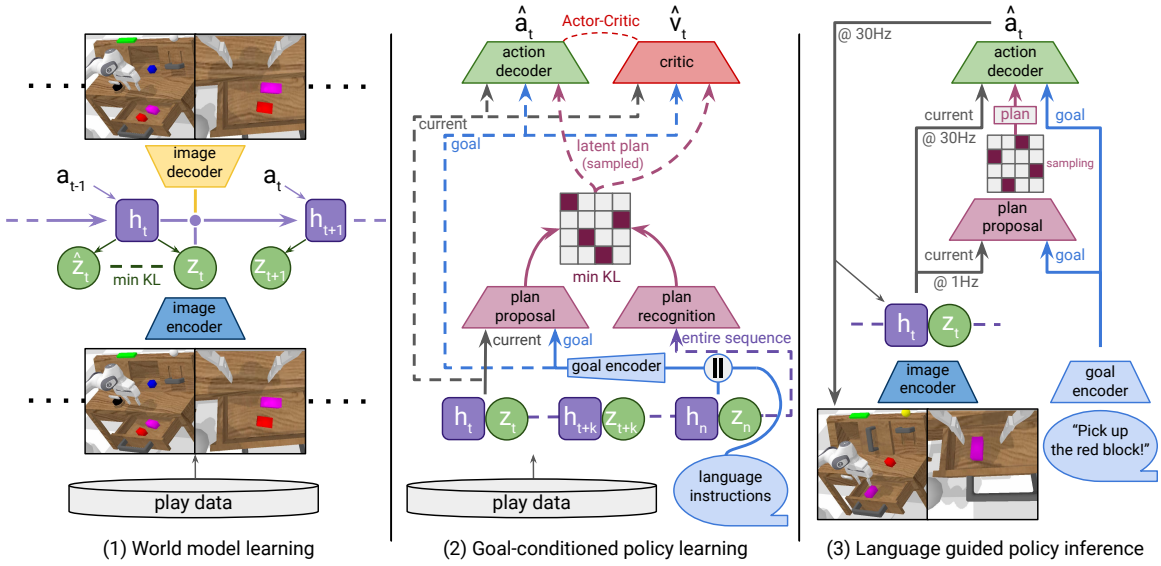

Figure 4.2: **LUMOS** learns a language-guided general-purpose policy within the latent space of a world model. (1) The world model, comprising an image encoder, a Recurrent State-Space Model (RSSM) for dynamics, and an image decoder, transforms play dataset experience into a predictive model that enables behavior learning in the latent state space. (2) The goal-conditioned policy samples latent trajectories and uses *either* a language annotation *or* the final latent state as the goal, with plan recognition and proposal networks being trained to identify and organize behaviors in a latent plan space. The action decoder is intrinsically rewarded by matching the expert's latent trajectory. (3) During inference, the policy acts based on the latent state inferred by the world model from the current observation and is guided by a user's language command.

choice of the world-model architecture, we follow DITTO by adopting the DreamerV2 architecture (Hafner, Lillicrap, Norouzi, et al., 2021) as the backbone for LUMOS, which includes an image encoder, a recurrent state-space model (RSSM) for learning transition dynamics, and a decoder for reconstructing observations from latent states. To learn robotic manipulation skills, we use observations from static and gripper-mounted cameras, with separate CNN encoders and transposed CNN decoders for each. These encodings are concatenated before being passed to the RSSM, which generates a sequence of deterministic recurrent states $(h_t)_{t=0}^{T}$. Each state defines two distributions over stochastic hidden states: the stochastic posterior state z_t, determined by the current observation x_t and recurrent state h_t, and the stochastic prior state $\hat{z}_t$, trained to approximate the posterior without the current observation. By predicting $\hat{z}_t$, the model learns the environment dynamics. The combined model state, comprising deterministic and stochastic components $\hat{s}_t = (h_t, z_t)$, reconstructs the observation.

The RSSM includes the following modules:

$$\begin{aligned} &\text{Recurrent state:} && h_t = f_\phi(\hat{s}_{t-1}, a_{t-1}) && \text{Representation model:} && z_t \sim q_\phi(z_t \mid h_t, x_t) \\ &\text{Dynamics predictor:} && \hat{z}_t \sim p_\phi(\hat{z}_t \mid h_t) && \text{Image decoder:} && \hat{x}_t \sim p_\phi(\hat{x}_t \mid \hat{s}_t). \end{aligned} \tag{4.1}$$

The prior and posterior models predict categorical distributions, optimized using straight-through gradient estimation (Bengio, Léonard, and Courville, 2013). All modules are implemented as neural networks parameterized by ϕ, and are optimized jointly by minimizing the negative variational lower bound (Kingma and Welling, 2013):

$$\min_\phi \mathbb{E}_{q_\phi(z_{1:T} \mid a_{1:T}, x_{1:T})} \left[\sum_{t=1}^{T} -\log p_\phi(x_t \mid \hat{s}_t) + \beta D_{\mathrm{KL}}(q_\phi(z_t \mid \hat{s}_t) \parallel p_\phi(\hat{z}_t \mid h_t)) \right]. \tag{4.2}$$

After training, the world model can operate without input observations by using the prior $\hat{z}$ instead of the posterior z. This allows the model to generate unlimited imagined trajectories of the form $\{(h_t, \hat{z}_t, a_t)_{t=0}^{H}\}$, where H is the time horizon for imagination.

4.3.2 Behavior Learning

LUMOS employs an actor-critic agent to learn long-horizon, language-conditioned behaviors within the latent space of its world model. It learns a goal-conditioned policy $\pi_\theta(a_t \mid s_t, g)$ and a value function $v_\psi(s_t, g)$ from the dataset $\mathcal{D}_{\text{play}}$, where s_t denotes the current latent state and $g \in \mathcal{G}$ represents a free-form language instruction l *or* a latent goal state s_g. In this section, we outline the design choices of our model, explain how to intrinsically reward the agent for matching expert latent state-action pairs over a trajectory, and detail the optimization process for the actor and critic.

Observation and Action Spaces Our goal-reaching policy builds on the HULC architecture by Mees, Hermann, and Burgard (2022), enhancing language-conditioned imitation learning from unstructured data. Unlike HULC, which uses high-dimensional image observations, our method utilizes the latent state space of the world model for its compact representations. In addition to encoding RGB images from static and gripper cameras, LUMOS encodes compact states of latent trajectories from the current to the goal state. The action space is a 7-DoF continuous space, including relative XYZ positions, Euler angles, and a gripper action.

Latent Plan Encoding Learning control with free-form imitation data faces the challenge of multiple valid trajectories connecting the same (s_t, s_g) pairs. To address

this, we use a sequence-to-sequence conditional variational auto-encoder (seq2seq CVAE) (Lynch, Khansari, et al., 2019) to encode contextual latent trajectories into a latent "plan" space. This space is determined by two stochastic encoders: one for recognizing plans during training and the other for proposing plans during inference. The plan recognition encoder identifies the performed behavior from the entire sequence, while the plan proposal encoder generates possible behaviors from the initial and final states (see the second column of Figure 4.2). Minimizing the KL divergence between these encoders, referred to as $\mathcal{L}_{\text{KL}}$, ensures the plan proposal encoder accurately reflects observed behaviors. Similar to HULC, we use a multimodal transformer encoder to develop a contextualized representation of latent trajectories, mapping them into a vector of multiple latent categorical variables, optimized with straight-through gradients.

Semantic Alignment of Latent Trajectories and Language Addressing the symbol grounding problem (Harnad, 1990) requires relating language instructions to the robot's perception and actions. Aligning instructions with visual observations is crucial for distinguishing similar objects like colored blocks. Unlike HULC, which aligns visual features with language features, we align the world model's latent features with language features. This involves maximizing cosine similarity between latent and language features while minimizing similarity with unrelated instructions, using a contrastive loss similar to CLIP (Radford et al., 2021) and incorporating in-batch negatives for efficient training. We denote this objective as $\mathcal{L}_{\text{contrast}}$ (see Appendix for details).

Reward Unlike behavior cloning approaches, which rely exclusively on supervised action labels for training, LUMOS leverages a simple intrinsic reward defined within the latent space of the world model to address the problem of covariate shift. To align the agent's behavior with the expert, we employ the reward function proposed in the previous chapter that encourages the agent to match the expert's state-action pairs in the latent space. Again, this intrinsic reward is defined as a modified inner product on the latent state representations, encouraging similarity without requiring exact matches. This lets the agent explore different trajectories in the world-model latent space, while aligning its behavior to the demonstrator over the entire training horizon. The reward at each step t is given by:

$$r_t^{int}(s_t^E, s_t^\pi) = \frac{s_t^E \cdot s_t^\pi}{\max(\|s_t^E\|, \|s_t^\pi\|)^2} \tag{4.3}$$

Action and value models Our stochastic actor, parameterized by θ, samples actions based on the current latent state s_t and a latent goal g, aiming to maximize

expected rewards (Eqn. 4.3). The deterministic critic, parameterized by ψ, predicts the expected discounted future rewards, providing value estimates for the same latent state and goal. The models are defined as:

$$\text{Actor:} \quad a_t \sim \pi_\theta(a_t \mid s_t, g), \quad \text{Critic:} \quad v_\psi(s_t, g) \approx \mathbb{E}_{\pi_\theta, p_\phi}\left[\sum_{t=0}^{H} \gamma^t r_t^{int}\right] \tag{4.4}$$

Fully-connected neural networks are used for the actor and critic. The action model outputs a tanh-transformed Gaussian (Haarnoja et al., 2018), facilitating reparameterized sampling for backpropagation. The world model is fixed during behavior learning, ensuring that the agent gradients do not alter its representations.

Learning Objective The objective in training the critic is to regress the computed value estimate to the λ-target (Richard S. Sutton and Barto, 2005), denoted V^λ (see Appendix for the definition):

$$\mathcal{L}(\psi) \doteq \mathbb{E}_{\pi_\theta, p_\phi}\left[\sum_{t=1}^{H-1} \tfrac{1}{2}(v_\psi(\hat{s}_t) - sg(V_t^\lambda))^2\right] \tag{4.5}$$

where $sg(\cdot)$ is the stop-gradient operator. Moreover, a target network, which updates its weights slowly, is used to provide stable value bootstrap targets (Mnih, Kavukcuoglu, et al., 2015). The actor's objective function uses backpropagation through the learned dynamics to maximize value estimates while minimizing the KL divergence between plan encoders and the contrastive loss for aligning latent trajectories with language. For sampled windows without language annotations, the contrastive loss is omitted:

$$\mathcal{L}(\theta) \doteq \mathbb{E}_{\pi_\theta, p_\phi}\left[\sum_{\tau=t}^{t+H} (-V_\lambda(s_\tau) + \alpha_1 \mathcal{L}_{KL} + \alpha_2 \mathcal{L}_{\text{contrast}})\right] \tag{4.6}$$

4.4 Experimental Results

We evaluate LUMOS in learning language-conditioned imitation learning in both simulated and real-world settings. The objectives of our experiments are threefold: (i) to determine the model's performance in executing long-horizon language-specified tasks, (ii) to ablate components of the objective function and analyze their impact, and (iii) to assess its scalability to real-world robotics tasks.

Method	LH-MTLC					
	No. Instructions in a Row (1000 chains)					
	1	2	3	4	5	Avg. Len.
GCBC 2019	56.7% (3.4)	20.9% (1.9)	7.3% (2.9)	1.6% (0.5)	0.04% (0.0)	0.63 (0.4)
MCIL 2021	70.3% (3.1)	40.2% (2.9)	21.6% (4.1)	11.1% (1.9)	5.4% (1.0)	1.48 (0.3)
HULC 2022	77.6% (1.1)	58.05% (0.3)	41.6% (1.3)	29.8% (0.9)	20.0% (1.5)	2.27 (0.05)
LUMOS (ours)	**80.7**% (1.0)	**59.3**% (1.4)	**42.6**% (1.8)	**30.7**% (1.2)	**21.1**% (0.8)	**2.34** (0.05)
No DITTO	71.8% (1.3)	45.6% (1.5)	27.8% (1.9)	14.2% (1.8)	8.7% (0.7)	1.68 (0.02)
No latent plan	76.5% (2.5)	50.5% (0.9)	28.7% (1.3)	15.9% (0.8)	11.0% (0.7)	1.81 (0.01)
No alignment	73.3% (1.5)	52.6% (0.9)	39.8% (2.0)	27.3% (1.3)	17.2% (0.9)	2.05 (0.06)

Table 4.1: Performance of all models on environment D of the CALVIN Challenge with an ablation study of key components evaluated over three seeds. All models receive 64×64 resolution RGB image inputs from both a static and gripper camera. Standard deviation in parentheses.

4.4.1 Experiments in Simulation

We conduct our simulation experiments in environment D of the CALVIN Challenge (Mees, Hermann, Rosete-Beas, et al., 2022), which offers six hours of unstructured tele-operated play data to train a 7-DoF Franka Emika Panda robot arm. CALVIN features 34 distinct subtasks and evaluates 1000 unique instruction chain sequences. The agent's objective is to sequentially solve up to five language instructions using only onboard sensors, requiring it to transition between subgoals based solely on language cues. Notably, only 1% of this dataset is annotated with language instructions through crowd-sourcing.

Evaluation Protocol We compare our model with the following language-conditioned robotic imitation learning approaches over unstructured data: **GCBC** (Lynch, Khansari, et al., 2019): Goal-conditioned behavior cloning does not model latent plans but instead maximizes the likelihood of each action as reconstruction loss at each time step. **MCIL** (Lynch and Sermanet, 2021): Multi-context imitation learning employs a sequential CVAE to predict the next actions from image or language-based goals by modeling reusable latent sequences of states as latent plans. **HULC** (Mees, Hermann, and Burgard, 2022): Hierarchical universal language conditioned policy provides a strong baseline by building upon MCIL, introducing several improvements, such as adding a self-supervised contrastive loss for alignment of video and language representation. Our approach differs from the above models by not regressing action labels. Instead, we learn a world model from the data, before training an agent to match the expert's latent trajectory in the world model. To reduce the computational

burden of LUMOS' world model, we standardize the input resolution of all images to 64×64. For fair comparison, all baselines are trained at this resolution, which sometimes lowers the original implementations' resolutions (e.g., 200×200 for static and 84×84 for the gripper camera). For an analysis of the baselines' performance at full resolutions, please refer to (Mees, Hermann, and Burgard, 2022).

Table 4.1 contains quantitative results of the average success rate of each model over 1,000 instruction chains across three seeded runs. LUMOS successfully performs multi-stage, long-horizon, language-conditioned robotic manipulation tasks, and outperforms all the baselines. This experiment shows that our agent can optimize its intrinsic reward, which measures its deviation from expert latent trajectories. This capability allows the agent to recover from its mistakes over multiple time steps and successfully chain more tasks together in sequence.

In our first ablation, we examine the impact of our latent matching reward function. Specifically, we supervise our agent using behavior cloning with mean squared error instead of optimizing for intrinsic rewards. This ablation demonstrates a significant drop in our model's performance under these conditions and indicates that our reward formulation effectively reduces covariate shift. For the second ablation study, we explore the effect of learning a latent plan space on long-horizon behavior. For this test, we removed the plan proposal and recognition components from the actor's architecture. The results show that, although the agent starts the trials relatively well, it struggles to complete longer-horizon tasks. Lastly, in our final ablation, we omit the alignment objective between language and latent state features. While this variant maintains some long-horizon performance, it performs slightly worse at each step compared to our full model. This is evident in its occasional manipulation of

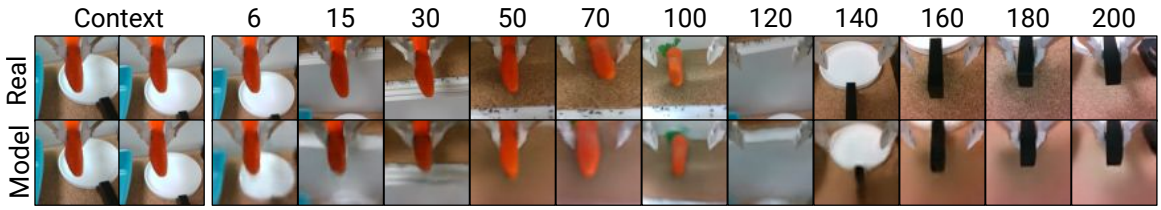

Figure 4.3: **World Model Long-Horizon Predictions.** Using the first five images of a hold-out trajectory as context, the world model predicts the next 195 steps using its latent dynamics, given only the actions. Trained on sequences of horizon 50, our model achieves precise long-term predictions, enabling effective behavior learning in a compact latent space. Here, we only visualize the reconstructions of the gripper camera.

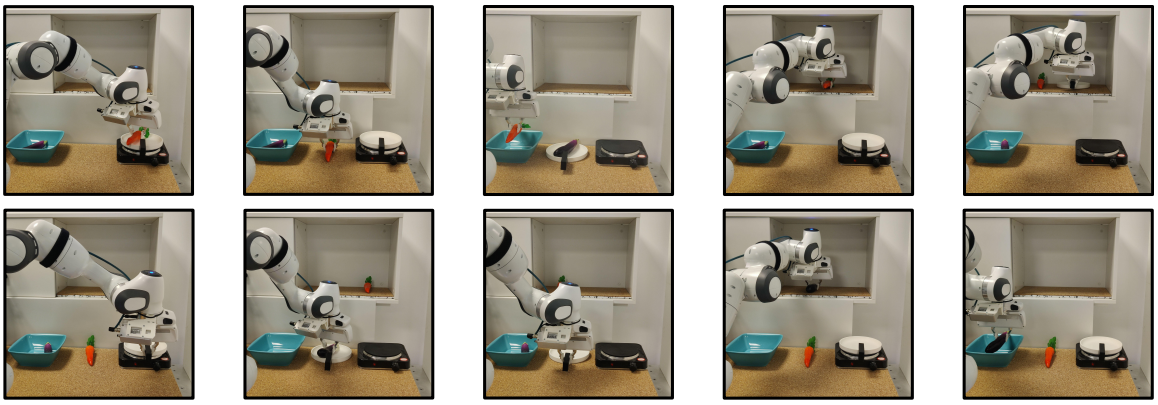

Figure 4.4: **Real-world Manipulation Tasks.** Examples shown from left to right are: placing the carrot onto the pan, lifting the carrot from the table, dropping the carrot into the bowl, storing the carrot in the cabinet, placing the pan into the cabinet, setting the pan onto the stove, adding the eggplant to the pan, picking up the pan from the table, taking the eggplant from the cabinet, and putting the eggplant into the bowl.

blocks of the wrong color due to a suboptimal association between instructions and the scene.

4.4.2 Experiments in Real-World

To assess the performance of LUMOS at handling various real-world tasks, we conduct experiments using a Franka Emika Panda robot arm within a 3D tabletop environment. This environment includes a table with a stove, a bowl, a cabinet, a carrot, and an eggplant. We collected a play dataset by recording three hours of teleoperated interactions with the robot arm, using a VR controller to guide its movements. During recording, we captured RGB observations from both static and gripper cameras. We used language annotation for less than 1% of the data (∼2800 random windows). Annotators described behaviors in randomly sampled portions of the interaction data. This dataset encompasses over 22 unique tasks for manipulating the available objects.

First, we used the offline play dataset to learn a generative world model. As shown in Figure 4.3 our model demonstrates competence in long-horizon predictions. To assess its long-term prediction abilities, we applied the representation model to the first five images of hold-out trajectories to establish context. Given an action trajectory, we then tasked the model with predicting forward for 195 steps using its latent dynamics, despite being trained only with a horizon length of 50. The accuracy of the reconstructed images decoded from the latent trajectory of our recurrent state-

Task	Success Rate
Lift the vegetable from the table	67.5%
Lift the vegetable from the pan	75%
Lift the vegetable from the bowl	72.5%
Lift the vegetable from the cabinet	55%
Lift the pan from the table	65%
Lift the pan from the stove	80%
Lift the pan from the cabinet	35%
Place the vegetable onto the table	90%
Place the vegetable onto the pan	62.5%
Place the vegetable into the bowl	85%
Place the vegetable into the cabinet	70%
Place the pan onto the table	70%
Place the pan onto the stove	65%
Place the pan into the cabinet	55%
Average over tasks	67.68%

Table 4.2: The average success rate of our language-conditioned general-purpose policy in the real world.

space model indicates that our world model is well-suited for learning behaviors within its compact latent space. We trained our language-conditioned policy within the latent space of the world model and subsequently assessed each task through 20 rollouts. We utilized various neutral starting positions to avoid biasing the policy toward any specific task. This neutral setup eliminates the correlation between the robot's initial state and the queried task and ensures that the agent depends entirely on language cues to comprehend and execute the tasks. The success rate of our model is reported in Table 4.2. For brevity, we combined the success rates for tasks involving either the carrot or eggplant under the category "vegetable". However, each task was evaluated individually (i.e., if the robot picks up the carrot when the instruction was to pick up the eggplant, it does not count as a success). Finally, we evaluate our method's real-world performance in consecutive tasks. See Appendix D, Section F for detailed results. Overall, our findings highlight the efficacy of our language-conditioned multi-task policy.

4.5 Conclusion

In this chapter, we proposed *Language-Conditioned Imitation Learning via World Models* (LUMOS), which leverages a learned world model's latent space for fully offline, multi-step on-policy learning. Practicing skills in long-horizon model-based rollouts allows our model to recover expert performance without any online interaction. Our approach demonstrates superior performance on the CALVIN benchmark, achieving robust long-horizon manipulation guided solely by language instructions. LUMOS serves

as a proof of concept, demonstrating that dynamics and policies learned offline within a world model can successfully transfer zero-shot to real environments.

Chapter 5

Conclusion

In this thesis, we have considered a variety of generalization phenomena in machine learning. In Chapter 2, we showed how the generalization of learning systems is predicted by their intrinsic complexity, and demonstrated a complexity phase transition in deep neural networks. In Chapter 3, we proposed an offline imitation learning algorithm that improves policy generalization by simulating on-policy training in the latent space of a learned world model. Finally, in Chapter 4, we demonstrated that a policy trained solely in the latent space of a world model can successfully generalize to a real robot system. Together, these findings suggest that generalization arises when learning dynamics mirror the structure of their environment.

We have emphasized an algorithmic viewpoint rather than purely statistical framings of learning. This algorithmic framing invites new connections between learning theory, information geometry, and dynamical systems. What else can the algorithmic frame help us understand in machine learning?

Petersen et al. (2022) study neural networks parameterized by distributions over logic gates with continuous relaxations. After training, a circuit is sampled from the distribution with minimal performance loss. In addition to their high inference speed, reduced power use, and high compression ratios relative to standard networks, these logic-gate networks can be analyzed as discrete systems. Methods like these let us directly interpret learned networks as symbolic systems and might admit much stronger generalization guarantees through formal verification and compilation down to minimal representations.

Another interesting application of AIT for machine learning is due to Theis (2024), who formalizes realism in the context of generative image modeling using the concept of *randomness deficiency*, which measures the difference between the Shannon information

of a string x under a proposal distribution P, and the Kolmogorov complexity of x:

$$U(x) = -\log P(x) - K(x) \tag{5.1}$$

To understand this quantity, consider the following sequences:

```
H, T, H, T, T, H
```
(5.2)

```
H, H, H, H, H, H
```
(5.3)

Both sequences are equiprobable under a fair coin, with $P(n) = 2^{-n}$ and entropy $H(n) = n$ bits. Despite being equally likely, one might feel cheated if sequence 5.3 appeared in a betting game. Randomness deficiency quantifies this discrepancy and helps us understand when we ought to treat systems statistically. Theis (2024) showed how these complexity-theoretic concerns cannot be avoided when defining realism in generative modeling, where samples must appear random—i.e., sampled from a particular distribution.

These developments illustrate how algorithmic notions of simplicity may enable stronger interpretability, formal guarantees, and improved realism.

Recall that the manifold hypothesis suggests that apparently high-dimensional data lie near a low-dimensional manifold. The tools of information geometry quantify this hypothesis statistically, but a more precise formulation might be possible using algorithmic concepts. It is already known that the effective Hausdorff dimension d_{eff} of a set X can be understood in terms of its Kolmogorov complexity via:

$$d_{\text{eff}} = \liminf_n \frac{K(X \mid n)}{n} \tag{5.4}$$

This relationship shows how effective dimensions arise from irreducibly complex descriptions. Machine learning must evolve to account for algorithmic regularities beyond statistical description.

Appendix A

Information Theory

Shannon's source coding theorem connects the description length of an object to its information content, as measured by its entropy. His landmark work *A Mathematical Theory of Communication* (1948) laid out the foundations of information theory, which we briefly review before introducing the coding theorem.

In Shannon's formulation, information is a measure of surprise. When we observe that which was expected to occur, we gain little information. Concretely, the information we gain when we observe an event with probability p is

$$I(p) = -\log p \tag{A.1}$$

The choice of the logarithm follows from two conditions: First, events which are more likely are less surprising, and so contain less information. This implies that information should be a monotonically decreasing function of probability, i.e. $p > q \implies I(p) < I(q)$. Second, the information we gain from observing two independent events should be the sum of their individual information, i.e. $I(pq) = I(p) + I(q)$. Only the logarithm function satisfies these conditions.

Information also has units, which are determined by the base of the logarithm. Throughout, we will use a base of 2, which gives information in units of bits. For example, the outcomes of a fair coin toss are $\{H, T\}$ with $P(H) = P(T) = 1/2$, so that the information we gain from observing the outcome is $I = -\log_2(1/2) = 1$ bit. Similarly, if we observe an outcome which we were certain would occur ($p = 1$), we gain $I = -\log(1) = 0$ bits; no information.

In information theory, entropy is nothing more than the expected amount of information gain from observing a random variable X with a known probability distribution $P(X = x_i) = p_i$:

$$H(X) = -\sum_i p_i \log p_i \tag{A.2}$$

With the basic concepts in place, we can state Shannon's source coding theorem. Let $\mathcal{X}$ be a finite set of symbols called the *source alphabet*, and let A be a finite set of symbols called the *code alphabet* (typically $A = \{0, 1\}$ is binary). Let X be a discrete random variable over $\mathcal{X}$, and $(X_1, X_2, \ldots, X_n) \overset{\text{iid}}{\sim} X$. Our task is to encode each source sequence into a corresponding string over the code alphabet A^*, the set of all finite strings over A.

Theorem A.0.1 (Shannon's Source Coding Theorem)**.** *Let $\mathcal{X}$ be a finite source alphabet and A a finite code alphabet. A (uniquely decodable) code is a mapping*

$$C : \mathcal{X} \to A^*.$$

The expected codeword length is $\mathbb{E}[|C(X)|]$. Then

$$\mathbb{E}[|C(X)|] \geq \frac{H(X)}{\log |A|}.$$

Furthermore, for every $\epsilon > 0$, there exists a sequence of (prefix-free) block codes

$$C_n : \mathcal{X}^n \to A^*$$

such that, for sufficiently large n,

$$\frac{1}{n}\mathbb{E}\big[\, |C_n(X^n)| \,\big] \leq \frac{H(X)}{\log |A|} + \epsilon.$$

That is, the average code length per source symbol can be made arbitrarily close to the entropy of the source.

Shannon's theorem sets a limit on the compressibility of iid random data, determined by the entropy of its source. Coding schemes which approach this limit are referred to as entropy coding methods. When we minimize entropy objective functions in machine learning, this corresponds to finding a model which best compresses the training data. Still, a fundamental limitation of information theory to describe learning systems is the assumption that data is random, and drawn from a stationary distribution.

Appendix B

The Complexity Dynamics of Grokking

Table B.1: Hyperparameters

Description	**Symbol**	**Value**
Modulus	p	113
Dataset size	N	$p^2 = 12769$
Modular arithmetic tasks	$\{+, \times, -, \div\}$	-
Train data fraction per task	-	$(+, \times : 20\%), (-, \div : 30\%)$
Transformer layers	n_{layers}	$(+, \times : 1), (-, \div : 2)$
Transformer hidden dimension	d_{model}	128
Transformer heads dimension	d_{head}	32
Transformer head count	n_{head}	4
Transformer MLP dimension	d_{mlp}	512
Noise regularization scale	δ	$(+, \times : 10^{-2}), (-, \div : 10^{-3})$
Spectral entropy scalar	β	10^{-1}
Optimizer	-	AdamW
Weight decay scalar	-	1
Learning rate	-	10^{-3}

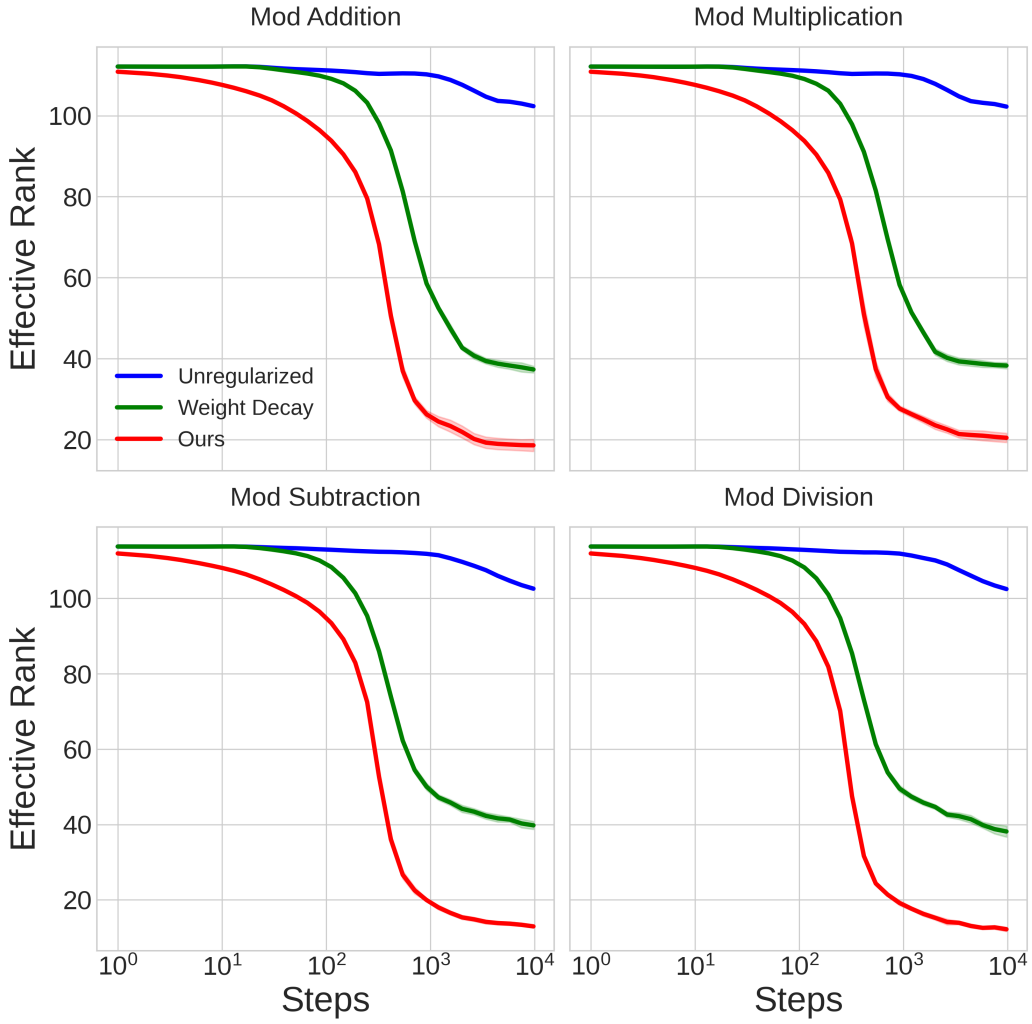

Figure B.1: Effective rank ($\text{rank}_{\text{eff}} = e^{H_{\text{svd}}}$) of models throughout training. We see that the regularized models fall in effective rank when generalizing, while the unregularized model remains near full rank throughout training. Our regularizer explicitly penalizes this measure, and results in both the lowest rank and most compressible model amongst those we study.

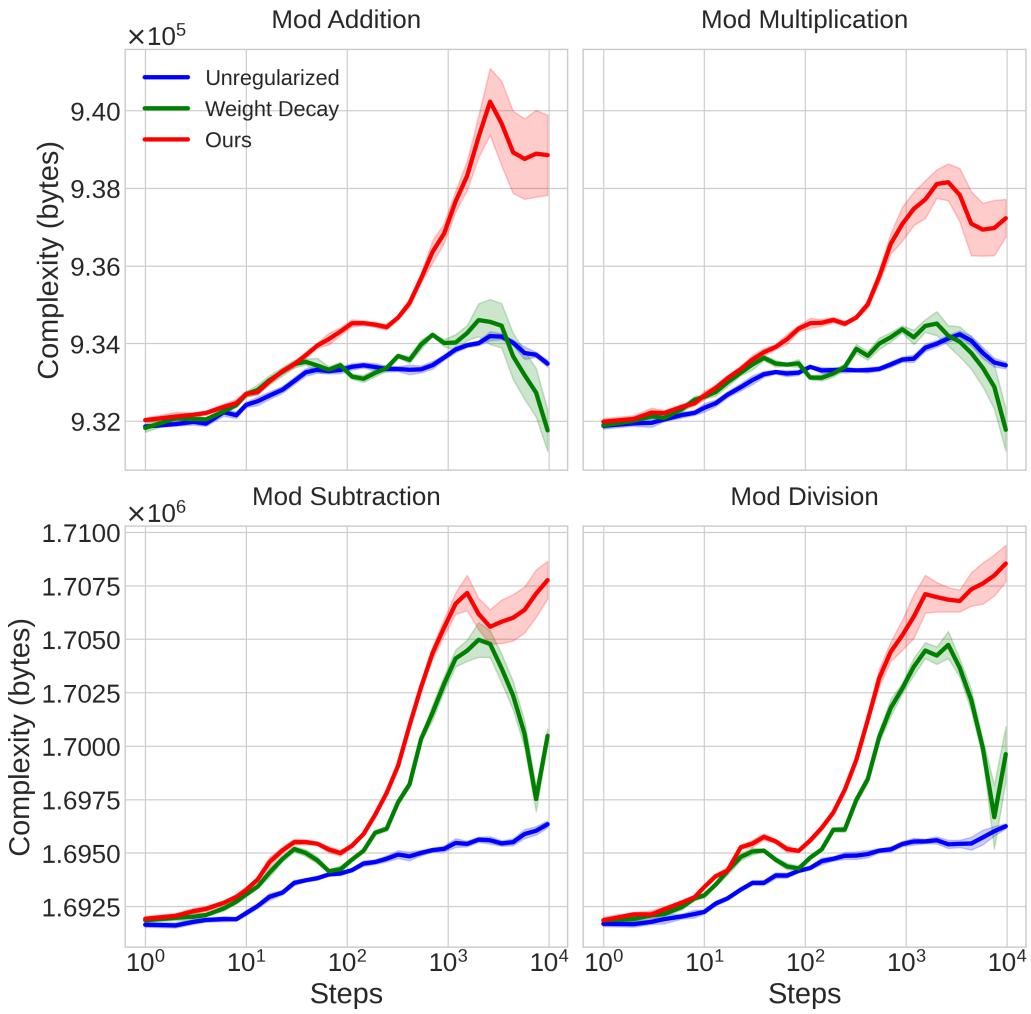

Figure B.2: In this figure, we simply plot the naïve `bzip2` compression of the model weights without any coarse-graining (note the y-axis scales). The range of variation in these complexity estimates is very small throughout training—if we plotted it on the same scale as other complexity plots, it would simply look like a flat line from the starting estimate. The dynamics here may correspond to weight patterns induced by the different regularizers that the compression algorithm is particularly suited or unsuited to compressing. Intriguingly, the dips in the complexity bounds of the regularized models around 10^3 steps corresponds to generalization, giving some slight hint that some change has occurred in the model. To fully reveal this phenomenon, the coarse-graining procedure is required.

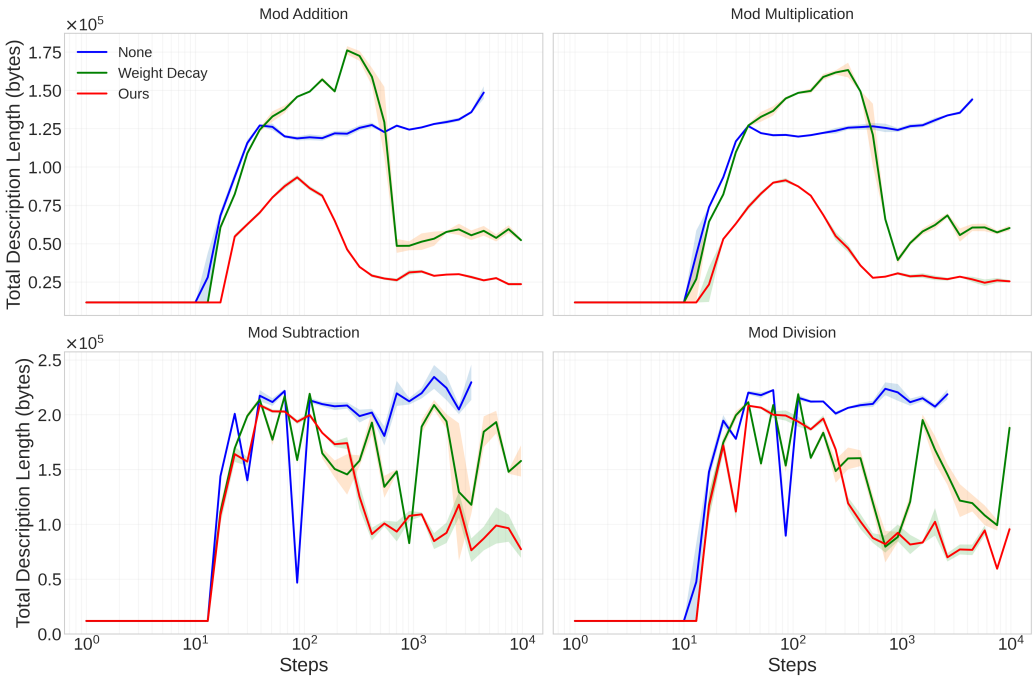

Figure B.3: Total description length (complexity + entropy) vs training steps. Models trained with our regularization (red) have a smaller total description length compared to baselines. The unregularized model collapses during memorization in the subtraction and division tasks. Apparent complexity computed for $\epsilon = 1$.

Appendix C

Generalization with World Models

C.1 Proof of divergence reward bound

We prove a corollary of proposition 1 from Ciosek (2022). Ciosek (2022) uses many intermediate results and definitions, so we encourage the reader to reference their work while reading to understand this proof.

Corollary C.1.1. *Suppose we also have another imitation learner, which uses the same data-set of size N, and still satisfies Assumption 3, but instead trains on some other intrinsic reward, R'_{int} which satisfies (for some $\epsilon > 0$):*

$$R'_{int}(s,a) = 1, \forall (s,a) \in D$$
$$0 \leq R'_{int}(s,a) \leq 1 - \epsilon, \textit{otherwise}$$

Let ρ^J be the limiting state-action distribution of this imitation learner. Then:

$$||\rho^J - \rho^E||_{TV} \leq \frac{\eta}{\epsilon}$$

$$\mathbb{E}_{\rho^J}[R] \geq \mathbb{E}_{\rho^E}[R] - \frac{\eta}{\epsilon}$$

Proof. Lemma 5 trivially still holds with R'_{int} instead of R_{int}, as $R'_{\text{int}} \geq R_{\text{int}}$ always, $\forall \rho, \mathbb{E}_\rho[R'_{\text{int}}] \geq \mathbb{E}_\rho[R_{\text{int}}]$. Hence the bound holding true for $\mathbb{E}_{\rho^I}[R_{\text{int}}]$ implies it holds for $\mathbb{E}_{\rho^I}[R'_{\text{int}}]$ too.

Lemma 7 holds with κ replaced by $\frac{\kappa}{\epsilon}$, so the result is $\mathbb{E}_{\rho^I}[R] \geq (1-\frac{\kappa}{\epsilon})\mathbb{E}_{\rho^E}[R] - 4\tau_{\text{mix}}\frac{\kappa}{\epsilon}$. We do this by considering their proof in Appendix D. The properties of the intrinsic reward are utilised in just one paragraph, after equation 25. This is done in stating that $\sum_\ell \frac{\ell M_\ell}{T} \to \mathbb{E}_{\rho^I}[R_{\text{int}}]$ and $\frac{B+1}{T} \to 1 - \mathbb{E}_{\rho^I}[R_{\text{int}}]$. This is not true for R'_{int}. Let p_a be

We acknowledge Sebastian Towers for assistance with this result.

the limiting chance of the expert agreeing with the R'_{int} imitation agent. Almost by definition, $\sum_\ell \frac{\ell M_\ell}{T} \to p_a$ and $\frac{B+1}{T} \to 1 - p_a$.

Note that $\mathbb{E}_{\rho^I}[R'_{\text{int}}] \leq p_a + (1 - p_a)(1 - \epsilon)$; we yield a reward of 1 every time we agree, and at most $1 - \epsilon$ if we disagree. Hence, using $1 - \kappa = \mathbb{E}_{\rho^I}[R'_{\text{int}}]$, we have $1 - \kappa \leq p_a + (1 - p_a)(1 - \epsilon) = 1 - \epsilon + p_a\epsilon$, hence $p_a \geq 1 - \frac{\kappa}{\epsilon}$.

So, taking limits as done in the original proof, we have:

$$\begin{aligned}
\mathbb{E}_{\rho^I}[R] &\geq p_a \mathbb{E}_{\rho^E}[R] - (1 - p_a)4\tau_{\text{mix}} - 0 \\
&= p_a \geq p_a(\mathbb{E}_{\rho^E}[R] + 4\tau_{\text{mix}}) - 4\tau_{\text{mix}} \\
&\geq (1 - \frac{\kappa}{\epsilon})(\mathbb{E}_{\rho^E}[R] + 4\tau_{\text{mix}}) - 4\tau_{\text{mix}}
\end{aligned}$$

Now, combining these lemmas is exactly as in section 4.4 in Ciosek (2022). The factor of $\frac{1}{\epsilon}$ carries forward, yielding $\mathbb{E}_{\rho^J}[R] \geq \mathbb{E}_{\rho^E}[R] - \frac{\eta}{\epsilon}$ as required.

□

C.2 Hyperparameters

Table C.1: Experimental hyperparameters

Description	**Symbol**	**Value**
Number of world model training episodes	N	1000
Number of expert training episodes	N^E	$\{4, 8, 15, 30, 60, 125, 250, 500, 1000\}$
World model training batch size	B_{wm}	50
World model training sequence length	L	50
Agent training batch size	B_{ac}	512
Agent training horizon	H	15
Discount factor	γ	0.95
$TD(\lambda)$ parameter	λ	0.95
KL-Balancing weight	β	0.1
KL-Balancing trade-off parameter	δ	0.8
Actor-critic entropy weight	η	5×10^{-2}
Behavior cloning entropy weight	η_{BC}	0.1
Optimizer	-	Adam
All learning rates	-	3×10^{-4}
Actor-critic target network update rate	-	100 steps

Appendix D

Generalization to the Real World

A Hyperparameters and Training Details of the World Model

To develop the world model of LUMOS, we used DreamerV2 (Hafner, Lillicrap, Norouzi, et al., 2021) as our foundation. While DreamerV2 models Atari game environments, our world model is specifically tailored for robotic manipulation. As a result, LUMOS incorporates distinct encoders and decoders for the static and gripper cameras. We concatenate their encodings and employ a fully-connected layer to fuse them before passing the features to the RSSM. This results in our model comprising approximately 40 million parameters. As described in (Eqn. 2), all components of the world model are trained jointly using a modified ELBO objective. Following (Hafner, Lillicrap, Norouzi, et al., 2021), we incorporate KL balancing in this training process, which is used to control the regularization of the prior and posterior relative to each other using a parameter δ:

$$D_{\mathrm{KL}}(q \parallel p) = \delta \underbrace{D_{\mathrm{KL}}(q \parallel sg(p))}_{\text{posterior regularizer}} + (1-\delta) \underbrace{D_{\mathrm{KL}}(sg(q) \parallel p)}_{\text{prior regularizer}} \tag{D.1}$$

where $sg(\cdot)$ is the stop-gradient operator. KL balancing is necessary because the prior and posterior should not be regularized at the same rate; the prior must update more quickly towards the posterior, which encodes more information. This approach ensures the prior better approximates the aggregate posterior, enhancing the model's performance.

Name	Symbol	Value
Batch size	B	50
Sequence length	L	50
Deterministic latent state dimensions	—	1024
Discrete latent state dimensions	—	32
Discrete latent state classes	—	32
Latent dimensions	k	2048
KL loss scale	β	0.3
KL balancing	δ	0.8
RSSM reset probability	ζ	0.01
World model learning rate	—	$3 \cdot 10^{-4}$
Gradient clipping	—	100
Adam epsilon	ϵ	10^{-5}
Weight decay (decoupled)	—	$5 \cdot 10^{-2}$

Table D.1: hyperparameters for learning the world model.

To achieve uniform coverage of various segments of play data while modeling the world, we randomly sample the start index of each training sequence within the episode and clip it to prevent exceeding the episode length minus the training sequence length.

Since our approach focuses on learning from offline, unstructured teleoperated play data, it includes very long-horizon episodes and consequently very few reset states. To address this, we randomly reset the hidden state of the RSSM during training with a probability ζ. This ensures that the world model learns to attend to both the current observation and the trajectory's history while learning from long-horizon streams of data.

In all simulated and real-world experiments, we use similar hyperparameters to learn the world model, as outlined in Table D.1.

The latent state consists of deterministic and stochastic components, $\hat{s}_t = (h_t, z_t)$. Flattening the stochastic sample from 32 categorical distributions, each with 32 classes, results in a sparse binary vector of length 1024. Consequently, the latent size for each time step becomes $k = 2048$.

B Hyperparameters and Training Details of the Agent

Since our goal-reaching policy is inspired by the HULC architecture (Mees, Hermann, and Burgard, 2022), we maintain similar design choices wherever possible for a fair comparison. Any deviations from the original design are specified in this section. Notably, HULC operates on raw observations, whereas our model first infers the latent

state trajectories by the learned world model and then takes latent trajectories as input.

Perceptual Encoder While HULC encodes the sequence of visual observations from both static and gripper cameras $X_{\{static,gripper\}} \in \mathbb{R}^{T\times H\times W\times 3}$ using separate convolutional encoders, LUMOS first leverages its learned world model to infer the latent state trajectories from the camera observations and then encodes the sequence of latent states $S \in \mathbb{R}^{T\times k}$ using a fully connected encoder with two layers, featuring a hidden size of 1024 and an output size of 128 units with ReLU activations. Both methods map to the same output space, forming a perceptual representation $V \in \mathbb{R}^{T\times d}$, where T represents the sequence length and d the feature dimension.

Goal Encoder The goal-conditioned policy samples latent trajectories and uses *either* a language annotation *or* the final latent state as the goal. For each goal type, we employ a goal encoder similar to HULC. Notably, to transform raw text into a semantically meaningful vector space, we use the paraphrase-MiniLM-L3-v2 model (Reimers and Gurevych, 2019). This model is trained on paraphrase datasets primarily sourced from Wikipedia. It supports a vocabulary of $30,522$ words and converts any length of sentence into a 384-dimensional vector. Both goal encoders map their respective inputs to a latent goal representation in $\mathbb{R}^g$, where g is the latent goal dimension.

Plan Recognition and Proposal While MCIL (Lynch and Sermanet, 2021) proposed using bidirectional recurrent neural networks (RNNs) as the plan recognition network to encode randomly sampled play sequences into a latent Gaussian distribution, HULC demonstrated improved performance with a multimodal transformer encoder (Vaswani et al., 2017), incorporating positional embeddings to capture temporal information, and representing latent plans as vectors of multiple categorical variables (Hafner, Lillicrap, Norouzi, et al., 2021). We adopt HULC's method by passing the encoded perceptual representation V to the transformer, leveraging its scaled dot-product attention mechanism, which allows elements within a sequence to attend to each other. This transformer encoder consists of two blocks, eight self-attention heads, and a hidden size of 2048 units. The plan proposal network is implemented using four fully connected layers, each with 2048 units and ReLU activations. To minimize the KL divergence between these encoders, referred to as $\mathcal{L}_{\text{KL}}$, we employ the KL balancing technique (Eqn. D.1) again, ensuring that the KL loss decreases faster with respect to the prior than the posterior.

Alignment of Latent Trajectories and Language LUMOS maximizes the cosine similarity between the latent trajectory features of sequence i and its corre-

Name	Symbol	Value
Batch size	B	512
Sequence length	T	32
Minimum window length	—	20
Maximum window length	—	32
Feature dimensions	d	128
Latent goal dimensions	g	32
Discrete latent plan dimensions	—	32
Discrete latent plan classes	—	32
KL loss scale	α_1	0.1
Contrastive loss scale	α_2	3.0
KL balancing	δ	0.8
Discount factor	γ	0.995
$TD(\lambda)$ parameter	λ	0.95
Optimizer	—	Adam
Actor learning rate	—	$2 \cdot 10^{-4}$
Critic learning rate	—	$3 \cdot 10^{-4}$
Slow critic update interval	—	100

Table D.2: hyperparameters for training the actor and critic of LUMOS.

sponding language features while minimizing the similarity with unrelated language instructions within the same batch. This is achieved using a contrastive loss similar to CLIP (Radford et al., 2021). First, the latent trajectory features (x_i) and language features (y_j) are normalized, and the cosine similarity between each pair is computed. The logits for each batch form an $M \times M$ matrix, where each entry is defined as:

$$\text{logit}(x_i, y_j) = \cos\_\text{sim}(x_i, y_j) \cdot \exp(\tau), \quad \forall (i, j) \in \{1, 2, \ldots, M\},$$

with τ being a trainable temperature parameter. Only the diagonal entries of this matrix, which represent true pairs, are considered positive examples. The contrastive loss is computed as follows:

$$\mathcal{L}_{\text{contrast}} = \frac{1}{2} \left(\begin{array}{l} \displaystyle\sum_{i=1}^{M} \text{CrossEntropy}(\textit{logits\_per\_latent}(i), i) \\ + \displaystyle\sum_{i=1}^{M} \text{CrossEntropy}(\textit{logits\_per\_text}(i), i) \end{array} \right) \tag{D.2}$$

where *logits_per_latent* represents the matrix of logits with rows corresponding to latent features and columns to language features, and *logits_per_text* is the transpose of this matrix.

Action Decoder The actor decodes the actions to be executed at each time step, conditioned on the current and goal latent states as well as the sampled latent plan. These three inputs are concatenated and passed as input to the action decoder. While HULC parameterizes the action decoder as a recurrent neural network (RNN) that outputs parameters of a discretized logistic mixture distribution (Salimans et al., 2017), our action decoder is significantly different: it is not recurrent and it outputs parameters of a tanh-transformed Gaussian distribution (Haarnoja et al., 2018). HULC's action decoder consists of 2 RNN layers, each with a hidden dimension of 2048, utilizing 10 distributions to predict 10 classes per dimension. This design results in their action decoder having approximately 15 million parameters. In contrast, our action decoder is composed of eight fully connected layers, each with a hidden size of 256 units. This architecture results in a more compact model with approximately 1 million parameters. Unlike HULC, which needs a recurrent network to capture temporal patterns due to its reliance on action labels at each time step, our action decoder leverages the learned world model that encapsulates the temporal structure of latent state-action trajectories. This allows LUMOS to maximize intrinsic rewards and recover from mistakes across multiple time steps by matching the expert latent trajectory, thereby eliminating the need for a recurrent architecture.

Critic The critic of LUMOS consists of eight fully connected layers, each with a hidden size of 1024 units. It predicts the discounted sum of future rewards (state value) achieved by the actor, given the current and goal latent states. As detailed in Section 4, we use temporal-difference (TD) learning (Richard S Sutton, 1988), training the critic towards a value target constructed from intermediate rewards and future state predictions. While 1-step targets are common, our model leverages the ability to generate multi-step on-policy trajectories, allowing us to use n-step targets for faster reward incorporation. This is achieved using the λ-target (Richard S. Sutton and Barto, 2005):

$$V_t^\lambda = r_t + \gamma \left((1-\lambda) v_\psi(\hat{s}_{t+1}) + \lambda V_{t+1}^\lambda \right), \quad V_{t+H}^\lambda = v_\psi(\hat{s}_{t+H}).$$

The hyperparameter λ determines the horizon for TD learning. When $\lambda = 0$, it results in 1-step TD learning. A value of $\lambda = 1$ corresponds to unbiased Monte Carlo returns. Intermediate values of λ provide an exponentially weighted sum of n-step returns. We set $\lambda = 0.95$ to prioritize long-horizon targets over short-horizon targets.

As reported in Table D.2, during training, we randomly sample windows with lengths between 20 and 32 and pad them to a maximum length of 32 to set the training horizon for the agent.

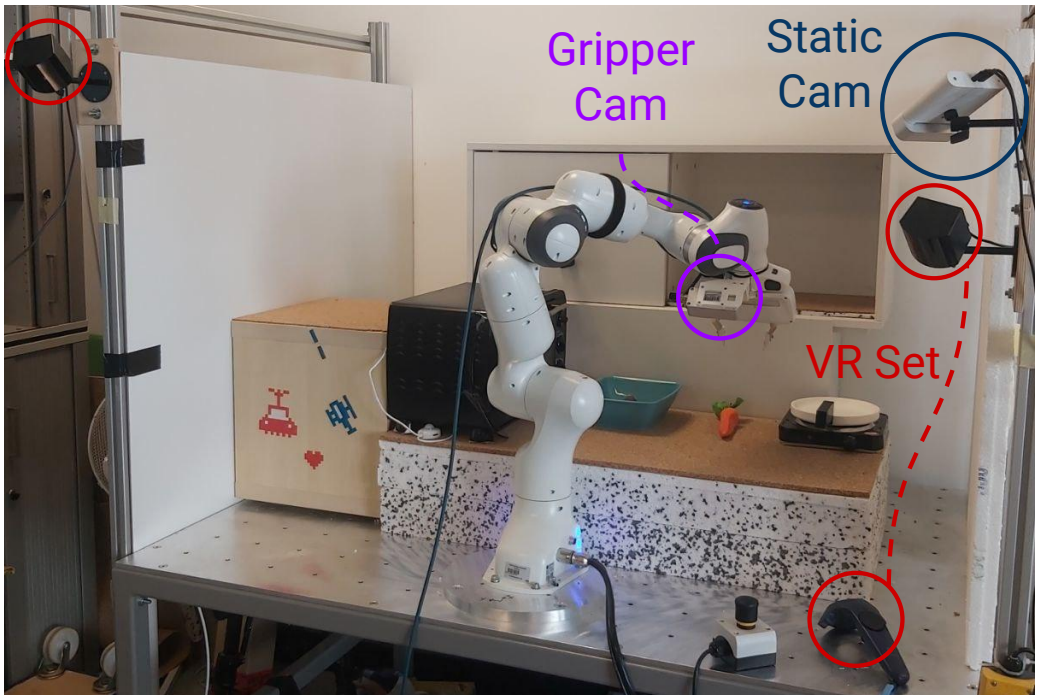

Figure D.1: Visualization of the complete real-world robot setup. This figure highlights the static and gripper cameras, as well as the VR controller and tracking system.

C Modifications to Baselines for Fair Comparisons

To efficiently handle the computational demands associated with increasing the resolution of LUMOS' world model, we standardize the input resolution of all images to 64×64. Consequently, all baselines are trained at this same resolution for a fair comparison. This reduces the resolution from the original implementations (e.g., 200×200 for the static camera and 84×84 for the gripper camera). To adapt the vision networks of our baselines to 64×64 images, we made the following changes to their convolutional layers:

- The first convolutional layer was changed from a 8×8 kernel with stride 4 to a 4×4 kernel with stride 2.
- The second convolutional layer was adjusted from a 4×4 kernel with stride 2 to a 2×2 kernel with stride 2.
- The third convolutional layer was modified from a 3×3 kernel with stride 1 to a 2×2 kernel with stride 2.

While our model does not use any image augmentation techniques, our baselines require data augmentation of image observations to facilitate learning. Specifically,

during training, we apply stochastic shifts of 0-3 pixels to both the gripper and static camera images, modifying the original shifts of 0-4 pixels and 0-10 pixels, respectively, as in Yarats *et al.* (Yarats et al., 2021). Additionally, we apply bilinear interpolation to the shifted images, replacing each pixel with the average of the nearest pixels. This augmentation technique is used consistently across all baselines.

D 7-DoF Action Framework

In all our experiments, including simulated and real-world scenarios, we employ a 7-dimensional action framework:

$$[\delta x, \delta y, \delta z, \delta\phi, \delta\theta, \delta\psi, \mathit{gripperAction}]$$

The first three dimensions, δx, δy, and δz, represent translations along the x, y, and z axes, respectively, to control the end-effector's position in 3D space. The following three dimensions, $\delta\phi$, $\delta\theta$, and $\delta\psi$, are Euler angles which specify changes in the end-effector's orientation relative to the robot's base frame. All six of these parameters accept continuous values within the range $[-1, 1]$. The final dimension, *gripperAction*, controls the gripper state. Unlike the others, it accepts discrete values: -1.0 for opening and 1.0 for closing. LUMOS treats *gripperAction* as a continuous variable between -1.0 and 1.0. To ensure compatibility with the environment, we apply thresholding before sending this value to the robot. Values greater than 0 trigger an open action, while values less than or equal to 0 cause the gripper to close.

E Data Collection Details

For our real-world experiment, we collected three hours of play data by teleoperating a Franka Emika Panda robot with a HTC VIVE Pro headset in a 3D tabletop environment (Figure D.1). The environment contained various everyday objects for interaction, including a stove, a bowl, a cabinet, and toy vegetables like a carrot and an eggplant. During data collection, we recorded a comprehensive set of sensor measurements from the robot. These measurements included proprioceptive data, which provide information about the robot's joint positions and the pose of its end-effector. Additionally, we captured static RGB images (200×200 resolution) using an Azure Kinect camera to capture the surrounding environment. We also acquired high-resolution RGB images (200×200 resolution) from a FRAMOS Industrial Depth D435e mounted on the robot's gripper, providing detailed information about the

Figure D.2: Visualization of the real-world environment observations across two time steps. At each time step, the RGB image from the static camera is presented on the left, and the RGB image from the gripper camera is shown on the right.

objects being manipulated (Figure D.2). Lastly, we logged the commanded absolute actions sent to the robot for control purposes.

For training, we derived relative actions by computing the differences in absolute actions between successive time steps. Additionally, we resized the observations to a resolution of 64×64. Given the minimal movement between frames, the original recording frequency of 30 Hz was downsampled to 15 Hz.

F Additional Experiments in Real-World

To evaluate the performance of LUMOS in executing long-horizon language-conditioned tasks in a real-world setup, we compared it with HULC (Mees, Hermann, and Burgard, 2022). We began by assessing each model's performance on 22 unique tasks (for brevity, we grouped the success rates for tasks involving either the carrot or eggplant under the single category of "vegetable"). Each task was evaluated through 20 rollouts, starting from various neutral positions to prevent bias towards any specific task. This setup ensured that the agent relied entirely on language cues to understand and execute the tasks. Our language-conditioned policy was trained within the latent space of a world model learned from the collected play data in the real world. The success rates of both models are reported in Table D.3. Moreover, we analyzed the action trajectories predicted and executed by LUMOS in the real environment by passing them to the learned world model to simulate their outcomes. The observed performance gap between LUMOS in the real environment and the world model is due to inaccuracies in the world model. This gap is typically higher for tasks with lower success rates in the real environment. We believe that training the world model with additional data that more comprehensively covers the state space will enhance

Task Method	LUMOS (ours)	HULC (Mees, Hermann, and Burgard, 2022)	World Model
Lift the vegetable from the table	67.5%	62.5%	72.5%
Lift the vegetable from the pan	75%	72.5%	77.5%
Lift the vegetable from the bowl	72.5%	67.5%	75%
Lift the vegetable from the cabinet	55%	50%	70%
Lift the pan from the table	65%	75%	70%
Lift the pan from the stove	80%	70%	75%
Lift the pan from the cabinet	35%	35%	60%
Place the vegetable onto the table	90%	85%	92.5%
Place the vegetable onto the pan	62.5%	65%	77.5%
Place the vegetable into the bowl	85%	75%	80%
Place the vegetable into the cabinet	70%	60%	77.5%
Place the pan onto the table	70%	65%	80%
Place the pan onto the stove	65%	55%	85%
Place the pan into the cabinet	55%	50%	70%
Average over tasks	**67.68**%	63.39%	75.89%
Average no. of sequential tasks	**2.05**	1.90	-

Table D.3: The average success rate of language-conditioned general-purpose policies in the real-world setup.

its robustness and reduce the observed performance gap between simulated and real environments.

Additionally, we assessed our model and HULC on 20 unique chains of 5 instructions in a row to evaluate their multi-stage long-horizon capabilities. We reported the average number of successfully completed sequential tasks, as shown in Table D.3. Our findings indicate that LUMOS outperforms HULC in both individual and multi-stage long-horizon tasks, demonstrating superior performance in recovering from its own mistakes across multiple time steps and reducing covariate shift.

These real-world experiments demonstrate the capability of LUMOS to learn language-conditioned continuous visuomotor control for a real-world robot within an offline world model. Nonetheless, our model's long-horizon performance in the real-world setting does not match its performance in the CALVIN benchmark. We attribute this discrepancy primarily to the smaller size of the real-world dataset, which is nearly half the size of the dataset used in the CALVIN simulation. To address this, we plan to collect more play data in our real-world setup, focusing on increasing the dataset size and incorporating more diverse interactions, including objects such as the oven and the sliding door of the cabinet. Additionally, we intend to open-source this real-world dataset once it is fully collected.

G Image-Conditioned Imitation Learning with LUMOS

As an additional ablation study, we train an actor-critic agent that learns long-horizon behavior conditioned solely on goal images. In this experiment, we omit the hindsight

Method	LH-MTVis			
	No. Instructions in a Row (100 chains)			
	1	2	3	Avg. Len.
LUMOS	**89.34**%±3.5	**72.34**%±4.5	**53**%±4.6	**2.54**±0.10
LUMOS (No DITTO)	78.67%±6.6	58.34%±9.4	38%±2.6	2.25%±0.18
LUMOS (No latent plan)	85.67%±4.5	67.34%±5.7	44.34%±4.5	2.43±0.12

Table D.4: Performance of LUMOS variants on environment D of the CALVIN Challenge running 100 chains per three different random seeds, using intermediate sub-goal images.

language annotation and rely exclusively on hindsight goal image relabeling during training. Consequently, this ablation excludes the semantic alignment loss ($\mathcal{L}_{\text{contrast}}$) from the agent's objective function. We follow the evaluation protocol suggested by Rosete-Beas et al. (2023) to assess our approach in the CALVIN environment. Specifically, we attempt to solve 100 unique chains of three image-based goals in succession. For each subtask, the policy is conditioned on the current sub-goal images from static and gripper cameras, transitioning to the next sub-goal only upon successful completion of the current task. This evaluation, termed long-horizon multitask with visual observations (LH-MTVis), assesses the agent's capability to perform multiple tasks sequentially.

Table D.4 presents the quantitative results of the average success rate for each variant of our model over 100 instruction chains, averaged across three seeded runs. LUMOS successfully executes multi-stage, long-horizon, vision-conditioned robotic manipulation tasks. Furthermore, similar to our language-conditioned experiments, we observe a significant drop in performance when removing our intrinsic reward function (DITTO), indicating that our reward formulation effectively mitigates covariate shift. Additionally, the removal of the plan proposal and recognition components from the LUMOS architecture results in poorer performance on longer-horizon tasks, highlighting the importance of these components for successful task completion.

Bibliography

Aaronson, Scott, Sean M. Carroll, and Lauren Ouellette (2014). *Quantifying the Rise and Fall of Complexity in Closed Systems: The Coffee Automaton*. arXiv: 1405.6903 [cond-mat.stat-mech]. URL: https://arxiv.org/abs/1405.6903.

Agarwal, Rishabh et al. (2021). "Deep Reinforcement Learning at the Edge of the Statistical Precipice". In: *Neural Information Processing Systems*.

Aghajanyan, Armen, Luke Zettlemoyer, and Sonal Gupta (2020). "Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning". In: *ArXiv* abs/2012.13255.

Andrychowicz, Marcin et al. (2017). "Hindsight experience replay". In: *NeurIPS*.

Baez, John and Mike Stay (2012). "Algorithmic thermodynamics". In: *Mathematical Structures in Computer Science* 22.5, pp. 771–787. DOI: 10.1017/S0960129511000521.

Bartlett, Peter L. and Shahar Mendelson (2003). "Rademacher and Gaussian Complexities: Risk Bounds and Structural Results". In: *J. Mach. Learn. Res.* 3, pp. 463–482.

Bengio, Yoshua, Nicholas Léonard, and Aaron C. Courville (2013). "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation". In: *ArXiv* abs/1308.3432.

Black, Kevin et al. (2023). "Zero-shot robotic manipulation with pretrained image-editing diffusion models". In: *arXiv preprint arXiv:2310.10639*.

Blau, Yochai and Tomer Michaeli (2019). "Rethinking Lossy Compression: The Rate-Distortion-Perception Tradeoff". In: *International Conference on Machine Learning*.

Brampton, Christopher (1964). "Nominalism and the Law of Parsimony". In: *The Modern Schoolman* 41, pp. 273–281.

Brantley, Kianté, Wen Sun, and Mikael Henaff (2020). "Disagreement-Regularized Imitation Learning". In: *ICLR*.

Brock, Andrew, Jeff Donahue, and Karen Simonyan (2019). "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=B1xsqj09Fm.

Burda, Yuri et al. (2019). "Exploration by Random Network Distillation". In: *ArXiv* abs/1810.12894.

Chaitin, Gregory J. (1969). "On the Length of Programs for Computing Finite Binary Sequences". In: *Journal of the ACM* 16.1, pp. 145–159. DOI: 10.1145/321495.321502.

Chi, Cheng et al. (2023). "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion". In: *Proceedings of Robotics: Science and Systems (RSS)*.

Ciosek, Kamil (2022). "Imitation Learning by Reinforcement Learning". In: *ICLR*. URL: https://openreview.net/forum?id=1zwleytEpYx.

Crooks, Gavin E. (Sept. 1999). "Entropy production fluctuation theorem and the nonequilibrium work relation for free energy differences". In: *Phys. Rev. E* 60 (3), pp. 2721–2726. DOI: 10.1103/PhysRevE.60.2721. URL: https://link.aps.org/doi/10.1103/PhysRevE.60.2721.

Dettmers, Tim et al. (2023). "QLoRA: Efficient Finetuning of Quantized LLMs". In: *ArXiv* abs/2305.14314.

Ebtekar, Aram and Marcus Hutter (Jan. 2025). "Foundations of algorithmic thermodynamics". In: *Phys. Rev. E* 111 (1), p. 014118. DOI: 10.1103/PhysRevE.111.014118. URL: https://link.aps.org/doi/10.1103/PhysRevE.111.014118.

Eddington Arthur Stanley, Sir (1928). *The Nature of the Physical World*. New York, The Macmillan Company, 1928, p. 392. URL: https://www.biodiversitylibrary.org/item/26990.

Fu, Justin, Katie Luo, and Sergey Levine (2018). "Learning Robust Rewards with Adversarial Inverse Reinforcement Learning". In: *ArXiv* abs/1710.11248.

Garg, Divyansh et al. (2021). "IQ-Learn: Inverse soft-Q Learning for Imitation". In: *ArXiv* abs/2106.12142. URL: https://api.semanticscholar.org/CorpusID:235606033.

Ghasemipour, Seyed Kamyar Seyed, Richard S. Zemel, and Shixiang Shane Gu (2019). "A Divergence Minimization Perspective on Imitation Learning Methods". In: *CoRL*.

Goldblum, Micah et al. (2023). "The No Free Lunch Theorem, Kolmogorov Complexity, and the Role of Inductive Biases in Machine Learning". In: *ArXiv* abs/2304.05366.

Goodfellow, Ian J. et al. (2014). "Generative Adversarial Nets". In: *NIPS*.

Grimm, Christopher et al. (2020). "The Value Equivalence Principle for Model-Based Reinforcement Learning". In: *ArXiv* abs/2011.03506. URL: https://api.semanticscholar.org/CorpusID:226278317.

Groot, S. R. de and P. Mazur (1962). *Non-Equilibrium Thermodynamics*. Amsterdam: North-Holland Publishing Company.

Ha, David and Jürgen Schmidhuber (2018). "World Models". In: *arXiv preprint arXiv:1803.10122*.

Ha, David R and Jürgen Schmidhuber (2018). "Recurrent World Models Facilitate Policy Evolution". In: *ArXiv* abs/1809.01999.

Ha, Huy, Pete Florence, and Shuran Song (2023). "Scaling up and distilling down: Language-guided robot skill acquisition". In: *Conference on Robot Learning*.

Haarnoja, Tuomas et al. (2018). "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *arXiv preprint arXiv:1801.01290*.

Hafner, Danijar, Timothy P. Lillicrap, Jimmy Ba, et al. (2020). "Dream to Control: Learning Behaviors by Latent Imagination". In: *ArXiv* abs/1912.01603.

Hafner, Danijar, Timothy P. Lillicrap, Mohammad Norouzi, et al. (2021). "Mastering Atari with Discrete World Models". In: *ArXiv* abs/2010.02193.

Hafner, Danijar, Jurgis Pasukonis, et al. (2023). "Mastering diverse domains through world models". In: *arXiv preprint arXiv:2301.04104*.

Harnad, Stevan (1990). "The Symbol Grounding Problem". In: *Physica D: Nonlinear Phenomena*.

Henighan, Tom et al. (2020). "Scaling Laws for Autoregressive Generative Modeling". In: *ArXiv* abs/2010.14701.

Hester, Todd et al. (2017). "Learning from Demonstrations for Real World Reinforcement Learning". In: *ArXiv* abs/1704.03732.

Hinton, Geoffrey E. and Drew van Camp (1993). "Keeping the neural networks simple by minimizing the description length of the weights". In: *Annual Conference Computational Learning Theory*.

Ho, Jonathan and Stefano Ermon (2016). "Generative Adversarial Imitation Learning". In: *NIPS*.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989). "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5, pp. 359–366. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8. URL: https://www.sciencedirect.com/science/article/pii/0893608089900208.

Hu, J. Edward et al. (2021). "LoRA: Low-Rank Adaptation of Large Language Models". In: *ArXiv* abs/2106.09685.

Hwangbo, Jemin et al. (2019). "Learning agile and dynamic motor skills for legged robots". In: *Science Robotics* 4.26, eaau5872. DOI: 10.1126/scirobotics.aau5872. eprint: https://www.science.org/doi/pdf/10.1126/scirobotics.aau5872. URL: https://www.science.org/doi/abs/10.1126/scirobotics.aau5872.

Jarzynski, C. (Apr. 1997). "Nonequilibrium Equality for Free Energy Differences". In: *Phys. Rev. Lett.* 78 (14), pp. 2690–2693. DOI: 10.1103/PhysRevLett.78.2690. URL: https://link.aps.org/doi/10.1103/PhysRevLett.78.2690.

Jaynes, E. T. (May 1957a). "Information Theory and Statistical Mechanics". In: *Phys. Rev.* 106 (4), pp. 620–630. DOI: 10.1103/PhysRev.106.620. URL: https://link.aps.org/doi/10.1103/PhysRev.106.620.

— (Oct. 1957b). "Information Theory and Statistical Mechanics. II". In: *Phys. Rev.* 108 (2), pp. 171–190. DOI: 10.1103/PhysRev.108.171. URL: https://link.aps.org/doi/10.1103/PhysRev.108.171.

Kanervisto, Anssi, Joonas Pussinen, and Ville Hautamäki (2020). "Benchmarking End-to-End Behavioural Cloning on Video Games". In: *2020 IEEE Conference on Games (CoG)*, pp. 558–565.

Kaplan, Jared et al. (2020). *Scaling Laws for Neural Language Models*. arXiv: 2001.08361 [cs.LG]. URL: https://arxiv.org/abs/2001.08361.

Ke, Tsung-Wei, Nikolaos Gkanatsios, and Katerina Fragkiadaki (2024). "3d diffuser actor: Policy diffusion with 3d scene representations". In: *arXiv preprint arXiv:2402.10885*.

Kidambi, Rahul, Jonathan Chang, and Wen Sun (2021). "MobILE: Model-Based Imitation Learning From Observation Alone". In: *NeurIPS*.

Kim, Geon-Hyeong et al. (2022). "DemoDICE: Offline Imitation Learning with Supplementary Imperfect Demonstrations". In: *International Conference on Learning Representations*. URL: https://api.semanticscholar.org/CorpusID:251647997.

Kingma, Diederik P and Max Welling (2013). "Auto-Encoding Variational Bayes". In: *arXiv preprint arXiv:1312.6114*.

Kolchinsky, Artemy and David H. Wolpert (Aug. 2020). "Thermodynamic costs of Turing machines". In: *Phys. Rev. Res.* 2 (3), p. 033312. DOI: 10.1103/

PhysRevResearch.2.033312. URL: https://link.aps.org/doi/10.1103/PhysRevResearch.2.033312.

Kolmogorov, A. N. (1965). "Three Approaches to the Quantitative Definition of Information". In: *Problems of Information Transmission* 1.1, pp. 1–7.

Kostrikov, Ilya, Ofir Nachum, and Jonathan Tompson (2019). "Imitation Learning via Off-Policy Distribution Matching". In: *ArXiv* abs/1912.05032. URL: https://api.semanticscholar.org/CorpusID:209202457.

Langford, John and Matthias Seeger (2001). "Bounds for averaging classifiers." In: *Carnegie Mellon*. URL: https://www.cs.cmu.edu/~jcl/papers/averaging/averaging_tech.pdf.

Laplace, Pierre Simon (1825). *Essai philosophique sur les probabilités*. Bachelier.

Li, Chunyuan et al. (2018). "Measuring the Intrinsic Dimension of Objective Landscapes". In: *ArXiv* abs/1804.08838.

Li, Ming and Paul M. B. Vitányi (1993). "An Introduction to Kolmogorov Complexity and Its Applications". In: *Graduate Texts in Computer Science*.

Liu, Ziming, Ouail Kitouni, et al. (2022). "Towards Understanding Grokking: An Effective Theory of Representation Learning". In: *ArXiv* abs/2205.10343.

Liu, Ziming, Eric J. Michaud, and Max Tegmark (2022). "Omnigrok: Grokking Beyond Algorithmic Data". In: *ArXiv* abs/2210.01117.

Lotfi, Sanae et al. (2024). "Non-Vacuous Generalization Bounds for Large Language Models". In: *ICML 2024*. arXiv: 2312.17173 [stat.ML]. URL: https://arxiv.org/abs/2312.17173.

Luo, Yicheng et al. (2023). "Optimal Transport for Offline Imitation Learning". In: *ArXiv* abs/2303.13971. URL: https://api.semanticscholar.org/CorpusID:257757426.

Lynch, Corey, Mohi Khansari, et al. (2019). "Learning latent plans from play". In: *CoRL*.

Lynch, Corey and Pierre Sermanet (2021). "Language Conditioned Imitation Learning Over Unstructured Data". In: *RSS*.

Ma, Shuming et al. (2024). "The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits". In: *ArXiv* abs/2402.17764.

Ma, Yecheng Jason et al. (2022). "Versatile Offline Imitation from Observations and Examples via Regularized State-Occupancy Matching". In: *International Conference on Machine Learning*. URL: https://api.semanticscholar.org/CorpusID:249921233.

Maddox, Wesley J., Gregory W. Benton, and Andrew Gordon Wilson (2020). "Rethinking Parameter Counting in Deep Models: Effective Dimensionality Revisited". In: *ArXiv* abs/2003.02139.

Mees, Oier, Jessica Borja-Diaz, and Wolfram Burgard (2023). "Grounding Language with Visual Affordances over Unstructured Data". In: *ICRA*. London, UK.

Mees, Oier, Lukas Hermann, and Wolfram Burgard (2022). "What matters in language conditioned robotic imitation learning over unstructured data". In: *IEEE Robotics and Automation Letters*.

Mees, Oier, Lukas Hermann, Erick Rosete-Beas, et al. (2022). "CALVIN: A Benchmark for Language-Conditioned Policy Learning for Long-Horizon Robot Manipulation Tasks". In: *IEEE Robotics and Automation Letters (RA-L)*.

Micikevicius, Paulius et al. (2017). "Mixed Precision Training". In: *ArXiv* abs/1710.03740.

Mitchell, Tom M. (1980). *The Need for Biases in Learning Generalizations*. Tech. rep. New Brunswick, NJ: Rutgers University. URL: http://dml.cs.byu.edu/~cgc/docs/mldm_tools/Reading/Need%20for%20Bias.pdf.

Mnih, Volodymyr, Adrià Puigdomènech Badia, et al. (2016). "Asynchronous Methods for Deep Reinforcement Learning". In: *ICML*.

Mnih, Volodymyr, Koray Kavukcuoglu, et al. (2015). "Human-level control through deep reinforcement learning". In: *Nature* 518, pp. 529–533.

Nakkiran, Preetum et al. (2019). "Deep double descent: where bigger models and more data hurt". In: *Journal of Statistical Mechanics: Theory and Experiment* 2021.

Nanda, Neel et al. (2023). "Progress measures for grokking via mechanistic interpretability". In: *ArXiv* abs/2301.05217.

Narayanan, Hariharan and Sanjoy Mitter (2010). "Sample Complexity of Testing the Manifold Hypothesis". In: *Advances in Neural Information Processing Systems*. Ed. by J. Lafferty et al. Vol. 23. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2010/file/8a1e808b55fde9455cb3d8857ed88389-Paper.pdf.

Peng, Xue Bin et al. (2017). "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization". In: *ICRA*. Brisbane, Australia.

Petersen, Felix et al. (2022). *Deep Differentiable Logic Gate Networks*. arXiv: 2210.08277 [cs.LG]. URL: https://arxiv.org/abs/2210.08277.

Pomerleau, Dean A. (1989). "ALVINN: An Autonomous Land Vehicle in a Neural Network". In: *NeurIPS*.

Power, Alethea et al. (2022). "Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets". In: *ArXiv* abs/2201.02177.

Radford, Alec et al. (2021). "Learning transferable visual models from natural language supervision". In: *arXiv preprint arXiv:2103.00020*.

Rafailov, Rafael et al. (2021). "Visual Adversarial Imitation Learning using Variational Models". In: *ArXiv* abs/2107.08829.

Raffin, Antonin (2020). *RL Baselines3 Zoo*. https://github.com/DLR-RM/rl-baselines3-zoo.

Reddy, Siddharth, Anca D. Dragan, and Sergey Levine (2020). "SQIL: Imitation Learning via Reinforcement Learning with Sparse Rewards". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=S1xKd24twB.

Reimers, Nils and Iryna Gurevych (2019). "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *EMNLP*.

Reuss, Moritz, Maximilian Li, et al. (2023). "Goal-conditioned imitation learning using score-based diffusion policies". In: *arXiv preprint arXiv:2304.02532*.

Reuss, Moritz, Ömer Erdinç Yağmurlu, et al. (2024). "Multimodal Diffusion Transformer: Learning Versatile Behavior from Multimodal Goals". In: *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*.

Rissanen, Jorma (1978). "Modeling By Shortest Data Description*". In: *Autom.* 14, pp. 465–471.
Rosete-Beas, Erick et al. (2023). "Latent plans for task-agnostic offline reinforcement learning". In: *Conference on Robot Learning*.
Ross, Stéphane and J. Andrew Bagnell (2010). "Efficient Reductions for Imitation Learning". In: *AISTATS*.
Ross, Stéphane, Geoffrey J. Gordon, and J. Andrew Bagnell (2011). "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning". In: *AISTATS*.
Roy, Olivier and Martin Vetterli (2007). "The effective rank: A measure of effective dimensionality". In: *2007 15th European Signal Processing Conference*, pp. 606–610.
Rudin, Nikita et al. (2022). "Learning to walk in minutes using massively parallel deep reinforcement learning". In: *Conference on Robot Learning*. PMLR.
Salimans, Tim et al. (2017). "Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications". In: *arXiv preprint arXiv:1701.05517*.
Sasaki, Fumihiro and Ryota Yamashina (2021). "Behavioral Cloning from Noisy Demonstrations". In: *ICLR*.
Schrittwieser, Julian et al. (2020). "Mastering atari, go, chess and shogi by planning with a learned model". In: *Nature*.
Schrödinger, Erwin (1946). *What is life? : the physical aspect of the living cell*. New York, The Macmillan Company.
Schulman, John et al. (2017). "Proximal Policy Optimization Algorithms". In: *ArXiv* abs/1707.06347.
Shalev-Shwartz, Shai and Shai Ben-David (2014). "Understanding Machine Learning - From Theory to Algorithms". In.
Shannon, Claude E. (1948). "A mathematical theory of communication". In: *Bell Syst. Tech. J.* 27, pp. 623–656.
Sharma, Utkarsh and Jared Kaplan (2020). "A Neural Scaling Law from the Dimension of the Data Manifold". In: *ArXiv* abs/2004.10802. URL: `https://api.semanticscholar.org/CorpusID:216080945`.
Solomonoff, Ray J. (1964a). "A Formal Theory of Inductive Inference. Part I". In: *Inf. Control.* 7, pp. 1–22.
— (1964b). "A Formal Theory of Inductive Inference. Part II". In: *Inf. Control.* 7, pp. 224–254.
Sutton, Richard S (1988). "Learning to predict by the methods of temporal differences". In: *Machine learning*.
Sutton, Richard S. and Andrew G. Barto (2005). "Reinforcement Learning: An Introduction". In: *IEEE Transactions on Neural Networks*.
Theis, Lucas (2024). "What makes an image realistic?" In: *ArXiv* abs/2403.04493.
Vapnik, Vladimir Naumovich (1991). "Principles of Risk Minimization for Learning Theory". In: *Neural Information Processing Systems*.
Vapnik, Vladimir Naumovich and Alexey Chervonenkis (1971). "On the uniform convergence of relative frequencies of events to their probabilities". In.
Varma, Vikrant et al. (2023). "Explaining grokking through circuit efficiency". In: *ArXiv* abs/2309.02390.

Vaswani, Ashish et al. (2017). "Attention is all you need". In: *NeurIPS*.

Vereshchagin, Nikolai K. and Paul M. B. Vitányi (2004). "Rate Distortion and Denoising of Individual Data Using Kolmogorov Complexity". In: *IEEE Transactions on Information Theory* 56, pp. 3438–3454.

Wang, Ruohan et al. (2019). "Random Expert Distillation: Imitation Learning via Expert Policy Support Estimation". In: *ICML*.

Wigner, Eugene P. (1960). "The unreasonable effectiveness of mathematics in the natural sciences. Richard courant lecture in mathematical sciences delivered at New York University, May 11, 1959". In: *Communications on Pure and Applied Mathematics* 13.1, pp. 1–14. DOI: https://doi.org/10.1002/cpa.3160130102. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpa.3160130102. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.3160130102.

Williams, Ronald J. (2004). "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine Learning* 8, pp. 229–256.

Wolpert, David H. and William G. Macready (1997). "No free lunch theorems for optimization". In: *IEEE Trans. Evol. Comput.* 1, pp. 67–82.

Wu, Philipp et al. (2023). "Daydreamer: World models for physical robot learning". In: *Conference on Robot Learning*. PMLR.

Xian, Zhou et al. (2023). "Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation". In: *Conference on Robot Learning*.

Yarats, Denis et al. (2021). "Mastering Visual Continuous Control: Improved Data-Augmented Reinforcement Learning". In: *arXiv preprint arXiv:2107.09645*.

Zhang, Edwin et al. (2024). "Language Control Diffusion: Efficiently Scaling through Space, Time, and Tasks". In: *The Twelfth International Conference on Learning Representations*.

Zhang, Ruohan et al. (2020). "Atari-HEAD: Atari Human Eye-Tracking and Demonstration Dataset". In: *Proceedings of the ... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence* 34 4, pp. 6811–6820.

Zhou, Hongkuan et al. (2023). "Language-conditioned imitation learning with base skill priors under unstructured data". In: *arXiv preprint arXiv:2305.19075*.

Zwanzig, R. (2001). *Nonequilibrium Statistical Mechanics*. Oxford University Press. ISBN: 9780198032151.