Exploring the Intersection of Physics Modeling and Representation Learning

by

Ouail Kitouni

B.Sc. Physics, University of Rochester, 2019

Submitted to the Department of Physics
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN PHYSICS, STATISTICS, AND DATA SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2024

Authored by: Ouail Kitouni
Department of Physics
August 14, 2024

Certified by: Mike Williams
Professor of Physics, Thesis Supervisor

Accepted by: Lindley Winslow
Professor of Physics
Associate Department Head, Department of Physics

Exploring the Intersection of Physics Modeling and Representation Learning

by

Ouail Kitouni

Submitted to the Department of Physics
on August 14, 2024 in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN PHYSICS, STATISTICS, AND DATA SCIENCE

ABSTRACT

Representation Learning has evolved into a multi-purpose tool capable of solving arbitrary problems provided enough data. This thesis focuses on two primary directions: (1) Harnessing the power of deep learning for applications in fundamental physics and (2) using physics-inspired tools to improve and shed some light on otherwise large-scale, inscrutable black-box algorithms. We explore a collection of applications that improve different aspects of nuclear and particle physics research across its many stages, from online data selection to offline data analysis. We also tease out how deep learning can open up entirely new avenues of research through the lens of mechanistic interpretability to (re)derive fundamental theory as well as new ways to reinterpret physics measurements. Lastly, we study how physics tools can be useful to better understand the dynamics of deep learning as well as provide a solid foundation for algorithms and training paradigms that expand the frontier of machine learning.

Thesis supervisor: Mike Williams
Title: Professor of Physics

Acknowledgments

First, I extend my gratitude to my advisor, Mike Williams, for his insightful mentorship, many words of wisdom, and for always being an inspiration in numerous ways that extend well beyond academics. Mike fights hard for his students and leaves no stone unturned to help them achieve their goals. Research is a long and perilous journey that can be more treacherous (and depressing) without supportive and reliable collaborators. Thus, I thank Niklas Nolte, my comrade-in-arms, for never failing to answer the call whenever I had a crazy new idea and for being responsible for some of my best times at MIT. I am grateful for the difficult problems, long hours, and sleepless nights that turned into productive and enjoyable experiences while working together. I am thankful to Nik, along with Camila Pazos, for all the fun times we had together and for being great roommates and exploration companions to the literal ends of the earth.

I owe much to the many scientists I had the pleasure of working with throughout the years: Ben Nachman, Constantin Weisser, Eric Michaud, Ziming Liu, Max Tegmark, Phil Harris, Mark Ibrahim, Mike Rabbat, Adina Williams, Bhaskar Mitra, James Hensman, and Sokratis Trifinopoulos. I also owe much to Jesse Thaler for his leadership and for helping create the interdisciplinary degree in Physics, Statistics, and Data Science, as well as the Institute for AI and Fundamental Interactions, both of which have been crucial in my Ph.D. journey.

I am thankful to the people who have made my experience at MIT more fun and engaging, in particular, my friends from the 2019 Course 8 Ph.D. class Nick Kamp, Andrew Tan, Sam Alipour-fard, Caolan John, Patrick Oar, David Rower, and Tri Nguyen, who has been a friend and classmate since our undergraduate days. I am thankful to my housemates Matt Vernacchia, Daniel Shaar, and Alexander Siegenfeld, from whom I learned so much through numerous interesting conversations, especially during the COVID lockdown, which they have made a lot more bearable.

I sincerely thank my parents, Yacine Kitouni and Zakia Berkani, who taught me at a young age the value of hard work and that, as cliché as that may sound, you really can achieve anything if you set your mind to it. Their unconditional support and full faith in me allowed me to cross continents in pursuit of my goals, and for that, I am forever grateful. I'm also grateful for my siblings, for their strength and wisdom, and for generally being awesome.

Importantly, the best thing that happened to me during these Ph.D. years, and what I am most grateful for, is marrying my best friend, Sara Matmatte. She has always been (and continues to be) a true inspiration in her dedication and hard work, which incidentally has motivated much of this thesis. Thank you for being willing and eager to listen to my many complaints, wild theories, and research grumblings. Thank you for your love and support for

so many years. And finally, I'm deeply grateful to you and the entire Matmatte household for welcoming me so warmly into the family.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

I would like for this thesis to be as self-contained as possible. To do so, I will endeavor to define *most* of the things I will talk about. Some of these concepts already have definitions, but I will not necessarily regurgitate them here. Instead, I will focus on my own interpretations, which I hope will be more relevant to better understanding the thesis as a whole. Let's lay down the basics.

What is representation learning? Representation learning is the idea that we can train, fit, or otherwise optimize models to extract useful features from some data to solve arbitrary problems. This feature extraction is essentially a byproduct of the training paradigm and a happy little coincidence through which we have been able to train models that *generalize*. Representation learning is the primary idea that has been spearheading all the recent progress in *artificial intelligence*, which—for the purposes of this thesis—will be taken to broadly refer to the process of extracting structure from data using modern neural networks trained on large-scale data to achieve a specific goal.

Neural networks enjoy the so-called *universal approximation theorem*, or the fact that they can fit any data arbitrarily well. Despite—or thanks to, I'm not entirely sure—their ability to fit anything, neural networks tend to do so in a way that obeys some formulation of Occam's razor: they find solutions that generalize to unseen data.

What does this have to do with physics? Quite a lot, actually. Neural networks are complex systems and, as such, are fascinating test subjects for physicists who cannot help but work on every hard problem they can get their hands on. At the same time, neural networks are so capable that they can, at the very least, make the study of the universe a slightly more straightforward undertaking. In this thesis, I will focus on three ways representation learning can make a physicist's job easier:

1. Algorithmic advances to improve specialized studies.
2. Novel ways to probe physics data.
3. Automated understanding.

Due to the current limitations of representation learning, we will constrain our topics to highly specialized applications. However, I suspect that in the not-so-far future, significant chunks of physics and all scientific research will be automated by machines. The final chapters of this thesis will contain some of the more speculative portions of my research on improving machine learning models' capability to reason and plan, which incidentally rely heavily on physics-inspired tools. Hopefully, our machine learning models will be able to plan and reason well enough to begin this process of scientific automation soon.

1.1 What is Deep Learning?

Much of the current advances in artificial intelligence at the time of writing this thesis are due to scaling. Scaling laws are a tool physicists are intimately familiar with, and they have been a guiding principle in the development of frontier AI systems. Training bigger models with more parameters on more data appears to be a true panacea, predictably and systematically curing the limitations of prior generations of models with no sign of stopping. Perhaps we are indeed beginning to chip away at some universal law linked to the nature of intelligence. However, something AI researchers have yet to borrow from the physicist's toolkit is the study of limiting behavior. It is currently unclear how long we can keep riding this power law through ever-expanding frontiers of intelligence. Leading labs have exhausted the Internet's data, expended hundreds of millions on single training runs, used enough energy to power dozens of small towns, and continued to race towards the first trillion-dollar compute cluster. Evidently, continuing down this path will require orders of magnitude improvements in efficiency, and many of the current pipelines (algorithms, data curation, supply chains, *etc.*) are far from optimal. While I firmly believe future generations of models will use drastic algorithmic changes that would bring all of these costs closer to the evolutionary upper bound of 20 Watts/human brain, it seems the current recipe works. Hence, it is at least worth a few sections of the thesis.

This chapter will review an abridged history of deep learning and the main components leading us to current frontier models. Sprinkled throughout, we will find hints of physics ideas and concepts that justify physicist's involvement in the field. At this point, it seems appropriate to define two terms I've been using liberally and interchangeably: neural networks and deep learning. The classic image of neural networks is an affine transformation (matrix multiplication and vector addition) followed by a non-linearity (activation function). For a d-dimensional input $x \in \mathbb{R}^d$, we can obtain an o-dimensional output $h \in \mathbb{R}^o$ as follows

$$h = \sigma(Wx + b), \tag{1.1}$$

for some function σ and parameters $W \in \mathbb{R}^{o \times d}$ and $b \in \mathbb{R}^o$. Input/output nodes are referred to as neurons, and by stacking many of them, we obtain a *neural network*. Stacking just two such layers is sufficient to fit arbitrarily complex functions provided you have enough neurons *i.e.*, have a shallow but wide network. Stacking many leads to the regime of *deep learning*. Equation (1.1) is but one possible formulation. As we will see, there are many variations and extensions, each with desirable properties. For instance, the parameters in the layers could be different or shared in specific ways, they could be constrained to certain

values, they can be fixed or functions of their inputs, the layers need not be stacked and can be connected arbitrarily, they can be reapplied recursively on their outputs, the activation function need not be point-wise and can affect neurons differently or contain high-order cross-terms, *etc.* We will explore some of these esoteric choices in later chapters, but for now we will focus on the condensed history of conventional approaches to creating and training deep neural networks.

1.2 An Abridged History of Neural Networks

Some will have you believe that there is nothing new under the sun, that neural networks have been around for decades, or that *grokking* (a topic we will revisit later) was well-known years before OpenAI published a paper on it in 2022 [1]. While there is some truth to this, I don't believe it to be the whole picture. The early neural networks are quite different from today's deep learning. Technically speaking, linear models optimized with least squares are neural networks, and they date back to Gauss and Legendre (1800s). The first non-learning recursive neural network (RNN) was introduced in the 1920s by physicists Ernst Ising and Wilhelm Lenz [2]. And in 1958, Psychologist Frank Rosenblatt introduced the first perceptron [3], ushering in the golden age of AI. Public excitement about artificial neural networks led to a boom in funding from the US government. Herbert Simon [4] writes, "There are now in the world machines that think, that learn and that create. Moreover, their ability to do these things is going to increase rapidly until—in a visible future—the range of problems they can handle will be coextensive with the range to which the human mind has been applied."

Unfortunately, progress in AI research stagnated, and funding dwindled as the community realized that current computer systems lacked the power to train useful neural networks [5]. In the meantime, the eastern world was still making some progress. Alexey Ivakhnenko implemented the first feed-forward deep neural network in 1965 [6], [7]. None of the works up to this point used backpropagation as we know it today. It was not until 1967 that Shun'ichi Amari published the first multi-layer perceptron (MLP) [8] trained with stochastic gradient descent. Still, only in 1982 did Paul Werbo [9] first implement back-propagation (the Leibniz chain rule) to train an MLP in what will become standard in modern-day deep learning.

In the 1990s, Yann LeCun trained CNNs for image recognition on handwritten digits [10], which were later applied by several banks. Around the same time, Juergen Schmidhuber and Sepp Hochreiter worked on a variety of neural architectures, including the widely cited Long Short-Term Memory (LSTM) [11] recursive neural networks. In 2009, neural networks began winning prizes at various machine learning competitions [12]–[15], primarily thanks to GPUs unlocking the training of deep models with back-propagation. The new AI golden age began.

Researchers built a wide array of approaches and architectures on the "NN-backprop" foundation, including Genartive Adversarial Networks [16], Variational Autoencoders [17], Normalizing Flows [18], Residual connections [19] for deep networks, and many others culminating in the introduction of the Transformer. This architecture, introduced by Vaswani et al. [20], officially cemented deep learning as a general-purpose approach to artificial intelligence. Pre-training large transformers became a staple across disciplines, including vision, code generation, and, importantly, language. The lines between these historically distinct subfields of machine learning begin to blur as we enter the age of multi-modal

foundation models. We will discuss transformers' ability to reason in the language modality towards the end of the thesis, but for now, we will build up deep learning intuition focusing on inductive biases and physics applications.

1.3 Neural Network Fundamentals

Neural networks are parameterized functions that map an input x into an output y. Neural network parameters are usually denoted by θ, and they are optimized to minimize some loss function ℓ. The inputs and outputs can be almost anything, including numerical or categorical values, sequences, tables, images, *etc.* For now, we will focus on the simple case where we have access to a ground-truth scalar label $y \in \mathbb{R}$ and where the input is a vector $x \in \mathbb{R}^n$. The neural network output is then $f(x;\theta)$ which can be fit to some empirically measured dataset $\mathcal{D} = \{(x^i, y^i)\}_{i=1}^N$ via so-called empirical risk minimization

$$f^* = \arg\min_f \mathcal{L}(f, y) = \arg\min_f \frac{1}{N} \sum_i \ell[(f(x^i), y^i], \tag{1.2}$$

where $\mathcal{L}$ is the empirical risk. The dependence on θ is implicit. Alternatively, we can write it in terms of the parameters directly as follows

$$\theta^* = \arg\min_\theta \mathcal{L}[(f(x;\theta), y]. \tag{1.3}$$

This is a valuable framework to develop intuition, but deep learning can take many forms. Sometimes, models are trained without access to explicit labels. For instance, contrastive language-image pre-training (CLIP) [21] aims to increase alignment between semantically similar inputs (usually different modalities of the same object) while decreasing alignment with negatives (other inputs that are meaningfully different). CLIP, or contrastive learning more broadly, does not fit the paradigm outlined above because of the interaction between different samples in the dataset. Still, Equation (1.3) is general enough to cover most cases encountered in the rest of this thesis.

1.3.1 Gradient descent

During training, neural network parameters are updated following the basic rule

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\mathcal{D}; \theta_t) \tag{1.4}$$

This is "full-batch" gradient descent with learning rate η. In practice, smaller batches are used by uniformly sampling points from the training dataset without replacement. This is called stochastic gradient descent or SGD (sometimes, this term specifically refers to gradient descent with batch size 1). SGD is primarily used due to memory constraints. GPUs are usually not large enough to store the entire training set in memory. However, there has been much recent work that shows that SGD has desirable theoretical properties related to the specific inductive biases it promotes.

This is as good a time as any to talk about inductive biases. What is an inductive bias? As discussed earlier, neural networks are capable of modeling arbitrary functions, but for

most problems, there are many admissible solutions (this is especially true for small datasets). An inductive bias is a model's biased preference for particular solutions or classes of functions. For instance, when dealing with sets, a useful inductive bias for a neural network is the invariance to permutations of the input elements. Inductive biases are important to avoid overfitting (*i.e.*, obtaining solutions that perform well on the training dataset but fail to generalize beyond it), in particular, in low data regimes. We will visit examples of inductive biases in later chapters.

Obtaining gradients with respect to each parameter uses the chain rule, and for neural networks with many submodules with non-trivial interactions (*e.g.*, recursion), gradient computation can quickly get out of hand. Popular "backprop" packages such as PyTorch [22] effectively democratized deep learning and made experimentation with different architectures, loss functions, and training paradigms much more straightforward.

Nowadays, the update rule in Equation (1.4) is replaced by so-called adaptive methods. Different parameters in the network can have vastly different magnitudes, which in turn usually leads to gradients of vastly different sizes. To remedy this issue—and ensure different parameters learn sufficiently during each update—the effective learning rate for each parameter is scaled by the norm of that parameter's gradient (usually estimated by a moving average). Another popular modification is adding a momentum term. As the name implies, this term effectively adds some inertia to the training dynamics. The parameters continue along the direction dictated by prior updates, mostly unaffected by potentially noisy gradients in random directions. This change was demonstrated empirically to improve convergence on most problems. Adam [23] combines both of these changes and reigns supreme as the defacto optimizer in modern deep learning. It may not be optimal for most cases, but it works out of the box in most scenarios, which makes it highly practical.

Why not use higher-order methods? The sheer number of parameters used in a neural network makes higher-order terms impractical to compute. As we will see next, scaling the number of parameters is a cornerstone of deep learning, so the community prioritizes methods that scale well with parameter count.

1.3.2 Scaling recipes

This section will discuss the holy grail of deep learning: scalable, stable training. Large training runs, such as the one that gave us ChatGPT, are more of an art than a science—at least outside of the big scaling labs. That's not to say we have no idea how to make things work. The basic principle is that one wants to avoid vanishing/exploding gradients and activations at initialization and throughout training.

Residual Connections: The first step in the scaling recipe is residual connections [19], which have allowed for training very deep networks. The key insight here is that it is easier to optimize residual functions on top of an input than it is to optimize unreferenced functions. One reason for that is that gradient signals can propagate easily through the residual connection from the end of the network to the earliest weights without any interruption. To see this, let's write our L-layer deep standard (feed-forward) neural network function in terms

of individual layers as follows

$$h^{(l)} = \sigma(W^l h^{(l-1)} + b^{(l)}) \quad \text{for } l = 1, 2, \ldots, L, \tag{1.5}$$

where h_0 is the input x and $h^{(L)}$ is the output $f(x;\theta)$. The gradient of $W^{(1)}$ has all sorts of dependencies on subsequent layers, and if any of them breaks *e.g.*, due to vanishing gradients, so will $\nabla_{W^{(1)}}$. Let's rewrite it with a residual connection instead

$$h^{(l)} = \sigma(W^{(l)} h^{(l-1)} + b^{(l)}) + h^{(l-1)} \quad \text{for } l = 1, 2, \ldots, L. \tag{1.6}$$

This innocuous change makes a big difference. If we were to unroll the entire computation, we would have something like

$$f(x) = \sigma(W^{(L)}(\sigma(W^{(L-1)}\sigma(\cdots) + b^{(L-1)})) + b^{(L)}) + \cdots + \sigma(W^{(1)}x + b^{(1)}), \tag{1.7}$$

which directly connects the first layer parameters to the output. This is what allows more stable gradients and, thus, more stable training of deep networks with residual connections.

A well-known result in deep learning is that layer depth improves "computational depth" *i.e.*, deeper models can perform more complex computations compared to shallow ones. Early empirical evidence for this was the ResNet architecture winning various machine vision competitions in 2015, cementing deep learning as a strong paradigm for machine learning and artificial intelligence.

We will use residual connections in much of the later chapters because they can also have various other interesting properties. For instance, they can implement useful inductive biases like monotonicity (Chapter 2), and they allow for more interpretable representations (Chapter 5).

Normalization: The second step in the scaling recipe is proper normalization. At least in the case of standard parameterizations, the training procedure leads to model weights "aligning" with the inputs such that activations become too large or deviate from the regime where things are stable and learnable. This is somewhat hand-wavy as no established definitions exist currently for this "alignment". An intuitive picture can be drawn by taking the linear regression case. Suppose we optimize the mean-squared-error $\mathcal{L}$ of a linear model $w \in \mathbb{R}^d$ given n d-dimensional inputs $X \in \mathbb{R}^{n \times d}$ on some targets $Y \in \mathbb{R}^n$.

$$\mathcal{L} = (Xw - Y)^2. \tag{1.8}$$

Optimizing the weights with gradient descent gives the following update rule

$$w^t = w^{t-1} - \eta \nabla_w \mathcal{L}, \tag{1.9}$$

where $\nabla_w \mathcal{L} \propto X^T X w - X^T Y = (\sum_i e_i s_i^2 e_i^T) w + X^T Y$ where e_i and s_i are the ith singular vector and singular value, respectively (I didn't distinguish between right and left singular vectors for simplicity). This tells us that w gets updates weighted by the singular values of X, s_i, and the more aligned it becomes with the top singular vector e_1, the larger the update gets. Stacking multiple such layers creates more opportunities for this alignment to occur, and the stack is only as strong as its weakest link—or largest singular value in this

case. If the largest singular value is too large, training can become unstable. This is a loose explanation, but it gives intuition as to why activations tend to increase during training and why deep network optimization fails if one is not careful.

To remedy this, practitioners tend to use different normalization schemes like BatchNorm [24], which normalizes activations by the standard deviation over a batch of elements. BatchNorm was popular in the early days of modern deep learning. It was later deprecated by LayerNorm [25] and RMSNorm, which normalize across the activation dimension.

Scaling Laws: This is probably my pick for the single most important physics contribution to deep learning. Diffusion is obviously a strong contender, but scaling laws actually introduced a practical and much-needed level of formalism to deep learning. Of course, learning theory can be formal, but not in the same way physics is formal. One is platonic, while the other is practical. I will not reproduce the plethora of scaling results here, but it suffices to say that one can accurately study and predict model performance along a variety of metrics as functions of different scaling quantities, most importantly, *compute*. This study can be made so precise as to produce closed-form expressions describing model performance across several orders of magnitude in scaling quantities.

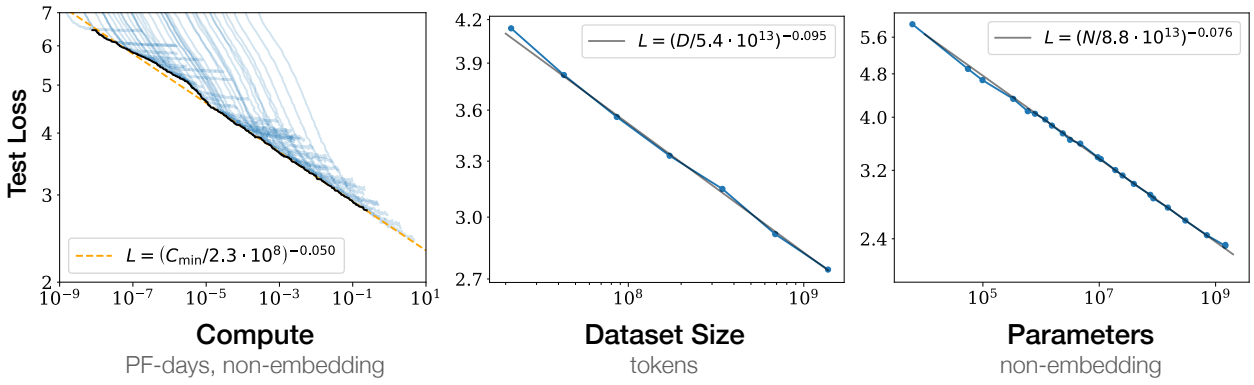


Figure 1.1: (Figure from Kaplan et al. [26]) Language modeling performance improves smoothly as we increase the model size, dataset size, and amount of compute used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

Kaplan et al. [26] first introduced the power-law scaling expression of large language models and effectively started the scaling laws subfield in large-scale deep learning. The expressions are shown in Figure 1.1. The Chinchilla laws [27] further refined the exponents in the expression and showed that data and model size are equally important to train compute-optimal large language models. More sophisticated studies concerning other aspects of large-scale pre-training, such as specific data mixtures (not all data sources are created equal) and submodule scaling (sparse and composite models like mixtures of experts) are the reason the "scaling labs" [1] found so much success in training models at the frontiers of feasible compute.

[1] Primarily OpenAI and Anthropic early on then followed by Google and Meta.

Stable Parameterizations: In my opinion, the real secret sauce in scaling to trillion parameter models is parameterization. Parameterization is a prescription for scaling some critical quantities with respect to scaling dimensions or, more crudely, for changing the learning rate and initialization of the random weights of a neural network as you change its size. If a parametrization is well understood, the prescription allows safe scaling along the dimensions of interest. Conversely, assigning incorrect values to any quantity, like the weight initialization scale or per-layer learning rate, can lead to a mismatch that is often hard to detect in small-scale experiments. This mismatch makes it such that the evidently true and provable statement "bigger models always outperform smaller ones" becomes challenging to reproduce in practice.

Relying solely on the extrapolation of empirical results is a perilous endeavor in the era of huge language models with training runs that cost hundreds of millions of dollars. Every aspect of the training procedure needs to be thoroughly de-risked. Parametrizations can allow for hyper-parameter transfer by training many small models to find an optimal recipe that can readily and confidently be used in training a much larger model. These approaches generally take a simple form

$$\begin{matrix}\text{parameterized}\\ \text{quantity}\end{matrix} = \begin{matrix}\text{empirical}\\ \text{constant}\end{matrix} \cdot \left(\begin{matrix}\text{scaling}\\ \text{dimension}\end{matrix}\right)^{\begin{matrix}\text{theoretical}\\ \text{exponent}\end{matrix}}$$

The empirical constant is determined with experiments at small scale. Extensive parameter search at these scales finds the optimal setting, which can then be transferred using the above formula. The first approach to hyper-parameter transfer was introduced in the Maximal Update Parameterization (μP) [28], which is defined in terms of a few desiderata. First, activations should be constant as the width is scaled *i.e.*, $||h||_2 = \Theta(\sqrt{n})$ where n is the width dimension (note this result is obtained because h is n-dimensional with each $\Theta(1)$ coordinates). Second, each parameter update must induce a constant change in the activation *i.e.*, $\Delta||h||_2 = \Theta(\sqrt{n})$. From this desiderata one can workout the following scaling rules for each layer l

$$\sigma_l = \Theta\big(\frac{1}{\sqrt{n_{l-1}}}\min\{1, \frac{\sqrt{n_l}}{\sqrt{n_{l-1}}}\}\big); \qquad \eta_l = \Theta(\frac{\sqrt{n_l}}{\sqrt{n_{l-1}}}) \tag{1.10}$$

Chapter 2

Representation Learning for Physics: Improving Online Data Selection at LHCb

Imagine a torrent of data so vast it defies conventional processing—over 100 terabytes per second, more than a zettabyte per year.[1] This is the reality faced by the LHCb experiment at CERN's Large Hadron Collider. In this environment, identifying the rare, interesting events that might hint at new physics is akin to finding not just a needle in a haystack, but a specific atom in a planet-sized heap of straw.

The challenge before us is twofold. First, our algorithms need to be capable of making decisions to match a frequency of 40 MHz (or 25 ns bunch crossing rate) on what data to keep and what to discard, a process known in the field as "triggering." Second, Our algorithms must not only be fast and accurate but also robust to the noise in experimental data and interpretable to the scientists who rely on them. We must ensure that these algorithms respect the physical principles and domain knowledge that underpin the very experiments they serve.

In this chapter, we will explore a neural network architecture that rises to meet these challenges. We call it the Monotonic Lipschitz Network, a name that encapsulates its two key innovations. The Lipschitz constraint ensures robustness, guaranteeing that small perturbations in input features—whether from experimental noise or imperfections in our simulations—lead to correspondingly small changes in output. The enforced monotonicity allows us to bake in physical intuition, ensuring that our model's behavior aligns with our understanding of the underlying physics, even in regions where our training data might be sparse or nonexistent.

The chapter [2] is structured as follows:

[1] These are raw data numbers which can include a lot of "zeros" that can be cleaned quickly.

[2] This chapter is based on research originally presented in Refs. [29], [30]. The work was conducted in collaboration with Niklas Nolte and Mike Williams.

1. We begin by formalizing the concept of Lipschitz-constrained networks and present our method for enforcing this constraint during training. 2. We then introduce the monotonic residual connection, which allows for selective monotonicity in input features. 3. We demonstrate the effectiveness of our approach on toy problems, showcasing its robustness and monotonicity properties. 4. Finally, we present a real-world application: classifying heavy-flavor particle decays in the LHCb experiment at CERN.[3]

As discussed in Chapter 1, deep learning is extremely powerful, and most problems we tackle using it are underspecified in some capacity. This is precisely why we need inductive biases, and this is precisely why the monotonicity property is both beneficial and crucial in this setting. Monotonic Lipschitz Networks have been adopted at many stages of the trigger-selection algorithm for LHCb in Run 3 of the LHC.

2.1 Introduction

The sensor arrays of the LHC experiments produce more than 100 TB/s of data, more than a zettabyte per year. After drastic data reduction performed by custom-built readout electronics, the annual data volumes are still $O(100)$ exabytes, which cannot be stored indefinitely. Therefore, each experiment processes the data in real time and decides whether each proton-proton collision event should remain persistent or be permanently discarded, referred to as *triggering* in particle physics. Trigger classification algorithms must be designed to minimize the impact of effects like experimental instabilities that occur during data taking—and deficiencies in simulated training samples. (If we knew all of the physics required to produce perfect training samples, there would be no point in performing the experiment.) The need for increasingly complex discriminators for the LHCb trigger system [31]–[33] calls for the use of expressive models which are both robust and interpretable. Here we present an architecture based on a novel weight normalization technique that achieves both of these requirements.

Robustness A natural way of ensuring the robustness of a model is to constrain the Lipschitz constant of the function it represents, defined such that for every pair of points on the graph of the function, the absolute value of the slope of the line connecting them is not greater than the Lipschitz constant. To this end, we developed a new architecture whose Lipschitz constant is constrained by design using a novel layer-wise normalization which allows the architecture to be more expressive than the current state-of-the-art with more stable and faster training.

Interpretability An important inductive bias in particle detection at the LHC is the idea that particular collision events are more *interesting* if they are outliers, *e.g.*, possible evidence of a particle produced with a longer-than-expected (given known physics) lifetime would

[3]Other standard datasets for applications where monotonicity is a desired property are shown in Appendix A.

definitely warrant further detailed study. The problem is that outliers are often caused by experimental artifacts or imperfections, which are included and labeled as background in training; whereas the set of all possible *interesting* outliers is not possible to construct *a priori*, thus not included in the training process. This problem is immediately solved if *outliers are better* is implemented directly using an expressive monotonic architecture. Some work was done in this regard [34]–[36] but most implementations are either not expressive enough or provide no guarantees. We present Monotonic Lipschitz Networks which overcome both of these problems by building an architecture that is monotonic in any subset of the inputs by design, while keeping the constraints minimal such that it still offers significantly better expressiveness compared to current methods.

2.2 Monotonic Lipschitz Networks

The goal is to develop a neural network architecture representing a scalar-valued function

$$f(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R} \tag{2.1}$$

that is provably monotonic in any subset of inputs and whose gradient (with respect to its inputs) has a constrained magnitude in any particular direction. In an experimental setting, this latter property is a measure of robustness to small changes in experimental conditions or to small deficiencies in the training samples.

Constraints with respect to a particular L_p metric will be denoted as Lip^p. We start with a model $g(\boldsymbol{x})$ that is Lip^1 with Lipschitz constant λ if $\forall\, \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ (we show below how to train such a model)

$$|g(\boldsymbol{x}) - g(\boldsymbol{y})| \leq \lambda \|\boldsymbol{x} - \boldsymbol{y}\|_1 \,. \tag{2.2}$$

The choice of 1-norm is crucial because it allows a well defined maximum directional derivative for each input regardless of the gradient direction. This has the convenient side effect that we can tune the robustness requirement for each input individually. Note that rescaling the inputs x_i allows for λ directional dependence.

2.2.1 Enforcing Monotonicity

Assuming we have trained a model that satisfies Equation (2.2), we can make an architecture with built-in monotonicity by adding a term that is linear (or has gradient λ) in each direction in which we want to be monotonic:

$$f(\boldsymbol{x}) = g(\boldsymbol{x}) + \lambda \sum_{i \in I} x_i, \tag{2.3}$$

where I denotes the set of indices of the input features for which we would like to be monotonic. This residual connection enforces monotonicity:

$$\frac{\partial f}{\partial x_i} = \frac{\partial g}{\partial x_i} + \lambda \geq 0 \quad \forall\, i \in I \,. \tag{2.4}$$

Note that the construction presented here only works with Lip^1 constraints as $\text{Lip}^{p\neq 1}$ functions introduce dependencies between the partial derivatives. In addition, we stress that monotonicity is defined via partial derivatives. The value of f is guaranteed to increase when x_i is increased while keeping all $x_{\neq i}$ constant. It is therefore advisable to look out for ill defined edge cases. For instance, let $x_2 \equiv -x_1$ in the training data and define $I = \{1, 2\}$. This is incompatible with the architecture and produces unwanted results unless $\lambda = 0$ for both x_1 and x_2 (otherwise the problem is ill posed).

To the best of our knowledge, the only use of residual connections in the literature when trying to learn monotonic functions is in the context of invertible ResNets [37]. Instead, the state-of-the-art approach for learning monotonic functions involves penalizing negative gradients in the loss, then certifying the final model is monotonic, rather than enforcing it in the architecture (*e.g.* in [34]).

2.2.2 Enforcing Lipschitz Constraints

Ideally, the construction $g(\boldsymbol{x})$ should be a universal approximator of Lip^1 functions. Here, we discuss possible architectures for this task.

Lip^1 constrained models Fully connected networks can be Lipschitz bounded by constraining the matrix norm of all weights [38], [39]. Given the fully connected network with activation σ

$$g(\boldsymbol{x}) = W^m \sigma(W^{m-1}\sigma(...\sigma(W^1\boldsymbol{x} + b^1)...) + b^{m-1}) + b^m, \tag{2.5}$$

where W^m is the weight matrix of layer m, $g(\boldsymbol{x})$ satisfies Equation (2.2) if

$$\prod_{i=0}^{m} \|W^i\|_1 \leq \lambda \tag{2.6}$$

and σ has a Lipschitz constant less than or equal to 1. There are multiple ways to enforce Equation (2.6). Two possibilities that involve scaling by the operator norm of the weight matrix [38] are

$$W^i \to W'^i = \lambda^{1/m} \frac{W^i}{\max(1, \|W^i\|_1)} \qquad \text{or} \qquad W^i \to W'^i = \frac{W^i}{\max(1, \lambda^{-1/m} \cdot \|W^i\|_1)}. \tag{2.7}$$

In our studies thus far, the latter variant seems to train slightly better. However, in some cases it might be useful to use the former to avoid the scale imbalance between the neural network's output and the residual connection used to induce monotonicity.

In order to satisfy Equation (2.6), it is not necessary to divide the entire matrix by its 1-norm. It is sufficient to ensure that the absolute sum over each column is constrained:

$$W^i \to W'^i = W^i \text{diag}\left(\frac{1}{\max\left(1, \lambda^{-1/m}\sum_j |W^i_{jk}|\right)}\right). \tag{2.8}$$

This normalization scheme tends to give even better training results in practice. While Equation (2.8) is not suitable as a general-purpose scheme, *e.g.* it would not work in convolutional networks, its performance during the training phase of this study motivates further examination in future work.

The constraints in Equations (2.7) and (2.8) can be applied in different ways. For example, one could normalize the weights directly before each call such that the induced gradients are propagated through the network like in [39]. While one could come up with toy examples for which propagating the gradients in this way hurts training, it appears that this approach is what usually is implemented for spectral norm [39] in PyTorch and TensorFlow. Alternatively, the constraint could be applied by projecting any infeasible parameter values back into the set of feasible matrices after each gradient update as in Algorithm 2 of [38]. Algorithm 1 summarizes our approach.

Algorithm 1 Training with enforced Lipschitz constraint using weight-norming

Require: $\{\mathbb{D}_i\}_{i=1}^n$, a collection of n training batches.
Require: w, the non-normalized weight parameter at some layer. ▷ These are the optimized leaf parameters
Require: **Norm**, the function used to normalize the weights, *e.g.* as given by Equation (2.8).
Require: **Cost**, the loss computed using a neural network with weight parameters $\hat{w}$ on a given batch.

$\hat{w} \leftarrow \mathbf{Norm}(w)$ ▷ This is the weight used in the neural network matrix multiplication
while not converged **do**
  for i from 1 to n **do**
    $L \leftarrow \mathbf{Cost}(\mathbb{D}_i, \hat{w})$
    $w \leftarrow w - \nabla_w \hat{w} \cdot \nabla_{\hat{w}} L$
    $\hat{w} \leftarrow \mathbf{Norm}(w)$
  end for
end while ▷ At inference time, only $\hat{w}$ is used.

Preserving expressive power Some Lipschitz network architectures (*e.g.* [39]) tend to overconstrain the model in the sense that these architectures cannot fit all functions λ-Lip[1] due to *gradient attenuation*. For many problems this is a rather theoretical issue. However, it becomes a practical problem for the monotonic architecture since it often works on the edges of its constraints, for instance when partial derivatives close to zero are required. The authors of [40] showed that ReLU networks are unable to fit the function $f(x) = |x|$ if the layers are norm-constrained with $\lambda = 1$. The reason lies in the fact that ReLU, and most other commonly used activations, do not have unit gradient with respect to the inputs over their entire domain.

While element-wise activations like ReLU cannot have unit gradient over the whole domain without being exactly linear, the authors of [41] explore activations that introduce nonlinearities by reordering elements of the input vector. They propose the following activation

function:

$$\sigma = \textbf{GroupSort}, \tag{2.9}$$

which sorts its inputs in chunks (groups) of a fixed size. This operation has gradient 1 with respect to every input and gives architectures constrained with Equation (2.6) increased expressive power. In addition, we have found that using this activation function also results in achieving sufficient expressiveness with a small number of weights, making the networks ideal for use in resource-constrained applications.

2.3 Example Applications to Simple Models

Before applying our new architecture to real-time data-processing at the LHC, we first demonstrate that it behaves as expected on some simple toy problems.

2.3.1 Robustness to Outliers

We will demonstrate the robustness that arises from the Lipschitz constraint by making a simple toy regression model to fit to data sampled from a 1-dimensional function with one particularly noisy data point. The underlying model that we sample from here has the form

$$y = \sin(x) + \epsilon(x), \tag{2.10}$$

where $\epsilon(x)$ is Gaussian noise with unit variance for one data point and 0.01 otherwise. While this toy problem will explicitly show that the Lipschitz network is more robust against outliers than an unconstrained network due to its bounded gradient, it also serves as a proxy for any scenario with deficiencies in the training data. *N.b.*, due to its bounded gradient a Lipschitz network is also more robust against adversarial attacks and data corruption than an unconstrained model.

Figure 2.1 shows that the unconstrained model overfits the data as expected, whereas applying our approach from Section 2.2 does not. The Lipschitz model effectively ignores the outlier, since there is no way to accommodate that data point while respecting its built-in gradient bound. In addition, we see that the Lipschitz constraint enforces much smoother functions over the full range—the degree of this smoothness determined by us via the chosen Lipschitz constant.

2.3.2 Monotonic Dependence

To demonstrate monotonicity, we will make a simple toy regression model to fit to data sampled from the following 1-dimensional function:

$$f(x) = \log(x) + \epsilon(x), \tag{2.11}$$

where ϵ is a Gaussian noise term whose variance is linearly increasing in x. In this toy model, we will assume that our prior knowledge tells us that the function we are trying to fit must be monotonic, despite the non-monotonic behavior observed due to the noise. This situation

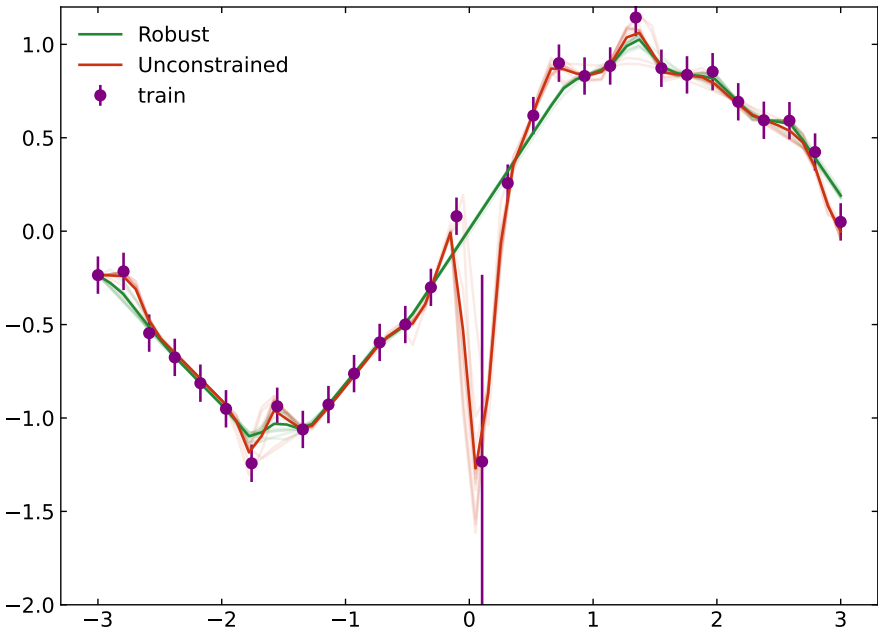


Figure 2.1: (From Kitouni et al. [30]) robust and unconstrained models using a realization (purple data points) of the toy model in Equation (2.10). The error barscanvas are the known standard deviation of the noise term $\epsilon(x)$ in Equation (2.10). Each model is trained using 10 random initialization seeds. The dark lines are averages over the seeds, which are each shown as light lines. The unconstrained models exhibit overfitting of the noisy outlier, whereas the Lipschitz networks are robust. In addition, the Lipschitz constraint produces much smoother models as expected. *N.b.*, here we set the Lipschitz constant to be $\lambda = 1$, whereas the slope of the true model is $\cos x$. This allows for more variation in the fit model than the true model. In this exercise we assumed that all we know is that the slope is bounded by unity. If we did have more precise *a priori* information about the slope, we could easily employ this by rescaling x as discussed in Section 2.2.

is ubiquitous in real-world applications of AI/ML, but is especially prevalent in the sciences (see, *e.g.*, Section 2.4).

First, we train standard (unconstrained) neural networks on several different samples drawn from Equation (2.11). Here, we also consider two generic situations where the training data are missing: one that requires extrapolation beyond the region covered by the training data, and another that requires interpolation between two occupied regions. Figure 2.2 shows that the unconstrained models overfit the data as expected, resulting in non-monotonic behavior. Furthermore, when extrapolating or interpolating into regions where training data were absent, the unconstrained models exhibit highly undesirable and in some cases unpredictable behavior. (This problem is exacerbated in higher dimensions and sparser data.) In the case of extrapolation, the behavior of the unconstrained model is largely driven by the noise in the last one or two data points. The interpolation scenario is less predictable.

While the overfitting observed here could be reduced by employing some form of strong regularization, such an approach would not (in general) lead to monotonic behavior, nor would it formally bound the gradient. Applying our approach from Section 2.2 does both.

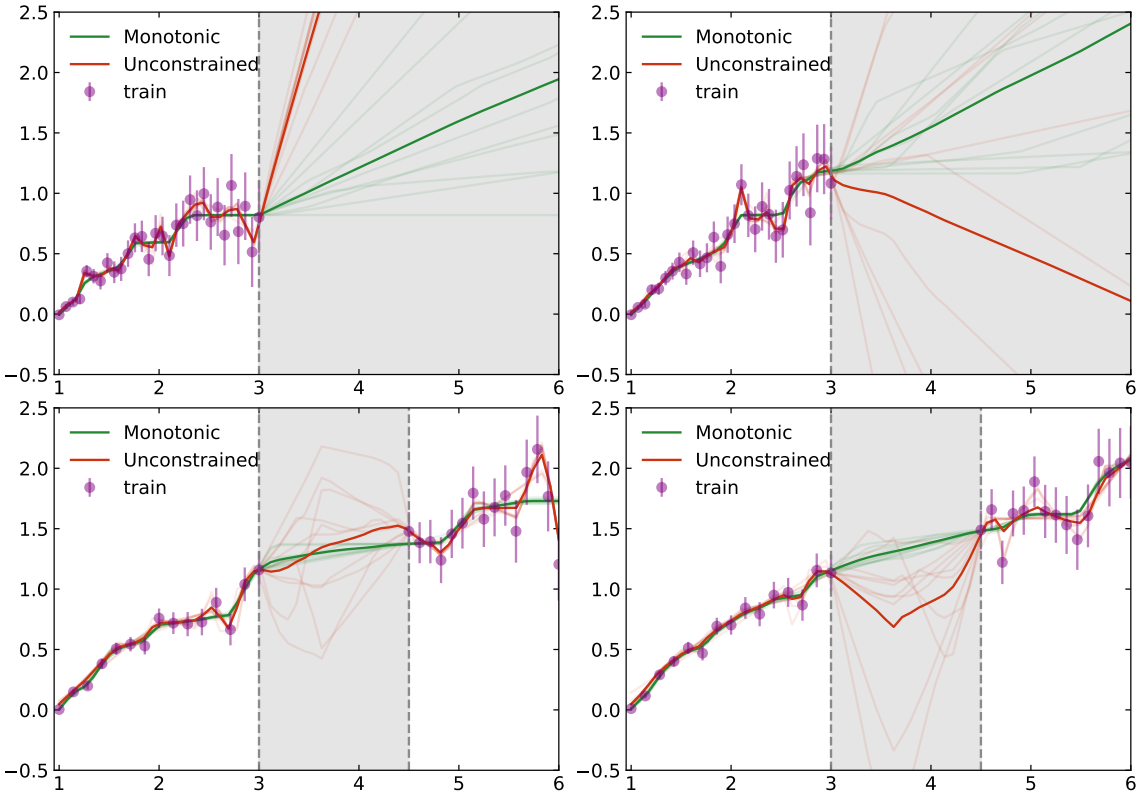


Figure 2.2: (From Kitouni et al. [30]) Training monotonic and unconstrained models using four realizations (purple data points) of the toy model in Equation (2.11). The error bars are the known standard deviation of the noise term ϵ from Equation (2.11). The shaded regions represent the (top) extrapolation or (bottom) interpolation regions of interest, where training data are absent. Each panel presents a different realization of the Gaussian noise. Each model is trained using 10 random initialization seeds. The dark lines are averages over the seeds, which are each shown as light lines. The unconstrained models exhibit overfitting of the noise and non-monotonic behavior, and when extrapolating or interpolating into regions where training data were absent, these models exhibit highly undesirable and unpredictable behavior. Conversely, the monotonic Lipschitz models always produce a monotonic function, even in scenarios where the noise is strongly suggestive of non-monotonic behavior. In addition, the Lipschitz constraint produces much smoother models as expected. *N.b.*, here we set the Lipschitz constant to be $\lambda = 1$, whereas the slope of the true model is $1/x$. This allows for more variation in the fit model than the true model. In this exercise we assumed that all we know is that the slope is bounded by unity. If we did have more precise *a priori* information about the slope, we could easily employ this by rescaling x as discussed in Section 2.2.

Figure 2.2 demonstrates that our method always produces a monotonic function, even in the extrapolation scenario where the slope of the noise terms in the last few data points is strongly suggestive of non-monotonic behavior. In addition, the Lipschitz constraint produces much smoother models than in the unconstrained case. Therefore, we conclude that the monotonicity and Lipschitz constraints do act as strong regularization against fitting random non-monotonic noise as expected.

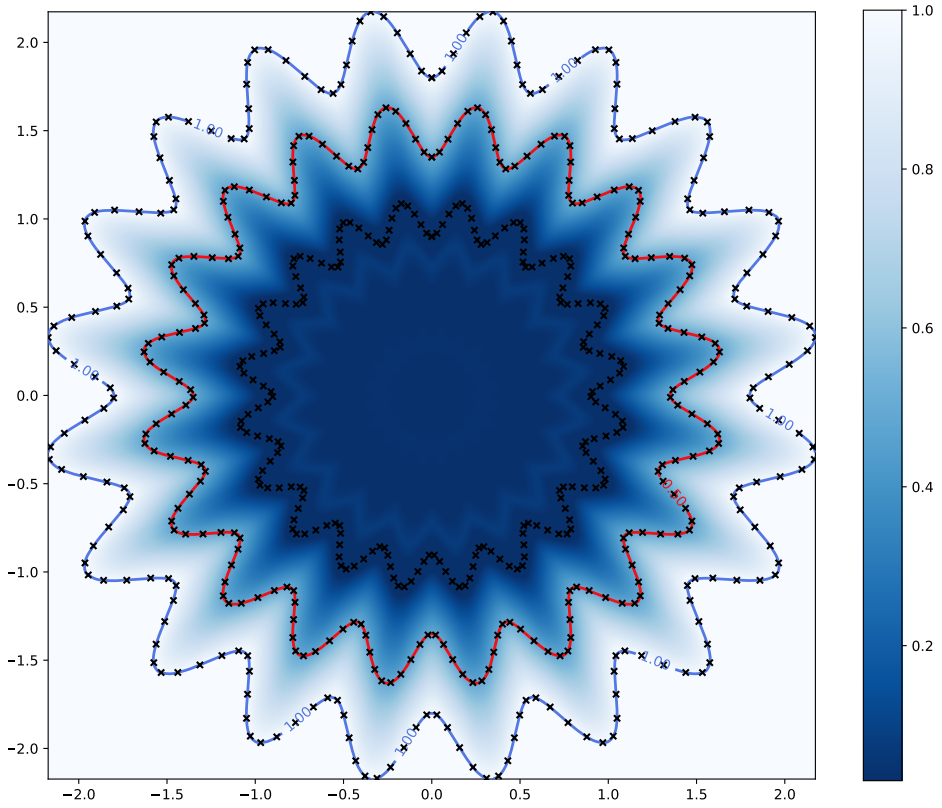


Figure 2.3: (From Kitouni et al. [30]) Regression example to emulate a complex decision boundary in two dimensions. The training data points are shown in black (the inner radius is labeled 0, the middle is labeled 0.5, and the outer radius is labeled 1), while the output of the network is shown in color. The contour lines of the network output are shown in blue and red for the values of 1.0 and 0.5, respectively, which properly trace out the curves populated by the outer and middle sets of data points.

2.3.3 Expressiveness

GroupSort weight-constrained neural networks can describe arbitrarily complex decision boundaries in classification problems provided the proper objective function is used in training (the usual cross entropy and MSE losses may be sub-optimal for Lipschitz models in some scenarios [42], see Section 2.5). Here we will directly regress on a synthetic boundary to emulate a classification problem. The boundary is the perimeter of circle with oscillating radius and is given by

$$\partial = \{r + \alpha[\cos\omega\theta,\, \sin\omega\theta]) \mid \theta \in [0, 2\pi]\}\,, \tag{2.12}$$

where r and α are chosen to be 1.5 and 0.18, respectively. Figure 2.3 shows an example where this complicated decision boundary is learned by a Lipschitz network (as defined in Section 2.2) trained on the boundary while achieving zero loss, demonstrating the expressiveness that is possible to obtain in these models.

2.4 Example Application: The LHCb inclusive heavy-flavor Run 3 trigger

The architecture presented in Section 2.2 has been developed with a specific purpose in mind: The classification of the decays of heavy-flavor particles produced at the Large Hadron Collider, which are bound states that contain a beauty or charm quark that live long enough to travel an observable distance $\mathcal{O}(1\,\text{cm})$ before decaying. The dataset used here is built from simulated proton-proton (pp) collisions in the LHCb detector. Charged particles that survive long enough to traverse the entire detector before decaying are reconstructed and combined pairwise into decay-vertex (DV) candidates.

The task concerns discriminating between DV candidates corresponding to the decays of heavy-flavor particles versus all other sources of DVs. The signatures of a heavy-flavor DV are substantial separation from the pp collision point, due to the relatively long heavy-flavor particle lifetimes, and sizable transverse momenta, p_T, of the component particles, due to the large heavy-flavor particle masses. There are three main sources of background DVs. The first involves DVs formed from particles that originated directly from the pp collision, but where the location of the DV is measured to have non-zero displacement due to resolution effects. These DVs will typically have small displacements and small p_T. The second source of background DVs arises due to particles produced in the pp collision interacting with the LHCb detector material, creating new particles at a point in space far from the pp collision point. Such DVs will have even larger displacement than the signal, but again have smaller p_T. The third source involves at least one *fake* particle, *i.e.* a particle inferred from detector information that did not actually exist in the event. Since the simplest path through the detector (a straight line) corresponds to the highest possible momentum, DVs involving fake particles can have large p_T.

In the first decision-making stage of the LHCb trigger, a pre-selection is applied to reject most background DVs, followed by a classifier based on the following four DV features: $\sum p_\text{T}$, the scalar sum of the p_T of the two particles that formed the DV; $\min[\chi^2_\text{IP}]$, the smaller of the two increases observed when attempting to instead include each component particle into the pp-collision vertex fit, which is large when the DV is far from the pp collision point; the quality of the DV vertex fit; and the spatial distance between the DV and pp-collision locations, relative to their resolutions. *N.b.*, the threshold required on the classifier response when run in real time during data taking is fixed by the maximum output bandwidth allowed from the first trigger stage.

Unfortunately, extremely large values of both displacement and momentum are more common for backgrounds than for heavy-flavor signals. For the former, this is easily visualized by considering a simplified problem using only the two most-powerful inputs, $\sum p_\text{T}$ and χ^2_IP. Figure 2.4 (left) shows that an unconstrained neural network learns to reject DVs with increasing larger displacements, corresponding to the lower right corner in the figure. Figure 2.5 (left) shows that this leads to a dependence of the signal efficiency on the lifetime of the decaying heavy-flavor particle. Larger lifetimes are disfavored since few heavy-flavor particles live more than $\mathcal{O}(10\,\text{ps})$. While rejecting DVs with the largest displacements does maximize the integrated signal efficiency in the training sample, this is undesirable because in many cases studying the longest-lived heavy-flavor particles is of more interest than simply

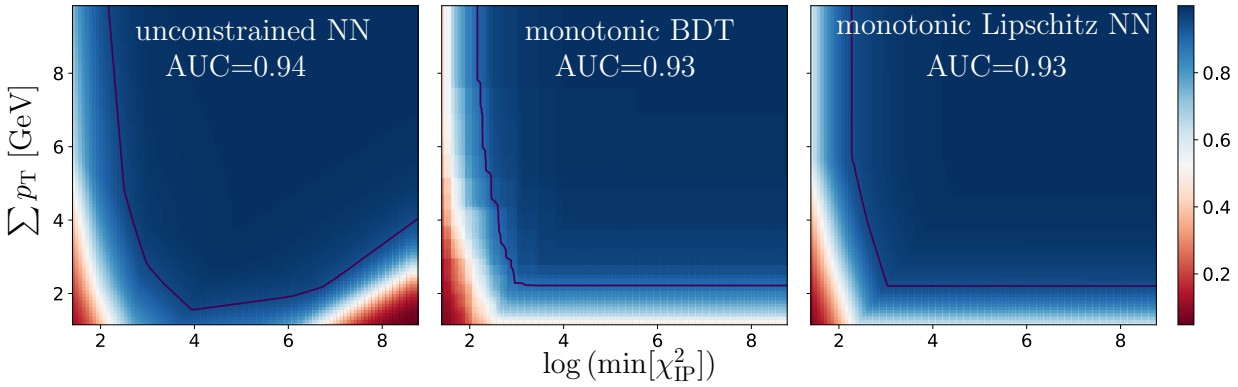


Figure 2.4: (From Kitouni et al. [30]) Simplified version of the LHCb inclusive heavy-flavor trigger problem using only 2 inputs, which permits displaying the response everywhere in the feature space; shown here as a heat map with more signal-like (background-like) regions colored blue (red). The dark solid line shows the decision boundary predicted to give the required output bandwidth in Run 3.

collecting the largest decay sample integrated over lifetime (see, *e.g.*, [43]). Furthermore, many proposed explanations of dark matter and other types of new physics predict the existence of new particles with similar properties to heavy-flavor particles, but with longer lifetimes [44], [45]. This classifier would reject these particles because it is unaware of our inductive bias that highly displaced DVs are worth selecting in the trigger and studying in more detail later.

Since the LHCb community is generally interested in studying highly displaced DVs for many physics reasons, we want to ensure that a larger displacement corresponds to a more signal-like response. The same goes for DVs with higher $\sum p_T$. Enforcing a monotonic response in both features is thus a desirable property, especially because it also ensures the desired behaviour for data points that are outside the boundaries of the training data. Multiple methods to enforce monotonic behavior in BDTs already exist [46], and Figure 2.4 (middle) and Figure 2.5 (middle) show that this approach works here. However, the jagged decision boundary can cause problems, *e.g.*, when measuring the heavy-flavor p_T spectrum. Specifically, the jagged BDT decision boundary can lead to sharp changes in the selection efficiency. If there is not perfect alignment of where these changes occur with where the interval boundaries of the spectrum are defined, then correcting for the efficiency can be challenging. Figure 2.4 (right) shows that our novel approach, outlined in Section 2.2, successfully produces a smooth and monotonic response, and Figure 2.5 (right) shows that this provides the monotonic lifetime dependence we wanted in the efficiency.

Not only does our architecture guarantee a monotonic response in whatever features the analyst wants, it is guaranteed to be robust with respect to small changes to the inputs as governed by the constrained Lipschitz constant. Because calibration and resolution effects play a role in obtaining the features during detector operation, robustness is a necessary requirement for any classification performed online. Downstream analyses of these data depend on their stability. Figure 2.6 shows that the cost in terms of signal efficiency loss of enforcing monotonicity and robustness is small, even under the unrealistic assumption that

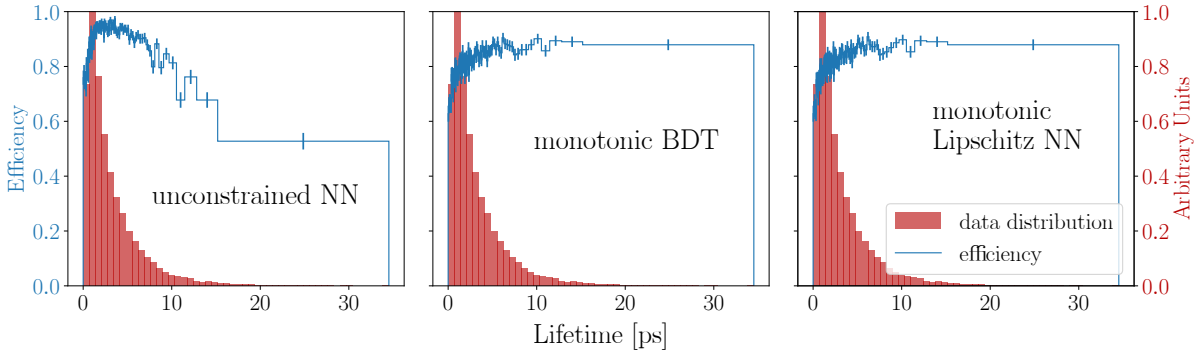


Figure 2.5: Efficiency of each model shown in Figure 2.4 at the expected Run 3 working point versus the proper lifetime of the decaying heavy-flavor particle selected. The monotonic models produce a nearly uniform efficiency above a few ps at the expense of a few percent lifetime-integrated efficiency. Such a tradeoff is desirable because, in many cases, studying the longest-lived heavy-flavor particles is of more interest than collecting the largest decay sample integrated over lifetime. In addition, many proposed hypothetical particles have similar properties to heavy-flavor particles, but with longer lifetimes.

the training data were, in fact, perfect. Therefore, the actual cost is likely negligible, while the benefits of the guarantees provided is hard to quantify but immediately obvious to the LHCb collaboration. Our algorithm runs in the LHCb trigger software stack and has been chosen to replace Refs. [47], [48] as the primary trigger-selection algorithm used by LHCb in Run 3. Due to its guaranteed robustness—and excellent expressiveness even for small networks—this architecture is being explored for other uses[4] within the LHCb trigger system for Run 3,since robustness and monotonicity are ubiquitous inductive biases in experimental particle physics.

Experiment details The default LHCb model shown here is a 4-input, 3-layer (width 20) network with GroupSort activation (here, all outputs are sorted), $\lambda = 2$, constrained using Equation (2.8). Inference times in the fully GPU-based LHCb trigger application [31] are 4 times faster than the Run 3 trigger BDT that was the baseline algorithm before ours was chosen to replace it (the BDT baseline was based on the model used during data taking in Run 2 [47], [48]). We performed $\mathcal{O}(1000)$ runs with different seeds but the differences were negligible, at the level of $\mathcal{O}(0.1\%)$. For the unconstrained network, we use the same architecture but without the linear term and without the weight constraints during training. The depth and width are the same as used for the monotonic Lipschitz network. The BDT is a LightGBM [49] gradient boosted classifier with 1000 base trees and a maximum 25 leaves per tree. Monotonicity is enforced there via the built-in `monotone_constraints` keyword. Code for the monotonic network implementation of the architecture developed here can be found at https://github.com/niklasnolte/MonotonicNetworks.

[4] See discussion.

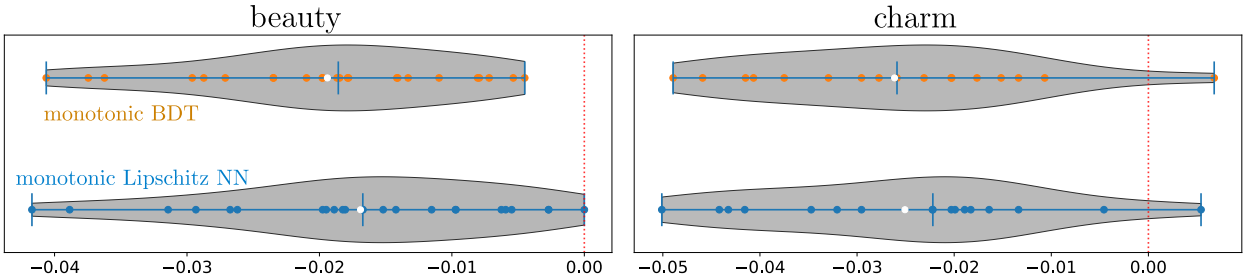


Figure 2.6: (From Kitouni et al. [30]) Difference in signal efficiency (true positive rate) relative to the unconstrained NN at the expected Run 3 working point for the (left) 24 beauty and (right) 17 charm decays currently being used to benchmark this trigger. Each colored data point shows the change in efficiency for a given decay, while the shaded bands represent the local density of points. The white points show the mean values for each set of points, and the vertical blue bars represent the extreme values and the median.

2.5 Limitations and Potential Improvements

In many scenarios, Lipschitz-constrained architectures are considered inferior to unconstrained architectures because of their inability to offer competitive performance on standard benchmarks. This low performance is partly due to the fact that standard losses (such as cross-entropy) are not an adequate proxy of the metric of interest (accuracy) for the Lipschitz-constrained models. At a fundamental level, for any maximally accurate unconstrained classifier $f(x)$ with Lipschitz constant λ, there exists a Lipschitz 1 classifier that replicates the former's decision boundary, namely, $f(x)/\lambda$. In the following, we will demonstrate a basic toy setting in which a maximally accurate Lipschitz classifier exists but cannot be obtained using standard losses.

To understand the effect of the choice of objective function, we train a Lipschitz-constrained model to separate the *two-moons* dataset as shown in Figure 2.7. This example is special in that the two samples do not overlap and can be completely separated by a Lipschitz-bounded function; however, that function cannot return the true label values for any data points due to the Lipschitz bound. Therefore, a loss function that penalizes any difference of the model output to the true label now faces a misalignment of the optimization target and the actual goal: While the classification goal is to have high accuracy, *i.e.* correct output sign, the optimization target is to minimize deviations of the output from the true label. This misalignment becomes irrelevant for a function with unbounded Lipschitz constant. We will show below that for examples such as this there is an important dependence on the objective used and its hyperparameters.

First, we note that losses with exponential tails (in the sense that they require large weights to reach zero) are in general not suitable for maximizing accuracy. In practice, this can be remedied in cross-entropy by increasing the temperature. Note that cross-entropy with temperature τ is defined as

$$\mathcal{L}^{\mathrm{BCE}}_{\tau}(y, \hat{y}) = \mathcal{L}^{\mathrm{BCE}}(y, \tau\hat{y}), \tag{2.13}$$

where $\mathcal{L}^{\mathrm{BCE}}(y, \hat{y})$ is the usual binary cross-entropy loss on targets y and predictions $\hat{y}$.

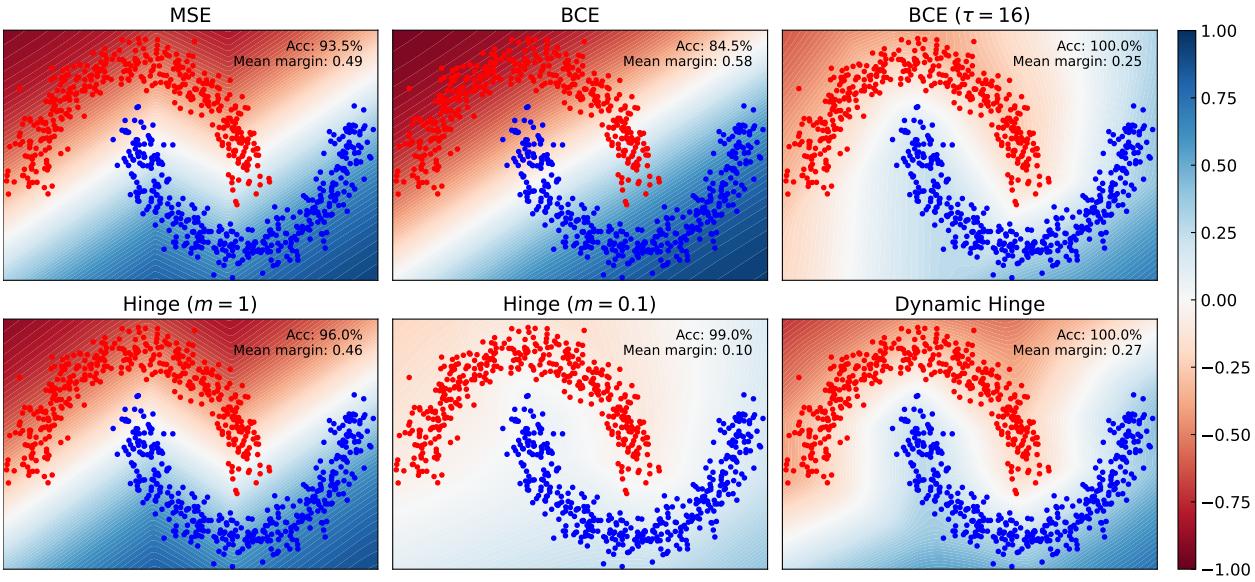


Figure 2.7: (From Kitouni et al. [30]) A Lipschitz network trained to classify the Two-Moons dataset using different objects. Ordered from left to right and from top to bottom: Mean Squared Error, Binary Cross Entropy, Binary Cross Entropy with high temperature ($\tau = 16$), Hinge loss with margin 1, Hinge loss with margin 0.1, and Hinge with dynamic margin. The network is evaluated on a uniform grid and its output is shown as a heatmap. The average absolute prediction (mean margin) on the validation set is also shown.

Following PyTorch conventions, $\hat{y}$ are logits which will be normalized before computing the negative log-likelihood.

An accurate classification boundary comes at the expense of reduced margins when classes have small separation. A maximally robust accurate classifier will, however, have optimal margins if trained using the appropriate objective. In the case of separable data (*i.e.* when classes have disjoint support), the maximally robust accurate Lipschitz classifier is the signed-distance function (SDF) [42] defined, in the binary case as

$$\mathrm{SDF}(x) = \mathrm{sign}\left[d(C^{+1}, x) - d(C^{-1}, x)\right] \cdot d(B, x), \tag{2.14}$$

where C^{+1} and C^{-1} are the sets of points for which x has label $+1$ and -1, respectively, and B is the boundary between classes defined as $B \equiv \left\{x \,\middle|\, d(C^{+1}, x) = d(C^{-1}, x)\right\}$. For a closed set S, the distance to x is defined as $d(S, x) = \min_{y \in S} d(y, x)$.

A naive objective minimized by the SDF is the hinge loss with margin given by $d(B, x)$. Because we do not have access to the true decision boundary *a priori*, as a proxy, we use the following objective:

$$\mathcal{L}^{\mathrm{DynamicHinge}}(y, \hat{y}, x) = \mathcal{L}^{\mathrm{Hinge}}_{\delta(x|y)}(y, \hat{y}), \tag{2.15}$$

where $\delta(x|y) = \frac{d(C^{-y}, x)}{2}$ and $\mathcal{L}^{\mathrm{Hinge}}_m$, with margin m, is defined as

$$\mathcal{L}^{\mathrm{Hinge}}_m(y, \hat{y}) = \max\left(0, m - y\hat{y}\right). \tag{2.16}$$

While this objective produces the highest margins for an accurate classifier, as depicted in Figure 2.7, it may encounter scalability issues when applied to higher-dimensional problems

due to the unavoidable sparsity of the training data. There are many possible alternative approaches that could resolve this issue, though this remains an open problem. For lower-dimensional problems with overlapping datasets—as studied in the various examples above and the most common scenario in scientific applications—this non-optimal loss issue does not appear to be relevant.

Another factor that restricts the perceived expressiveness of Lipschitz architectures is the lack of access to standard techniques that improve convergence like in unconstrained networks. For example, normalization cannot be directly used with Lipschitz architectures. If the variance is too small, it may exceed the Lipschitz bound, and if it is too large, it can reduce the effective Lipschitz constant substantially.

2.6 Summary & Discussion

The Lipschitz constant of the map between the input and output space represented by a neural network is a natural metric for assessing the robustness of the model. We developed a new method to constrain the Lipschitz constant of dense deep learning models that can also be generalized to other architectures. Our method relies on a simple weight normalization scheme during training that ensures the Lipschitz constant of every layer is below an upper limit specified by the analyst. A simple monotonic residual connection can then be used to make the model monotonic in any subset of its inputs, which is useful in scenarios where domain knowledge dictates such dependence.

Our implementation of Lipschitz constrained networks is minimally constraining compared to other weight-normed models. This allows the underlying architecture to be more expressive and easier to train while maintaining explicit robustness guarantees. We showed how the algorithm was used to train a powerful, robust, and interpretable discriminator for heavy-flavor decays in the LHCb trigger system. Furthermore, thanks to the expressive capacity of the architecture, we were able to shrink the number of model parameters to meet the memory and latency requirements of the LHCb trigger, which allows for faster event selection. This translates to higher sensitivity to the elusive physics phenomena we aim to observe. Our algorithm has been adopted for use as the primary data-selection algorithm in the LHCb trigger in the current LHC data-taking period known as Run 3. More specifically, it is currently in use in the two most important triggers in LHCb's High-Level Trigger (HLT): HLT1 generic multi-track lines and HLT2 topological B trigger.

Our architecture could also be used in various applications in which robustness is required such as safety-critical environments and those which need protection against adversarial attacks. Monotonicity is a desirable property in various applications where fairness and safety are a concern. There are many scenarios in which models which are not monotonic are unacceptable. For example, in Ref. [29] we showed that our algorithm achieves state-of-the-art performance on benchmarks in medicine, finance, and other applications with monotonic inductive biases. In addition, in Ref. [50] and as we will see in Chapter 4, we presented a new and interesting direction for the architecture developed here: Estimation of the Wasserstein metric (Earth Mover's Distance) in optimal transport by employing the Kantorovich-Rubinstein duality to enable its use in geometric fitting applications.

Chapter 3

Representation Learning for Physics: Improving Offline Data Analysis

After data selection, physicists search the large amounts of data stored in the hope of finding various interesting events and phenomena. In this chapter, we explore a neural network-based method to improve this offline data analysis. Yet again, we find that neural networks are too powerful, which can lead to undesired results when the problem is underspecified. A key challenge in searches for resonant new physics is that classifiers trained to enhance potential signals must not induce localized structures. Such structures could result in a false signal when the background is estimated from data using sideband methods. A variety of techniques have been developed to construct classifiers which are independent from the resonant feature (often a mass). Such strategies are sufficient to avoid localized structures, but are not necessary. We develop a new set of tools using a novel moment loss function (Moment Decomposition or MoDe)[1], which relaxes the assumption of independence without creating structures in the background. By allowing classifiers to be more flexible, we enhance the sensitivity to new physics without compromising the fidelity of the background estimation.

3.1 Introduction

Searching for new phenomena associated with localized excesses in otherwise featureless spectra, often referred to as bump hunting, is one of the most widely used approaches in particle and nuclear physics, dating back at least to the discovery of the ρ meson [52], and used continuously since, including recently in the discovery of the Higgs boson [53], [54]. In the present day, such searches reach the multi-TeV scale [55], [56] and span high energy particle and nuclear physics experiments [57]–[63]. A key feature of these searches is that they are relatively background model agnostic since sidebands in data can be used to estimate the background under a potential localized excess. These sideband fits are possible because

[1]This chapter is based on research originally presented in Ref. [51]. The work was conducted in collaboration with Ben Nachman, Constantin Weisser, and Mike Williams.

the background data can be well-approximated either with simple parametric functions or smooth non-parametric techniques such as Gaussian processes [64].

Sideband methods for background estimation are often combined with relatively simple and robust event selections in order to ensure broad coverage of new physics model space. However, there is a growing use of modern machine learning techniques to enhance signal sensitivity [65]–[69]. For example, both ATLAS [70] and CMS [71] have developed machine-learning-based W jet taggers that improve the sensitivity of searches involving Lorentz-boosted and hadronically decaying W bosons. Boosted electroweak bosons are common in searches for models with a significant mass hierarchy between the primary resonance mass and the W boson mass [72]–[83], and boosted W-like particles are a feature of searches for low-mass dark matter mediators [84]–[90].

A key challenge with complex event selections like those involved in boosted W tagging is that they can invalidate the smoothness assumption of the background. In particular, if classifiers can infer the mass of the parent resonance, then selecting signal-like events will simply pick out background events with a reconstructed mass near the target resonance mass. Many techniques have been developed that modify or simultaneously optimize classifiers so that their responses are independent of a given resonance feature [91]–[108]. For machine learning classifiers, the proposed solutions include modifications to loss functions that implicitly or explicitly enforce independence. These methods have been successfully deployed in bump hunts; see, *e.g.*, Refs. [74], [83]–[88], [90], [109]–[119]. A variety of similar proposals under the monikers of domain adaptation and fairness have been proposed in the machine learning literature (see e.g. Ref. [120], [121] and Ref. [122], [123]).

Ensuring that a classifier is independent from a given resonant feature is sufficient for mitigating sculpting, but it is not necessary. The original requirement is simply that a selection using the classifier does not introduce localized features in the background spectrum, which is a much looser requirement than enforcing independence. For example, if a classifier has a linear dependence on the resonant feature, then there would be a strong correlation. However, a threshold requirement on such a classifier would not sculpt any bumps in the background-only case. This example motivates a new class of techniques that allow classifiers to depend on the resonant feature in a controlled way. In the limit that constant dependence is required, then the classifier and the resonant feature will be independent. The advantage of relaxing the independence requirement is that the resulting classifiers can achieve superior performance because they are allowed to be more flexible.

In this chapter, we present a new set of tools that allow for controlled dependence on a resonant feature. This new approach is called *Moment Decomposition* (MoDe). Using MoDe, analysts can require independence, linear dependence, and quadratic dependence. In addition, analysts can place bounds on the slope of the linear dependence, and restrict quadratic dependence to be monotonic. Extending MoDe to allow for arbitrarily higher-order dependence is straightforward. This chapter is organized as follows. Section 3.2 briefly reviews existing decorrelation methods and then introduces MoDe. Numerical results using a simplified model and a physically motivated example are presented in Section 3.3. Finally, we present conclusions and outlook in Section 3.4.

3.2 Methods

3.2.1 Existing decorrelation methods

We will consider the binary classification setting in which examples are given by the triplet (X, Y, M), where $X \in \mathcal{X}$ is a feature vector, $Y \in \mathcal{Y} := \{0, 1\}$ is the target label, and finally, $M \in \mathcal{M}$ is the resonant feature (or protected attribute) whose spectrum will be used in the bump hunting. Throughout this chapter, we take M to be mass, though it could be any feature. The feature vector X can either contain M directly as one of its elements or contain other features that are arbitrarily indicative of M. We are interested in finding a mapping $f : \mathcal{X} \to \mathcal{S}$ where $s \in \mathcal{S}$ are scores used to obtain predictions $\hat{y} \in \mathcal{Y}$ with the additional constraint that f be conditionally independent of (or uniform with) M in the sense that

$$p(f(X) = s|M = m, Y = y) = p(f(X) = s|Y = y) \; \forall \; m \in \mathcal{M} \text{ and } \forall \; s \in \mathcal{S}, \tag{3.1}$$

for one or more values y, although typically, Equation (3.1) is required only for the background.

Existing decorrelation methods used in particle physics that simultaneously train a classifier $f(x) : \mathbb{R}^n \to [0, 1]$ and decorrelate from a resonant feature m use the following loss function:

$$\mathcal{L}[f(x)] = \sum_{i \in \mathrm{S}} L_{\text{class}}(f(x_i), 1) + \sum_{i \in \mathrm{B}} w(m_i)\, L_{\text{class}}(f(x_i), 0) + \lambda \sum_{i \in \mathrm{B}} L_{\text{decor}}(f(x_i), m_i)\,, \tag{3.2}$$

where $S = \{i|\, y_i = 1\}$ and $B = \{i|\, y_i = 0\}$ denote signal and background, respectively, L_{class} is the usual classification loss such as the binary cross entropy $L_{\text{BCE}}(f(x), y) = y \log(f(x)) + (1 - y) \log(1 - f(x))$, w is a weighting function, λ is a hyperparameter, and L_{decor} generically denotes some form of decorrelation loss. Standard classification corresponds to $w(m) = 1$ and $\lambda = 0$. Decorrelation methods include:

- Planing [106], [124]: $\lambda = 0$ and $w(m_i) \approx p_S(m)/p_B(m)$ so that the marginal distribution of m is non-discriminatory after the reweighting.
- Adversaries [91], [95], [100], [107]: $w(m) = 1$, $\lambda < 0$, and L_{decor} is the loss of a second neural network (adversary) that takes $f(x)$ as input and tries to learn some properties of m or its probability density.
- Distance Correlation (DisCo) [98], [105]: $w(m) = 1, \lambda > 0$, and the last term in Equation (3.2) is the *distance correlation* [125]–[128] between $f(x)$ and m for the background.[2]
- Flatness [102]: $w(m) = 1$, $\lambda > 0$, and $L_{\text{decor}} = \sum_m b_m \int |F_m(s) - F(s)|^2 \, \mathrm{d}s$ where the sum runs over mass bins, b_m is the fraction of candidates in bin m, F is the cumulative distribution function, and $s = f(x)$ is the classifier output.

Decorrelation methods have proven to be useful additions to the toolkit of the bump hunter.

[2]Technically, the term L_{decor} is applied at the level of a batch because it requires computing expectation values over pairs of events.

3.2.2 Moment decorrelation

First, we will derive a new decorrelation method based on moments. While this technique achieves state-of-the-art decorrelation performance, along with being robust, simple, and fast, its true value is that it is trivially extended to allow for controlled dependence beyond just decorrelation.

We begin by noting that the uniformity constraint in Equation (3.1) can be written in terms of the conditional cumulative distribution function (CDF) of scores at s, $F(s|M,Y)$, as

$$F(f(X) = s|Y = y) = F(f(X) = s|M = m, Y = y) \; \forall \; m \in \mathcal{M} \text{ and } \forall \; s \in \mathcal{S}. \tag{3.3}$$

This is the same observation that lies at the heart of the flatness loss defined in Ref. [102]. Here, we will consider the conditional CDFs in bins of mass and only on the background, which allows us to adopt the following more compact notation

$$F(f(X) = s|M = m, Y = y) \to F_m(s), \tag{3.4}$$

where now m is discrete and indexes the mass bins. We leave the exploration of similar unbinned approaches for future work. Furthemore, we assume that some transformation is performed on m such that $\mathcal{M} \to [-1, 1]$. This could be a simple linear transformation but does not have to be, discussion on this point is provided later.

The uniformity constraint of Equation (3.3) can be imposed on the learned function by defining the decorrelation loss using[3]

$$L_{\text{decor}} \to L^0_{\text{MoDe}} \equiv \sum_m \int |F_m(s) - F^0_m(s)|^2 \mathrm{d}s. \tag{3.5}$$

Here, F^0_m is based on the 0^{th} Legendre[4] moment of $F_m(s)$ in m, c_0, and polynomial, $P_0(x) = 1$, as

$$F^0_m(s) = c_0(s)P_0(\tilde{m}) = \frac{1}{2}\int_{-1}^{+1} P_0(m')F(s|m')\mathrm{d}m' \approx \frac{1}{2}\sum_{m'} \Delta_{m'} F_{m'}(s), \tag{3.6}$$

where Δ_m denotes the width of bin m, and $\tilde{m}$ is its central mass value. Note that the loss in Equation (3.5) is clearly minimized when

$$F_m(s) = F^0_m(s) = c_0(s) \; \forall \; m, \tag{3.7}$$

which implies that Equation (3.3) holds and $f(X)$ and M are indeed independent. Note that in the limit that all bins have equal width and occupancy, it is straightforward to show that the loss function in Equation (3.5) is the same as the flatness loss of Ref. [102]; however, when the underlying background distribution is highly nonuniform, these loss functions are drastically different resulting in MoDe outperforming Ref. [102] in such cases.

[3]We do not presume to know what the analyst is going to do with the trained model; therefore, we weight all score values equally seeking to achieve decorrelation for any score threshold. If additional information is available about how the model will be used, another choice of weighting function of the form $\mathrm{d}s \to w(s)\mathrm{d}s$ could be used instead, though it would be important to ensure that the functional derivative of the MoDe loss can still be calculated precisely; see Section 3.2.4.

[4]Any choice of orthogonal polynomials would work here.

3.2.3 Beyond decorrelation: Moment decomposition

We will now generalize moment decorrelation to allow for controllable mass dependence in the form of an ℓ^{th} order polynomial, where ℓ is a hyperparameter chosen by the analyst. The generalized MoDe loss is given by

$$\mathcal{L}[f] = L_{\text{class}} + \lambda L^{\ell}_{\text{MoDe,}} \tag{3.8}$$

where

$$L^{\ell}_{\text{MoDe}} \equiv \sum_m \int |F_m(s) - F^{\ell}_m(s)|^2 \mathrm{d}s. \tag{3.9}$$

Here, F^0_m in Equation (3.5) has been replaced by

$$F^{\ell}_m(s) = \sum_{l=0}^{\ell} c_l(s) P_l(\tilde{m}), \tag{3.10}$$

and the Legendre moments are given by

$$c_l(s) = \left[\frac{2l+1}{2}\right] \int_{-1}^{1} P_l(m') F(s|m') \mathrm{d}m' \approx \left[\frac{2l+1}{2}\right] \sum_{m'} \Delta_{m'} P_l(\tilde{m}') F_{m'}(s). \tag{3.11}$$

We note that setting $\ell = 0$ reduces the generalized MoDe loss of Equation (3.9) down to the moment decorrelation of Equation (3.5).

The MoDe loss in Equation (3.9) is optimal when $F_m(s) = F^{\ell}_m(s)\ \forall\ m, s$, which clearly occurs when the mass dependence of the classifier is at most an ℓ^{th} order polynomial. For example, taking $\ell = 0$ drives the classifier to be independent of mass. More interestingly, choosing $\ell = 1$ allows for a linear mass dependence, $\ell = 2$ quadratic dependence, *etc.* Furthermore, making the replacement[5]

$$c_1(s) \to c_1^{\max} c_0(s) \tanh\left(\frac{c_1(s)}{c_1^{\max} c_0(s)}\right) \tag{3.12}$$

in Equation (3.10) places an upper limit $c_1^{\max} c_0(s) > 0$ on the magnitude of the linear slope (the first Legendre moment is the coefficient of the $\tilde{m}$ term), allowing the analyst to control this aspect of the mass dependence through a hyperparameter, $c_1^{\max}$. In addition, for the case where $\ell = 2$ is selected, it is straightforward to show that as long as $3|c_2(s)| \leq |c_1(s)|$ the derivative of $F^{\ell}_m(s)$ is nonzero on $(-1, 1)$. Therefore, making the replacement

$$c_2(s) \to \frac{c_1(s)}{3} \tanh\left(\frac{3c_2(s)}{c_1(s)}\right) \tag{3.13}$$

in Equation (3.10) results in monotonic mass dependence. This option can be turned on or off in MoDe, and can be used in conjunction with $c_1^{\max}$ if desired. Finally, controlled higher-order mass dependence can be achieved by extending these ideas to larger ℓ values.

[5]The hyperbolic tangent function has several beneficial properties which motivate its usage here—its range is $(-1, 1)$, it is differentiable, monotonic, and odd—although other functions could be substituted.

3.2.4 Computational details

Computing the MoDe loss and its gradient is straightforward using a few approximations. At the batch level,

$$F_m(s) \approx \frac{1}{n_m} \sum_{i=1}^{n} \Theta(s - s_i)\delta_{m,m_i}, \tag{3.14}$$

where n is the number of samples in the batch, n_m is the number of samples in bin m, $s_i \equiv f(x_i)$ is the score of sample i, and Θ is the Heaviside function: $\Theta(x) = 1$ if $x > 0$ and $\Theta(x) = 0$ otherwise. Minimizing the loss function requires calculating the functional derivative of L^{ℓ}_{MoDe} with respect to f. This requires specifying how the MoDe loss changes due to variations of the score of each sample in the batch:

$$\delta L^{\ell}_{\text{MoDe}} = \delta s_i \sum_m \int 2\left[F_m(s) - F^{\ell}_m(s)\right] \left[\frac{\partial F_m}{\partial s_i} - \frac{\partial F^{\ell}_m}{\partial s_i}\right] \mathrm{d}s, \tag{3.15}$$

where from Equation (3.14)

$$\frac{\partial F_m}{\partial s_i} \approx -\frac{1}{n_{m_i}} \delta(s - s_i)\delta_{m,m_i}. \tag{3.16}$$

In addition, using this result, along with Equations (3.10) and (3.11), we obtain

$$\frac{\partial F^0_m}{\partial s_i} \approx \frac{1}{2}\Delta_{m_i} \frac{\partial F_{m_i}}{\partial s_i} = -\frac{\Delta_{m_i}}{2n_{m_i}} \delta(s - s_i), \tag{3.17}$$

$$\frac{\partial F^1_m}{\partial s_i} \approx \frac{\partial F^0_m}{\partial s_i} + \frac{3}{2}\tilde{m} \cdot \tilde{m}_i \Delta_{m_i} \frac{\partial F_{m_i}}{\partial s_i} = -\frac{\Delta_{m_i}}{2n_{m_i}} \delta(s - s_i)\left[1 + 3\tilde{m} \cdot \tilde{m}_i\right], \tag{3.18}$$

$$\vdots$$

where the sum over mass bins in Equation (3.11) is no longer needed, since changes to the score for sample i only affect the CDF in bin m_i. The factors of $\delta(s - s_i)$ eliminate the integral over s resulting in relatively simple gradient terms, *e.g.*, for $\ell = 1$ we obtain

$$\delta L^1_{\text{MoDe}} \approx -\delta s_i \sum_m \frac{1}{n_{m_i}} \left[F_m(s) - F^1_m(s)\right] \left[2\delta_{m,m_i} - \Delta_{m_i}(1 + 3\tilde{m} \cdot \tilde{m}_i)\right]. \tag{3.19}$$

The fact that terms like Equation (3.19) do not depend on how the integral over s is approximated yields high-precision gradients, which is a big advantage when performing gradient descent.

The results in this subsection are easily generalized for weighted samples. The CDFs in Equation (3.14) become

$$F_m(s) \approx \frac{1}{w_m} \sum_{i}^{n} w_i \Theta(s - s_i)\delta_{m,m_i}, \tag{3.20}$$

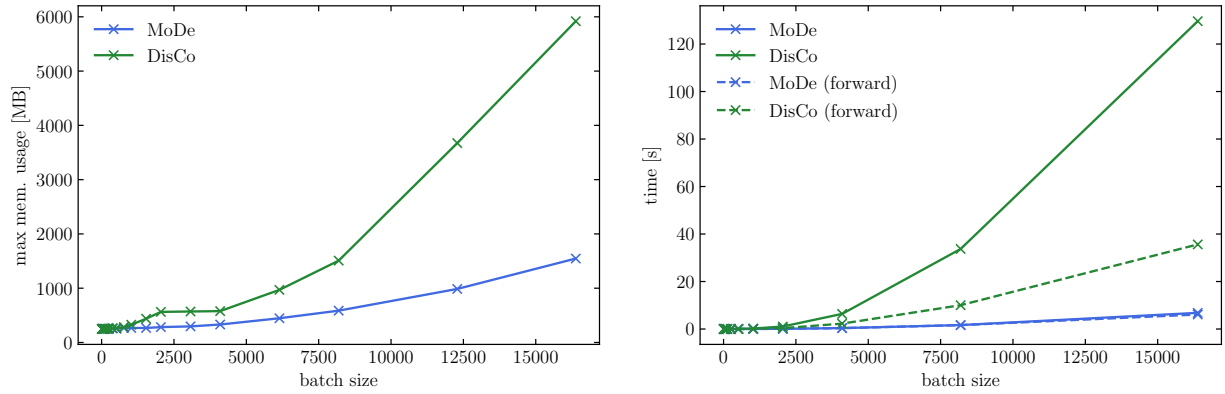


Figure 3.1: Maximum memory usage and CPU time for different batch sizes of triples (s_i, m_i, y_i).

where w_i are the per-sample weights and

$$w_m = \sum_i^n w_i \delta_{m,m_i} \tag{3.21}$$

is the sum of the weights in bin m.Equation (3.16) then becomes

$$\frac{\partial F_m}{\partial s_i} \approx -\frac{w_i}{w_{m_i}} \delta(s - s_i) \delta_{m,m_i}, \tag{3.22}$$

and updating the rest of the results follows accordingly.

Finally, we address the topic of scalability. While the optimization of the MoDe loss works well stochastically with few examples every step, its performance increases greatly with larger batch sizes. This is not surprising due to the *global* nature of the MoDe constraint. Fortunately, all of the calculations scale well with batch size (see Figure 3.1 for time and memory performance as functions of the number of inputs). Most computational costs occur in the forward direction, where MoDe scales linearly with the number of inputs n (batch size) and the number of steps chosen for the integral in s, n_s. In addition, dynamic binning sorts m_i and reindexes s_i are required, and so MoDe runs in $\mathcal{O}(n_s \times n + n \log n)$ time. In the forward direction, we also compute and cache the residual $F_m(s_i) - \tilde{F}_m(s_i)$ which is used in the backward pass. Since the CDFs are evaluated at every s_i, this contributes an $\mathcal{O}(n \times n_m)$ component. In theory, this could be improved, *e.g.*, if the CDFs at s_i were instead approximated using nearest neighbor interpolation. Finally, MoDe takes $\mathcal{O}(n \times n_m)$ extra memory (beyond what is required to store the data) when calculating the gradients, since the CDF is computed for each input for each bin. It is worth noting that, at particularly small batch sizes, MoDe might be susceptible to slow convergence due to mini-batch statistics not accurately reflecting the full-batch statistics. That is why we recommend using MoDe with a sizeable fraction of the full sample.

3.3 Results

In this section, we will demonstrate how MoDe performs on a simple model problem, and on the W-jet tagging problem used in the decorrelation studies of Ref. [97], [98]. All of the numerical results reported in this section are obtained using the PyTorch framework [22].

3.3.1 Simple Model

We first consider a binary classification example composed of a signal and two types of background. Each sample $X \in \mathcal{X}$ has 2 features:

$$x_1 \sim \begin{cases} \mathcal{N}(1,1) & \text{when } Y = 1, \\ \mathcal{N}(0,1) & \text{when } Y = 0 \text{ for background type 1}, \\ \mathcal{N}(-4,1) & \text{when } Y = 0 \text{ for background type 2}, \end{cases} \tag{3.23}$$

$$x_2 = \exp\left[-\frac{(m-0.2)^2}{2 \cdot 0.1^2}\right] \quad \text{when } Y = 0 \text{ or } Y = 1, \tag{3.24}$$

where $\mathcal{N}$ denotes the normal distribution. The mass, m, is drawn from $\mathcal{N}(0.2, 0.1)$ and $\mathcal{U}(-1,1)$ at equal rates when $Y = 1$. For the backgrounds, we sample a uniform random variable, $U \sim \mathcal{U}(0,1)$, then define $m = 1 - 2\sqrt{U}$ and $m = -1 + 2\sqrt{U}$ for background types 1 and 2, respectively. This simple-model data is shown in Figure 3.2.

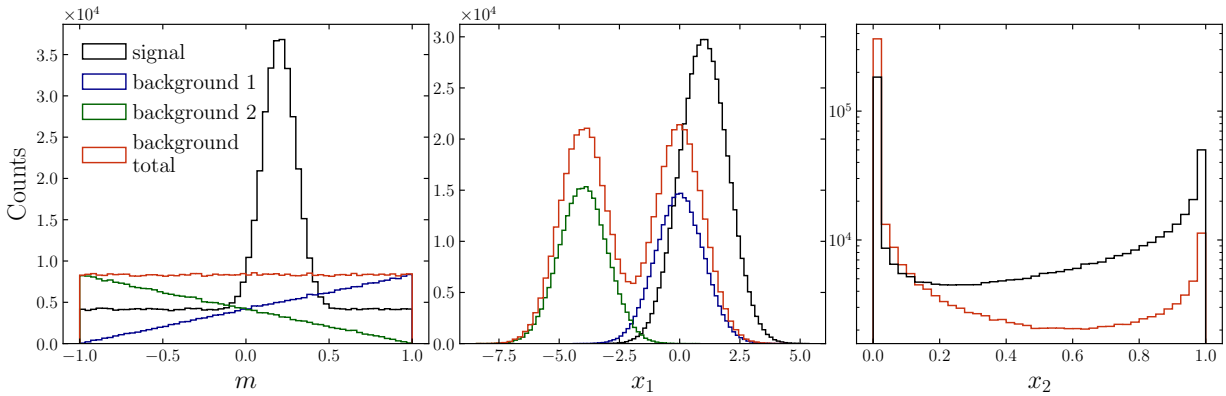


Figure 3.2: (From Kitouni et al. [51]) Simple model distributions.

In this scenario, an unconstrained classifier with sufficient capacity will learn the underlying mass distribution (due to the explicit mass dependence of x_2) and use it to discriminate between signal and background. Figure 3.3 shows how such a classifier favors regions near $m = 0.2$, leading to extreme peak-sculpting in the background. It would be difficult to employ this classifier in a real-world analysis and obtain an unbiased signal estimator. Figure 3.3 also shows that MoDe[0] successfully decorrelates the classifier response from mass producing a viable classifier for such an analysis.

In this simple example, we can easily choose to only use information not explicitly indicative of mass by removing x_2 from $\mathcal{X}$. Figure 3.3 shows that the resulting *mass agnostic* classifier is linearly correlated with mass. Indeed, we ensured this via our choice of x_2, *i.e.* we

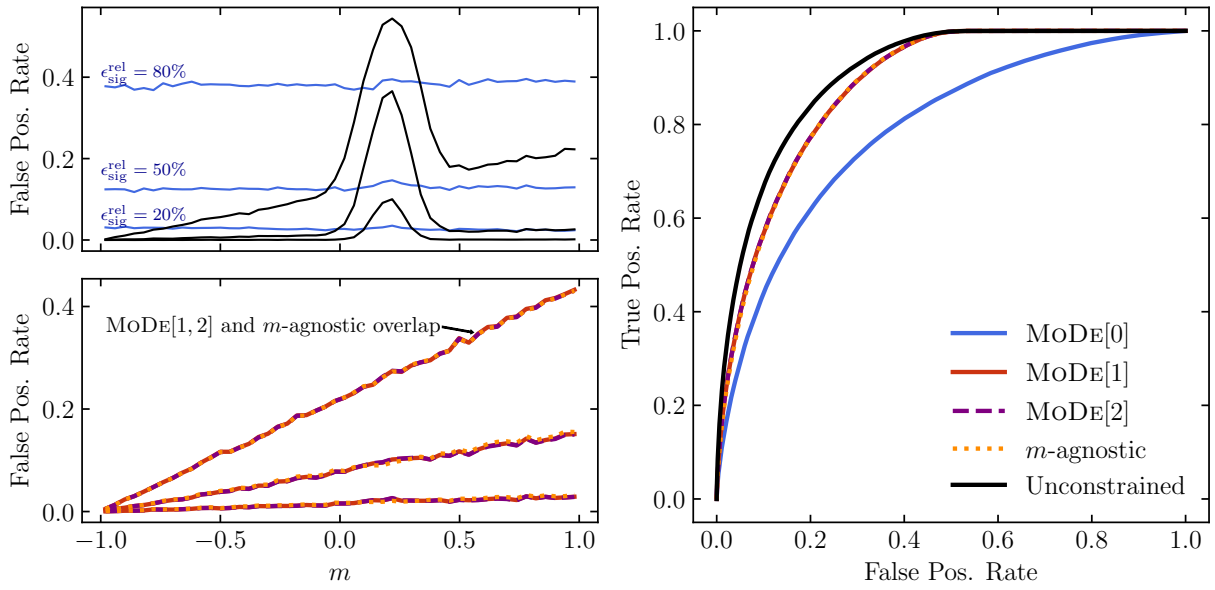


Figure 3.3: (From Kitouni et al. [51]) **Left:** The false positive rate versus mass for various models at signal efficiencies $\epsilon_{\text{sig}}^{\text{rel}} = 80, 50, 20\%$ (each set of 3 identically colored and stylized lines correspond to the same model but with selection thresholds chosen to achieve the 3 desired signal efficiencies). The bottom panel shows that MoDe[1] and MoDe[2] completely overlap with the m-agnostic model for this simple example, which is expected because the optimal classifier here has linear dependence on mass (see text). **Right:** ROC curves for MoDe[0], MoDe[1], and MoDe[2] compared to the m-agnostic model and a model with unconstrained mass dependence. As in the left panel, we see that MoDe[1], MoDe[2], and the m-agnostic ROC curves are nearly identical because the optimal classifier has linear mass dependence in this simple example.

configured this toy example such that the optimal classifier, the likelihood ratio, is linearly correlated with mass and obtained without the use of x_2. However, a classifier that enforces decorrelation must accept backgrounds 1 and 2 at equal rates to keep $p(s|m)$ flat, which as shown in Figure 3.3 produces performance that is far from optimal. By relaxing the flatness constraint, MoDe[1] is able to reject background 2 at a higher rate, while producing the expected linear dependence on mass. This linear mass dependence, which will not sculpt out any fake peaks from the background, allows MoDe[1] to achieve better classification power than is possible using decorrelation. In this case, it is able to achieve the same performance as the optimal mass-agnostic classifier, since the optimal performance here is linear. Figure 3.3 shows that even though MoDe[2] is given the freedom to find quadratic mass dependence, it also produces the same optimal linear mass dependence in this case.

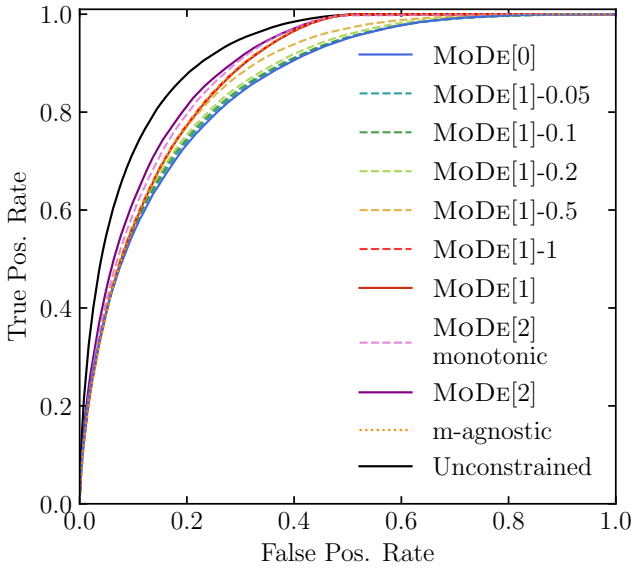


Figure 3.4: (From Kitouni et al. [51]) Same as the right panel of Figure 3.3 but with the simple-model modification of Equation (3.25). In addition, a monotonic version of MoDe[2] and several versions of MoDe[1] with constrained maximum slope values are also shown.

As discussed in Section 3.2, the MoDe package provides an even higher level of control over the response of a model, including allowing the analyst to define the maximum linear slope and to require that the quadratic dependence is monotonic. To demonstrate these features, we make the following minor change to the simple model:

$$x_2 \to \exp(m) + 2m. \tag{3.25}$$

In this case, the optimal classifier is no longer linear. Figure 3.4 shows that here the additional freedom given to MoDe[2] does improve the classification performance. Figure 3.5 shows that the MoDe[2] solution does indeed have quadratic mass dependence. The MoDe[2] response is not monotonic by default, but we also show in Figure 3.5 that we can apply such a constraint. As can be seen in Figure 3.4, there is a small decrease in classification performance; whether this is acceptable is a problem-specific decision left to the analyst. Finally, Figure 3.6 shows that the analyst can exert full control over the maximum linear

slope. This could be desirable in cases where the signal mass is not known, and similar—but not necessarily equivalent—performance across the mass range is viewed as beneficial.

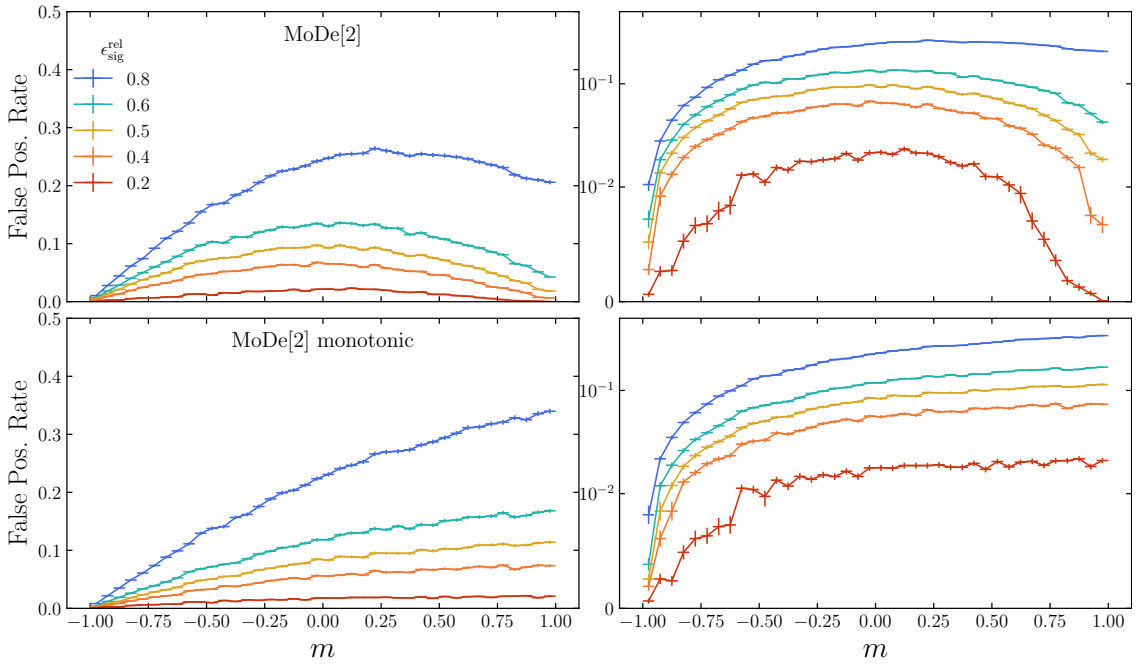


Figure 3.5: (From Kitouni et al. [51]) **Top:** False positive rate versus mass at various signal efficiencies for non-monotonic MoDe[2] on the modified simple-model example; see Equation (3.25). **Bottom:** False positive rate for monotonic MoDe[2]. *N.b.*, the right panels show the same curves as the left but on log scales.

3.3.2 Boosted hadronic W tagging

As mentioned in Section 3.1, highly lorentz boosted, hadronically decaying W bosons commonly arise in extensions of the Standard Model. The boost causes the decay products of these bosons to be collimated in the lab frame and to be mostly captured by a single large-radius jet. Various features of the substructure of these jets can be used to distinguish the boosted bosons from generic quark and gluon jets.

A bump hunt is performed either in the mass of the W candidate jet, m_J, or the mass of one W candidate jet and another (possibly W candidate) jet, m_{JJ}. The challenge with substructure classifiers is that they can introduce artificial bumps into the mass spectrum because substructure is correlated with the jet mass and the jet kinematic properties (which are related to m_{JJ}). For this reason, boosted W tagging has become a benchmark process for studying decorrelation methods at the LHC.

The simulated samples used in this section are the same as in Ref. [98] (intended to emulate the study in Ref. [97]) and are briefly summarized here. In particular, boosted W bosons (signal) and generic multijet (background) events are generated with Pythia 8.219 [129], [130] and a detector simulation is provided by Delphes 3.4.1 [131]–[133]. Jets are clustered using the anti-k_t algorithm [134] with $R = 1.0$ implemented in FastJet 3.0.1 [135], [136]. The

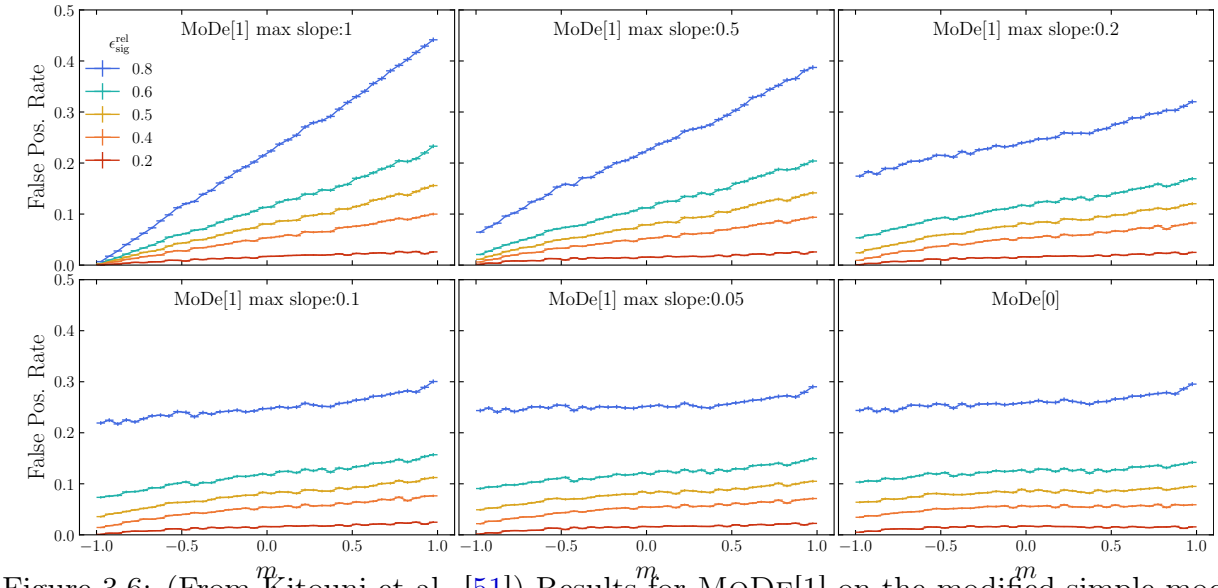


Figure 3.6: (From Kitouni et al. [51]) Results for MoDe[1] on the modified simple-model example requiring various maximum slope values.

selected jets for this study have 300 GeV $< p_T <$ 400 GeV and 50 GeV$< m_J <$ 300 GeV. Ten representative jet substructure features are computed for each jet and used for classification. This list is the same as in Ref. [97] (based on Ref. [137]) and includes the energy correlation ratios C_2 and D_2 [138], the N-subjettiness ratio τ_{21} [139], the Fox-Wolfram moment $R_2^{\rm FW}$ [140], planar flow $\mathcal{P}$ [141], the angularity a_3 [142], aplanarity A [143], the splitting scales $Z_{\rm cut}$ [144] and $\sqrt{d_{12}}$ [145], and the k_t subjet opening angle $KtDR$ [146]. Detailed explanations of these features can be found in the references.

Classifier Details

MoDe and DisCo: We use a simple 3-layer neural network with a similar architecture to that described in Ref. [98]. However, unlike Refs. [98] and [97], after each of the 3 fully connected 64-node layers, we use Swish activation [147] as it provides a notable performance increase. We also use a batch normalization layer after the first fully connected layer. The output layer has a single node with a sigmoid activation. Both MoDe and DisCo are trained with the ADAM optimizer [148] using a 1cycle learning rate policy [149] with a starting learning rate of 10^{-3} and a maximum learning rate (lr) of 10^{-2}, which is reached using a cosine annealing strategy [150] and decayed to 10^{-5} during the last few iterations. Momentum is cycled in the inverse direction from 0.95 to a minimum of 0.85. These hyperparameters were selected through a learning rate range test. Training is done using large batches of 10–20% of the training data. Note that large batch sizes do not necessarily make training more difficult especially when coupled with the 1cycle learning policy; see Ref. [151].

Adversarial Decorrelation: The same classifier used for MoDe and DisCo is trained against a Gaussian Mixture Network (GMN) [152] that parametrizes a Gaussian mixture model with 20 components, *i.e.* its outputs are the means, variances, and mixing coefficients of

20 normal distributions. We follow a similar adversarial setup to that referenced in Refs. [97] and [98]. We use one hidden layer with 64 nodes with ReLu activation connected to 60 output nodes. These outputs are then used to model the posterior probability density function $p_{\theta_{\text{adv}}}(m|f(X)=s)$, where θ_{adv} are the parameters of the GMN. The adversary is optimized by maximizing the likelihood of the data given by

$$L_{\text{adv}} = \mathbb{E}_{s\sim f(X)}\mathbb{E}_{m\sim M|s}\left[-\log\, p_{\theta_{\text{adv}}}(m|s)\right]. \tag{3.26}$$

The training procedure starts by training the classifier alone for 20 epochs with $lr = 10^{-4}$ followed by 20 epochs of adversarial training only with $lr = 5\cdot 10^{-3}$. Finally, both networks are trained simultaneously by optimizing the following loss function

$$\arg\min_{\theta_{\text{class}}}\ \max_{\theta_{\text{adv}}}\left[L_{\text{class}(\theta_{\text{class}})} - \lambda L_{\text{adv}}(\theta_{\text{class}},\theta_{\text{adv}})\right], \tag{3.27}$$

where θ_{class} and θ_{adv} are the parameters of the classifier and the adversary, respectively. To control the classification-decorrelation trade-off, we vary λ between 1 and 100. The non-convex nature of the loss makes training considerably more difficult; the hyperparameters must be chosen carefully.

Decorrelation

First, we will show that MoDe[0] achieves state-of-the-art decorrelation performance. Following Ref. [98], we quantify the classification and decorrelation performance using the following metrics: R50, the background rejection power (inverse false positive rate) at 50% signal efficiency; and 1/JSD, where the Jensen-Shannon divergence (JSD) is a symmetrized version of the Kullback–Leibler divergence. Here, JSD is used to compare the mass distributions of backgrounds that pass and fail the classifier-based selections, with the relative entropy measured in bits.

Figure 3.7 shows that, as expected, without imposing a strong constraint on mass decorrelation, the classifier learns to select samples near the W-boson mass, which sculpts a fake peak in the background. Figure 3.7 also shows that MoDe[0] successfully decorrelates its response from mass (if the decorrelation hyperparameter λ is chosen to be sufficiently large). Figure 3.8 shows that the existing state-of-the-art decorrelation methods discussed in Section 3.2 perform similarly to MoDe[0] on this W-tagging problem.

More precisely, as observed in Ref. [98], the adversary method performs slightly better, but is considerably more difficult to train, since it requires carefully tuning two neural networks against each other. The optimal solution is a saddle point, where the classification and adversarial losses are minimized and maximized, respectively, which makes the training inherently unstable. Conversely, both DisCo and MoDe[0] minimize convex loss functions, making their training robust and stable, and both only introduce one additional hyperparameter in the loss function. The performances of MoDe[0] and DisCo are comparable in these metrics (Figure 3.1 shows that MoDe[0] is less resource intensive), though decorrelation is not our primary objective.

Beyond Decorrelation

Moving beyond decorrelation the 1/JSD metric is no longer relevant. Figure 3.7 shows that neither MoDe[1] nor MoDe[2] sculpts a peaking structure in the background, but their

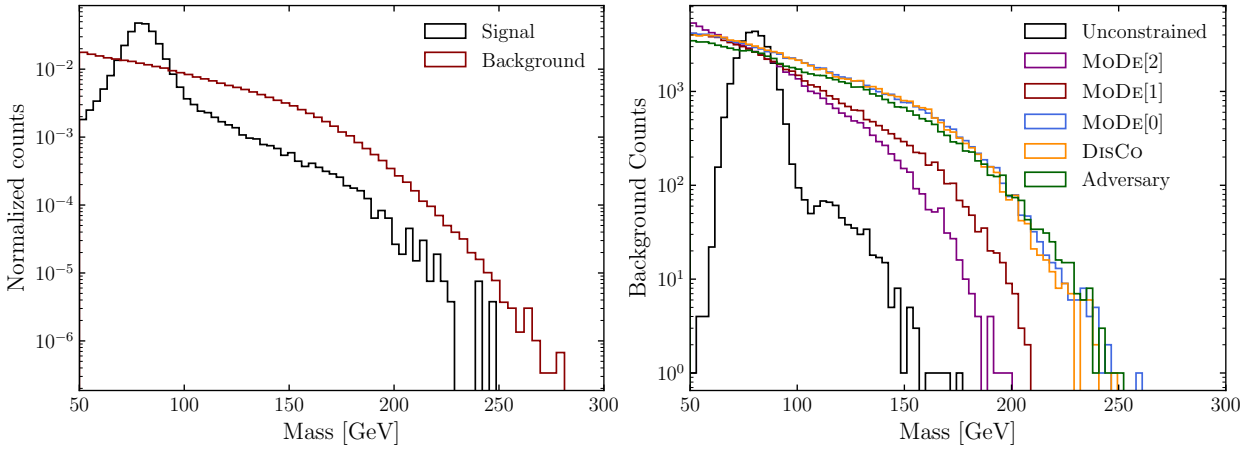


Figure 3.7: (From Kitouni et al. [51]) **Left:** Distributions of signal and background events without selection. **Right:** Background distributions at 50% signal efficiency (true positive rate) for different classifiers. The unconstrained classifier sculpts a peak at the W-boson mass, while other classifiers do not.

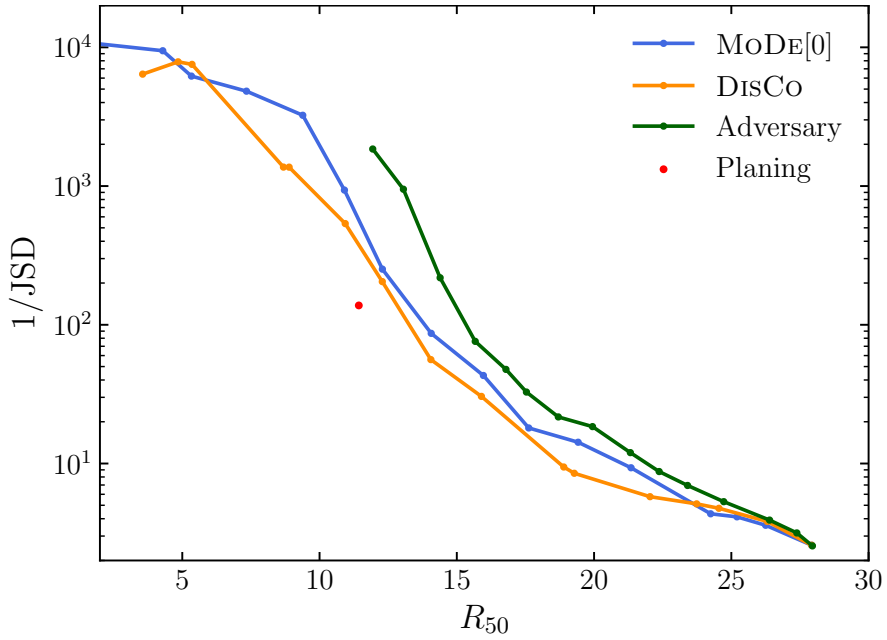


Figure 3.8: (From Kitouni et al. [51]) Decorrelation versus background-rejection power showing that MoDe[0] performs similarly to existing state-of-the-art decorrelation methods.

1/JSD values are small since neither seeks to decorrelate from the mass. Therefore, we replace the 1/JSD metric with the signal bias induced by the classifier selection, which is what actually matters when searching for resonant new physics. Specifically, we use the signal estimators obtained by fitting the selected background-only samples to a simple polynomial function as proxies for the signal biases. These are divided by their uncertainties such that values of roughly unity are consistent with no bias, while values significantly larger than unity indicate substantial bias that could result in false claims of observations.

Figure 3.9 shows that the DisCo and MoDe[0] decorrelation methods provide unbiased signal estimators for R50 $\lesssim 9$, which from Figure 3.8 corresponds to 1/JSD $\gtrsim 1000$. While achieving higher decorrelation values is possible, this does not provide any tangible gains in the bump-hunt analysis. Figure 3.9 also shows that the flexibility to go beyond decorrelation provided by MoDe[1] and MoDe[2] results in achieving unbiased signal estimators at larger background-rejection power. This would directly translate to improved sensitivity in a real-world analysis. For example, since it is likely that only unbiased classifiers would be considered, Figure 3.9 can be used to estimate the improvement in the signal cross-section sensitivity for the W-tagging analysis, which scales roughly like $\sqrt{R_{50}}$, using the classifier of each type with the largest R_{50} value that is consistent with being unbiased. MoDe[1] and MoDe[2] provide roughly 5% improved sensitivity over the adversary method, which recall is considerably more difficult to train, and 10–20% improvements over the other decorrelation methods. We note, however, that how much is gained will strongly depend on the specifics of the problem, *e.g.*, how large of a mass range is considered and whether the signal mass is known or if a scan in mass will be done.

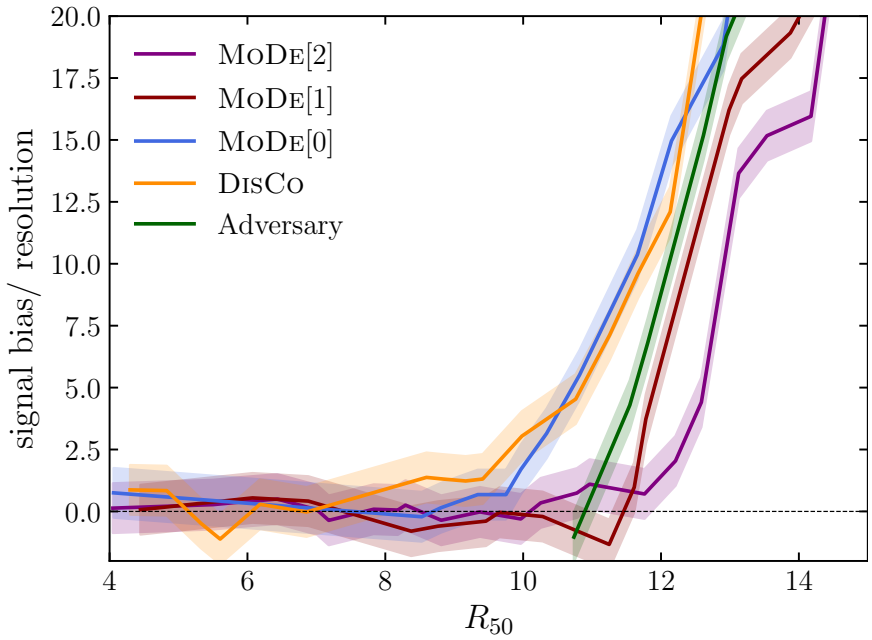


Figure 3.9: (From Kitouni et al. [51]) Signal bias relative to resolution, which is roughly the square root of the background in the signal region, versus background-rejection power. The flexibility beyond simple decorrelation provided by MoDe[1] and MoDe[2] result in improved performance, *i.e.* larger rejection power.

3.4 Summary & Discussion

In summary, a key challenge in searches for resonant new physics is that classifiers trained to enhance potential signals must not induce localized structures. In particular, if classifiers can infer the mass of the parent resonance, then selecting signal-like events will simply pick out background events with a reconstructed mass near the target resonance mass creating an artificial structure in the background. Such structures could result in a false signal when the background is estimated from data using sideband methods. A variety of techniques have been developed to construct classifiers which are independent from the resonant feature (often a mass). Such strategies are sufficient to avoid localized structures, but are not necessary.

In this chapter, we presented a new set of tools using a novel moment loss function (Moment Decomposition or MoDe) which relax the assumption of independence without creating structures in the background. Using MoDe, analysts can require independence, linear dependence, quadratic dependence, *etc.* In addition, analysts can place bounds on the slope of the linear dependence, and restrict higher-order dependence to be monotonic. By allowing classifiers to be more flexible, we enhance the sensitivity to new physics without compromising the fidelity of the background estimation.

Code and Data

An implementation for MoDe in PyTorch as well as Keras/Tensorflow is available at https://github.com/okitouni/MoDe, along with example code used to produce the results presented here. The simulated W and QCD samples are available from Zenodo at Ref. [153].

Chapter 4

Representation Learning for Physics: Improving Searches by Translating New Theoretical Insights

In this chapter, we present an interesting application for networks that enforce a strict upper bound on their Lipschitz constant building on the results from Chapter 2: geometrical fitting through differentiable estimation of the Earth Mover's Distance. This is a completely self-supervised paradigm that deviates from the standard input/label setup described in Chapter 1. This work is an example of the broad utility and flexibility of neural networks trained with gradient descent as a general learning paradigm. We focus on high-energy physics, where it has been shown that a metric for the space of particle-collider events can be defined with the Earth Mover's Distance, referred to in this context as Energy Mover's Distance (EMD). This metrization has the potential to revolutionize data-driven collider phenomenology. The work presented here represents a step towards realizing this goal by providing a differentiable way of directly calculating the EMD. We show how the flexibility that our approach enables can be used to develop novel clustering algorithms.[1]

4.1 Introduction

The Earth Mover's Distance, otherwise referred to as Wasserstein-1 distance, is a metric defined between two probability measures. In the field of high-energy particle physics, a modified version of the Earth Mover's distance, the Energy Mover's Distance (EMD), serves as a metric for the space of collider events by defining the *work* required to rearrange the radiation pattern of one event into another [154]. In particular, the EMD is intimately connected to the structure of *infrared- and collinear-safe* observables used in the ubiquitous task of clustering particles into *jets* [155], and is foundational in the SHAPER tool for developing geometric collider observables [156].

[1]This chapter is based on research originally presented in Ref. [50]. The work was conducted in collaboration with Niklas Nolte and Mike Williams.

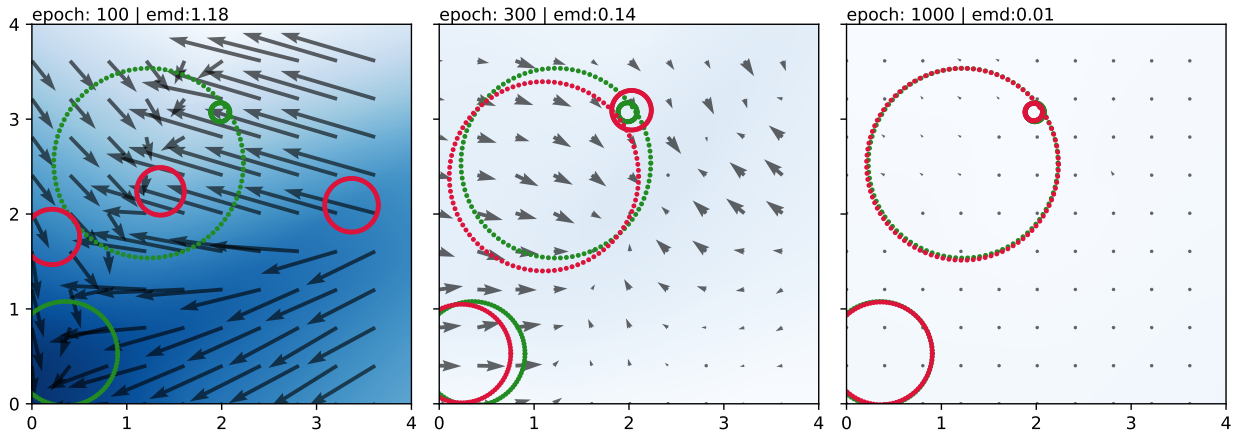


Figure 4.1: (From Kitouni et al. [50]) Fitting three synthetic clusters (green) with three circles (red) using NEEMo (see Section 4.3). The heatmap is the Kantorovic potential, parameterized as a Lipschitz-bounded network, which induces forces on the circles (shown as arrows) that drive them into perfect alignment with the target distribution (only a few steps in the evolution of the fit are shown).

Recently, a novel neural architecture was developed that enforces an exact upper bound on the Lipschitz constant of the model by constraining the norm of its weights in a minimal way, resulting in higher expressiveness than other methods [30], [41]. Here, we employ this architecture—leveraging its improved expressiveness for 1-Lipschitz continuous networks—to replace the ϵ-Sinkhorn estimation of the EMD in SHAPER [156], [157] by directly calculating the EMD using the Kantorovic-Rubenstein (KR) dual formulation (though there are already other implementations of Sinkhorn that use the KR duality). The KR duality casts the optimal transport problem as an optimization over the space of 1-Lipschitz functions, which we parameterize with dense neural networks using the architecture from [30]. With small modifications to the KR dual formulation, we are able to reliably and accurately obtain the EMD and Kantorovic potential in a differentiable way, without any ϵ approximations. This makes it possible to run gradient-based optimization procedures over the exact EMD (see Figure 4.1). In addition, we expect these improvements could potentially have a major impact on jet studies at the future Electron-Ion Collider, where traditional clustering methods are not optimal [158], and more broadly in optimal transport problems.

4.2 Lipschitz Networks and the Energy Mover's Distance

Lipschitz Networks Fully connected networks can be Lipschitz bounded by constraining the matrix norm of all weights [30], [38], [39]. Constraints with respect to a particular L^p norm will be denoted as Lip^p. We start with a model $f(\boldsymbol{x})$ that is Lip^p with Lipschitz constant λ *i.e.*, $\forall\, \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$:

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq \lambda \|\boldsymbol{x} - \boldsymbol{y}\|_p \,. \tag{4.1}$$

Without loss of generality, we take $\lambda = 1$ (rescaling the inputs would be equivalent to changing λ). We recursively define the layer l of the fully connected network of depth D with activation

σ as

$$\boldsymbol{z}^l = \boldsymbol{W}^l \sigma(\boldsymbol{z}^{l-1}) + \boldsymbol{b}^l, \tag{4.2}$$

where $\boldsymbol{z}^0 = \boldsymbol{x}$ is the input and $f(\boldsymbol{x}) = z^D$ is the output of the neural network. We have that $f(\boldsymbol{x})$ satisfies equation 4.1 if

$$\|\boldsymbol{W}^i\|_\infty \leq 1 \quad \text{when} \quad 2 \leq i \leq D \quad \text{and} \quad ||\boldsymbol{W}^1||_{p,\infty} \leq 1 \tag{4.3}$$

and σ has a Lipschitz constant less than or equal to 1. Here, $||\boldsymbol{W}||_{p,q}$ denotes the operator norm with norm L^p in the domain and L^q in the co-domain. It is shown in [41] that when using the **GroupSort** activation, $f(\boldsymbol{x})$ can approximate any Lip^p function arbitrarily well, making weight-normed networks universal approximators.

Energy Mover's Distance The EMD is a metric between probability measures $\mathbb{P}$ and $\mathbb{Q}$. Using the standard Wasserstein-metric notation, the EMD is defined as

$$\text{EMD}(\mathbb{P}, \mathbb{Q}) = \inf_{\gamma \in \Pi(\mathbb{P}, \mathbb{Q})} \mathbb{E}_{(x,y)\sim\gamma}\big[||x - y||_2\big], \tag{4.4}$$

where $\Pi(\mathbb{P}, \mathbb{Q})$ is the set of all joint probability measures whose marginals are $\mathbb{P}$ and $\mathbb{Q}$. The EMD optimization problem can be cast as an optimization over Lipschitz continuous functions using the Kantorovich-Rubinstein duality:

$$\text{EMD}(\mathbb{P}, \mathbb{Q}) = \sup_{||f||_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}}\big[f(x)\big] - \mathbb{E}_{x\sim\mathbb{Q}}\big[f(x)\big], \tag{4.5}$$

where f is Lip^2 continuous, *i.e.,* $||\nabla f||_2 \leq 1$. In high-energy particle collisions, the EMD is defined by using the energies of individual particles in place of probabilities, with their momentum directional coordinates representing the supports of the probability distribution. For more details, including on how unequal total energies are handled, see [154]. By performing optimizations over a constrained set of $\mathbb{P}$s, one can use the EMD to define observables over $\mathbb{Q}$.

4.3 NEEMo: Neural Estimation of the Energy Mover's Distance

Algorithm Consider a high-energy particle-collision event with n particles. Let E^i be the energy of particle i, $\boldsymbol{x}^i$ be the direction of its momentum, and $\mathbb{Q} = \{(E^i, \boldsymbol{x}^i)\}_{i=1}^n$ be the set of all particles in the event. Following the SHAPER prescription [156] for defining an observable $O(\mathbb{Q})$, we first define $\mathbb{P}_{\boldsymbol{\theta}} = \{w^i_{\boldsymbol{\theta}}, \boldsymbol{y}^i_{\boldsymbol{\theta}}\}_{i=1}^m$ to be any collection of points parameterized by $\boldsymbol{\theta}$, *e.g.*, these points can be sampled from any geometric object with any density distribution. The EMD between the event $\mathbb{Q}$ and the geometric object $\mathbb{P}_{\boldsymbol{\theta}}$ can be computed with equation 4.5 as

$$\text{EMD}(\mathbb{P}_{\boldsymbol{\theta}}, \mathbb{Q}) = \max_{\boldsymbol{\phi}} \left[\sum_{i=1}^n E^i f_{\boldsymbol{\phi}}(\boldsymbol{x}^i) - \sum_{i=1}^m w^i_{\boldsymbol{\theta}} f_{\boldsymbol{\phi}}(\boldsymbol{y}^i_{\boldsymbol{\theta}}) \right], \tag{4.6}$$

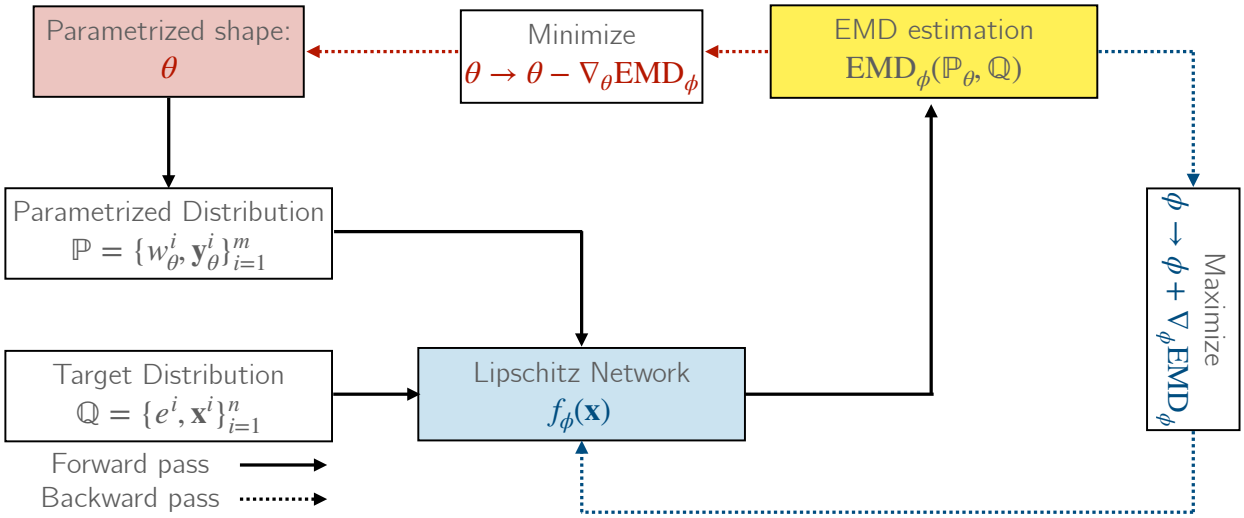


Figure 4.2: (From Kitouni et al. [50]) Training procedure to fit a parameterized shape $\mathbb{P}_{\boldsymbol{\theta}}$ to a distribution $\mathbb{Q}$. NEEMo replaces the ϵ-Sinkhorn estimation in the standard SHAPER procedure with a Lipschitz network that evaluates the Kantorovic potential to obtain the EMD.

where $f_{\boldsymbol{\phi}}(x)$ is a 1-Lipschitz neural network with parameters $\boldsymbol{\phi}$. At $\boldsymbol{\phi}^*$ the expression above is maximized and $f_{\boldsymbol{\phi}^*}$ is the Kantorovic potential from which the EMD is obtained as the RHS of equation 4.6. Since f is differentiable, the optimum can be obtained using standard gradient descent techniques. This is the key improvement of NEEMo over SHAPER, which can only estimate the Kantorovic potential and the EMD up to a specified order ϵ. Note that in equation 4.6 the expectation is computed exactly but optimization can also be done stochastically by sampling from the discrete distributions with probabilities $\{E^i\}_i$ and $\{w_\theta^i\}_i$ and using the empirical mean to estimate the EMD. This can improve convergence in some cases.

Given that all of our operations are differentiable, gradients can flow back to $\mathbb{P}_{\boldsymbol{\theta}}$. Therefore, one can also optimize the parameters $\boldsymbol{\theta}$ to obtain the best-fitting collection of points in that class. We obtain the following minimax optimization problem:

$$O(\mathbb{Q}) = \min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \left[\sum_{i=1}^{n} E^i f_{\boldsymbol{\phi}}(\boldsymbol{x}^i) - \sum_{i=1}^{m} w_{\boldsymbol{\theta}}^i f_{\boldsymbol{\phi}}(\boldsymbol{y}_{\boldsymbol{\theta}}^i) \right], \tag{4.7}$$

where $O(\mathbb{Q})$ quantifies how well the event $\mathbb{Q}$ is described by the class of geometric object $\mathbb{P}$ [155], [156].

Limitations Unlike the conventional clustering algorithms used in high-energy particle physics, NEEMo relies on nonconvex gradient-based optimization of a neural network and a set of geometric parameters. This results in the clustering procedure itself being relatively slow and not easily implemented in real time. This problem can be alleviated with powerful custom optimizers and initialization techniques to guarantee fast convergence, though whether

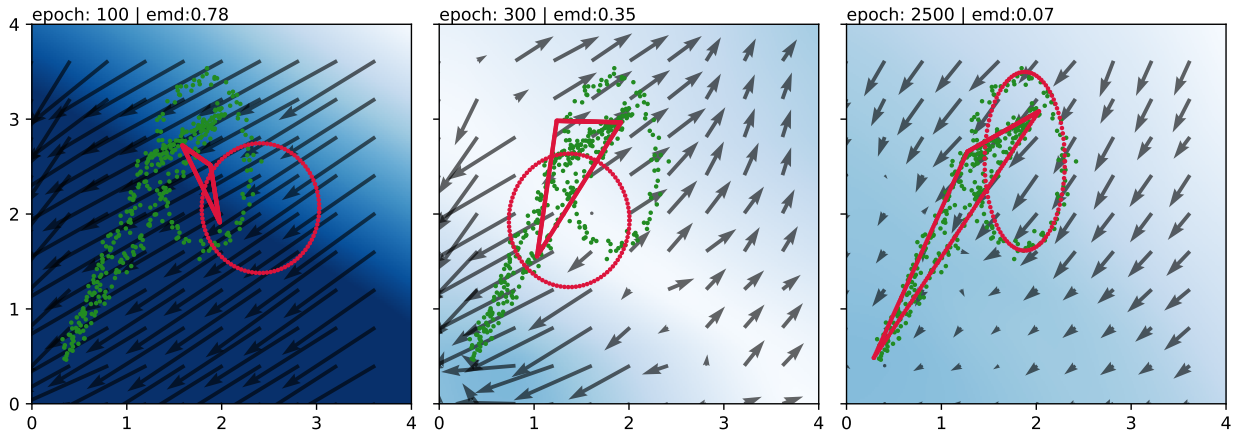


Figure 4.3: (From Kitouni et al. [50]) Same as Figure 4.1, but fitting to distributions parameterized by a triangle and an ellipse.

NEEMo could ever be run online during data taking is an open question. We note that for many potential applications, *e.g.* at the Electron-Ion Collider, this is not a problem since running online is not required.

4.4 Experiments

Synthetic Data We start with a few toy examples. First, consider an event consisting of three sets of particles distributed uniformly along the perimeters of circles. Here, we know the exact parameterization of our target distribution and use NEEMo to fit three randomly initialized circles to the event. Figure 4.1 shows a few steps in the fit evolution. The Kantorovic potential given by the Lipschitz-constrained network induces forces on the parameters of $\mathbb{P}$, which drive it to evolve from its random initialization to perfect alignment with the target distribution. In this example, $O(\mathbb{Q})$ in equation 4.7 quantifies the *3-circliness* of the event $\mathbb{Q}$, an observable first defined in [156]. To highlight the flexibility, we next consider an event with two sets of particles distributed along the perimeters of a triangle and ellipse, respectively. Figure 4.3 shows that $\mathbb{P}$ again evolves following the gradients of the Kantorovic potential to perfect alignment with the target distribution.

N-Subjets We now perform a model jet-substructure study, clustering synthetic data into *N-subjets*. First, we generate jets with 3, 4, or 5 subjet centers distributed uniformly. From each center we generate 10 particles drawn from a Gaussian distribution. We then use our algorithm to fit 3, 4, or 5 centers to the simulated jets. Figure 4.4 shows that our algorithm is able to estimate the correct number of subjets. The EMD of the N-subjet fit is clearly lowest for jets with N true clusters.

4.5 Summary & Discussion

In the framework developed in this chapter, any parameterized source distribution can be chosen to fit any target distribution using the EMD, without any ϵ-approximations. This

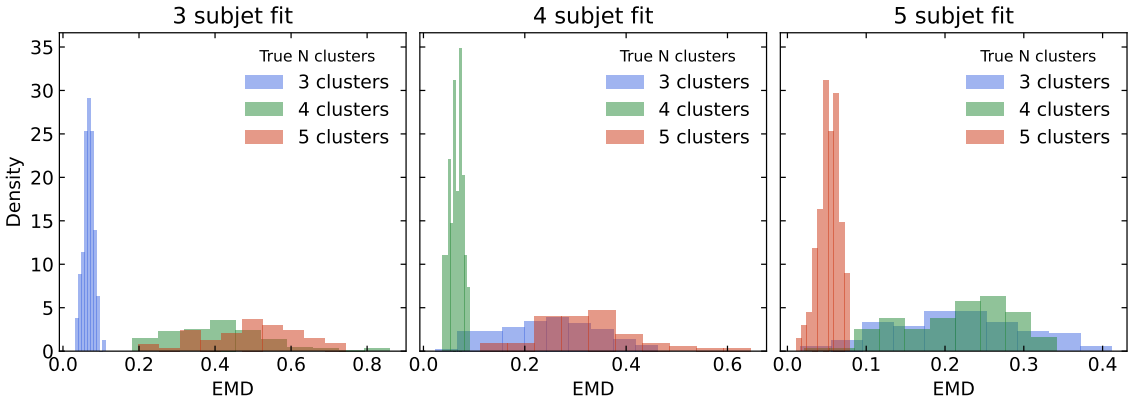


Figure 4.4: (From Kitouni et al. [50]) From left to right: Fit of N subjets (centers) to jets with 3, 4, or 5 number true subjets.

can be used, *e.g.*, for constructing precision jet observables that are sensitive to percent-level fluctuations for new physics searches at LHC experiments. In addition, NEEMo provides a more precise way to quantify event modifications due to hadronization and detector effects. Finally, the flexibility provided by NEEMo could potentially have a major impact on jet studies at the future Electron-Ion Collider, where traditional clustering methods are not optimal. Rather than modifying the metric used in a sequential-recombination algorithm as in [158], the jet geometry itself can be altered using NEEMo in an event-by-event unsupervised manner.

Chapter 5

Representation Learning for Physics: Towards Automating the Discovery Nuclear Laws

In this chapter, we tackle the third prong of representation learning for physics described in Chapter 1: *Automated understanding*. We are not quite capable of making scientific progress via fully automated AI/deep learning paradigms. However, this chapter explores a nascent, semi-automated approach as a case study.[1]

Mechanistic Interpretability (MI) promises a path toward fully understanding how neural networks make their predictions. Prior work demonstrates that even when trained to perform simple arithmetic, models can implement a variety of algorithms (sometimes concurrently) depending on initialization and hyperparameters. Does this mean neuron-level interpretability techniques have limited applicability? In this chapter, we argue that high-dimensional neural networks can learn low-dimensional representations of their training data that are useful beyond simply making good predictions.

Such representations can be *understood* through the mechanistic interpretability lens and provide insights that are surprisingly faithful to human-derived domain knowledge. This indicates that such approaches to interpretability can be useful for deriving a new understanding of a problem from models trained to solve it. As a case study, we extract nuclear physics concepts by studying models trained to reproduce nuclear data.

5.1 Introduction

The scientific process involves understanding high-dimensional phenomena, often with large-scale data, and deriving low-dimensional theories that can accurately describe and predict the

[1]This chapter is based on research originally presented in Ref. [159]. The work was conducted in collaboration with Niklas Nolte, Sokratis Trifinopoulos, Víctor Samuel Pérez-Díaz, and Mike Williams.

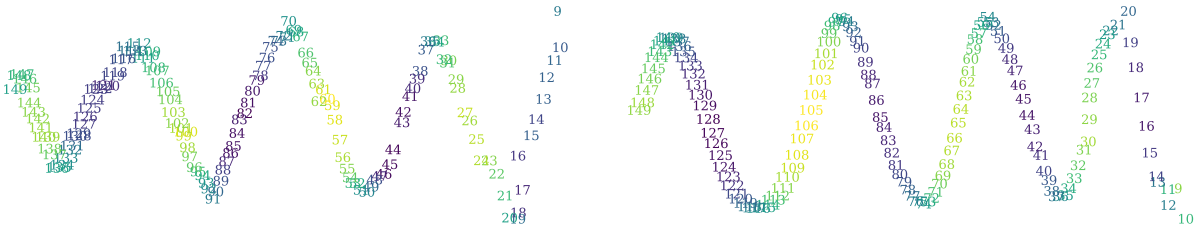


Figure 5.1: (From Kitouni et al. [159]) Projections of neutron number embeddings onto their first three principal components (PCs). Models were trained on nuclear data (left) or a human-derived nuclear theory (right). X-axis: 1st PC, Y-axis: 2nd PC, color: 3rd PC. Numbers indicate the neutron number (N) of each nucleus (see **Setup** in Section 5.3). The helix structure encodes insights about nuclear physics discussed in subsequent sections.

outcome of observations. There is mounting evidence that modern machine learning operates in a similar fashion, taking large-scale, high-dimensional data and deriving low-dimensional representations from them. For instance, recent work on the interpretability of deep learning has focused on understanding the low-dimensional representations learned by these models, with a particular emphasis on disentangled representations that separate the underlying factors of variation in the data [160]–[162]. Disentanglement aims to learn representations where each latent dimension corresponds to a semantically meaningful factor, such that varying one dimension while keeping others fixed produces interpretable changes in the input space [163]–[165].

Given the success of deep learning at modeling a wide variety of data, it seems plausible that interpretability can help us learn from these models that are effectively domain experts.[2] In this chapter, we investigate the ability of machine-learned algorithms to re-derive insights in human-developed understanding, taking nuclear theory as a case study of mechanistic interpretability.

Modern machine learning posits the manifold hypothesis [160], the idea that most natural data we tend to care about lives in a low-dimensional manifold embedded in the high-dimensional measurement space. This is observed across modalities and, more recently, in language modeling where low-rank representations are ubiquitous in fully-trained large language models [166]–[170]. Due to the nature of the data or the various implicit biases of the modern deep learning training procedures, neural networks learn compact representations that live in a small subspace of the inputs. Interpretability in deep learning has always been an active area of research [171], [172] but the process of understanding how neural networks operate to make particular predictions (macroscopic phenomena) by uncovering the algorithms they implement (microscopic phenomena), is a nascent field of deep learning built around the idea that neural networks, despite their scale and complexity, can be interpreted and understood [173], [174]. Here, we further posit that not only can they be understood, but they can also be used to say something useful about the nature of the problem they aim to solve. In the following, we will investigate whether mechanistic approaches can uncover scientific knowledge derived from the prediction task the model is trained on. In other words, we propose expanding the view on MI from "*How does a model make predictions?*" to include

[2]There are of course some caveats here such as the question of the robustness of learned representations.

"*What can the model tell us about the data?*"

In Section 5.2, we discuss prior work on MI in modular arithmetic and show an intuitive example of how it can be used to understand the algorithm that a simple MLP can learn to perform modular addition. Transitioning from modular arithmetic, Section 5.3 introduces the nuclear physics problem we will be tackling, explains the model architecture, and summarizes some key properties of the established physical models used by physicists. Then, in Section 5.4 we motivate and explain the approach we take to interpret the models trained on the nuclear physics data. Finally, in Section 5.5, we interpret and extract ubiquitous concepts from the model representations and show that these are similar to the most important human-derived concepts. For example, in Figure 5.1 we show a spiral pattern that emerges in the model's representation when trained on nuclear data is similar to the one that arises when training instead on pseudo data obtained from a human-derived nuclear theory.

5.2 Modular Arithmetic Primer

A recent wave of research in interpretability has focused on algorithmic tasks such as arithmetic or checking the parity of a sequence. This has good reason: These datasets are extremely clean, arbitrary in size, and non-trivial enough to show a variety of interesting phenomena. Models trained to perform modular arithmetic have been shown to yield relatively interpretable structures in their embeddings [175]. Prior work has shown that the algorithms by which the trained models perform the task can be recovered precisely by understanding the model mechanistically at the activation and neuron level. Furthermore, this interpretation can be used to provide progress measures for the model's ability to generalize [176]. Beyond these directions, we can leverage interpretability not only to understand models but also to extract knowledge from the training data. In this chapter, we explore this shift in perspective in a highly specialized domain.

First, we will revisit some of the mechanistic interpretability efforts for models trained to perform modular addition. In Figure 5.2 (left), we show the projection of the embeddings onto their first two principal components (PCs). Long after full generalization and circuit cleanup (see [176] for a definition), the algorithm learned by the network involves a simple vector average. This can be visualized easily by projecting the first layer activations down to the first two principal components, uniformly sampling points in a two-dimensional grid, and feeding them back into the network after a reverse transformation to the right space. This procedure, which we will henceforth refer to as *latent space topography* (LST), gives what the output of the network would have been as we move in a particular 2D subspace of the embeddings. As it turns out, this is quite informative. In Figure 5.2 (right), we overlay the 2D projections of the embeddings for each integer on top of our latent space map and find that in order to compute the modular sum of numbers, the network first computes the vector average between the embeddings and returns the index of the slice the resulting sum falls into. This fully explains the neural network solution to the problem but also sheds light on a new visual algorithm for modular addition. Simply arrange numbers around a circle, create slices between every two points, label the slices following the scheme given by the network in Figure 5.2, then finally obtain the sum of any two numbers by finding the mid point and reading off the label of the slice.

In the following sections, we demonstrate the feasibility of knowledge extraction beyond modular arithmetic, using nuclear physics as a case study. Researchers have invested significant effort in understanding and modeling this domain over several decades. By training models on such data, we investigate whether known physics concepts can be identified through inspection of their representations.

5.3 Beyond Arithmetic: A Physics Case Study

Why Nuclear Physics? We choose to explore nuclear physics as a case study for several compelling reasons. First, physicists have studied various aspects of this data for decades and have developed simple yet effective expressions and concepts that explain the data well. This provides a useful frame of reference and a plausible approximate "ground truth" for comparison. However, understanding the data remains a significant challenge, with several phenomena still unaccounted for by current theories and long-standing questions persisting. This combination of established knowledge and ongoing scientific challenges makes nuclear physics particularly interesting for interpretability research. To further motivate our choice, consider a simple principal component projection in Figure 5.1, extracted the same way as Figure 5.2 (left), but trained on nuclear physics. A surprisingly periodic and continuous helical structure emerges, suggesting an opportunity for insightful interpretation.

The remainder of this section will be organized as follows: First, we provide a description of the experimental process and the data to establish context. We also briefly discuss existing human-derived knowledge about the data. Next, we take a close look at the input embeddings. Embeddings have been shown to carry significant structure in modular arithmetic training [175] and are a promising first step for model interpretation. Finally, we study model features extracted from the penultimate layer activation and compare them to known physics terms to gauge similarities between model-derived and human-derived features.

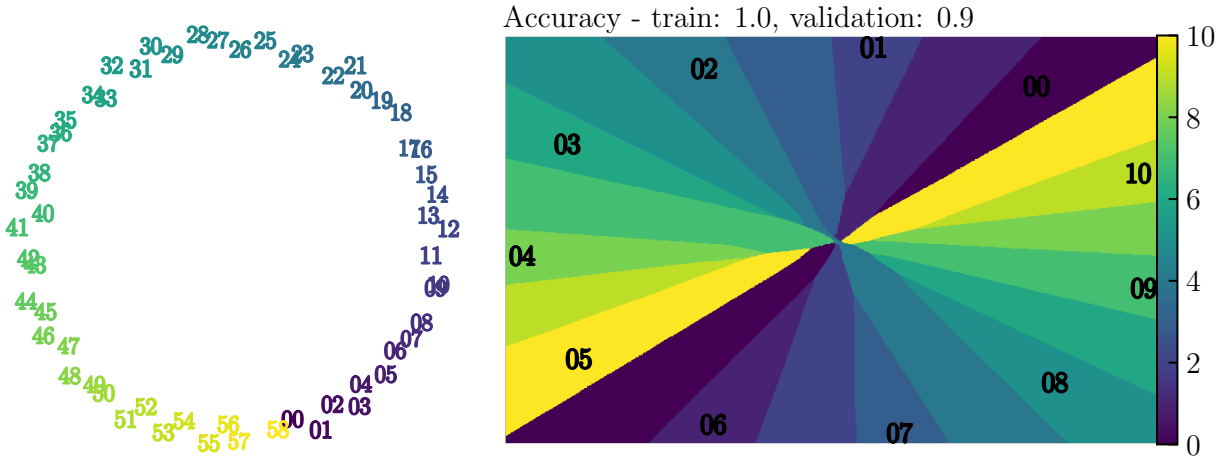


Figure 5.2: (From Kitouni et al. [159]) (left) Principal component projection of modular addition embeddings. The circular structure mirrors human-derived approaches used to teach modular arithmetic. (right) Model output in regions of the phase space. From [175].

Dataset and Nuclear Theory Nuclei, the cores of atoms, have an array of interesting properties that depend on their composition. Like elements in the periodic table, they can be visualized on a two-dimensional grid and are characterized by two integer-valued inputs: the number of protons (Z) and neutrons (N), ranging from 1 to 118 and 0 to 178, respectively. From these inputs, we aim to predict several continuous target properties of nuclei: binding energy (E_{B}), charge radius (R_{ch}), and various separation energies (Q_{A}, Q_{BM}, Q_{BMN}, Q_{EC}, S_{N}, S_{P}; see Appendix B.3.4 for more details). As a form of regularization, we often also predict the input values Z and N that are obscured during embedding. This creates a multivariate regression task across up to 10 target observables for 3363 total nuclei.

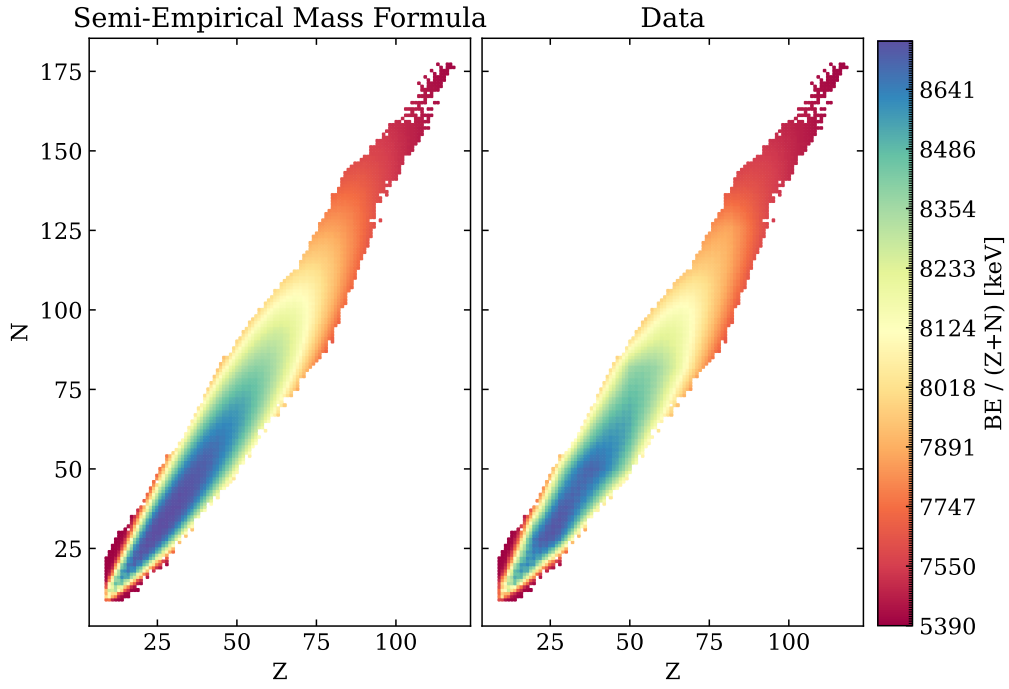


Figure 5.3: (From Kitouni et al. [159]) Binding energy per nucleon as given by the SEMF formula (left) and observed in measurements (right).

One of the most important nuclear observables is the binding energy. Many models have been developed in the literature with the liquid-drop model being the prototypical description of the nucleus. A consequence of the model is the renowned Semi-Empirical Mass Formula (SEMF) [177]:

$$E_{\mathrm{B}} = \underbrace{a_{\mathrm{V}} A}_{\text{Volume}} - \underbrace{a_{\mathrm{S}} A^{2/3}}_{\text{Surface}} - \underbrace{a_{\mathrm{C}} \frac{(Z^2 - Z)}{A^{1/3}}}_{\text{Coulomb}} - \underbrace{a_{\mathrm{A}} \frac{(N-Z)^2}{A}}_{\text{Asymmetry}} + \underbrace{\delta(N,Z)}_{\text{Pairing}} , \tag{5.1}$$

where $A = N + Z$ is the total nucleon number. The coefficients a_* are determined empirically. Appendix B.3 contains more detailed explanations of each term. This formula is fairly

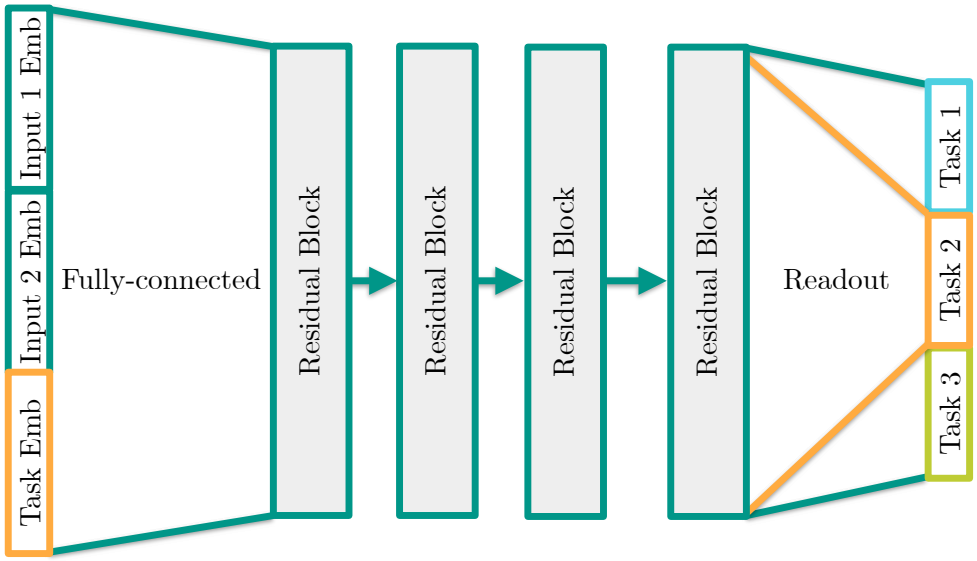


Figure 5.4: (From Kitoni et al. [180]) Model diagram: input-data and task embeddings are concatenated, projected, and passed through a sequence of residual blocks. Different readout heads output predictions for each task.

accurate and theoretically well-motivated. Figure 5.3 shows E_{B} for both the data and the SEMF.

Setup We are interested in making predictions of the form $T(Z, N) =?$, where T is the task or observable being considered, and Z and N are integers uniquely identifying a nucleus on which predictions will be made. Similar to the algorithmic tasks setup, inputs are tokenized and stacked in a sequence. Each token is embedded into a d-dimensional space. The sequence of embeddings (E_Z, E_N, E_T) is then fed into the model, which is tasked with completing the sequence using a numerical prediction. Specifically, the last token prediction is compared against the target numerical value and penalized with a mean-squared error loss. Similar to [178], we find that using attention provides a qualitatively different solution than input-independent attention ([179]). For the purposes of this chapter, we will focus on fixed attention where all tokens are attended to equally[3] (see Appendix B.2 and Figure 5.4).

In all our experiments, we will consider one or several observables to predict with various models. The performance of the models will generally be measured by a Root-Mean-Square error (RMS) on a holdout set.[4] We will also predict some useful unitless quantities such as the neutron and proton numbers.

Objectives Our goal will be to understand how the models' generalizing solutions work, extract useful representations from them, and compare those solutions to what is well-known in nuclear theory. To ascertain the source of the learned representations, we can train our model on different tasks and collect results from the following experiments: **(1)** Train multiple

[3]Without residual connections, this model could be written as a feedforward MLP.

[4]Error is in units of keV for energies and fm for lengths.

models with different seeds on different data splits to understand the properties of generalizing versus memorizing solutions. **(2)** Study the internal representations of models trained on different tasks to understand the mechanistic effects of multi-tasking on generalization *i.e.* what are the features of the representations that generalize and where do they come from? **(3)** Compare the neural network-derived concepts with human-derived models.

5.4 Are Principal Components Meaningful?

Principal Component Analysis (PCA) is a widely used dimensionality reduction technique due to its simplicity. However, it relies on several assumptions that, when violated, can result in erroneous conclusions. There is extensive literature discussing various PCA pitfalls, such as the complex relationship between oscillations and PCA ([181]–[184]). Remarkably, these studies reported instances where non-oscillatory data exhibited oscillatory principal components. If this phenomenon is prevalent across various types of data, it is crucial to ensure it does not affect our results.

5.4.1 Evidence 1: PCs Capture Most of the Performance

There is evidence in the literature that models operate on a much smaller subspace than their full dimension. Low-Rank adaptation ([166]) is an example showing that much of the performance gains from supervised fine-tuning can be obtained by training a low-rank approximation of the model. If the PCs extracted were meaningless, we should see large performance gaps between the original model and one that solely relies on a subset of the PCs in making predictions. However, we do indeed recover most of the performance with a relatively small number of PCs. Figure 5.5 shows the error as a function of principal components at different layers. To get this prediction, we project the activations (or the embeddings) onto their first k principal components (ordered by variance) and set higher order components to zero. Then we invert the initial projection and consider the result the new activation that is sent through the rest of the network.

The behaviour observed in Figure 5.5 seems to be fairly universal, albeit to varying degrees. For instance, [185] recently utilized PCA to increase sparsity in language models by projecting activations to their principal components without losing significant performance.

5.4.2 Evidence 2: Rich Structure

Phantom oscillations are sinusoidal patterns that can emerge in PCA even when the underlying data does not contain oscillations ([186]). They can arise due to noise, smoothness across a continuum like time or space, or small misalignments/shifts across observations. Phantom oscillations characteristically emerge at multiple frequencies, with each principal component exhibiting a distinct frequency and lower frequencies explaining more variance. In this chapter, we found that PC features exhibit unique patterns that differ from those expected in the case of noise. As observed in the previous section, highly informative structures emerge in the first two PCs of embeddings when learning modular arithmetic. Using Figure 5.2 as a reference, [175] and [178] hypothesized the complete algorithm used to perform the modular

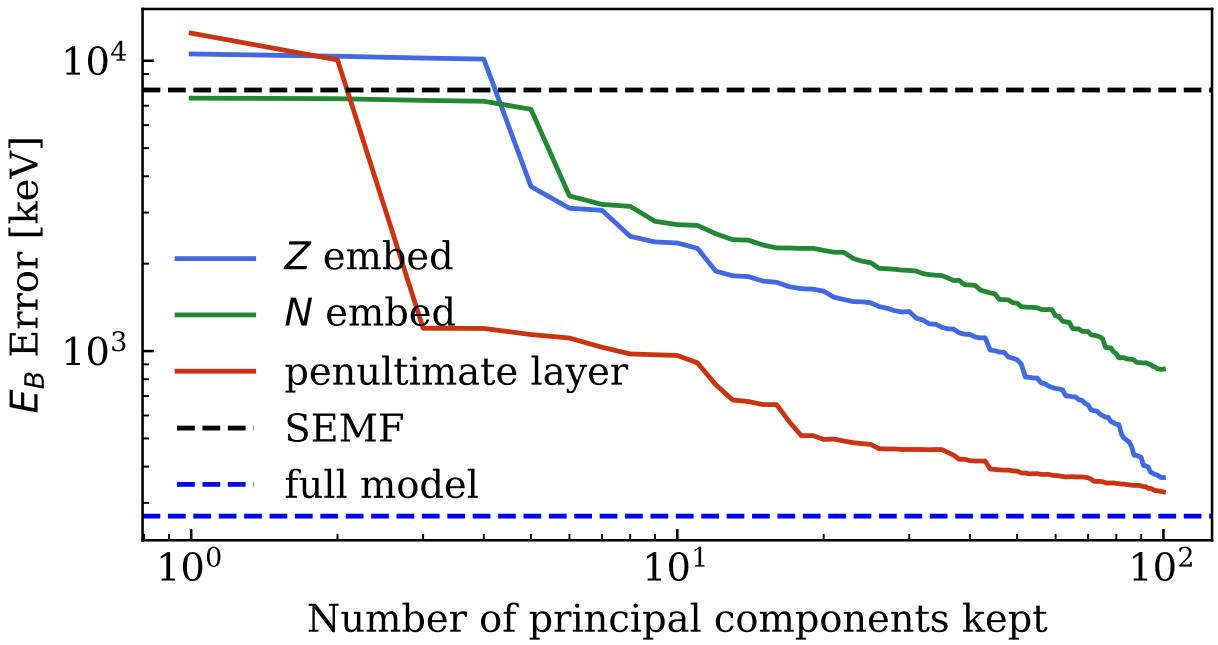


Figure 5.5: (From Kitouni et al. [159]) Binding energy prediction error as a function of number of PCs used at different layers.

addition task. In the context of nuclear physics, similarly rich structures emerge during training beyond what would be expected in the case of noise. Figure 5.6 displays the first two PCs of proton number embeddings extracted from a generalizing model. This clearly showcases features such as an even-odd split and periodicity, which we further explore in subsequent sections.

5.5 Experiments

5.5.1 Embeddings

Growing evidence, including studies on language model analogies(*e.g.*, the "$king - man + woman = queen$" analogy) ([187]) suggests the presence of interpretable and robust structures in the initial embedding layers of neural networks. We can reasonably expect similar phenomena to occur in nuclear physics, and thus we will closely examine the neutron and proton number embeddings for trained models.

Given the large dimensionality of the embeddings, we analyze the latent representations using a low-dimensional PCA projection, as motivated in Section 5.4. Figure 5.6 illustrates the three highest variance principal components of proton embeddings, plotted against each other. The observed structure, a helix (or spiral) pattern associated with increasing proton numbers, is one of the most striking features in the models trained. The color scheme transitions to lighter hues for higher numbers, emphasizing the clear numerical ordering

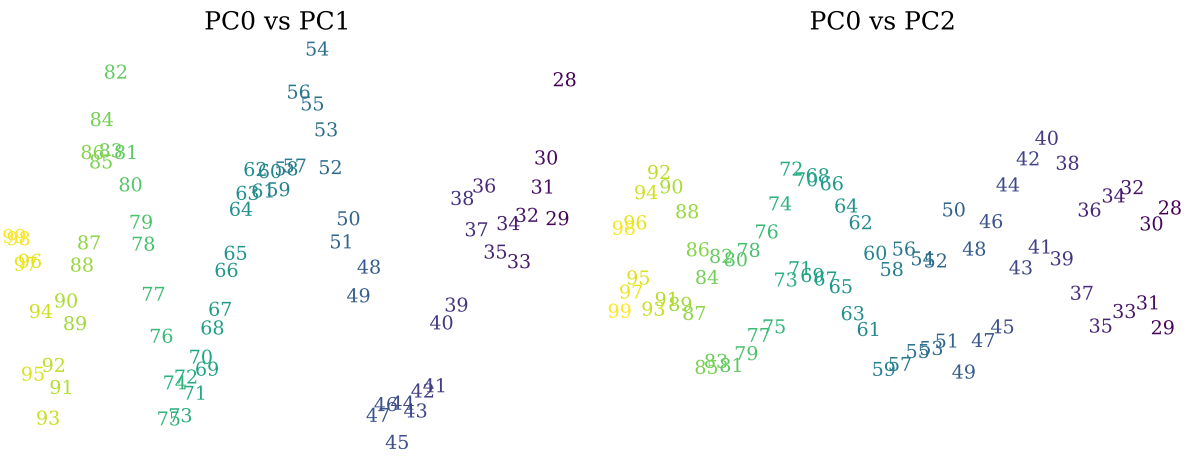


Figure 5.6: (From Kitouni et al. [159]) PC projections of Z embeddings from a model trained on all tasks. The color hue is a monotonic function of the proton number Z, to be able to quickly assess the presence of order.

observed.[5] This ordering is also apparent, and the helix structure is particularly pronounced, in the high-variance primary components of the neutron number embeddings from Figure 5.1. Note that the color in this case represents the third PC.

Notably, E_B has a strong correlation with both N and Z, as seen in the first term of the SEMF. Therefore, it seems plausible that the inductive bias of ordering neutron and proton numbers in the embedding space is particularly beneficial. To understand the model better, consider Figure 5.7, the latent space topography of Z embeddings, constructed similarly to Figure 5.2 for modular addition. It shows the predicted E_{B} as a colored background to the scatter plot of the two highest variance primary components in the Z embeddings for $N = 100$. The dominating effect is the monotonic increase in binding energy when moving from right to left in PC0, which corresponds to the fact that E_{B} scales as $A = Z + N$ to leading order (this is known as the *volume term* in the SEMF Equation (5.1)).

Properties of Models That Generalize Well Modifying the model architecture and hyperparameters significantly can result in different generalizing algorithms. We explore a small region of the algorithmic phase space and discover that generalizing solutions share a set of common properties, which we enumerate here.

1. Helicity We attempt to isolate the origin of the helix structure in the neutron and proton embeddings, and find that it represents a compelling geometric explanation of the data. Experiments reveal this structure appears when predicting binding energy. To elucidate how the model utilizes the helix, we parameterize it and perturb parameters to understand their effects (a detailed study with visualization is shown in Appendix B.1). We fit a helix to the visually most helix-like portion of 3D PCA projections as illustrated in Figure 5.9. The fits map to the projections well and enable us to isolate the effect of the different parameters

[5] While the number ordering could be expected for models where N and Z are among the prediction targets, it persists even in models where those targets are absent.

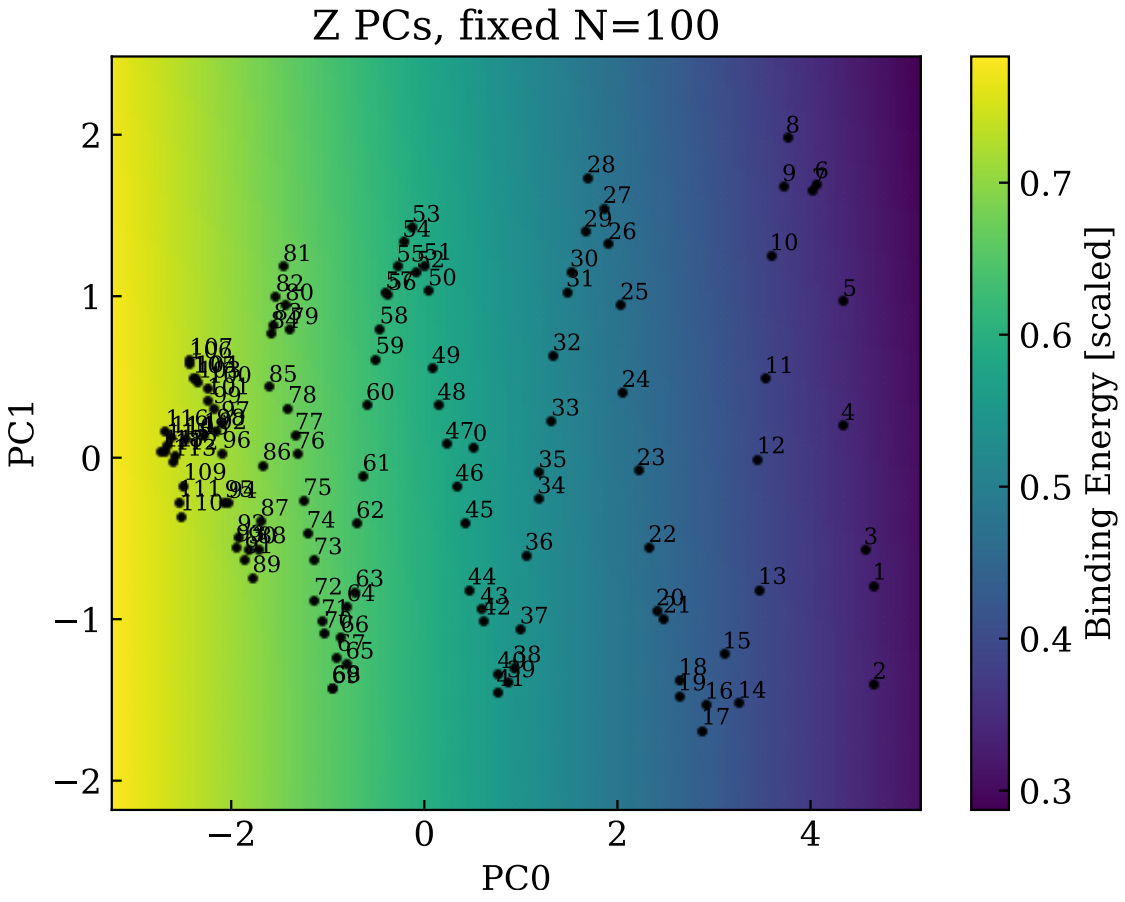


Figure 5.7: (From Kitouni et al. [159]) Projection of proton number (Z) embeddings onto the first two principal components (PCs), superimposed on the neural network's binding energy predictions. The binding energy LST is computed as a function of the first two PCs, while the remaining components are fixed at their mean values. Black dots indicate the positions of the Z embeddings in this space, with the corresponding proton numbers annotated next to each dot. The color scale represents the predicted binding energy values, with brighter hues denoting higher energies.

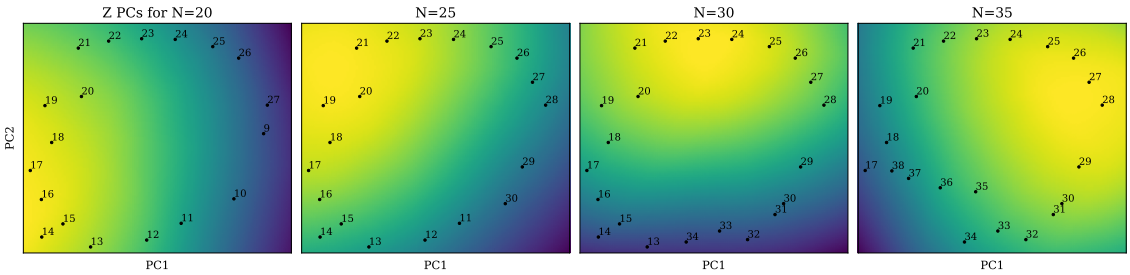


Figure 5.8: (From Kitouni et al. [159]) Z embeddings projected onto principal components 1 and 2 (counting from 0) given multiple fixed neutron numbers. For each N, only Z embeddings are shown for which actual nuclei exist. The background shows the binding energy prediction of the model as a function of PC1 and PC2, where other primary components are fixed to their mean value. Brighter means more E_B.

of the helix. For instance, we note that increasing the pitch (length of the central axis)

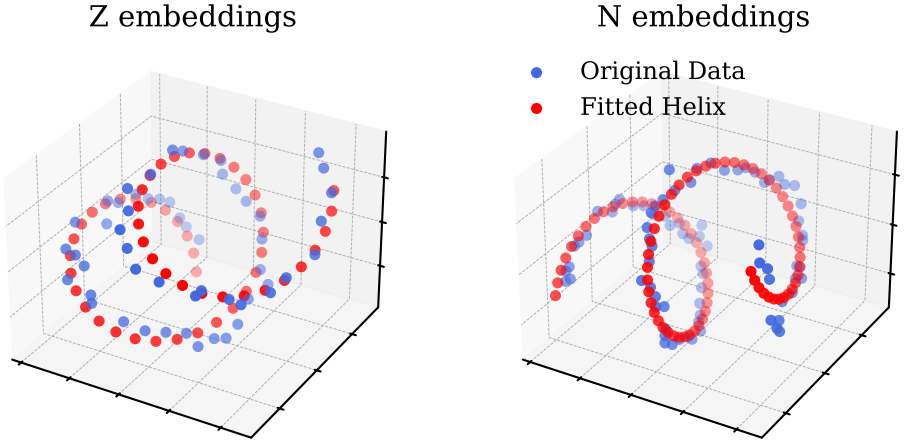


Figure 5.9: (From Kitouni et al. [159]) Fitting a helix to the PC-projected embeddings.

elongates the helix, causing a constant offset in predictions, similar to the *volume term* in the SEMF. Reducing the length has the opposite effect. Increasing the radius "sharpens" the downward arcs in predictions, likely linked to the SEMF's *asymmetry term*, with radius controlling the prefactor. The helix structure provides an interesting geometric explanation of how the model represents the data. In particular, it presents a complete description of the SEMF—itself motivated by geometry (Appendix B.3.2) and basic physics principles—and yields particularly accurate fits, as shown in Appendix B.1.

Figure 5.8 presents a complementary view to Figure 5.7, with the latent space topology displayed across the next two principal components (PC1 and PC2). This perspective is obtained by rotating the viewpoint by 90 degrees out-of-the-page compared to Figure 5.7. For each pane, the neutron number (N) is fixed to a different value, increasing in increments of 5 between adjacent panes. The proton number (Z) embeddings displayed in each pane

are limited to those corresponding to physically existing nuclei, *i.e.*, (Z, N) pairs present in the dataset. The background is produced by evaluating the model by varying PC1 and PC2, keeping all other primary components fixed at their mean. We also tried varying PC0 but, as anticipated, we observed that changes in PC0, which aligns with the helix axis, only influence the absolute values of the model's output. The relative values within each LST "slice" remain stable. Note that, since PC0 and N are fixed, the overarching near-linear trend of binding energy with respect to increasing N and Z does not play a leading role here.

To focus on the local variations, we consider the binding energy relative to the nucleon number A (E_{B}/A) for the following analysis. For each fixed N, there exists a specific Z value that corresponds to the highest E_{B}/A, representing the most stable element for that given N. As Z diverges from this optimal value, the E_{B}/A decreases smoothly. This trend can be observed in Figure 5.3, where for each slice along the N axis, there is a peak in E_{B}/A around a central Z value (and *vice versa* for slices along the Z axis). Consequently, for each N, there should be a continuous strip of Z embeddings, with one embedding marking the highest E_{B}/A value, corresponding to the most stable nucleus for that particular N. Since each N requires such a continuous strip, the entire sequence of Z embeddings should form a continuous structure.

This is where the helix structure, which can be viewed as stacked circles, offers a compact and efficient way of achieving this continuity. By arranging the Z embeddings along a helical path, the model ensures that for each N, there is a smooth progression of Z values, with the most stable element located at the optimal position within the latent space. The helical structure allows for a continuous representation of the binding energy landscape, capturing the local variations and the stability peaks across different N values.[6]

2. Orderedness We hypothesize that ordering numbers in the first few principal components is indicative of generalization and investigate the relationship between"orderedness" in embedding structures and generalization performance (see Appendix B.2.1 for the time evolution of this property). We train models with different train/validation splits (10% to 90% in 10% increments, 3 random seeds each), varying batch size for consistent total optimization steps, and keeping other hyperparameters constant. Given the clear structure observed in the previous section, we experiment with a simple measurement of ordering along the first PC dimension. It reveals a surprising correlation with generalization performance, see Figure 5.10. We define the quantity,

$$\text{orderedness} = \frac{1}{M}\sum_{i=1}^{M-1} \mathbf{1}(\tilde{\mathbf{E}}_0^i < \tilde{\mathbf{E}}_0^{i+1}) \ ,$$

where $\mathbf{1}$ is the indicator function,[7] $\tilde{\mathbf{E}}_0^i$ is the PC0 projection of the N or Z embedding, and M is the total number of embeddings. We will generally use the tilde ($\tilde{\cdot}$) to denote PC-projected vectors. It's important to note that all models fit the training data extremely well, with errors on the order of tens of keV. However, there is no correlation observed between train error and the degree of order.

[6] See Appendix B.6 for another example of continuity in the latent space.

[7] The direction of the order might be reversed.

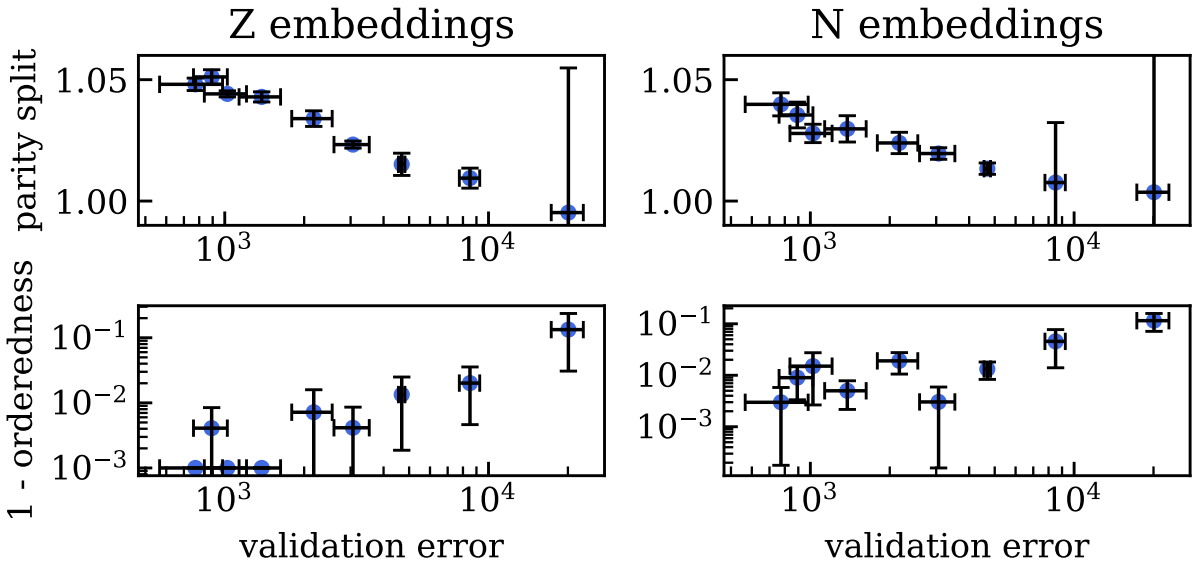


Figure 5.10: (From Kitouni et al. [159]) Parity split R_{P} (top row) and orderedness (bottom row) calculated on N and Z embeddings as a function of validation error. Zero values were clipped to 10^{-3} for visualization. Error bars are standard deviations and each point groups models trained with the same training fraction.

3. Parity In addition to orderedness, we explore another prominent feature in the embedding space: number parity. This feature is immediately apparent in the projection of PC0 and PC2 in Figure 5.6 where even Z embeddings are separated from odd Z embeddings along PC2. To measure the influence of parity on the embeddings, we introduce the following quantity:

$$R_{\mathrm{P}} = \frac{2 \cdot d(\mathrm{even}, \mathrm{odd})}{d(\mathrm{even}, \mathrm{even}) + d(\mathrm{odd}, \mathrm{odd})} \ ,$$

where $d(\cdot, \cdot)$ is the average pairwise L_2-distance between elements in the sets of even/odd N or Z. This quantity is the ratio of the average distance of embeddings of different parity to that of embeddings of the same parity. Figure 5.10 illustrates how R_{P}, calculated on proton embeddings, correlates with validation performance. The clear trend observed suggests that parity is an important indicator of model performance and possibly an important feature of the data.

It turns out that an important feature of nuclear properties is the tendency of nuclear constituents (both protons and neutrons) to form pairs.[8] Numerous characteristics depend on the parity (even/odd) of N and Z. This is evident in the *Pairing* term of the SEMF, which changes sign based on the parity.

[8]This is related to the so-called Pauli Exclusion Principle [188].

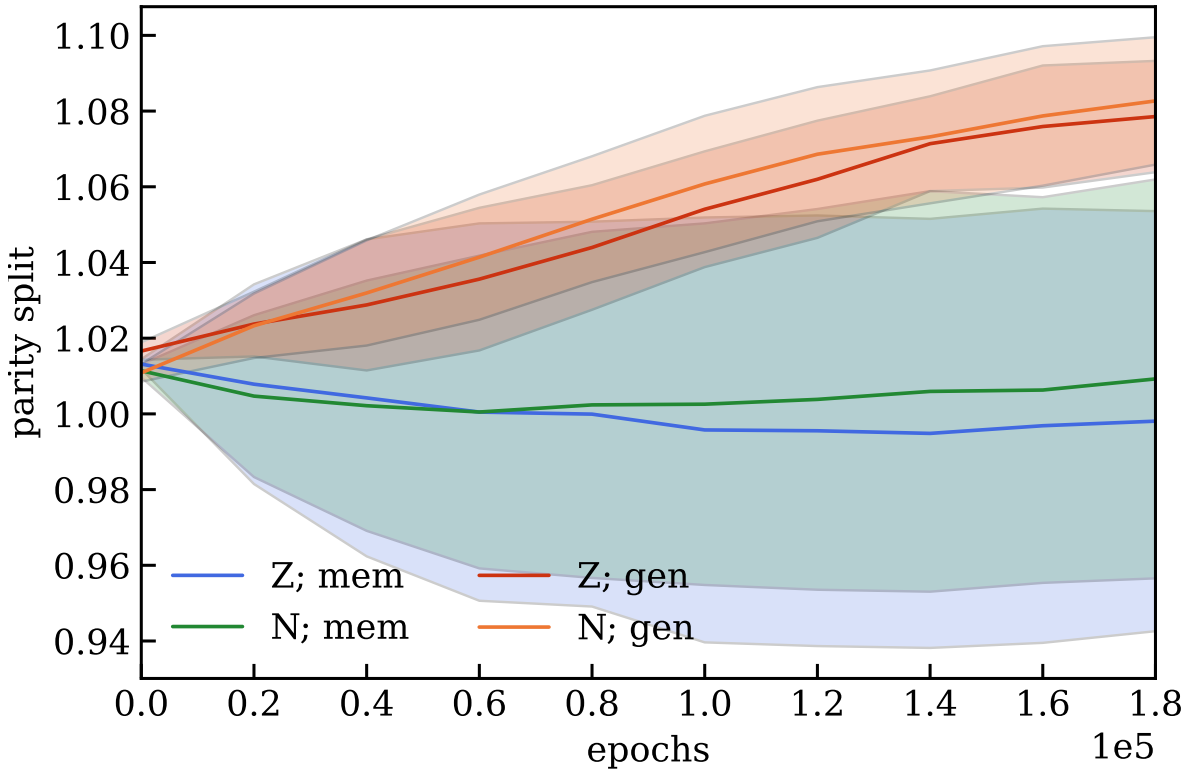


Figure 5.11: Parity split R_{P} as a function of training time for N and Z embeddings for memorizing and generalizing models. The uncertainties are computed over 3 data and initialization seeds.

5.5.2 Hidden Layer Features

In the previous subsection, we explored proton and neutron embeddings to extract valuable information about models that generalize well. We discovered some properties of these models and were able to map them to well-known physics concepts. However, the functional relationship between initial embeddings and the output is often unclear. Now we focus on the activations of the penultimate layer, which does not have this drawback since it maps linearly to the output. We continue to use PCA projections to visualize and analyze these high-dimensional features. As seen in Figure 5.5, we can recover much of a model's performance using just a few of these features. We observe that, similar to those we see in the embeddings, the principal components of the activations exhibit a rich structure, including terms that are smooth and slowly varying, others that have a high-frequency and small-scale, and some that are highly structured. Examples from each category are shown in the top row of Figure 5.12, and a larger collection of PCs can be found in Figure B.9 of the Appendix.

We aim to recover human-derived descriptions of the problem in these latent representations, and we will do so based on a simple matching heuristic. Let $\tilde{\mathbf{x}}_i$ be the i-th vector of the neural network's penultimate layer features (given by the i-th PC dimension) and $\mathbf{y}_j$ be the j-th physical term vector produced by evaluating the term at all values of N and Z (see Appendices B.3.2 and B.3.3 for all terms). We use the cosine similarity, defined as $\text{sim}(\tilde{\mathbf{x}}_i, \mathbf{y}_j) = \tilde{\mathbf{x}}_i \cdot \mathbf{y}_j / ||\tilde{\mathbf{x}}_i|| ||\mathbf{y}_j||$, to compare the two sets of vectors. We find that this heuristic recovers visually compelling matches and show a few examples in Figure 5.12 with the physical terms at the bottom and their matches in neural features at the top. We note the following:

- PC0 shows a strong trend towards higher values increasing Z and N. Since the model predictions are linear combinations of those features, we can deduce that PC0 is primarily responsible for the general upwards trend in the output. Note the striking consistency of that trend with the effect of the PC0 of input embeddings (seen in Figure 5.7) and the number ordering described in the previous section. The bottom left pane of Figure 5.12 shows the dominant volume term of the SEMF, closely matching our feature PC0.
- Unlike PC0, the contribution of PC6 is of smaller scale, characterized by a high-frequency periodicity in both N and Z. Interestingly, we can also match this feature quite distinctly to the pairing term in the SEMF, observing that both are predominantly a function of the parity of N and Z. Note again the close connection to the parity split observed in initial embeddings.
- Lastly, we take a look at PC4. This one stands out due to its obvious structure and the distinctive, staircase pattern. No term in the SEMF predicts this structure. As it turns out, a higher-order correction to the SEMF comes from the nuclear shell theory that predicts the significance of the so-called *magic numbers* in Z and N. The corresponding bottom-right pane in Figure 5.12 shows the predicted contribution from the shell theory with strikingly similar structure as our PC4.

Note the significance of this finding: there is a vast amount of possible ways in which a neural network could decompose the problem, and yet, despite the simple techniques we used to inspect the activations, we were able to recover a range of human-derived concepts. With all of the above, we have (re)discovered the liquid drop model of nuclear physics and found hints of more advanced corrections from the shell model, simply by studying the weights and

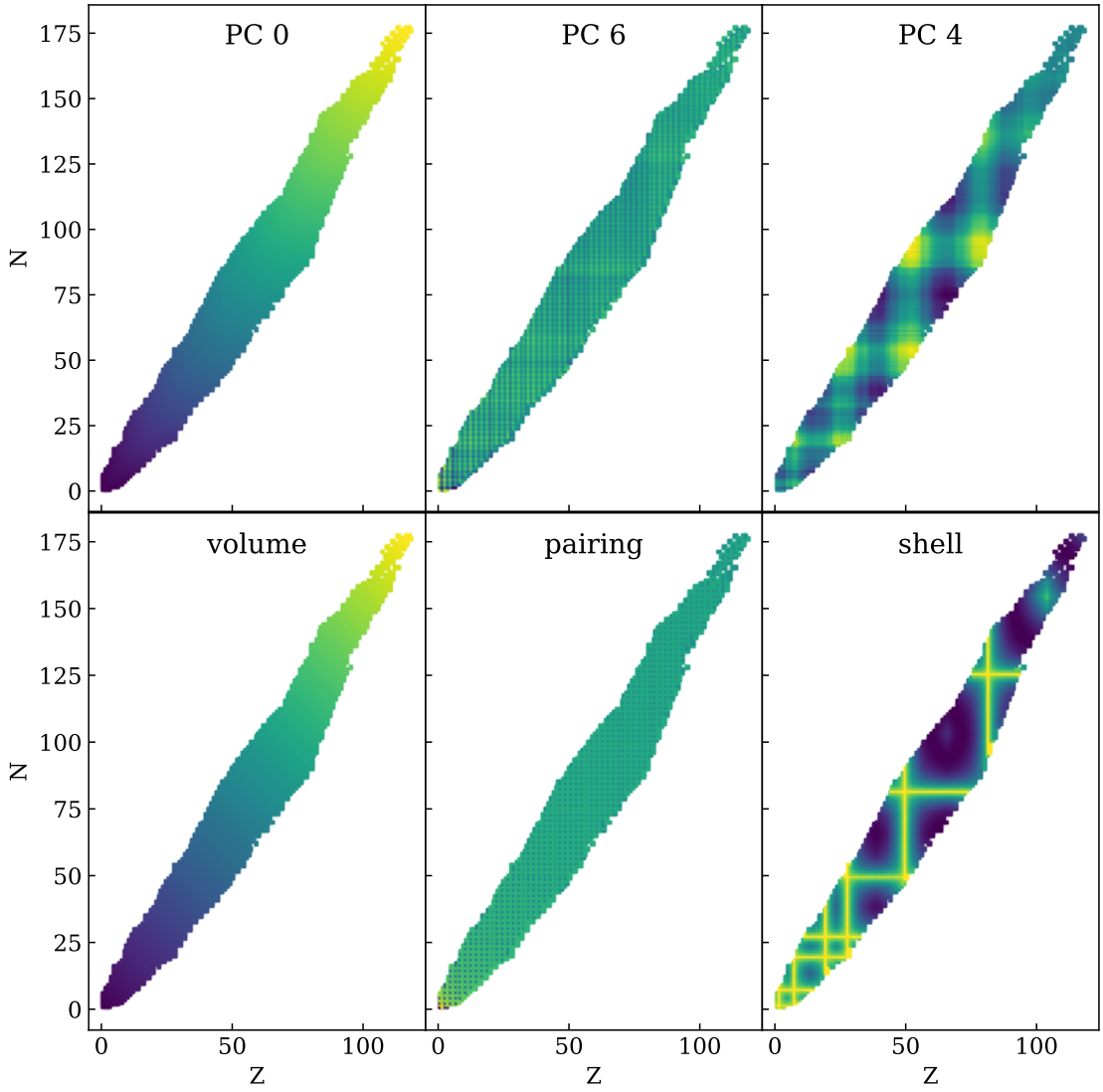


Figure 5.12: (From Kitouni et al. [159]) (Top) penultimate layer PCs and (bottom) physics terms with high similarity.

activations of a neural network trained on nuclear data. We are currently working on further decoding what the machine has learned into human-interpretable knowledge.

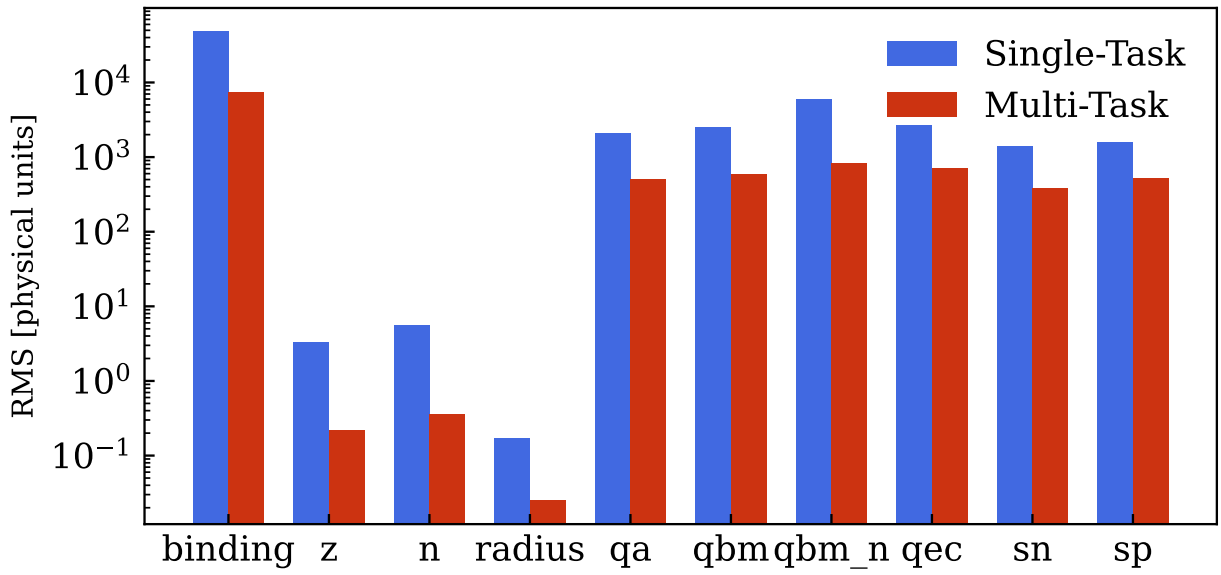


Figure 5.13: (From Kitouni et al. [159]) Test performance over different observables for models trained on a single task versus multiple tasks jointly.

Where Do These Representations Come From? Learning from more diverse datasets should yield higher quality models and lead to improved generalization, provided that the model has enough capacity and nothing goes wrong with the training procedure. Naturally, this is expected to reflect also in the quality of the representations. Figure 5.13 demonstrates that using the same representations to predict a variety of nuclear observables improves the performance on each of them individually. For this demonstration, we perform training runs with one feature at a time, or all at the same time, with 50% of the data held out as a validation set in each setting to gauge the generalization performance. We observe a consistent improvement on all observables when tackling the problem with a multi-task solution, utilizing more data.

But where do the prominent features we observed in the latent representations come from? We systematically compare the representations learned on individual tasks and note that binding energy is primarily responsible for helicity and is never observed elsewhere, parity is most pronounced when training on separation energies, ordering seems to be partially present in many cases, and Z and N do not produce particularly interesting structures (examples in Appendix B.4).

Symbolic Expressions for Discovering New Terms We can also use the latent representations to model what the neural network learned, and thus, extract a new physics model. We use symbolic regression to map to the features of the penultimate layer, and then apply a transformation that aligns to the binding energy. Using this pipeline we recover a predictive

symbolic expression. The new formula achieves a better performance than the SEMF, though is less interpretable. As a baseline, we also regress directly over the task. However, we were not able to recover a performance as good as the one obtained exploiting the neural network features. Though in general, results would depend on the data, the model trained, and the symbolic regressor itself, this result suggests that the model learns to decompose the problem into features that can make it easier to find interpretable symbolic expressions. This is inline with prior work that derives symbolic formulae from neural network features for physical systems ([189]). See Appendix B.7 for details.

5.6 Related Work

As an emerging field, mechanistic interpretability has recently focused on large language models (LLMs) [173], but it is also starting to gain relevance in scientific discovery [190]. Another relevant line of work studies whether models build internal *"world models"* [191]–[193]. Glimpses of more complex understanding have already emerged. For instance, LLMs have constructed (to some extent) knowledge in world geography [194], and meaningful representations of space and time [195], both of which have been studied since Word2Vec [187].

In computer vision, interpretability can take a more direct approach due to the visual nature of the data [171], [196]. Here, mechanistic interpretability was used to gain insights on and improve the effectiveness of convolutional networks [197]. A more microscopic approach to layer level interpretability on vision models was explored in [198].

5.7 Summary & Discussion

In this chapter, we explore the potential of using mechanistic interpretability to extract scientific knowledge from neural networks trained on physics data. We not only investigate *how* models make their predictions, but also *what* insights the model can provide about the data. Our analysis has revealed several findings. First, the learned embeddings of proton and neutron numbers exhibit interpretable structures such as the helix and parity splits, which are indicative of the models' generalization capabilities. These structures mirror known physics concepts like pairing effects, suggesting that the models are capable of learning and employing established scientific knowledge. Second, our inspection of hidden layer activations has uncovered components that resemble terms in established theories: the semi-empirical mass formula and the nuclear shell model. This similarity in both macroscopic trends and microscopic structures suggests that the models are learning physically meaningful representations. Finally, by employing latent space topography,[9] we were able to arrive at a full description of the algorithms used by the model to make accurate binding energy predictions. In particular, we found that the learned embeddings provide a geometric representation of the theoretically well-motivated SEMF. These findings provide a proof-of-concept that neural networks, when trained on scientific data, can learn useful representations that align with

[9]Example code is available here:
https://github.com/samuelperezdi/nuclr-icml

human knowledge. This opens up exciting possibilities for future research on richer data and more complex tasks, which may uncover new scientific insights.

On the Impact of Interpretability in Scientific Discovery

This section presents a brief overview of our vision for an MI-enhanced approach to the scientific endeavor. Throughout the history of science, natural laws have been discovered by domain scientists studying high-dimensional data and realizing that, in some cases, these data can be explained by a simple interpretable picture. These pictures were generated in the minds of the domain scientists, often based on a simplified geometrical model of the system being studied.

We present a new approach to generating interpretable models from scientific data: rather than having domain experts study the high-dimensional data directly, we propose to first determine if a low-rank structure can be found in a machine-learned model representation. If it can, human domain scientists can try and decode this structure into an interpretable model, rather than continuing to work directly with the high-dimensional data.

Here, we chose an example where a human-derived interpretable picture is known to exist—nuclear physics and its famous Shell Model—and find that representation learning (without any physics input), along with the use of PCA, does indeed discover a low-rank geometric structure. After further study, using the Shell Model as a known baseline solution, we see that the machine has learned the Shell Model—though with corrections that lead to more precise predictions than the Nobel Prize-winning human-discovered model. Therefore, the known interpretable human-discovered model is found by the machine and communicated to us, albeit in a different form that still needed decoding by domain experts.

As in the nuclear physics case studied here, most human-discovered interpretable scientific models are only approximately true. In such cases, our approach has the potential to derive corrections to the human-discovered model, represented as deviations in the low-rank structure. We see this with the nuclear data and are working on fully decoding these deviations into interpretable correction terms to the Shell Model.

Such interpretable corrections will have a huge impact on the field of nuclear physics. This is especially true for exotic nuclei far from the stability region, which are impossible to make and study in the lab. Yet, the properties of these nuclei are crucial for understanding nuclear processes in extreme environments, such as neutron stars. This understanding, in turn, enhances our knowledge of how heavy elements were produced in our universe. This is an out-of-distribution (OOD) problem from the ML perspective, hence finding interpretable corrections that can be trusted in the OOD region is crucial.

Most other known interpretable models (in other scientific domains) are also only approximate, and similar corrections could likely be found to improve scientific knowledge in those areas as well. Furthermore, in many scientific domains, humans have not been capable of developing any interpretable theories, even approximate ones, when studying high-dimensional data. Whether our approach could lead to discoveries in such fields is impossible to predict—interpretable models may not exist for some highly non-linear problems—but it is a direction worth pursuing. Hence, one of our goals is to encourage the ML community to work more closely with domain scientists on such problems, which can drive a disproportionate impact across disciplines.

In summary, our work underscores the value of interpretability in scientific exploration. By elucidating how models represent problems, interpretability becomes a powerful tool for scientific discovery. As we continue to develop and refine these techniques, we anticipate that they will play an increasingly important role in advancing human understanding in a wide range of domains.

Chapter 6

Physics for Representation Learning: An Effective Theory of Grokking

This chapter marks the second half of the thesis, where we switch gears from "how can deep learning help physics" to "What can physics do FOR deep learning". We aim to understand *grokking*, a phenomenon where models generalize long after overfitting their training set. We present both a *microscopic* analysis anchored by an effective theory and a *macroscopic* analysis of phase diagrams describing learning performance across hyperparameters. We find that generalization originates from structured representations whose training dynamics and dependence on training set size can be predicted by our effective theory in a toy setting. We observe empirically the presence of four learning phases: *comprehension*, *grokking*, *memorization*, and *confusion*. We find representation learning to occur only in a "Goldilocks zone" (including comprehension and grokking) between memorization and confusion. We find on transformers the grokking phase stays closer to the memorization phase (compared to the comprehension phase), leading to delayed generalization. The Goldilocks phase is reminiscent of "intelligence from starvation" in Darwinian evolution, where resource limitations drive discovery of more efficient solutions. This study not only provides intuitive explanations of the origin of grokking, but also highlights the usefulness of physics-inspired tools, e.g., effective theories and phase diagrams, for understanding deep learning.[1]

6.1 Introduction

Perhaps *the* central challenge of a scientific understanding of deep learning lies in accounting for neural network generalization. Power et al. [1] recently added a new puzzle to the task of understanding generalization with their discovery of *grokking*. Grokking refers to the surprising phenomenon of *delayed generalization* where neural networks, on certain learning problems, generalize long after overfitting their training set. It is a rare albeit striking phenomenon that violates common machine learning intuitions, raising three key puzzles:

[1]This chapter is based on research originally presented in Ref. [175]. The work was conducted in collaboration with Ziming Liu, Niklas Nolte, Eric Michaud, Max Tegmark, and Mike Williams.

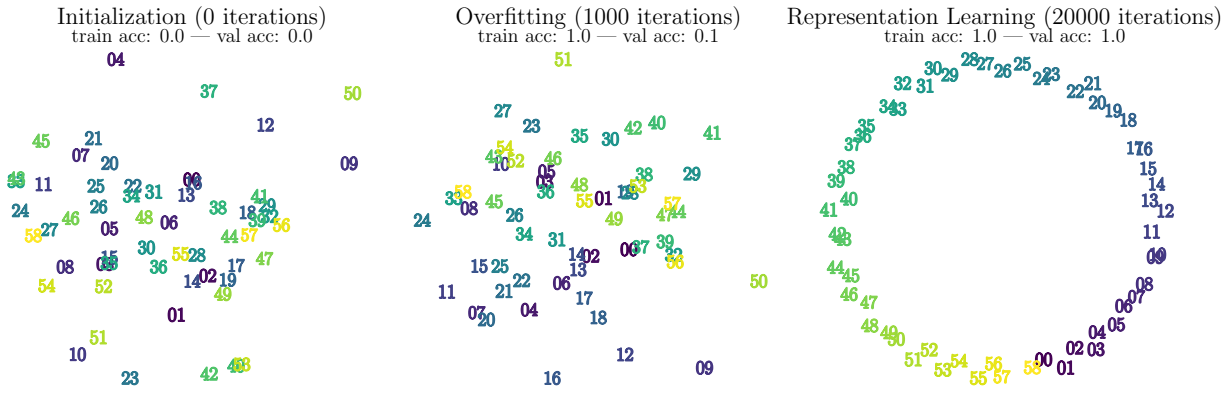


Figure 6.1: Visualization of the first two principal components of the learned input embeddings at different training stages of a transformer learning modular addition. We observe that generalization coincides with the emergence of structure in the embeddings. See Section 6.4.2 for the training details.

Q1 *The origin of generalization*: When trained on the algorithmic datasets where grokking occurs, how do models generalize at all?

Q2 *The critical training size*: Why does the training time needed to "grok" (generalize) diverge as the training set size decreases toward a critical point?

Q3 *Delayed generalization*: Under what conditions does delayed generalization occur?

We provide evidence that representation learning is central to answering each of these questions. Our answers can be summarized as follows:

A1 Generalization can be attributed to learning a good representation of the input embeddings, i.e., a representation that has the appropriate structure for the task and which can be predicted from the theory in Equation (6.3). See Figures 6.1 and 6.2.

A2 The critical training set size corresponds to the least amount of training data that can determine such a representation (which, in some cases, is unique up to linear transformations).

A3 Grokking is a phase between "comprehension" and "memorization" phases and it can be remedied with proper hyperparmeter tuning, as illustrated by the phase diagrams in Figure 6.6.

This chapter is organized as follows: In Section 6.2, we introduce the problem setting and build a simplified toy model. In Section 6.3, we will use an *effective theory* approach, a useful tool from theoretical physics, to shed some light on questions **Q1** and **Q2** and show the relationship between generalization and the learning of structured representations. In Section 6.4, we explain **Q3** by displaying phase diagrams from a grid search of hyperparameters and show how we can "de-delay" generalization by following intuition developed from the phase diagram. We discuss related work in Section 6.5, followed by conclusions in Section 6.6.[2]

[2]Project code can be found at: https://github.com/ejmichaud/grokking-squared

6.2 Problem Setting

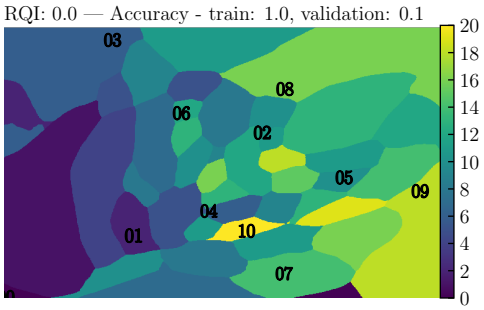
Power et al. [1] observe grokking on a less common task – learning "algorithmic" binary operations. Given some binary operation $\circ$, a network is tasked with learning the map $(a, b) \mapsto c$ where $c = a \circ b$. They use a decoder-only transformer to predict the second to last token in a tokenized equation of the form "<lhs> <op> <rhs> <eq> <**result**> <eos>". Each token is represented as a 256-dimensional embedding vector. The embeddings are learnable and initialized randomly. After the transformer, a final linear layer maps the output to class logits for each token.

Toy Model We primarily study grokking in a simpler toy model, which still retains the key behaviors from the setup of [1]. Although [1] treated this as a classification task, we study both regression (mean-squared error) and classification (cross-entropy). The basic setup is as follows: our model takes as input the symbols a, b and maps them to trainable embedding vectors $\mathbf{E}_a, \mathbf{E}_b \in \mathbb{R}^{d_{\text{in}}}$. It then sums $\mathbf{E}_a, \mathbf{E}_b$ and sends the resulting vector through a "decoder" MLP. The target output vector, denoted $\mathbf{Y}_c \in \mathbb{R}^{d_{\text{out}}}$ is a fixed random vector (regression task) or a one-hot vector (classification task). Our model architecture can therefore be compactly described as $(a, b) \mapsto \text{Dec}(\mathbf{E}_a + \mathbf{E}_b)$, where the embeddings $\mathbf{E}_*$ and the decoder are trainable. Despite its simplicity, this toy model can generalize to all abelian groups (discussed in Appendix C.2). In Sections 6.3 and 6.4.1, we consider only the binary operation of addition. We consider modular addition in Section 6.4.2 to generalize some of our results to a transformer architecture and study general non-abelian operations in Appendix C.8.

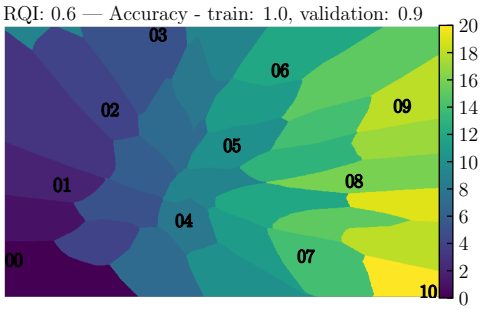
Dataset In our toy setting, we are concerned with learning the addition operation. A data sample corresponding to $i + j$ is denoted as (i, j) for simplicity. If $i, j \in \{0, \ldots, p-1\}$, there are in total $p(p+1)/2$ different samples since we consider $i + j$ and $j + i$ to be the same sample. A dataset D is a set of non-repeating data samples. We denote the full dataset as D_0 and split it into a training dataset D and a validation dataset D', i.e., $D \bigcup D' = D_0$, $D \bigcap D' = \emptyset$. We define *training data fraction* $= |D|/|D_0|$ where $|\cdot|$ denotes the cardinality of the set.

6.3 Why Generalization Occurs: Representations and Dynamics

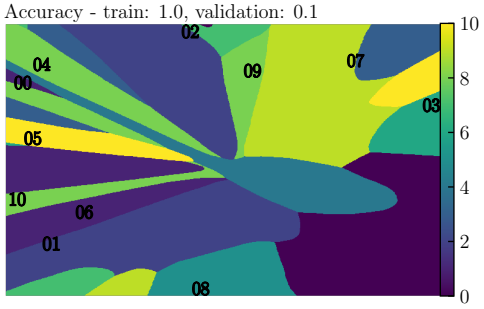
We can see that generalization appears to be linked to the emergence of highly-structured embeddings in Figure 6.2. In particular, Figure 6.2 (a, b) shows parallelograms in toy addition, and (c, d) shows a circle in toy modular addition. We now restrict ourselves to the toy addition setup and formalize a notion of representation quality and show that it predicts the model's performance. We then develop a physics-inspired *effective* theory of learning which can accurately predict the critical training set size and training trajectories of representations. The concept of an effective theory in physics is similar to model reduction in computational methods in that it aims to describe complex phenomena with simple yet intuitive pictures. In our effective theory, we will model the dynamics of representation learning not as gradient descent of the true task loss but rather a simpler effective loss function ℓ_{eff} which depends only on the representations in embedding space and not on the decoder.



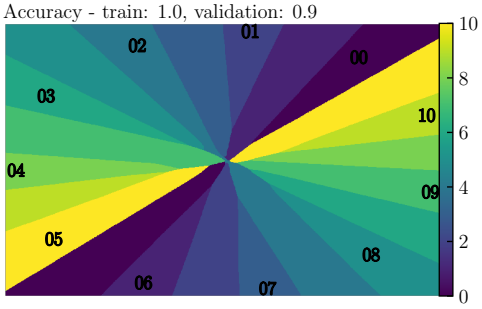
(a) Memorization in toy addition



(b) Generalization in toy addition



(c) Memorization in toy modular addition



(d) Generalization in toy modular addition

Figure 6.2: (From Ref. [175]) Visualization of the learned set of embeddings ($p = 11$) and the decoder function associated with it for the case of 2D embeddings. Axes refer to each dimension of the learned embeddings. The decoder is evaluated on a grid of points in embedding-space and the color at each point represents the highest probability class. For visualization purposes, the decoder is trained on inputs of the form $(\mathbf{E}_i + \mathbf{E}_j)/2$. One can read off the output of the decoder when fed the operation $i \circ j$ from this figure simply by taking the midpoint between the respective embeddings of i and j.

6.3.1 Representation quality predicts generalization for the toy model

A rigorous definition for *structure* in the learned representation is necessary. We propose the following definition,

Definition 1. (i, j, m, n) *is a* δ-***parallelogram*** *in the representation* $\mathbf{R} \equiv [\mathbf{E}_0, \cdots, \mathbf{E}_{p-1}]$ *if*

$$|(\mathbf{E}_i + \mathbf{E}_j) - (\mathbf{E}_m + \mathbf{E}_n)| \leq \delta.$$

In the following derivations, we can take δ, which is a small threshold to tolerate numerical errors, to be zero.

Proposition 1. *When the training loss is zero, any parallelogram* (i, j, m, n) *in representation* $\mathbf{R}$ *satisfies* $i + j = m + n$.

Proof. Suppose that this is not the case, i.e., suppose $\mathbf{E}_i + \mathbf{E}_j = \mathbf{E}_m + \mathbf{E}_n$ but $i + j \neq m + n$, then $\mathbf{Y}_{i+j} = \text{Dec}(\mathbf{E}_i + \mathbf{E}_j) = \text{Dec}(\mathbf{E}_m + \mathbf{E}_n) = \mathbf{Y}_{m+n}$ where the first and last equalities come from the zero training loss assumption. However, since $i + j \neq m + n$, we have $\mathbf{Y}_{i+j} \neq \mathbf{Y}_{n+m}$ (almost surely in the regression task), a contradiction. □

It is convenient to define the permissible parallelogram set associated with a training dataset D ("permissible" means consistent with 100% training accuracy) as

$$P_0(D) = \{(i, j, m, n) | (i, j) \in D,\ (m, n) \in D,\ i + j = m + n\}. \tag{6.1}$$

For simplicity, we denote $P_0 \equiv P_0(D_0)$. Given a representation $\mathbf{R}$, we can check how many permissible parallelograms actually exist in $\mathbf{R}$ within error δ, so we define the parallelogram set corresponding to $\mathbf{R}$ as

$$P(\mathbf{R}, \delta) = \{(i, j, m, n) | (i, j, m, n) \in P_0, |(\mathbf{E}_i + \mathbf{E}_j) - (\mathbf{E}_m + \mathbf{E}_n)| \leq \delta\}. \tag{6.2}$$

For brevity we will write $P(\mathbf{R})$, suppressing the dependence on δ. We define the representation quality index (RQI) as

$$\text{RQI}(\mathbf{R}) = \frac{|P(\mathbf{R})|}{|P_0|} \in [0, 1]. \tag{6.3}$$

We will use the term *linear representation* or *linear structure* to refer to a representation whose embeddings are of the form $\mathbf{E}_k = \mathbf{a} + k\mathbf{b}$ ($k = 0, \cdots, p-1; \mathbf{a}, \mathbf{b} \in \mathbb{R}^{d_{\text{in}}}$). A linear representation has RQI = 1, while a random representation (sampled from, say, a normal dstribution) has RQI = 0 with high probability.

Quantitatively, we denote the "predicted accuracy" $\widehat{\text{Acc}}$ as the accuracy achievable on the whole dataset given the representation $\mathbf{R}$ (see Appendix C.4 for the full details). In Figure 6.3, we see that the predicted $\widehat{\text{Acc}}$ aligns well with the true accuracy Acc, establishing good evidence that structured representation of input embeddings leads to generalization. We use an example to illustrate the origin of generalization here. In the setup of Figure 6.2 (b), suppose the decoder can achieve zero training loss and $\mathbf{E}_6 + \mathbf{E}_8$ is a training sample hence $\text{Dec}(\mathbf{E}_6 + \mathbf{E}_8) = \mathbf{Y}_{14}$. At validation time, the decoder is tasked with predicting a validation sample $\mathbf{E}_5 + \mathbf{E}_9$. Since $(5, 9, 6, 8)$ forms a parallelogram such that $\mathbf{E}_5 + \mathbf{E}_9 = \mathbf{E}_6 + \mathbf{E}_8$, the decoder can predict the validation sample correctly because $\text{Dec}(\mathbf{E}_5 + \mathbf{E}_9) = \text{Dec}(\mathbf{E}_6 + \mathbf{E}_8) = \mathbf{Y}_{14}$.

6.3.2 The dynamics of embedding vectors

Suppose that we have an ideal model $\mathcal{M}^* = (\text{Dec}^*, \mathbf{R}^*)$ such that:[3]

- (1) $\mathcal{M}^*$ can achieve zero training loss;
- (2) $\mathcal{M}^*$ has an injective decoder, i.e., $\text{Dec}^*(\mathbf{x}_1) \neq \text{Dec}^*(\mathbf{x}_2)$ for any $\mathbf{x}_1 \neq \mathbf{x}_2$.

Then Proposition 2 provides a mechanism for the formation of parallelograms.

Proposition 2. *If a training set D contains two samples (i, j) and (m, n) with $i + j = m + n$, then $\mathcal{M}^*$ learns a representation $\mathbf{R}^*$ such that $\mathbf{E}_i + \mathbf{E}_j = \mathbf{E}_m + \mathbf{E}_n$, i.e., (i, j, m, n) forms a parallelogram.*

Proof. Due to the zero training loss assumption, we have $\text{Dec}^*(\mathbf{E}_i + \mathbf{E}_j) = \mathbf{Y}_{i+j} = \mathbf{Y}_{m+n} = \text{Dec}^*(\mathbf{E}_m + \mathbf{E}_n)$. Then the injectivity of Dec^* implies $\mathbf{E}_i + \mathbf{E}_j = \mathbf{E}_m + \mathbf{E}_n$. □

[3] One can verify a posteriori if a trained model $\mathcal{M}$ is close to being an ideal model $\mathcal{M}^*$. Please refer to Appendix C.5 for details.

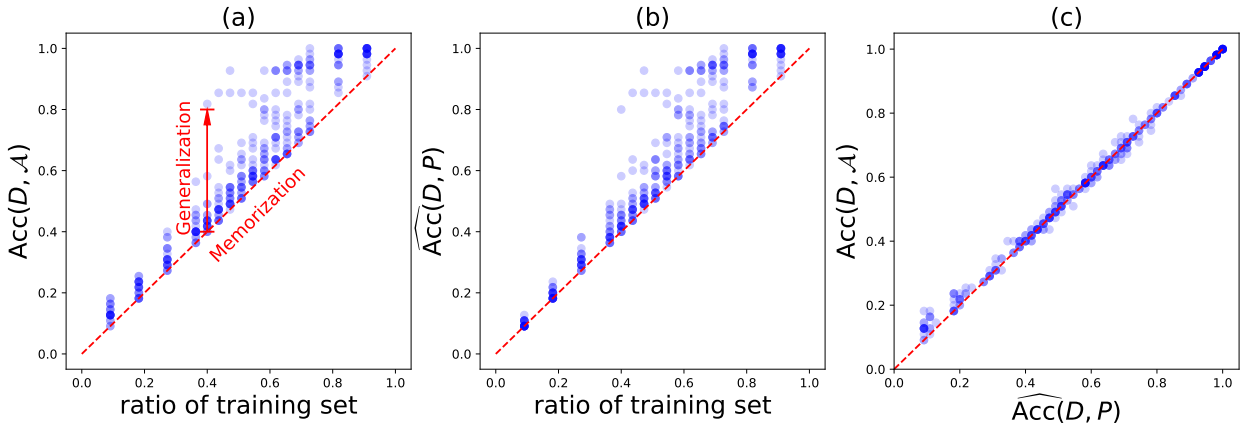


Figure 6.3: (From Ref. [175]) We compute accuracy (of the full dataset) either measured empirically Acc, or predicted from the representation of the embeddings $\widehat{\text{Acc}}$. These two accuracies as a function of training data fraction are plotted in (a)(b), and their agreement is shown in (c).

The dynamics of the trained embedding vectors are determined by various factors interacting in complex ways, for instance: the details of the decoder architecture, the optimizer hyperparameters, and the various kinds of implicit regularization induced by the training procedure. We will see that the dynamics of normalized quantities, namely, the normalized embeddings at time t, defined as $\tilde{\mathbf{E}}_k^{(t)} = \frac{\mathbf{E}_k^{(t)} - \mu_t}{\sigma_t}$, where $\mu_t = \frac{1}{p}\sum_k \mathbf{E}_k^{(t)}$ and $\sigma_t = \frac{1}{p}\sum_k |\mathbf{E}_k^{(t)} - \mu_t|^2$, can be qualitatively described by a simple effective loss (in the physics effective theory sense). We will assume that the normalized embedding vectors obey a gradient flow for an effective loss function of the form

$$\frac{d\tilde{\mathbf{E}}_i}{dt} = -\frac{\partial \ell_{\text{eff}}}{\partial \tilde{\mathbf{E}}_i}, \tag{6.4}$$

$$\ell_{\text{eff}} = \frac{\ell_0}{Z_0}, \quad \ell_0 \equiv \sum_{(i,j,m,n)\in P_0(D)} |\tilde{\mathbf{E}}_i + \tilde{\mathbf{E}}_j - \tilde{\mathbf{E}}_m - \tilde{\mathbf{E}}_n|^2 / |P_0(D)|, \quad Z_0 \equiv \sum_k |\tilde{\mathbf{E}}_k|^2, \tag{6.5}$$

where $|\cdot|$ denotes Euclidean vector norm. Note that the embeddings do not collapse to the trivial solution $\mathbf{E}_0 = \cdots = \mathbf{E}_{p-1} = 0$ unless initialized as such, because two conserved quantities exist, as proven in Appendix C.6:

$$\mathbf{C} = \sum_k \mathbf{E}_k, \quad Z_0 = \sum_k |\mathbf{E}_k|^2. \tag{6.6}$$

We shall now use the effective dynamics to explain empirical observations such as the existence of a critical training set size for generalization.

Degeneracy of ground states (loss optima) We define ground states as those representations satisfying $\ell_{\text{eff}} = 0$, which requires the following linear equations to hold:

$$A(P) = \{\mathbf{E}_i + \mathbf{E}_j = \mathbf{E}_m + \mathbf{E}_n | (i,j,m,n) \in P\}. \tag{6.7}$$

Since each embedding dimension obeys the same set of linear equations, we will assume, without loss of generality, that $d_{\text{in}} = 1$. The dimension of the null space of $A(P)$, denoted as

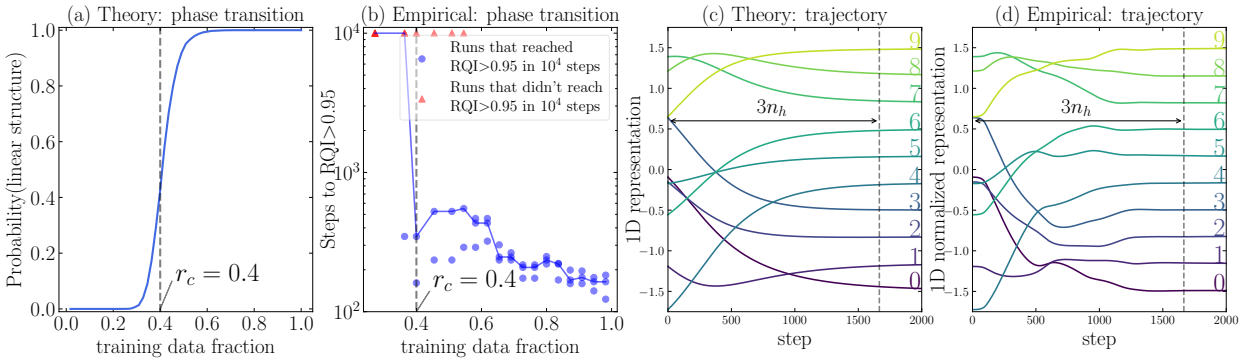


Figure 6.4: (From Ref. [175]) (a) The effective theory predicts a phase transition in the probability of obtaining a linear representation around $r_c = 0.4$. (b) Empirical results display a phase transition of RQI around $r_c = 0.4$, in agreement with the theory (the blue line shows the median of multiple random seeds). The evolution of 1D representations predicted by the effective theory or obtained from neural network training (shown in (c) and (d) respectively) agree creditably well.

n_0, is the number of degrees of freedom of the ground states. Given a set of parallelograms implied by a training dataset D, the nullity of $A(P(D))$ could be obtained by computing the singular values $0 \leq \sigma_1 \leq \cdots \leq \sigma_p$. We always have $n_0 \geq 2$, i.e., $\sigma_1 = \sigma_2 = 0$ because the nullity of $A(P_0)$, the set of linear equations given by all possible parallelograms, is $\text{Nullity}(A(P_0)) = 2$ which can be attributed to two degrees of freedom (translation and scaling). If $n_0 = 2$, the representation is unique up to translations and scaling factors, and the embeddings have the form $\mathbf{E}_k = \mathbf{a} + k\mathbf{b}$. Otherwise, when $n_0 > 2$, the representation is not constrained enough such that all the embeddings lie on a line.

We present theoretical predictions alongside empirical results for addition ($p = 10$) in Figure 6.4. As shown in Figure 6.4 (a), our effective theory predicts that the probability that the training set implies a unique linear structure (which would result in perfect generalization) depends on the training data fraction and has a phase transition around $r_c = 0.4$. Empirical results from training different models are shown in Figure 6.4 (b). The number of steps to reach RQI > 0.95 is seen to have a phase transition at $r_c = 0.4$, agreeing with the proposed effective theory and with the empirical findings in [1].

Time towards the linear structure We define the Hessian matrix of ℓ_0 as

$$\mathbf{H}_{ij} = \frac{1}{Z_0} \frac{\partial^2 \ell_0}{\partial \mathbf{E}_i \partial \mathbf{E}_j}, \tag{6.8}$$

Note that $\ell_{\text{eff}} = \frac{1}{2}\mathbf{R}^T\mathbf{H}\mathbf{R}$, $\mathbf{R} = [\mathbf{E}_0, \mathbf{E}_1, \cdots, \mathbf{E}_{p-1}]$, so the gradient descent is linear, i.e.,

$$\frac{d\mathbf{R}}{dt} = -\mathbf{H}\mathbf{R}. \tag{6.9}$$

If $\mathbf{H}$ has eigenvalues $\lambda_i = \sigma_i^2$ (sorted in increasing order) and eigenvectors $\bar{\mathbf{v}}_i$, and we have the initial condition $\mathbf{R}(t = 0) = \sum_i a_i \bar{\mathbf{v}}_i$, then we have $\mathbf{R}(t) = \sum_i a_i \bar{\mathbf{v}}_i e^{-\lambda_i t}$. The first two eigenvalues vanish and $t_h = 1/\lambda_3$ determines the timescale for the slowest component

to decrease by a factor of e. We call λ_3 the *grokking rate*. When the step size is η, the corresponding number of steps is $n_h = t_h/\eta = 1/(\lambda_3\eta)$.

We verify the above analysis with empirical results. Figure 6.4 (c)(d) show the trajectories obtained from the effective theory and from neural network training, respectively. The 1D neural representation in Figure 6.4 (d) are manually normalized to zero mean and unit variance. The two trajectories agree qualitatively, and it takes about $3n_h$ steps for two trajectories to converge to the linear structure. The quantitative differences might be due to the absence of the decoder in the effective theory, which assumes the decoder to take infinitesimal step sizes.

Dependence of grokking on data size Note that ℓ_{eff} involves averaging over parallelograms in the training set, it is dependent on training data size, so is λ_3. In Figure 6.5 (a), we plot the dependence of λ_3 on training data fraction. There are many datasets with the same data size, so λ_3 is a probabilistic function of data size.

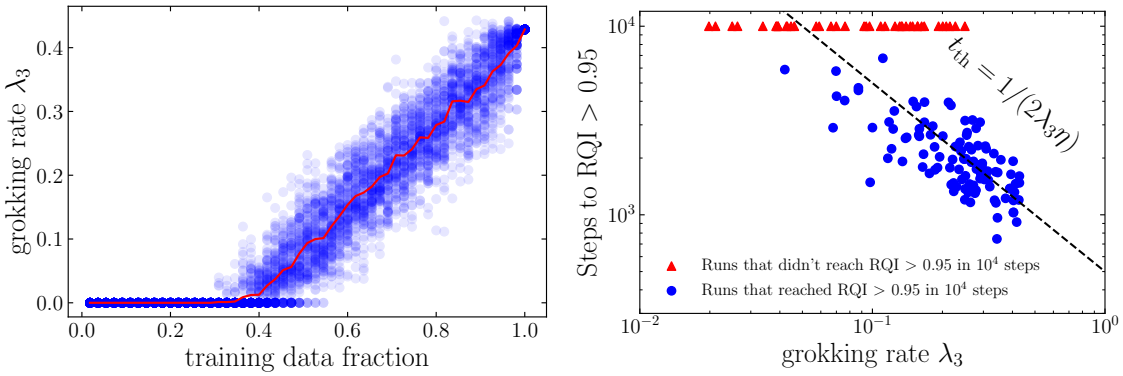
Two insights on grokking can be extracted from this plot: (i) When the data fraction is below some threshold (around 0.4), λ_3 is zero with high probability, corresponding to no generalization. This again verifies our critical point in Figure 6.4. (ii) When data size is above the threshold, λ_3 (on average) is an increasing function of data size. This implies that grokking time $t \sim 1/\lambda_3$ decreases as training data size becomes larger, an important observation from [1].

To verify our effective theory, we compare the grokking steps obtained from real neural network training (defined as steps to RQI > 0.95), and those predicted by our theory $t_{\text{th}} \sim \frac{1}{\lambda_3\eta}$ (η is the embedding learning rate), shown in Figure 6.5 (b). The theory agrees qualitatively with neural networks, showing the trend of decreasing grokking steps as increasing data size. The quantitative differences might be explained as the gap between our effective loss and actual loss.

Limitations of the effective theory While our theory defines an effective loss based on the Euclidean distance between embeddings $\mathbf{E}_i + \mathbf{E}_j$ and $\mathbf{E}_n + \mathbf{E}_m$, one could imagine generalizing the theory to define a broader notion of parallogram given by some other metric on the representation space. For instance, if we have a decoder like in Figure 6.2 (d) then the distance between distinct representations within the same "pizza slice" is low, meaning that representations arranged not in parallelograms w.r.t. the Euclidean metric may be parallelograms with respect to the metric defined by the decoder.

6.4 Delayed Generalization: A Phase Diagram

So far, we have (1) observed empirically that generalization on algorithmic datasets corresponds with the emergence of well-structured representations, (2) defined a notion of representation quality in a toy setting and shown that it predicts generalization, and (3) developed an effective theory to describe the learning dynamics of the representations in the same toy setting. We now study how optimizer hyperparameters affect high-level learning performance. In particular, we develop phase diagrams for how learning performance depends on the representation learning rate, decoder learning rate and the decoder weight decay. These parameters are of interest since they most explicitly regulate a kind of *competition* between the encoder and decoder, as we elaborate below.



(a) (b)

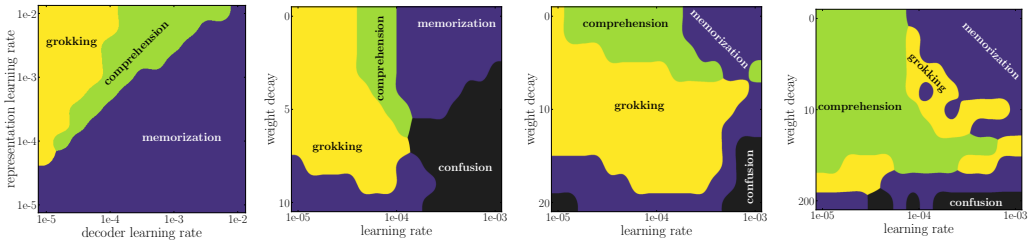
Figure 6.5: (From Ref. [175]) Effective theory explains the dependence of grokking time on data size, for the addition task. (a) Dependence of λ_3 on training data fraction. Above the critical data fraction (around 0.4), as data size becomes larger, λ_3 increases hence grokking time $t \sim 1/\lambda_3$ (predicted by our effective theory) decreases. (b) Comparing grokking steps (defined as RQI > 0.95) predicted by the effective theory with real neural network results. $\eta = 10^{-3}$ is the learning rate of the embeddings.

6.4.1 Phase diagram of a toy model

Training details We update the representation and the decoder with different optimizers. For the 1D embeddings, we use the Adam optimizer with learning rate $[10^{-5}, 10^{-2}]$ and zero weight decay. For the decoder, we use an AdamW optimizer with the learning rate in $[10^{-5}, 10^{-2}]$ and the weight decay in $[0, 10]$ (regression) or $[0, 20]$ (classification). For training/validation spliting, we choose 45/10 for non-modular addition ($p = 10$) and 24/12 for the permutation group S_3. We hard-code addition or matrix multiplication (details in Appendix C.8) in the decoder for the addition group and the permutation group, respectively.

For each choice of learning rate and weight decay, we compute the number of steps to reach high (90%) training/validation accuracy. The 2D plane is split into four phases: *comprehension*, *grokking*, *memorization* and *confusion*, defined in Table C.1 in Appendix C.1. Both comprehension and grokking are able to generalize (in the "Goldilocks zone"), although the grokking phase has delayed generalization. Memorization is also called overfitting, and confusion means failure to even memorize training data. Figure 6.6 shows the phase diagrams for the addition group and the permutation group. They display quite rich phenomena.

Competition between representation learning and decoder overfitting In the regression setup of the addition dataset, we show how the competition between representation learning and decoder learning (which depend on both learning rate and weight decay, among other things) lead to different learning phases in Figure 6.6 (a). As expected, a fast decoder coupled with slow representation learning (bottom right) lead to memorization. In the opposite extreme, although an extremely slow decoder coupled with fast representation learning (top left) will generalize in the end, the generalization time is long due to the inefficient decoder training. The ideal phase (comprehension) requires representation learning to be faster, but not too much, than the decoder.



(a) Addition, regression (b) Addition, regression (c) Addition, classification (d) Permutation, regression

Figure 6.6: (From Ref. [175]) Phase diagrams of learning for the addition group and the permutation group. (a) shows the competition between representation and decoder. (b)(c)(d): each phase diagram contains four phases: comprehension, grokking, memorization and confusion, defined in Table C.1. In (b)(c)(d), grokking is sandwiched between comprehension and memorization.

Drawing from an analogy to physical systems, one can think of embedding vectors as a group of particles. In our effective theory from Section 6.3.2, the dynamics of the particles are described *only* by their relative positions, in that sense, structure forms mainly due to inter-particle interactions (in reality, these interactions are mediated by the decoder and the loss). The decoder plays the role of an environment exerting external forces on the embeddings. If the magnitude of the external forces are small/large one can expect better/worse representations.

Universality of phase diagrams We fix the embedding learning rate to be 10^{-3} and sweep instead decoder weight decay in Figure 6.6 (b)(c)(d). The phase diagrams correspond to addition regression (b), addition classification (c) and permutation regression (d), respectively. Common phenomena emerge from these different tasks: (i) they all include four phases; (ii) The top right corner (a fast and capable decoder) is the memorization phase; (iii) the bottom right corner (a fast and simple decoder) is the confusion phase; (iv) grokking is sandwiched between comprehension and memorization, which seems to imply that it is an undesirable phase that stems from improperly tuned hyperparameters.

6.4.2 Beyond the toy model

We conjecture that many of the principles which we saw dictate the training dynamics in the toy model also apply more generally. Below, we will see how our framework generalizes to transformer architectures for the task of addition modulo p, a minimal reproducible example of the original grokking paper [1].

We first encode $p = 53$ integers into 256D learnable embeddings, then pass two integers to a decoder-only transformer architecture. For simplicity, we do not encode the operation symbols here. The outputs from the last layer are concatenated and passed to a linear layer for classification. Training both the encoder and the decoder with the same optimizer (i.e., with the same hyperparameters) leads to the grokking phenomenon. Generalization appears much earlier once we lower the effective decoder capacity with weight decay (full phase diagram in Figure 6.7).

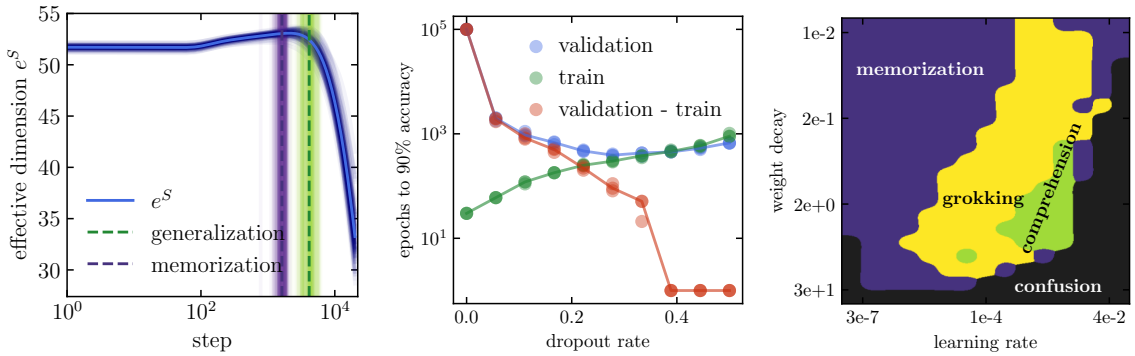


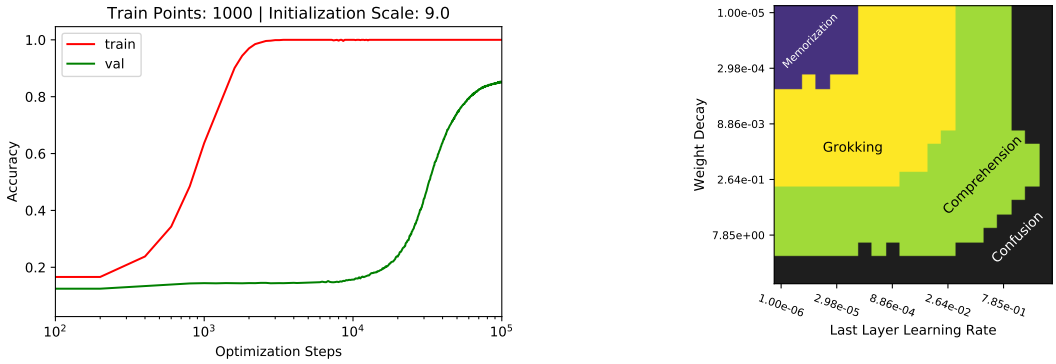
Figure 6.7: (From Ref. [175]) Left: Evolution of the effective dimension of the embeddings (defined as the exponential of the entropy) during training and evaluated over 100 seeds. Center: Effect of dropout on speeding up generalization. Right: Phase diagram of the transformer architecture. A scan is performed over the weight decay and learning rate of the decoder while the learning rate of the embeddings is kept fixed at 10^{-3} (with zero weight decay).

Early on, the model is able to perfectly fit the training set while having no generalization. We study the embeddings at different training times and find that neither PCA (shown in Figure 6.1) nor t-SNE (not shown here) reveal any structure. Eventually, validation accuracy starts to increase, and perfect generalization coincides with the PCA projecting the embeddings into a circle in 2D. Of course, no choice of dimensionality reduction is guaranteed to find any structure, and thus, it is challenging to show explicitly that generalization only occurs when a structure exists. Nevertheless, the fact that, when coupled with the implicit regularization of the optimizer for sparse solutions, such a clear structure appears in a simple PCA so quickly at generalization time suggests that our analysis in the toy setting is applicable here as well. This is also seen in the evolution of the entropy of the explained variance ratio in the PCA of the embeddings (defined as $S = -\sum_i \sigma_i \log \sigma_i$ where σ_i is the fractional variance explained by the ith principal component). As seen in Figure 6.7, the entropy increases up to generalization time then decreases drastically afterwards which would be consistent with the conjecture that generalization occurs when a low-dimensional structure is discovered. The decoder then primarily relies on the information in this low-dimensional manifold and essentially "prunes" the rest of the high-dimensional embedding space. Another interesting insight appears when we project the embeddings at initialization onto the principal axes at the end of training. Some of the structure required for generalization exists before training hinting at a connection with the Lottery Ticket Hypothesis. See Appendix C.11 for more details.

In Figure 6.7 (right), we show a comparable phase diagram to Figure 6.6 evaluated now in the transformer setting. Note that, as opposed to the setting in [1], weight decay has only been applied to the decoder and not to the embedding layer. Contrary to the toy model, a certain amount of weight decay proves beneficial to generalization and speeds it up significantly. We conjecture that this difference comes from the different embedding dimensions. With a highly over-parameterized setting, a non-zero weight decay gives a crucial incentive to reduce complexity in the decoder and help generalize in fewer steps. This is subject to

further investigation. We also explore the effect of dropout layers in the decoder blocks of the transformer. With a significant dropout rate, the generalization time can be brought down to under 10^3 steps and the grokking phenomenon vanishes completely. The overall trend suggests that constraining the decoder with the same tools used to avoid overfitting reduces generalization time and can avoid the grokking phenomenon. This is also observed in an image classification task where we were able to induce grokking. See Appendix C.10 for more details.

6.4.3 Grokking Experiment on MNIST



(a) (b)

Figure 6.8: (From Ref. [175]) Left: Training curves for a run on MNIST, in the setting where we observe grokking. Right: Phase diagram with the four phases of learning dynamics on MNIST.

We now demonstrate, for the first time, that grokking (significantly delayed generalization) is a more general phenomenon in machine learning that can occur not only on algorithmic datasets, but also on mainstream benchmark datasets. In particular, we exhibit grokking on MNIST in Figure 6.8 and demonstrate that we can control grokking by varying optimization hyperparameters. More details on the experimental setup are in Appendix C.10.

6.5 Related work

Relatively few works have analyzed the phenomenon of grokking. [199] describe the circuit that transformers use to perform modular addition, track its formation over training, and broadly suggest that grokking is related to the phenomenon of "phase changes" in neural network training. [200], [201] provided earlier speculative, informal conjectures on grokking [200], [201]. Our work is related to the following broad research directions:

Learning mathematical structures [202] trains a neural network to learn arithmetic operation from pictures of digits, but they do not observe grokking due to their abundant training data. Beyond arithmetic relations, machine learning has been applied to learn other mathematical structures, including geometry [203], knot theory [204] and group theory [205].

Double descent Grokking is somewhat reminiscent of the phenomena of "epoch-wise" *double descent* [206], where generalization can improve after a period of overfitting. [207] find that regularization can mitigate double descent, similar perhaps to how weight decay influences grokking.

Representation learning Representation learning lies at the core of machine learning [160], [208]–[210]. Representation quality is usually measured by (perhaps vague) semantic meanings or performance on downstream tasks. In our study, the simplicity of arithmetic datasets allows us to define representation quality and study evolution of representations in a quantitative way.

Physics of learning Physics-inspired tools have proved to be useful in understanding deep learning from a theoretical perspective. These tools include effective theories [211], [212], conservation laws [213] and free energy principle [214]. In addition, statistical physics has been identified as a powerful tool in studying generalization in neural networks [215]–[218]. Our work connects a low-level understanding of models with their high-level performance. In a recent work, researchers at Anthropic [219], connect a sudden decrease in loss during training with the emergence of *induction heads* within their models. They analogize their work to *statistical physics*, since it bridges a "microscopic", mechanistic understanding of networks with "macroscopic" facts about overall model performance.

6.6 Summary & Discussion

We have shown how, in both toy models and general settings, that representation enables generalization when it reflects structure in the data. We developed an effective theory of representation learning dynamics (in a toy setting) which predicts the critical dependence of learning on the training data fraction. We then presented four learning phases (comprehension, grokking, memorization and confusion) which depend on the decoder capacity and learning speed (given by, among other things, learning rate and weight decay) in decoder-only architectures. While we have mostly focused on a toy model, we find preliminary evidence that our results generalize to the setting of [1].

Our work can be viewed as a step towards a *statistical physics of deep learning*, connecting the "microphysics" of low-level network dynamics with the "thermodynamics" of high-level model behavior. We view the application of theoretical tools from physics, such as effective theories [220], to be a rich area for further work. Subsequent work of similar flavor includes Zhong et al. [178] and Liu et al. [221], which both include physics-inspired approaches to grokking "phenomenology". The broader impact of such work, if successful, could be to make models more transparent and predictable [219], [222], [223], crucial to the task of ensuring the safety of advanced AI systems.

Chapter 7

Physics for Representation Learning: Diffusion Models for Reasoning

In this chapter, we take a deep dive into language modeling and reasoning. The main algorithm described below is based on the pioneering work of physicists on Deep Unsupervised Learning using Nonequilibrium Thermodynamics [224], also known as *diffusion models*. There are various introductory materials on the topic, so I will not reproduce them here. Instead, we will use the basic observation that next-token prediction is a special case of a more general any-to-any loss to re-derive a discrete diffusion objective similar to that of Refs. [225], [226] and study its various benefits for language modeling.[1] We start from a fundamental failure model of current state-of-the-art left-to-right autoregressive models and find that it can be solved with a diffusion-based training paradigm. Specifically, today's best language models still struggle with *hallucinations*: factually incorrect generations, which impede their ability to reliably retrieve information seen during training. The *reversal curse*, where models cannot recall information when probed in a different order than was encountered during training, exemplifies this in information retrieval. We reframe the reversal curse as a *factorization curse*—a failure of models to learn the same joint distribution under different factorizations. Through a series of controlled experiments with increasing levels of realism including *WikiReversal*, a setting we introduce to closely simulate a knowledge intensive finetuning task, we find that the factorization curse is an inherent failure of the next-token prediction objective used in popular large language models. Moreover, we demonstrate reliable information retrieval cannot be solved with scale, reversed tokens, or even naive bidirectional-attention training. Consequently, various approaches to finetuning on specialized data would necessarily provide mixed results on downstream tasks, unless the model has already seen the right sequence of tokens. Across five tasks of varying levels of complexity, our results uncover a promising path forward: factorization-agnostic objectives such as masked diffusion can significantly mitigate the reversal curse and hint at improved

[1]This chapter is based on research originally presented in Ref. [227]. The work was conducted in collaboration with Niklas Nolte, Dianne Bouchacourt, Adina Williams, Mike Rabbat, and Mark Ibrahim.

knowledge storage and planning capabilities.

7.1 Introduction

Although today's best language models produce impressively cogent, articulate text by learning the statistics of language, they still struggle to reliably retrieve information seen during training. Models are known to suffer from hallucinations, potentially responding with fabricated content that differs from the knowledge present in training data. Hallucinations pose a significant hurdle to the adoption of language models, especially in domains where reliable knowledge retrieval is paramount [228]. A well-studied failure mode underlying hallucinations is the *reversal curse*, which ascribes this deficiency to the precise order of words presented to the model at train-time [229], [230]. For example, a model trained on sentences where *Paris* always appears as the subject of the sentence, such as *"Paris is the capital of France"*, can be tuned to answer *"Paris is the capital of which country?"* but not *"What is the capital of France?"*, even though these two formulations encode the same underlying information. Existing approaches aimed at mitigating the reversal curse have focused on data augmentations that involve training on both the forward and reversed tokens [231]. In this work, we focus on learning objectives.

In Section 7.2, we propose the *factorization curse*, a framework that characterizes the reversal curse as a failure to model the same joint distribution under different factorizations. We show the prevailing left-to-right next token prediction, autoregressive (AR) objective used in popular large models such as GPT [232] and Llama models [233], [234], underlies the reversal curse. We illustrate in Figure 7.1 how the factorization in AR training only encodes information based on prior context, thereby limiting how well the model can retrieve information based on *later context*. Through this lens, we show the reversal curse is not merely a failure to learn logical implications, but a more general problem related to learning objectives. Given this framework, we hypothesize in Section 7.2.1 that factorization agnostic models, *i.e.*, models trained in a manner that is less dependent on the specific token order while preserving the overall meaning, can store knowledge better and are less prone to the reversal curse. To validate our hypothesis and explore potential solutions, we conduct extensive experiments in controlled settings in Section 7.3.1, focusing on the effects of pretraining objectives on knowledge storage. Section 7.3.2 introduces *WikiReversal*, a realistic testbed based on Wikipedia knowledge graphs that closely replicates a knowledge-intensive finetuning application. In experiments with increasing levels of complexity and realism, we observe that scale, naive bidirectional objectives, and even left-to-right training do not resolve the reversal curse. These results suggest that finetuning strategies for downstream applications might not allow models to store knowledge adequately. Finally, in Section 7.4, we find that factorization-agnostic training is not only a promising initial solution for knowledge storage but also hints at improved planning capabilities in a minimal graph traversal task.

To summarize, our contributions are as follows:

1. We introduce the concept of the *factorization curse*, which posits that different objectives' decomposition of an input into context and prediction is a key factor underlying the reversal curse.

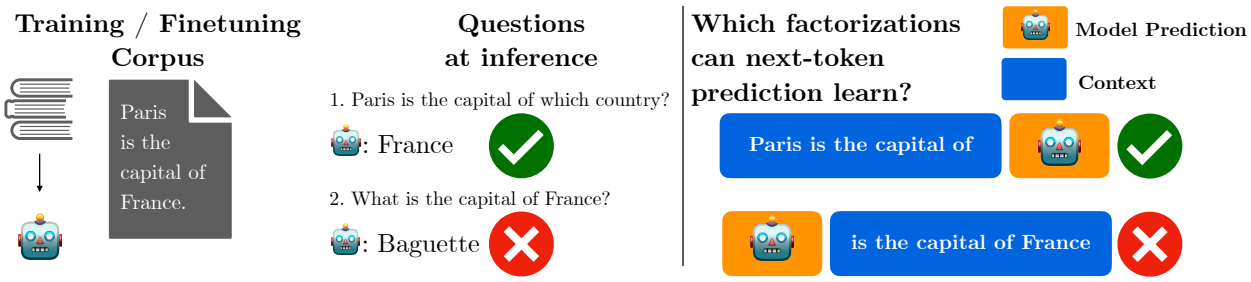


Figure 7.1: **(Left)** Reversal curse from training a model on sentences with *Paris* before *France*. **(Right)** Left-to-right objective does not learn how to predict early tokens from later ones even if the information content is the same. The model overfits to a specific factorization of the joint distribution over tokens, and is unable to answer questions that require reasoning about a different factorization.

2. We conduct empirical studies with increasing levels of complexity and realism to validate our framework, comparing strategies such as standard autoregressive training (AR), AR with reversed sequences, and masked language modeling (MLM) as a prototypical bidirectional objective.

3. Building on our factorization curse framework, we identify factorization-agnostic objectives that allow for making predictions using every possible context decomposition, as a strong baseline solution. We explore its effectiveness across all investigated settings, including the WikiReversal setting.

4. We show that factorization-agnostic strategies are promising not only for knowledge storage/retrieval but also for planning, suggesting potentially broader implications for our findings.

7.2 The Factorization Curse

The reversal curse highlights how language models struggle to reliably retrieve information seen during training given some context. Our aim is to understand this failure by probing how common training objectives factorize an input into context and prediction. We show how common training objectives, including popular left-to-right AR and masked modeling objectives, struggle to learn factorizations that can help the model generalize better on a given task, a challenge we label the *factorization curse*.

7.2.1 Hypothesis: Reversal Curse as an Instance of Factorization Curse

Let us define the *factorization curse* more formally. We first start with the usual left-to-right autoregressive model for a sequence $\boldsymbol{x}$ with D tokens. This is the standard formulation in

popular GPT-style [232], [235] models and its loglikelihood is given by

$$\log p(\boldsymbol{x}) = \sum_{t=1}^{D} \log p(x_t|\boldsymbol{x}_{<t}). \tag{7.1}$$

Each token is represented as x_t, where t is its index in the sequence. $\boldsymbol{x}_{<t}$ represents all tokens that precede the t-th token in the sequence. The log probability of the entire sequence $\boldsymbol{x}$ is computed as the sum of the log probabilities of each token x_t, given all the preceding tokens $\boldsymbol{x}_{<t}$. This is the *left-to-right factorization* of the joint distribution over tokens. Note that there are many factorizations ($D!$) of the joint distribution, each given by some permutation σ of the tokens in the sequence, which we can write as $\log p(\boldsymbol{x}) = \sum_{t=1}^{D} \log p(x_{\sigma(t)}|\boldsymbol{x}_{\sigma(<t)})$.

Example in Two Tokens For illustration purposes, let us walk through an example with $D = 2$. Suppose our goal is to model $p(\boldsymbol{x}) = p(x_2, x_1) = p(x_2|x_1)p(x_1)$. The left-to-right factorization loss optimizes

$$-\mathcal{L}_{AR} = \log p(\boldsymbol{x}) = \log p(x_2|x_1) + \log p(x_1). \tag{7.2}$$

Interestingly, we can readily see the reversal curse failure mode in $\mathcal{L}_{AR}$. A model p_θ that attributes high likelihood to $p_\theta(x_2|x_1)p_\theta(x_1)$ does not necessarily yield a high value for $p_\theta(x_1|x_2)p_\theta(x_2)$ (a right-to-left factorization) even though the two expressions should be equal due to the chain rule of probability. Note that here we make no statement about the sequential order of the random variables or their relationship. The only statement we make is that, unsurprisingly, p_θ is not necessarily capable of modeling the joint distribution when presented with a different factorization. This is the *factorization curse*.

Definition 2. *(Factorization Curse). A model p_θ for the joint distribution of a sequence $\boldsymbol{x}$ suffers from the factorization curse if, for a factorization order σ different from the "training" factorization order σ_0 (which depends on the objective and model details), we have*

$$\prod_t p_\theta(x_{\sigma(t)}|x_{\sigma(<t)}) \neq \prod_t p_\theta(x_{\sigma_0(t)}|x_{\sigma_0(<t)}). \tag{7.3}$$

In particular, the model may be optimal on σ_0, but perform poorly on a different factorization.

Implications This has a number of implications. First, by definition, a highly factorization-dependent LLM will struggle to retrieve knowledge from earlier in the context given later information. Second, simply training on additional sequences with all tokens reversed would likely not alleviate the issue. Indeed, if the information we seek to retrieve is composed of multiple tokens, the factorization the LLM needs to handle is not right-to-left, but instead reversed in chunks. Thus, in order for any reverse training strategy to work, one must first parse the entities of interest then train with sequences reversed in entity-preserving chunks (see Section 7.3 and Figure D.1).

Furthermore this explains why standard MLM approaches with fixed masking rates fail to address the issue, despite their bidirectionality, for two reasons: First, entities may span a larger number of tokens than the model masks, meaning there is never supervision signal to

make the prediction from the right context (without leaking parts of the entity). Second, training with a fixed rate does not yield a good generative model. While the model is used to predicting, *e.g.*, 15% of a full context-window during training, at inference, the model can fail to generalize [236] to the arbitrary-length sequences it encounters (see Figure 7.2). [237] suggest that encountering different length sequences during training encourages disentanglement and compositionality, which will be crucial for a good generative model.

Knowledge retrieval beyond reversal: A model that cannot learn how to retrieve information in reverse order will likely suffer from further downstream issues that are often ascribed to hallucination. For instance, let us take a model pretrained on entities in a database, say a list of soccer players with various attributes (statistics, game histories, *etc.*) with the name appearing before the attributes as follows $\boldsymbol{x}_{\text{name}}, \boldsymbol{x}_{\text{attributes}}$. The model may memorize the database perfectly, but when queried for players that match specific criteria (*e.g.*, played in a particular position, or have a specific nationality, *etc.*), the model can produce hallucinated answers that do not match the training distribution due to lack of direct supervision of the form $p(\boldsymbol{x}_{\text{name}}|\boldsymbol{x}_{\text{attributes}})$ during pretraining.

7.2.2 Factorization-Agnostic Training Strategies

To store and retrieve knowledge in "all directions" for arbitrary-length entities and without external intervention (entity pre-parsing, retrieval augmented generation, *etc.*), the model needs to be equally good at any factorization of the joint distribution. Below, we discuss two ways this can be achieved.

Permutation Language Modeling (PLM) A straightforward way to alleviate the factorization issue, is to write the autoregressive loss in a way that is independent of factorization by averaging over all permutations as follows

$$\log p(\boldsymbol{x}) = \log \mathbb{E}_{\sigma \sim \mathcal{U}(S_D)} \left[\prod_{t=1}^{D} p(x_{\sigma(t)}|\boldsymbol{x}_{\sigma(<t)}) \right] \geq \mathbb{E}_{\sigma \sim \mathcal{U}(S_D)} \left[\sum_{t=1}^{D} \log p(x_{\sigma(t)}|\boldsymbol{x}_{\sigma(<t)}) \right], \quad (7.4)$$

where σ is a permutation sampled uniformly at random from S_D, the permutation group on D tokens. The term $\boldsymbol{x}_{\sigma(<t)}$ represents all tokens that precede the t-th token in the permuted sequence. The log probability of the entire sequence $\boldsymbol{x}$ is then lower-bounded (using Jensen's inequality) by the expected sum of the log probabilities of each element $x_{\sigma(t)}$, given all its preceding tokens in the permuted sequence. Note that all we did here is average over all factorizations. This formulation is used in XLNet [238]. However, for practical reasons they end up training with a permutation on the last few tokens only. This partial prediction, as we argued above, can limit knowledge storage improvements because we do not know how to chunk tokens into entities a priori.

Uniform-Rate Masked Language Modeling (MLM-$\mathcal{U}$) An alternative factorization-agnostic objective is to predict any context from any other context uniformly at random. This includes next-token, previous-token, predictions spanning multiple future or past tokens, and

all other forms of contextual prediction. As it turns out, this generalization over objectives (amounting to something similar to masked language modeling with a randomly sampled masking rate $r \sim \mathcal{U}(0,1)$) is a discrete diffusion model with an absorbing masking state [225], [226]. This diffusion formulation can be used to make a factorization-order-independent autoregressive model. See Figure 7.2 for an illustration of the differences between the MLM-$\mathcal{U}$ objective and more standard MLM. Specifically, $\mathcal{L}_{CT}$ from [225]'s Proposition 1, which we will refer to as $\mathcal{L}_{\text{MLM-}\mathcal{U}}$ here, can be retrieved from Equation (7.4) as follows

$$\begin{aligned}
\mathcal{L}_{\text{MLM-}\mathcal{U}} &= -\mathbb{E}_{\sigma\sim\mathcal{U}(S_D)} \sum_{t=1}^{D} \log p(x_{\sigma(t)}|\boldsymbol{x}_{\sigma(<t)}) \\
&= -\mathbb{E}_{\sigma\sim\mathcal{U}(S_D)}\mathbb{E}_{t\sim\mathcal{U}(1,\cdots,D)} D \log p(x_{\sigma(t)}|\boldsymbol{x}_{\sigma(<t)}) \\
&= -\mathbb{E}_{\sigma\sim\mathcal{U}(S_D)}\mathbb{E}_{t\sim\mathcal{U}(1,\cdots,D)} \frac{D}{D-t+1} \sum_{\tau\in\sigma(\geq t)} \log p(x_\tau|\boldsymbol{x}_{\sigma(<t)})
\end{aligned} \tag{7.5}$$

where the last equality is possible because all $\tau \in \sigma(\geq t)$ tokens are equally likely to appear at position t as we average across all permutations, and so we can average over the predictions for each τ at this position. This approach can be implemented as a denoising process which recovers randomly masked tokens, like BERT [239], but with uniformly sampled masking rates. This key difference allows training a generative model with masked modeling.[2]

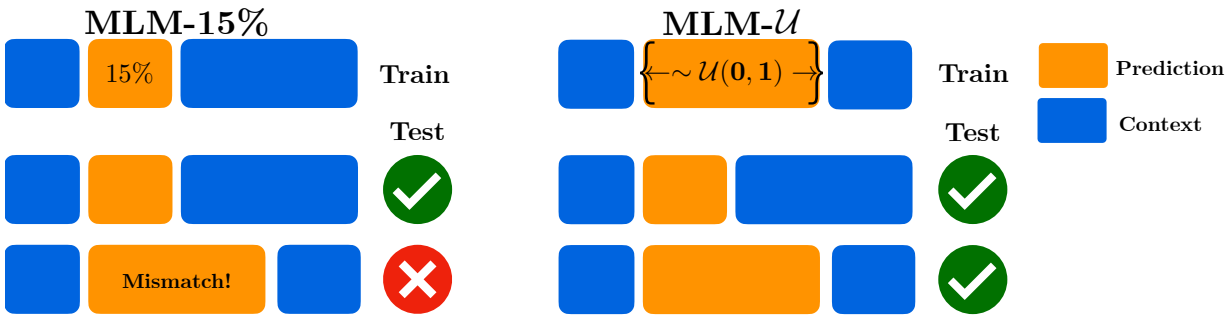


Figure 7.2: MLM struggles when entities span more tokens than the masked span. MLM-$\mathcal{U}$ encounters all possible masking fractions during training and does not suffer from this problem.

7.3 Experiments

We now investigate information retrieval capabilities across learning objectives through the lens of different factorizations of the joint sequence probability. Specifically, we compare

- **AR:** The standard autoregressive causal next-token prediction. Though all models generate tokens autoregressively, we use AR as a shorthand for left-to-right models, in line with Equation (7.1).

[2]Appendix D.2 shows a simple example illustrating the connection to permutation language modeling.

- **AR w/reverse:** AR prediction on sequences and their token-level reverse.[3]
- **MLM r:** BERT-like masked language modeling with fixed random masking rate, r.
- **MLM-$\mathcal{U}$:** MLM with $r \sim \mathcal{U}(0, 1)$. PLM results are similar, and are reported in the Appendix.

To ensure a fair comparison and allow each objective to perform optimally, we employ model architectures specifically designed for each objective. For autoregressive (AR) training, we use GPT-2 [232] and Mistral [240]. For masked language modeling (MLM), we use BERT [239]. Finally, for MLM-$\mathcal{U}$, we employ an encoder-decoder model[4] based on the GPT architecture (see Appendix D.7 for details).

We study these models across increasing levels of complexity and realism, beginning with a controlled retrieval task using synthetic tokens to a retrieval task using natural text from Wikipedia articles. In our evaluation, we find that the degree to which the learning objective factorizes the input reliably explains performance across this wide range of information retrieval tasks. Factorization-agnostic methods show improved knowledge retrieval capabilities.

7.3.1 Controlled Experiments in Factorization-Agnostic Training

Retrieval Task. We are particularly interested in models' ability to recall knowledge from data they were trained on. We will use a simple toy task, adapted from [231], to evaluate this capability. First, we generate a collection of key-value pairs which are each composed of a sequence $\{t_i\}^{i \in S}$ of tokens, *e.g.*, consider the key-value pair

$$t_0 t_1 : t_2 t_3.$$

Each key/value is unique and is composed of a unique set of tokens to control for any effects due to token statistics. Additionally, we generate two types of queries: (forward) "[value of] *key* : *value*" and (backward) "[key of] *value* : *key*". Models are trained on all key-value pairs and a subset of queries, and tested on unseen queries by completing tokens after the colon. Accuracy, measured using exact match, in Table 7.1 shows AR training does not retrieve keys from values and that reversing tokens does not improve backward retrieval. We observe entity-based reversing trivially solves this task. Additionally, while MLM does not suffer from a forward/backward disparity, its fixed masking rate causes poor overall results. Introducing a uniformly sampled rate via MLM-$\mathcal{U}$ solves the task perfectly.

Non-reciprocal Relationships. Are models employing incorrect reversal heuristics? A weakness of the retrieval task is that it could be solved by assuming all relations between keys and values to be symmetric/reciprocal. In language, this is not always the case: even though they contain many of the same words, the sentence *Alice likes Bob* does not necessarily imply that *Bob likes Alice*. To investigate whether models inappropriately rely on a reversal

[3] We obtained similar results when manipulating attention masks to train on an equal mix of causal and "anti-causal" sequences.

[4] We also ran our experiments with this architecture for all the objectives and found consistent results.

Table 7.1: Exact match accuracy of different training paradigms on **(Top)** the retrieval task and **(Bottom)** relationship task. Due to the non-reciprocal nature of the relationship, a model that swaps the subject and object will make errors (e.g., inferring B is A's child from A being B's child). Shown in the bottom row. Entity reversal without a delimiter is marked with a*. Maximum values are bold.

Retrieval Task	AR	AR w/reverse	MLM	MLM-$\mathcal{U}$
Forward ↑	**100**	**100**	21	**100**
Backward ↑	0	0	22	**100**
Relationship Task	AR w/reverse (entity)*	AR w/reverse (entity)	MLM	MLM-$\mathcal{U}$
Forward ↑	54	**100**	24	**100**
Backward ↑	53	**100**	19	**100**
Incorrect Inference ↓	**44**	0	0	0

heuristic, we extend the retrieval task to a third entity for each sample, yielding statements of the form *"$A \implies B \implies C$"*. Question answering (QA) samples are of the form (forward) *"$B \implies ?$"* and (backward) *"$B \impliedby ?$"* where the right answers are C and A, respectively. With a third entity in play, a model assuming symmetry would be unable to decide between A and C as the answer for either question.

The bottom of Table 7.1 shows that simply reversing entities (denoted with AR w/reverse (entity)*) leads to undesirable behaviour as can be seen from the large incorrect inference rate. However, adding simple delimiter tokens around entity reversed sequences (without asterisk) leads to more a robust model. Finally, MLM-$\mathcal{U}$ learns the asymmetric relations correctly.

BioS. Next, we investigate performance of the different objectives for more complex but still controlled data. BioS [241] is a synthetic dataset consisting of biographies for 10k fictional individuals. For each individual, biographical details (birth date, birth city, *etc.*) were randomly selected from a uniform distribution. The authors ensured each individual was assigned a unique full name. We reproduce some of their results on the `birth_date-to-full_name` task which aims to recover a person's full name from their birth date. Results are shown in Table 7.2. Again, the autoregressive, MLM and reversed-token training struggle to recover backward queries.

Training in a factorization-agnostic fashion leads to non-negligible backward performance. Interestingly, backward performance keeps improving over a long time (many times the number of epochs required for forward performance to reach 100%) (see Appendix D.6). If this delay is due to the low frequency of observing the right factorization in training, this could indicate that meth-

Table 7.2: BioS exact match accuracy for property retrieval in the backward direction (birth date to full name) and in the forward direction (full name to birthdate).

	AR	AR w/reverse	MLM	MLM-$\mathcal{U}$
Forward	**100**	**100**	8	**100**
Backward	0	0	0	**68**

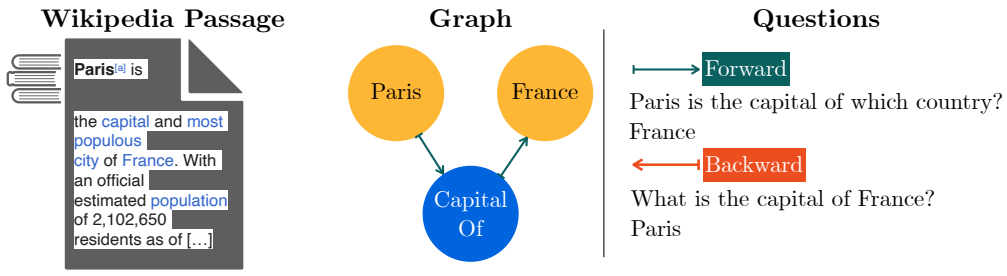


Figure 7.3: An example passage with a forward relation triple. The forward question queries the tail, backward queries the head. *WikiReversal* is a collection of passages and forward/backward QAs.

ods to automatically select data such as
RHO-LOSS [242] could have a disproportionate impact in improving factorization-agnostic methods compared to standard AR training.

7.3.2 Wikipedia Knowledge Graph Reversal

To bridge the gap between the controlled studies on synthetic datasets and more realistic evaluations, we introduce a new evaluation setup that combines the best of both approaches. Our setup involves finetuning a language model on real-world natural text from Wikipedia articles, along with a precise knowledge graph describing the relations and entities within them. This allows for principled experiments that mimic real-world use-cases where practitioners finetune pretrained models on domain-specific corpora.

Experiment Design We introduce a new closed-book QA dataset to evaluate the ability of models to reason about entities and relations in both forward and backward directions. The dataset is derived from the GenWiki corpus based on DBpedia [243], which contains 1.3 million text passages from Wikipedia, along with entity and relation annotations.

Extracting Relation Triples and Generating Questions For each passage P with annotated entities $E = e_1, e_2, ..., e_n$, we consider only "forward" relation triples (e_i, r, e_j), where e_i appears before e_j in the passage. For the example, in the passage *"Paris is the [...] capital of France [...]"* (Figure 7.3), the triplet (*Paris*, *capitalOf*, *France*) is a "forward" triplet. Had the triplet (*France*, *hasCapital*, *Paris*) been present in the graph, we would consider it a "backward" triplet. We filter the data to contain only triplets (and corresponding passages) for which the relation r exists at least in 500 different instances. We generate forward questions $F_r(e_i)$ querying the tail of the relation (e_j) and backward questions $B_r(e_j)$ querying the head (e_i) using predefined templates. We

Table 7.3: Wikireversal task exact match QA accuracies. MLM-$\mathcal{U}$, MLM and AR are are 100M parameter models trained from scratch.

	Mistral 7B	MLM	MLM-$\mathcal{U}$	AR
Forward	**21**	3.4	11	14
Backward	5.2	2.7	**7.9**	4.3

filter out ambiguous samples to ensure each question has a single unique answer. Algorithm 2 in the Appendix summarizes the dataset creation process.

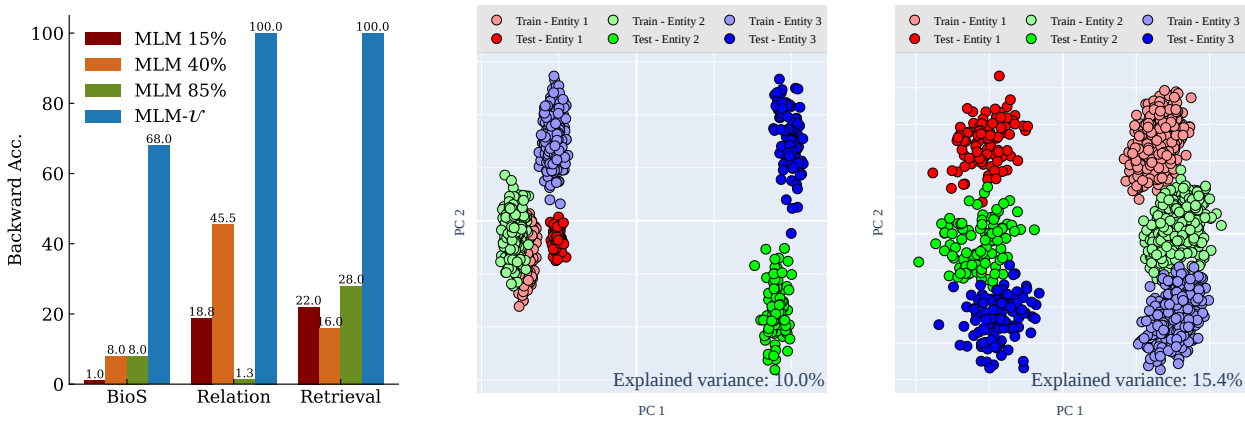
Table 7.3 reports the forward/backward performance disparity, particularly for autoregressive models. Mistral 7B, achieves a backward accuracy of around 5%, much lower than its forward accuracy. Interestingly, the model starts around the same backward accuracy in the beginnig of finetuning. This indicates there may still be backwards triplets present in a "forward fashion" within the model's training text. This could also explain the non-trivial backward performance of the AR model, despite its susceptibility to the reversal curse. MLM-$\mathcal{U}$ attains the highest backward accuracy among the evaluated models, demonstrating its robustness to the reversal curse. However, it still falls short of the AR model's forward performance, possibly due to the inherent difficulty of the task. Notably, significantly better results can be obtained by allowing models to leverage knowledge stored from the QAs themselves (see Appendix D.5.3 for details).

7.3.3 Analyzing Representations Learned via Factorization-Agnostic Training

We further examine factorization-agnostic training by first comparing the role of random masking in MLM-$\mathcal{U}$ versus standard masked language modeling. We also visualize the learned representations from MLM-$\mathcal{U}$ showing they contain more distinct entity structure compared to standard AR training.

Understanding the role of random masking To understand the importance of varying the masking ratio as introduced in MLM-$\mathcal{U}$ we compare MLM-$\mathcal{U}$ to MLM with various masking ratios (15%, 40%, 85%) based on prior work [244]). We find MLM exhibits much noisier performance that's consistently lower than MLM-$\mathcal{U}$ with uniformly random masking ratio as shown in Figure 7.4a. This suggests fixed masking ratios, whether with high or low values, are limited in what they can learn in contrast to MLM-$\mathcal{U}$.

Visualizing learned representations in the 3-entity relationship task To better probe the representations learned via MLM-$\mathcal{U}$ we plot in Figures 7.4b and 7.4c the PCA projections after training on the relationship task from Section 7.3.1 for AR and MLM-$\mathcal{U}$. Compared to AR, which learns disconnected components without apparent symmetry for entities never seen backwards during training, MLM-$\mathcal{U}$ seems to have learned a form of translation symmetry across train and test samples. This suggests MLM-$\mathcal{U}$ training leads to more structured entities in the model's representation space.



(a) Fixed-rate masked modeling is inconsistent.

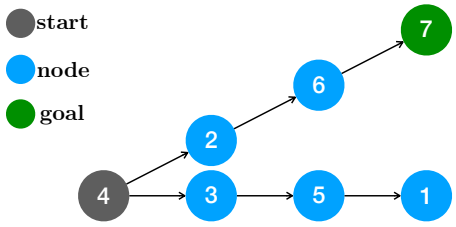
(b) AR model entities PCA projection.

(c) MLM-$\mathcal{U}$ model entities PCA projection.

Figure 7.4: In panel (a) we compare MLM with varying masking ratios to MLM-$\mathcal{U}$. In panels (b) and (c) we visualize the two main principal components of representations learned via AR versus MLM-$\mathcal{U}$.

7.4 On the Importance of Future Predictions for Planning

Prior work argues next-token prediction auto-regressive loss is not conducive to planning [245]–[247]. Specifically, [248] introduces a simple path finding task that requires basic planning: From a start node, multiple linear paths $p_1, p_2, \ldots, p_n$ extend outward. They are given as symbolic sequences of this form: $\underbrace{2,6|6,7|5,1|4,3|4,2|3,5}_{\text{edges}}$ / $\underbrace{4,7}_{\text{start,end}}$ = $\underbrace{4,2,6,7}_{\text{desired response}}$ A model is tasked to predict the sequence of nodes along path p_i that leads to a specified final node at the end of p_i. They show that when trained with a standard autoregressive (AR) next-token prediction objective, the model is unable to effectively learn this task. This failure is attributed, at least in part, to the teacher-forcing supervision used during training. As illustrated in Figure 7.5, from the second node $x_2 = 2$ onward along a path $p_i = (x_1, x_2, \ldots, x_m)$, the model can predict each "easy" token x_t for $t > 2$ by simply conditioning on the immediately previous teacher-forced token x_{t-1}, without requiring retention of the earlier path history or look-ahead planning, a pitfall referred to as the "Clever Hans" cheat (see Section 4.5 [248] and [249]).



(a) Accuracies of various training paradigms on the Star Graph Task. Randomly choosing a starting node in this setting (and employing the Clever Hans Cheat) results in 50% accuracy.

	AR	AR w/reverse	MLM-$\mathcal{U}$
Accuracy	50	49	**100**

Figure 7.5: Star Graph Task: Illustration and Performance Comparison. The illustration shows the "Clever Hans" failure mode with teacher-forced AR ([248] adapted).

[248] found that predicting multiple future tokens in a teacher-less setting helped mitigate the issue of discovering the algorithm to correctly predict the initial "difficult" token x_2. We identify this as an intermediate objective between standard next-token prediction and the factorization-agnostic objective studied in this chapter, which encourages planning capabilities via both far look-ahead and look-backward along the sequence. Figure 7.5a shows that the MLM-$\mathcal{U}$ objective enables the model to reliably solve the path-planning task by better capturing the planning requirements.

7.5 Related Work

The reversal curse was first introduced in [229]. Using text-overlap based heuristics for modeling inferences between sequence of text dates back nearly two decades in NlP [250], [251]. As our modeling approaches have improved, increasing work has drawn attention to models overapplying text-overlap heuristics ([252]–[257], i.a.). Perhaps most relevant is [258]'s evaluation, which used synthetic entity-based kinship data with multiple entities based on graph structures to expose model failures and is similar to our relationship task. Most recently, work aimed at mitigating the reversal curse by [230], [231] suggest using data augmentations by reversing both token sequences, or if available, entity orders by training both on the forward and augmented text. Related projects have also trained and/or finetuned RoBERTa [259] or BERT [239]-based models on input sequences with randomly shuffled word order [260]–[262]. [263] explore a fine-tuning objective with bidirectional attention and show that it can mitigate the reversal curse in the original synthetic setting from [229]. However, they employ fixed masking rates. In addition to the standard objectives we explored, much recent work has gone into a variety of pre-training objectives including span-based and hybrid objectives [236], [264], [265]. XLNet [238] utilizes a permutation language modeling objective, considering permutations of the input sequence during training. However, XLNet is not completely factorization-agnostic as it only predicts the last few tokens in each permutation.

Various benchmarks have been introduced to evaluate the reasoning capabilities of language models. [248] present a study on the limitations of next-token prediction in capturing reasoning abilities, arguing that the standard autoregressive training objective hinders models' ability to plan. In a similar vein, [245] investigate the limits of transformer LLMs across three compositional tasks: multi-digit multiplication, logic grid puzzles, and

a classic dynamic programming problem. Their findings suggest that transformer LLMs solve compositional tasks by reducing multi-step compositional reasoning into linearized subgraph matching, without necessarily developing systematic problem-solving skills. They also provide theoretical arguments on abstract multi-step reasoning problems, highlighting how autoregressive generations' performance can rapidly decay with increased task complexity.

7.6 Summary & Discussion

Limitations and Potential Extensions. MLM-$\mathcal{U}$ has a much more challenging objective since we approximate all possible partitions of the input into context and predictions. Learning curves show delayed generalization, especially on backward samples. The main limitation of factorization-agnostic approaches is the optimization difficulty due to task complexity. Predicting one token ahead is far easier than predicting the last word of a novel with limited context, due to increasing entropy along longer horizons. This requires better schedules/curricula that smoothly interpolate the difficulty increase from next-token prediction to the highest-complexity factorization the model can handle.

This work highlights how alternative objectives can address some of the issues with current state-of-the-art language models, which rely on left-to-right autoregressive generative decoder pretraining. Despite impressive capabilities with increasing scales, there are concerns about reaching a plateau due to fundamental limitations, computational constraints, or data scarcity. We find that factorization-agnostic training can learn "more" from the same data in the context of reversal curse. This presents a case for studying factorization-agnostic objectives and investing in approaches to scale them.

Chapter 8

Conclusion

In this thesis, we explored a wide variety of topics on developing deep learning approaches to explore questions in fundamental physics specifically in nuclear and particle physics. Neural networks are notorious for their black-box and whimsical nature, but they can be tamed into highly efficient and robust algorithms physicists can trust to make decisions on their behalf as we have seen with Lipschitz Monotonic Networks in Chapter 2. We have seen that their tendency to pick up biases present in data can be mitigated using simple changes in the training objective like MoDe in Chapter 3. This new regularization forces models to specific output distributions (uniform or otherwise) thereby avoiding peak-sculpting, a common pitfall in particle physics searches where models infer a hidden variable and incorrectly use it as a discriminator. We have also seen that neural networks can enable alternative approaches to studying collider events in Chapter 4. In Chapter 5, we gained a surprising insight: neural networks can learn interpretable features that align with human-derived understanding. Perhaps this is not so surprising if we consider the observation[1] that humans can solve two types of problems: (1) linear problems with arbitrarily high dimensions and (2) non-linear problems with low dimensions. Neural networks similarly like to linearize highly non-linear problems by expanding into a high-dimensional space, and when that's not possible, they tend to fall back to some uninterpretable long collection of numbers. This is precisely what we observe in many problems, including when training models on nuclear physics data, and this explains why PCA, a linear operation, does reasonably well at extracting human-interpretable terms out of neural features.

We found that neural networks are a powerful and highly versatile tool in the physicist's toolkit. We also found that this toolkit can be employed to better understand deep learning phenomena and develop new and powerful algorithms. The field of deep learning phenomenology is nascent and can take on many different forms, but I believe the physics-inspired approach will be highly fruitful. For one, it gave us neural scaling laws to which we owe a lot of the recent advancement of frontier models, but it also gives us new ways to characterize learning problems via phase diagrams and effective theories (Chapter 6). Physics has inspired learning paradigms for a long time, from spin glasses to diffusion. We formulated a diffusion paradigm that can be used to resolve a limitation of current frontier language models, the reversal curse, in Chapter 7. These are but a few examples of physics formalisms used to

[1] courtesy of my advisor Mike

advance deep learning but there can be many more.

Appendix A

Monotnic Networks

A.1 Public datasets with monotonic networks

In this section, we follow as closely as possible the experiments done in [34], and some experiments done in [266] to be able to directly compare to state-of-the-art monotonic architectures. [34] studied monotonic architectures on four different datasets: COMPAS ([267]), BlogFeedback ([268]), LoanDefaulter ([269]), and ChestXRay ([270]). From [266] we compare against one regression and one classification task: AutoMPG ([271]) and HeartDisease ([272]). Results are shown in Table A.1.

COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) refers to a commercial algorithm used by judges and police officers to determine the likelihood of reoffense. [267] discusses that the algorithm is racially biased and provides a dataset from a two-year study of the real-world performance of COMPAS. The task here is to predict the reoffense probability within the next two years. The dataset has 13 features, 4 of which have a monotonic inductive bias, and contains a total of 6172 data points.

BlogFeedBack This dataset contains 54270 data points with 276 dimensions describing blog posts. The task is to predict the number of comments following the post publication within 24 hours. 8 of the features have a monotonic inductive bias. Just like [34], we also only consider the 90% of the data points with the smallest targets so as to not let the RMSE be dominated by outliers.

LoanDefaulter The version of this dataset available on Kaggle was updated on a yearly basis up to 2015. [269] contains a link that is, we believe, a superset of the data used in [34]. Luckily, the authors have shared with us the exact version of the dataset they used in their studies for an appropriate comparison. The data is organized in 28 features and the task is to determine loan defaulters. The classification score should be monotonic in 5 features: non-decreasing in number of public record bankruptcies and Debt-to-Income ratio, non-increasing in credit score, length of employment and annual income.

ChestXRay This dataset contains tabular data and images of patients with diseases that are visible in a chest x-ray. The task is to predict whether or not the patient has such a disease. Just like [34], we send the image through an ImageNet-pretrained ResNet18 ([273]). The penultimate layer output concatenated with tabular data acts as input to the monotonic architecture. Two of the four tabular features are monotonic. In the bottom right table in A.1, there are two entries for our architecture. The *E-E* entry refers to end-to-end training with ResNet18, whereas the other experiment fixes the ResNet weights.

AutoMPG ([271]) This is a dataset containing 398 examples of cars, described by 7 numerical features and the model name. The target, MPG, is monotonically decreasing with 3 of the features. The name is not used as a feature.

HeartDisease ([272]) is a dataset of patients, described by 13 features. The task is to determine whether or not the patient has heart disease.

As can be seen in Table A.1, our Lipschitz monotonic networks perform competitively or better than the state-of-the-art on all benchmarks we tried.

COMPAS

Method	Parameters	↑↑ Test Acc
Certified	23112	$(68.8 \pm 0.2)\%$
LMN	**37**	$(\mathbf{69.3 \pm 0.1})\%$

BlogFeedback

Method	Parameters	↓↓ RMSE
Certified	8492	$.158 \pm .001$
LMN	**2225**	$\mathbf{.160 \pm .001}$
LMN mini	**177**	$\mathbf{.155 \pm .001}$

LoanDefaulter

Method	Parameters	↑↑ Test Acc
Certified	8502	$(65.2 \pm 0.1)\%$
LMN	**753**	$(\mathbf{65.44 \pm 0.03})\%$
LMN mini	**69**	$(\mathbf{65.28 \pm 0.01})\%$

ChestXRay

Method	Parameters	↑↑ Test Acc
Certified	12792	$(62.3 \pm 0.2)\%$
Certified E-E	12792	$(66.3 \pm 1.0)\%$
LMN	**1043**	$(\mathbf{67.6 \pm 0.6})\%$
LMN E-E	**1043**	$(\mathbf{70.0 \pm 1.4})\%$

Heart Disease

Method	↑↑ Test Acc
COMET	$(86 \pm 3)\%$
LMN	$(\mathbf{89.6 \pm 1.9})\%$

Auto MPG

Method	↓↓ MSE
COMET	$(8.81 \pm 1.81)\%$
LMN	$(\mathbf{7.58 \pm 1.2})\%$

Table A.1: We compare our method (in bold) against state-of-the-art monotonic models (we only show the best) on a variety of benchmarks. The performance numbers for other techniques were taken from [34] and [266]. In the ChestXRay experiment, we train one model with frozen ResNet18 weights (second to last) and another with end-to-end training (last). While our models can generally get quite small, we can achieve even smaller models when only taking a subset of all the features. These models are denoted with "mini".

It is also immediately apparent that our architecture is highly expressive. We manage to train tiny networks with few parameters while still achieving competitive performance. Given that some of these datasets have a significant number of features compared to our chosen network width, most parameters are in the weights of the first layer. We manage to build and train even smaller networks with better generalization performance when taking only a few important features. These networks are denoted with **mini** in Table A.1. Because all of the presented architectures are small in size, we show practical finite sample expressiveness for harder tasks and larger networks by achieving 100% training accuracy on MINST, CIFAR-10, and CIFAR-100 with real and random labels as well as an augmented version (i.e. with an additional monotonic feature added artificially) of CIFAR100 in Appendix A.2.

A.2 Expressive power of the architecture

Robust architectures like Lipschitz constrained networks are often believed to be much less expressive than their unconstrained counterparts [274]. Here we show that our architecture is capable of (over)fitting complex decision boundaries even on random labels in a setup simular to [275].

We show the finite sample expressiveness of the architecture in https://github.com/okitouni/Lipschitz-network-bench by fitting MNIST, CIFAR10, CIFAR100 with normal and random labels to 100% training accuracy. We also train on CIFAR100 with an additional "goodness" feature $x \in [0, 1]$ to showcase the monotonicity aspect of the architecture. This dataset is referred to as CIFAR101 below. The synthetic monotonicity problem is currently implemented such that samples with values above a critical threshold in the goodness feature $x > x_{\text{crit}}$ are labeled 0. An alternative implementation is to take label 0 with probability x and keep the original label (or assign a random one) with probability $1 - x$. Table A.2 summarizes the setup used for training. We use Adam with default hyper-parameters im all experiments.

Task	Width	Depth	LR	EPOCHS	Batchsize	Loss
MNIST	1024	3	10^{-5}	10^5	ALL	CE($\tau = 256$)
CIFAR10	1024	3	10^{-5}	10^5	ALL	CE($\tau = 256$)
CIFAR100/101	1024	3	10^{-5}	10^5	ALL	CE($\tau = 256$)

Table A.2: Training MNIST and CIFAR10/100 to 100% training accuracy with Lipschitz networks.

Appendix B

Automated Nuclear Physics

B.1 Why does the model learn a helix?

The helix structure observed in the embeddings of both neutron and proton embeddings presents one of the most striking features in the model trained on nuclear properties. In an effort to get to the bottom of it, we attempt to isolate where it comes from. From experiments in the multi-task *vs.*single-task settings, we notice that having the binding energy as a target is a strong predictor for the appearance of the helix. Therefore we will restrict ourselves to the prediction of binding energy. Our strategy for shedding light on how the model uses the helix structure to its advantage is parameterizing and then perturbing the helix parameters. We hope to be able to factorize contributions from different aspects to break the process into understandable pieces. We fit a helix with trainable parameters using the following parametric equation:

$$\vec{r}(t) = R\left[\cos(2\pi f t + \phi)\vec{u} + \sin(2\pi f t + \phi)\vec{v}\right] + P\vec{a}t + \vec{r}_0 \ , \tag{B.1}$$

where $\vec{u}$ and $\vec{v}$ are orthonormal unit vectors perpendicular to the central axis pointing towards the direction given by the unit vector $\vec{a}$. The shape parameters are: the length of central axis $\vec{P}$, the frequency f, the phase ϕ, the radius R, and the origin $\vec{r}_0$. The direction of the evolution is chosen to be towards the visually most helix-like portion of 3D PCA projections of both neutron and proton embeddings.

In an effort to maximize visual clarity, we show experiments for a model trained on binding energy predictions from the SEMF, where we find a cleaner helix structure than when training on real data, see Figure 5.1 (right). We constrain ourselves to $N \in [40, 120], Z \in [25, 80]$ to be able to fit the helix with a constant radius. The results of the fit can be found in Figure B.2. The fits match the PC projections well and we can now perturb helix parameters. For visualization, we provide three plots for each parameter change: First, a plot of the helix with and without the changed parameter. Second, the model prediction relative to $A = N + Z$ with and without the changed parameter as a function of N for a fixed value of Z. Third, the same plot with N and Z roles reversed. We find that plotting relative to A gives visually more informative results.

First, we increase the length parameter in Figure B.3a. This elongates the helix along its main direction. Similarly as depicted in Figure 5.7, we find that moving along the main

direction corresponds to a macroscopic term akin to the volume term in the SEMF. Since we plot relative to A, that term causes, in first order, a constant offset in the predictions. Figure B.3b shows a reduction of the length, resulting in a negative offset.

Next, we increase the radius parameter, see Figure B.3d. This causes the downwards facing arcs to “sharpen". Taking a closer look at the SEMF formula and the N vs. model output plot, we hypothesize that the depicted arcs are in fact the approximate parabola described by the third term and that the radius controls the prefactor of that parabola, causing the “sharpening", or, in case of a radius parameter reduction, the flattening depicted in Figure B.3c.

Lastly, we double the frequency parameter, see Figure B.3e. There is no clear correspondence to any one particular term in the SEMF, but it gives an indication about how the arc is created. Doubling the frequency doubles the frequency of a now periodic sequence of arcs. This can be understood intuitively when observing Figure 5.8. The ring structure with double frequency goes around twice and two periods appear in the model output. Figure B.3f shows that this trend is persistent also when increasing the frequency even more.

While we have made decent progress towards understanding how the embeddings map to the output of the model, the full picture is not completely clear yet. However, we are confident that an iterative approach can help us understand the story completely.

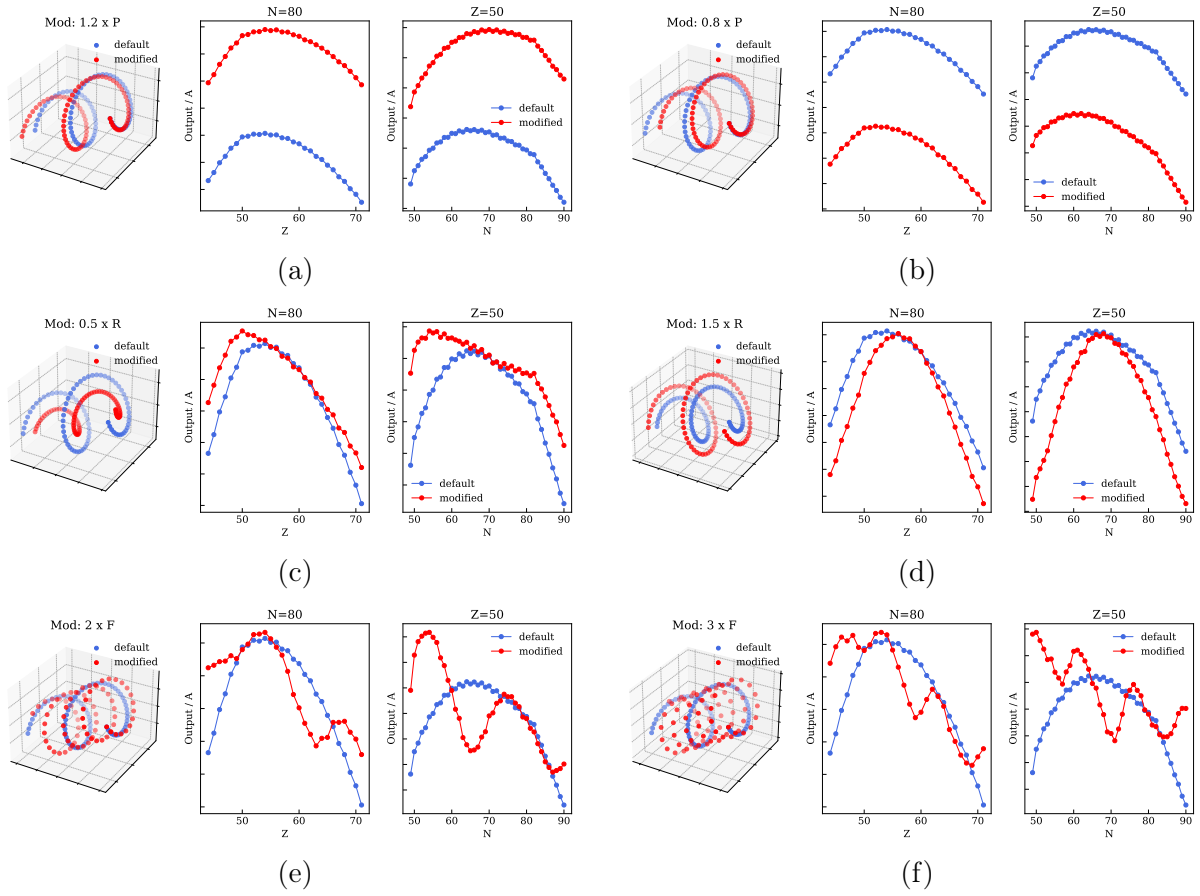


Figure B.1: Variations in helix parameters and their effects on predictions when: (a) increasing the length by 20%, (b) reducing the length by 20%, (c) reducing the radius by 50%, (d) increasing the radius by 50%, (e) multiplying the frequency by 2, (f) multiplying the frequency by 3. (Model trained on data).

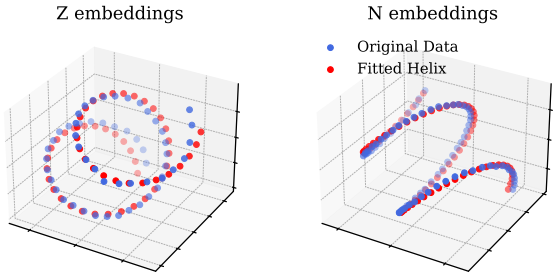


Figure B.2: Results of fitting the helix to the selected portions of N and Z embeddings. This model was trained on the SEMF.

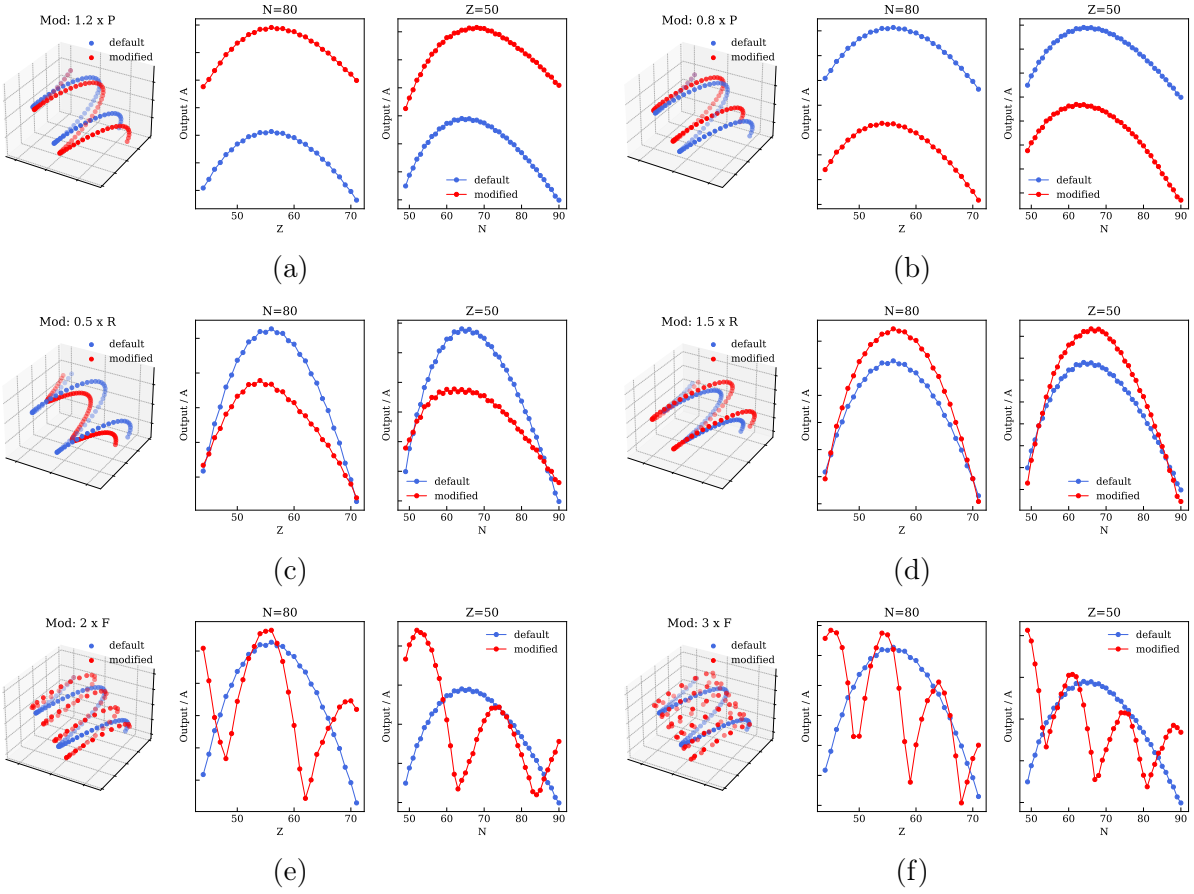


Figure B.3: Equivalent of Figure B.1, but for a model trained on the SEMF directly.

B.2 Training and model details

We use an attention ablated transformer with SiLU activations and residual connections. We experimented with different norms (RMS/Layer/Batch)Norm and the results seemed similar to having no norm at all (probably due to shallowness of the models used). Attention seems to matter a lot more despite the fact that model and context length are relatively small. Fixing attention in the way we do can be shown to simplify the model quite drastically [178]. We also found the embeddings to be easier to interpret so we focus on this setup throughout the paper. We use a linear readout layer at the top of the model to predict scalar values which we train with MSE loss. We also experimented with different weighting schemes for the tasks and settled on a "physics-informed" scheme based on expected measurement errors for each task.

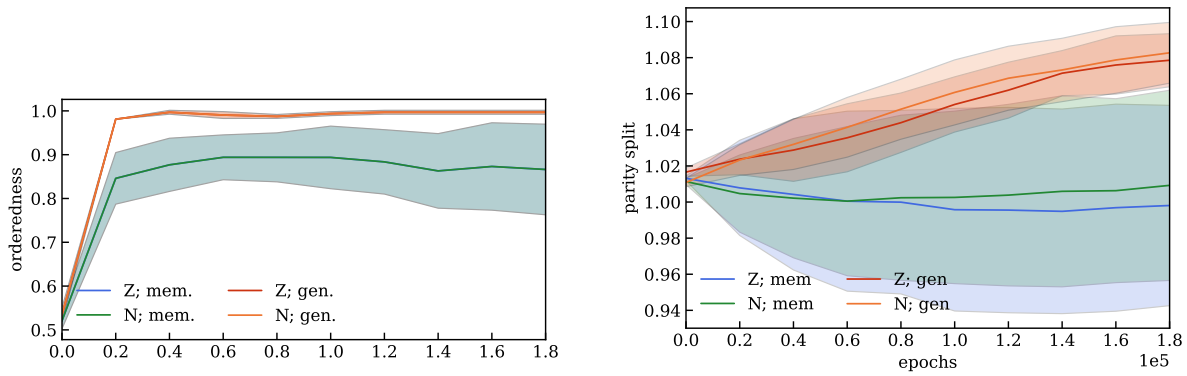
We use AdamW with mostly default parameters and experiment with a range of hyperparameters in our explorations learning rate $\in [10^{-4}, 10^{-3}]$, weight decay $\in [10^{-8}, 10^{-2}]$. The runs used to generate the embeddings and visualizations have the following parameters:

- `EPOCHS = 200,000`
- `HIDDEN_DIM = 2048`
- `LR = 0.0001`
- `WD = 0.01`
- `DEPTH = 2`
- `Seed = 0`

Most training runs were on Nvidia V100 GPUs with some done on Nvidia A6000 GPUs.

B.2.1 Structure evolution

Here we visualize the progress of our "strcuture measures" as a function of time for models that generalize well and models that memorize.



(a) Orderness in time for generalizing and memorizing models.

(b) Parity in time for generalizing and memorizing models.

Figure B.4: Progress of structure measures plotted against the number of epochs (normalized by 10^5).

B.3 Physics models and observables

B.3.1 Data

The data sources are: for the various energies the Atomic Mass Evaluation (AME) [276] and for the charge radii the Atomic Data and Nuclear Data Tables 99 (2013) [277]. We note that all the RMS metrics are calculated using the whole datasets, which include both experimental measurements as well as estimates, e.g. via the method of *trends from the mass surface* (TMS).

B.3.2 Liquid-Drop Model (LDM) - the theory behind the SEMF

While the properties of the nuclei share the same microscopic origin, namely the strong nuclear force and electromagnetism, experimentally we have access only to a set of macroscopic observables. The first and historically most important nuclear model is the macroscopic LDM, which treats the nucleus as a droplet of highly dense fluid, bound together by the strong nuclear force. The model explains why most nuclei have a spherical shape with a radius proportional to $\sim A^{1/3}$. Impressively, this dependence yields an excellent fit to the charge radius data.

Moreover, the LDM provides an estimation of the binding energy [177], [278], which is the fundamental observable in nuclear physics as it enters the calculations of most of the other quantities. It represents the energy required to break apart a nucleus into its individual nucleons and it is defined as

$$E_B(Z,N) \equiv Zm_p + Nm_n - M(Z,N) \ , \tag{B.2}$$

The LDM prediction for E_B is given by the SEMF (see equation 5.1). In the following, we briefly explain the phenomenological motivation for the terms that appear in the SEMF.

Volume Term $+a_V A$: Represents the bulk energy contribution. The nucleus's overall energy is directly proportional to its volume.

Surface Term $-a_S A^{2/3}$: Accounts for nucleons on the surface having fewer neighboring nucleons to bond with. It is proportional to the surface area of the nucleus and it is negative, since it corrects the additional contribution assumed for the volume term.

Coulomb Term $-a_C \frac{Z(Z-1)}{A^{1/3}}$: Reduces the total energy due the electrostatic repulsion between protons.

Asymmetry Term $-a_S \frac{(N-Z)^2}{A}$ Accounts for the Pauli exclusion principle, i.e. increased energy is required when neutrons and protons are present in unequal numbers, forcing one type of particle into higher energy states.

Pairing Term $\pm a_P A^{-1/2}$: This term is non-zero only for even A and reflects the stability gained through the pairing of protons and neutrons due to spin coupling. The contribution is either positive or negative if N and Z are both even or odd, respectively.

The SEMF is refined upon the inclusion of a number of additional terms: (i) exchange Coulomb term, (ii) Wigner term, (iii) surface symmetry term, (iv) curvature term, and (v) shell effects term. For detailed explanations of these terms, as well as the fits of all the coefficients a_* see [279]. The contributions of these additional terms are depicted in Figure B.10 (the refined SEMF is denoted as BW2).

B.3.3 Nuclear shell model

The failure of the SEMF at reproducing the measured values of masses for light nuclei and nuclei with certain numbers of nucleons, the *magic numbers*[1], led to the development of the *nuclear shell model* by Goeppert-Mayer and Jensen (Nobel Prize in Physics, 1963). According to this model, protons and neutrons are seperately arranged in shells, and magic numbers occur when shells are filled. Nuclei with either Z or N (or both) equal to a magic (or doubly magic) number exhibit enhanced stability, and thus the E_B spikes.

The various shell properties can be reproduced by approximating the nuclear potential with a three-dimensional harmonic oscillator plus a spin–orbit interaction. More advanced treatments include the usage of mean field potentials. However, a simple phenomenological term can be still be added to the SEMF and improve its performance. This term is: $a_{M1}P + a_{M2}P^2$, where $P = \frac{\nu_N \nu_Z}{\nu_N + \nu_Z}$ and $\nu_{N,Z}$ the numbers of the valence nucleons (i.e. the difference between the actual nucleon numbers, N and Z respectively, and the nearest magic numbers). The contribution of this term can be seen in Figure B.11.

B.3.4 Separation energies

The stability of a nuclide is determined by its separation energies, which refers to the energies needed to remove a specific number of nucleons from it. They reflect the changes in structure across the nuclear landscape and play a crucial role in understanding the energy requirements involved in nuclear reactions. The separation energies of an isotope can be determined in case the binding energies of neighboring isotopes on the $N - Z$ plane have been measured (and vice-versa). The one-neutron S_N, one-proton S_P separation energy, the energy released in α-decay $Q_{\rm A}$, β-decay $Q_{\rm BM}$, double β-decay $Q_{\rm BMN}$, and electron-capture process $Q_{\rm EC}$ are, respectively

$$\begin{aligned}
S_N(Z,N) &\equiv M(Z,N-1) + m_n - M(Z,N) \ , \\
S_P(Z,N) &\equiv M(Z-1,N) + m_p - M(Z,N) \ . \\
Q_{\rm A}(Z,N) &\equiv M(Z,N) - M(Z-1,N+1) - m_{^4_2{\rm He}} \\
Q_{\rm BM}(Z,N) &\equiv M(Z,N) - M(Z+1,N-1) \ , \\
Q_{\rm BMN}(Z,N) &\equiv M(Z,N) - m_n - M(Z+1,N-2) \ , \\
Q_{\rm EC}(Z,N) &\equiv M(Z,N) - M(Z-1,N+1) \ .
\end{aligned} \tag{B.3}$$

[1]The most widely recognized are $[2, 8, 20, 28, 50, 82, 126]$ and others are still debated.

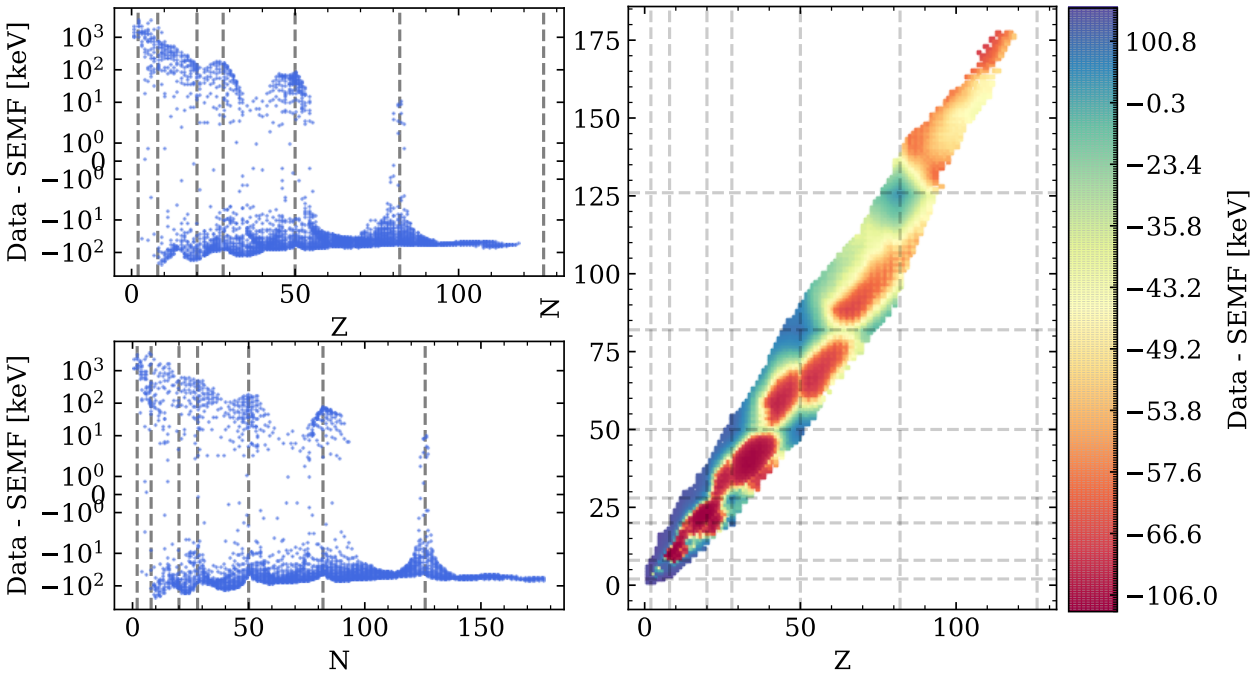


Figure B.5: Residual between data and the semi-empirical mass formula. Dashed lines are magic numbers.

B.4 Which representations come from which task?

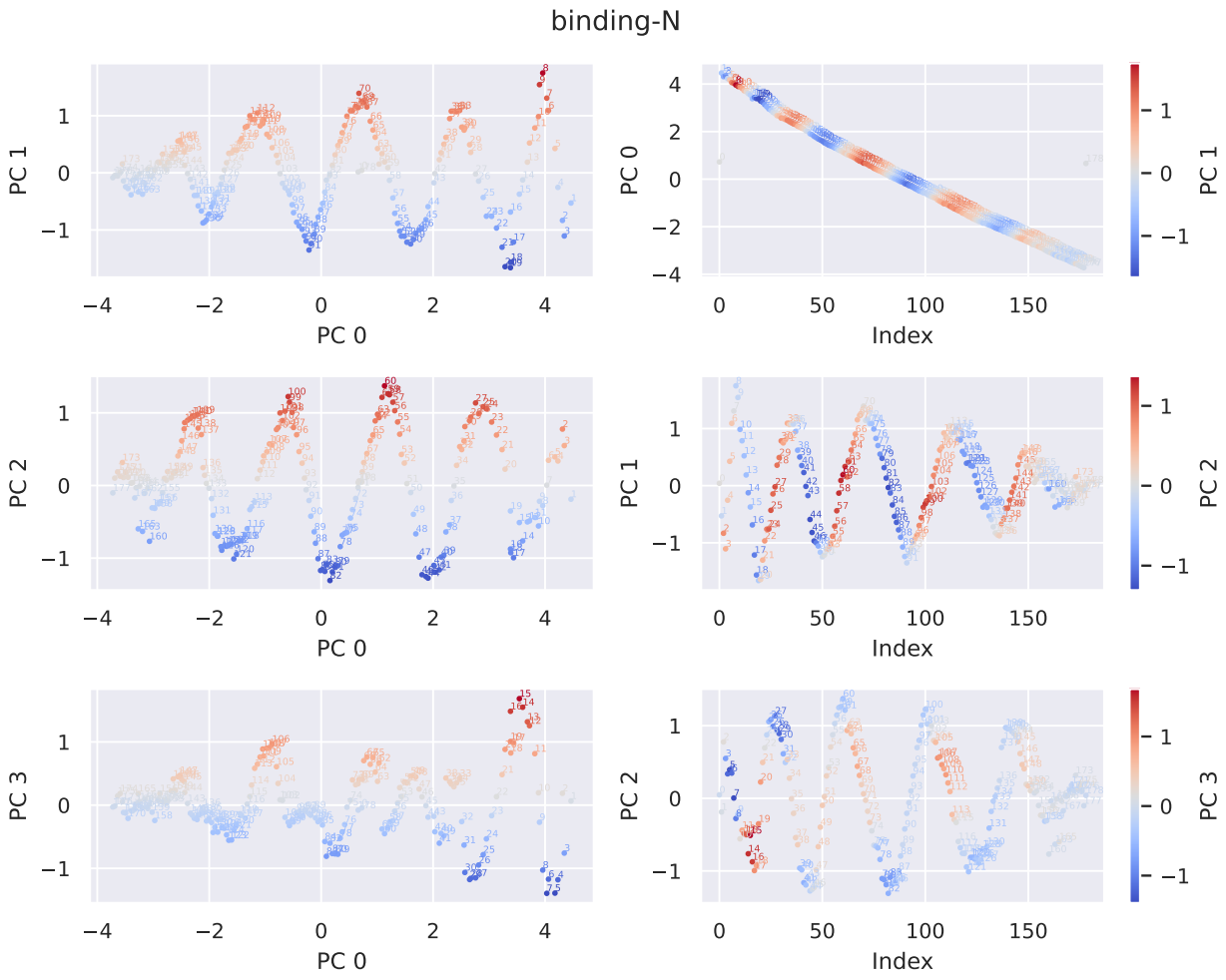


Figure B.6: First few PC projections of the N embeddings for a model trained on only binding energy. Index here refers to the token index or the value of N.

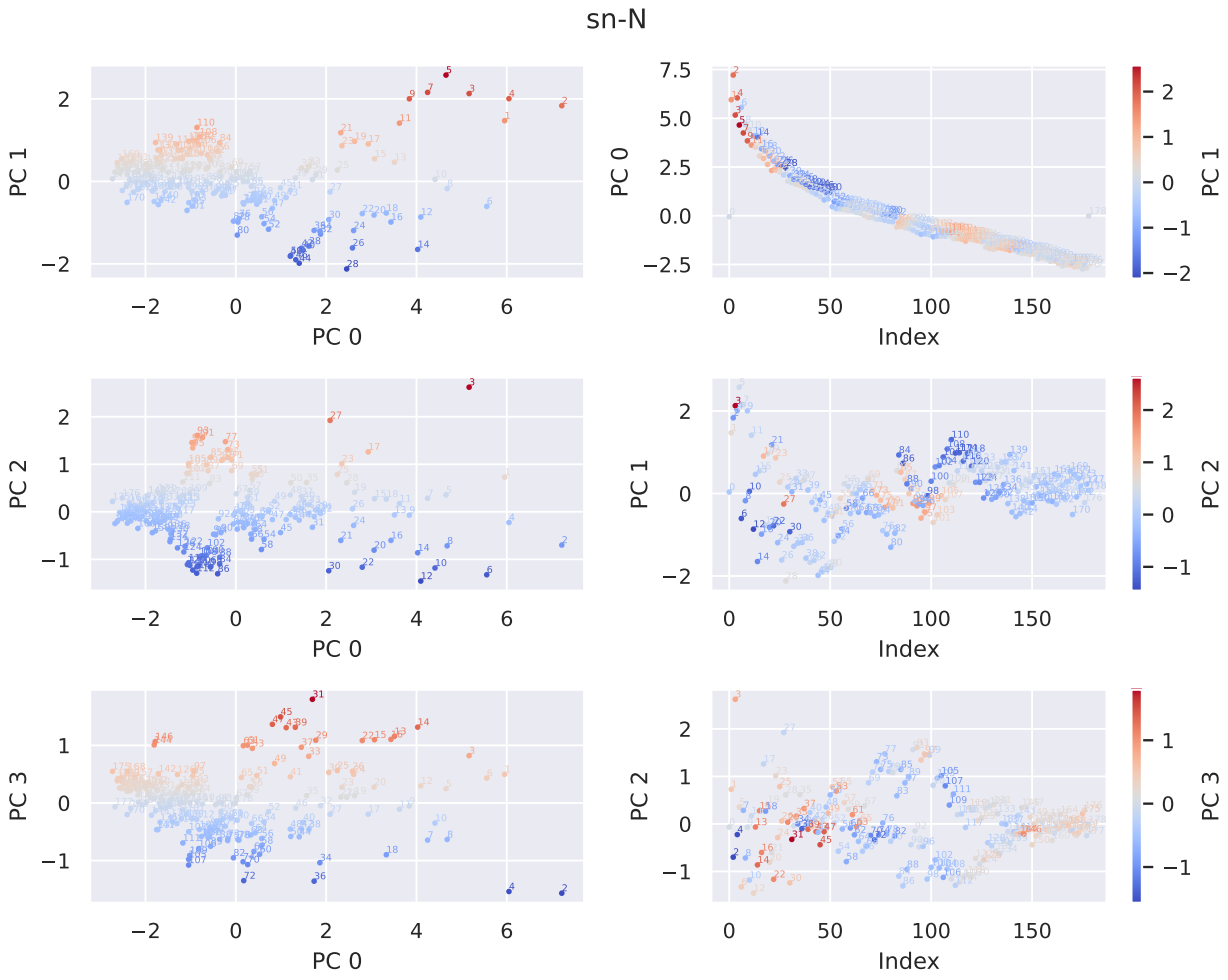


Figure B.7: First few PC projections of the N embeddings for a model trained on the target S_N only.

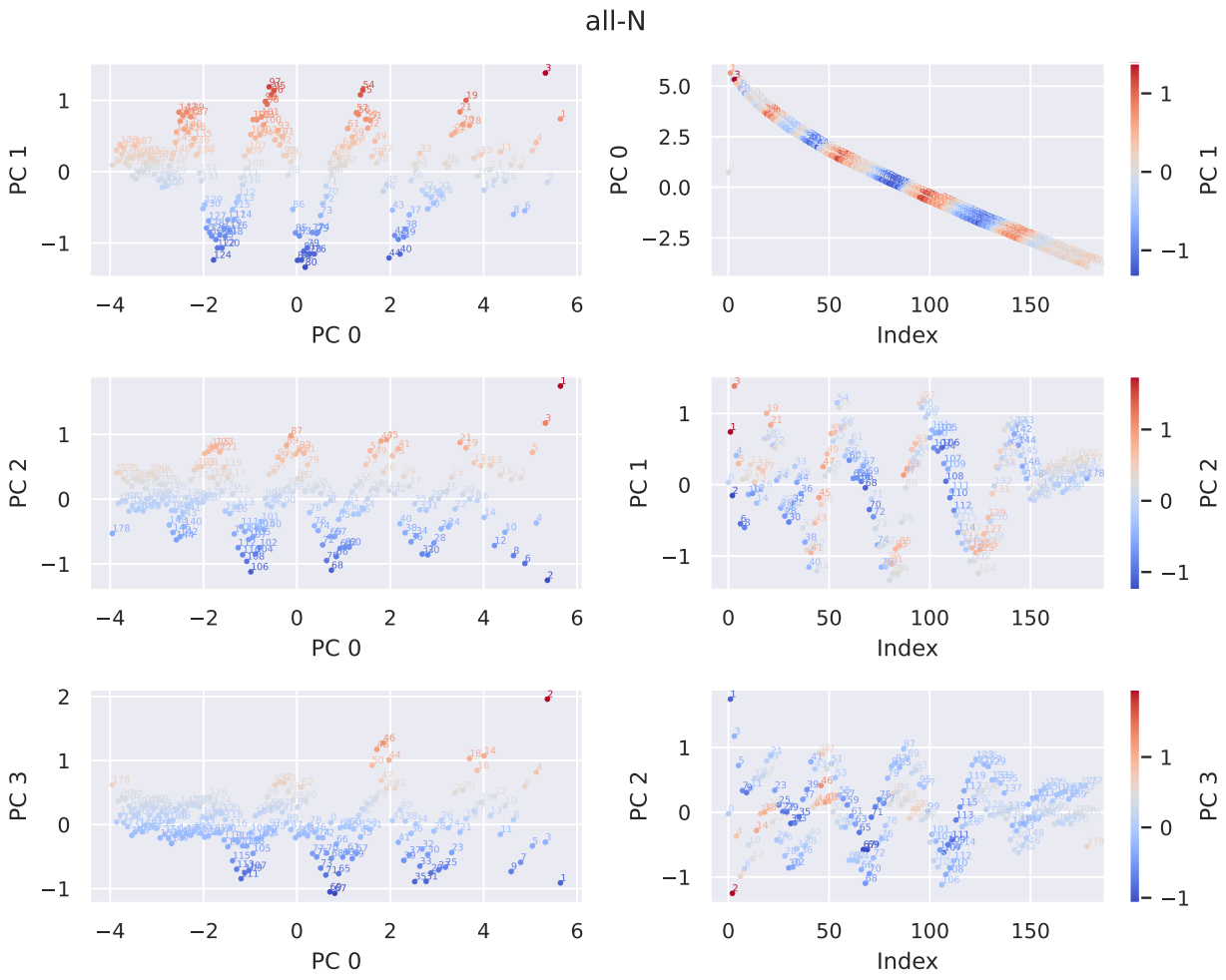


Figure B.8: First few PC projections of the N embeddings for a model trained on “all" data *i.e.*, in the multi-task setting.

B.5 Penultimate layer features

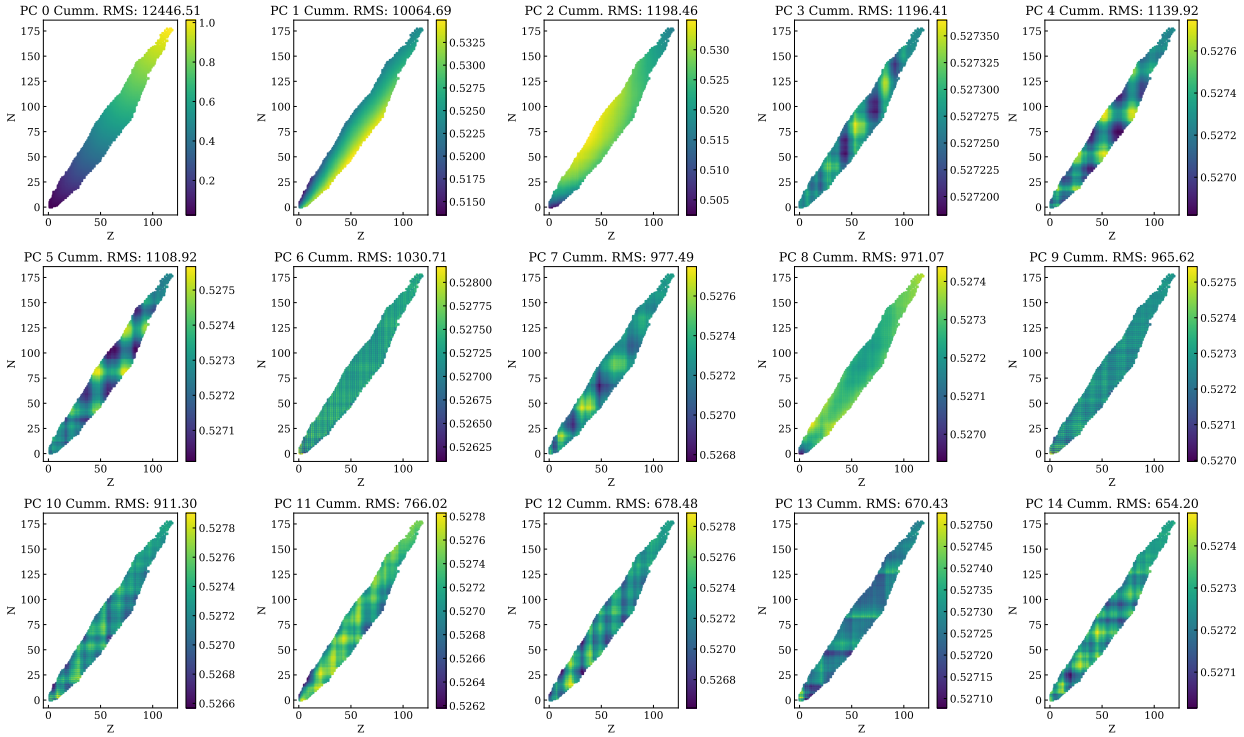


Figure B.9: Visualization of of a few penultimate layer PC features and their cumulative effect on the error in binding energy prediction (the error is computed up to and including the PC).

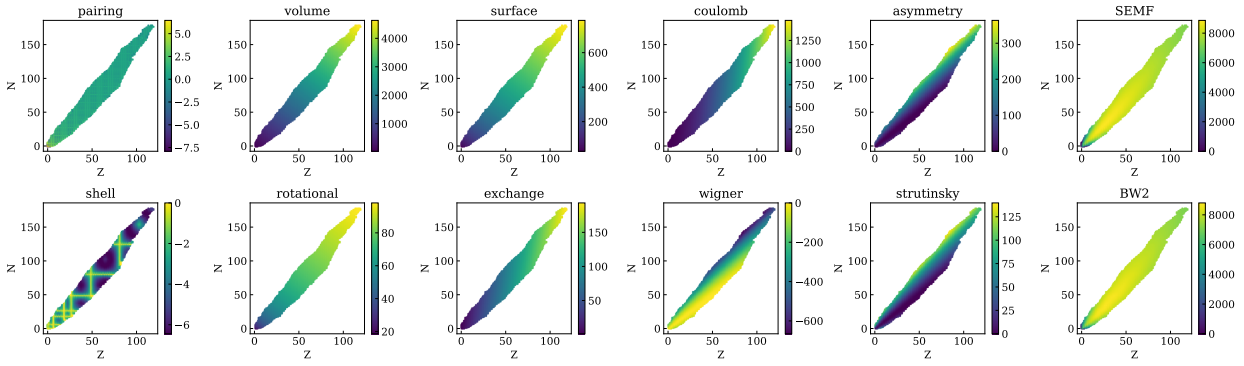


Figure B.10: Physics terms visualized. The top row are the terms from the SEMF. The bottom row includes nuclear shell model corrections (BW2 terms).

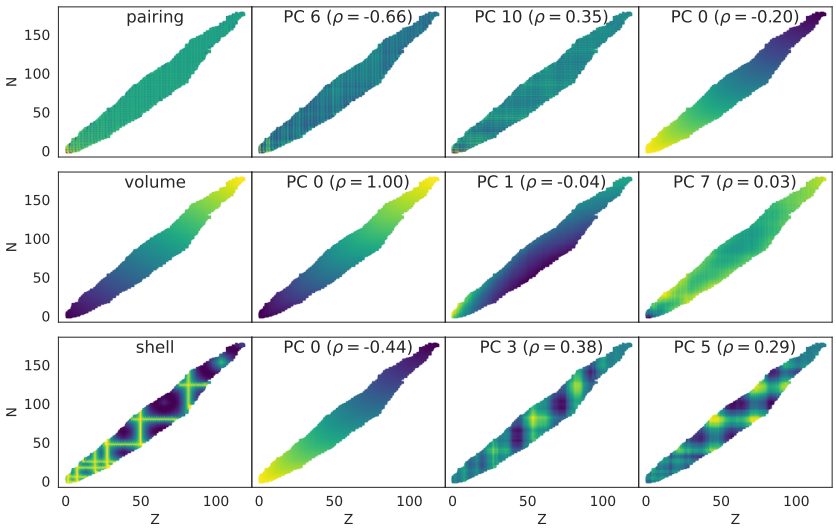


Figure B.11: Model penultimate features in the multi-task setting. Physical terms derived from the Nuclear Shell Model and their best matching PCs.

B.6 Other structures

We discussed how the helix structure (essentially stacked circles) is ideal to model the continuous spectrum of binding energies. However, continuity can be realized in other ways than in a circle (or helix when considering PC0), for instance by a simple line. In fact, we believe that the circular structure is chosen by the model because weight decay favors a continuous structure if it revolves around 0. A circular structure presents a good trade-off between embedding weight norm and sufficient distance between elements to form separate predictions for each Z or N without resorting to high weight norm in other layers. Figure B.12 shows N embedding projections from a model trained without weight decay, but with somewhat comparable test set performance. As hypothesized, a continuous structure emerges, but no helix. This behaviour is conceptually consistent over different random seeds.

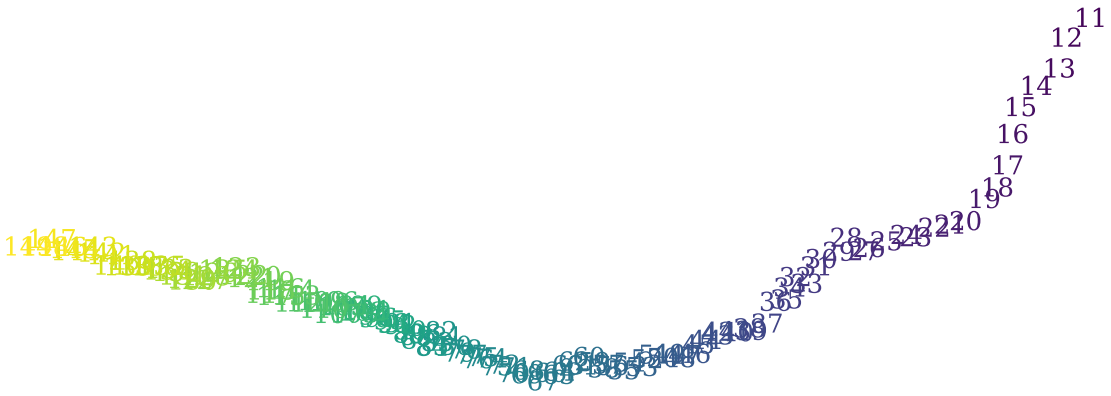


Figure B.12: Neutron embeddings projected into the first two PC from a model trained without weight decay.

B.7 Symbolic regression

We use symbolic regression to find functions $f^i_{\mathrm{PC}}(Z, N)$ that map from Z and N to the i-th feature extracted from the penultimate layer. We use the `PySR` library ([190]), which employs an evolutionary tree-based algorithm.[2],

Subsequently, we may write the new expression for the binding energy as $E_B = \sum_{i=1}^{n_F} a_i f^i_{\mathrm{PC}}(Z, N) + b$, where n_F is the number of PC features that are used. The coefficients a_i and the intercept b are determined using linear regression on the binding energy dataset without the TMS values. We find that the using the fits of solely PC0 and PC2, we can retain the bulk of the prediction. The new expression for binding energy reads,

$$E_B = a_1 \left(-0.09 + 10^{-6} Z^2\right) \left[A + 2.5 \sin\left(0.25 - 0.13N + 0.2Z\right)\right] + a_2 0.97^N + b \; . \tag{B.4}$$

[2]In the physical sciences, this method has proven useful for extracting symbolic formulas that reveal new physical patterns or reinterpret known physical laws [189], [280], [281].

where $a_1 = -88062.52$, $a_2 = -171331.53$ and $b = 95815.44$. This formula achieves an RMS of around 4600 keV. As a comparison, the performance of the SEMF over the same dataset is 8000 keV. Noteably, any direct regression on the data leads to considerably worse predictions for the same number of free parameters. We assess thus, that the analysis of the representation space of neural networks may streamline symbolic regression tasks.

B.8 Limitations

The interpretability of the extracted knowledge is not guaranteed. Even if the network finds a low-rank structure, it may not necessarily correspond to a simple, interpretable theory that provides clear insight to domain experts. The learned representations might capture complex, nonlinear interactions that are hard to distill into compact, explainable expressions. Moreover, there is currently a lack of quantitative metrics to assess the interpretability of the extracted knowledge. Developing such metrics is crucial, as that which is measured can be improved. Without a way to quantify interpretability, it becomes challenging to track progress and iterate on techniques to enhance the clarity and usefulness of the derived insights for domain experts. As seen in the attempts at symbolic regression, the expressions recovered from the neural features did not yield fully interpretable improvements over human-derived models. This limitation highlights the need for more rigorous metrics to guide the search for more explainable and meaningful representations of the learned knowledge.

Additionally, integrating MI into the scientific discovery workflow requires interdisciplinary collaborations and close partnerships between machine learning researchers and domain experts. Translating between the language of neural network components and the scientific concepts of a given field is a significant challenge that demands dedicated effort from both sides to have a real-world impact in driving scientific progress.

Appendix C

Grokking

C.1 Definitions of the phases of learning

Table C.1: Definitions of the four phases of learning

Phase	criteria		
	training acc $> 90\%$ within 10^5 steps	validation acc $> 90\%$ within 10^5 steps	step(validation acc$>90\%$) $-$step(training acc$>90\%$)$<10^3$
Comprehension	Yes	Yes	Yes
Grokking	Yes	Yes	No
Memorization	Yes	No	Not Applicable
Confusion	No	No	Not Applicable

C.2 Applicability of our toy setting

In the main paper, we focused on the toy setting with (1) the addition dataset and (2) the addition operation hard coded in the decoder. Although both simplifications appear to have quite limited applicability, we argue below that the analysis of the toy setting can actually apply to all Abelian groups.

The addition dataset is the building block of all Abelian groups A cyclic group is a group that is generated by a single element. A finite cyclic group with order n is $C_n = \{e, g, g^2, \cdots, g^{n-1}\}$ where e is the identify element and g is the generator and $g^i = g^j$ whenever $i = j \pmod n$. The modulo addition and $\{0, 1, \cdots, n-1\}$ form a cyclic group with $e = 0$ and g can be any number q coprime to n such that $(q, n) = 1$. Since algorithmic datasets contain only symbolic but no arithmetic information, the datasets of modulo addition could apply to all other cyclic groups, e.g., modulo multiplication and discrete rotation groups in 2D.

Although not all Abelian groups are cyclic, a finite Abelian group G can be always decomposed into a direct product of k cyclic groups $G = C_{n_1} \times C_{n_2} \cdots C_{n_k}$. So after training

k neural networks with each handling one cyclic group separately, it is easy to construct a larger neural network that handles the whole Abelian group.

The addition operation is valid for all Abelian groups It is proved in [282] that for a permutation invariant function $f(x_1, x_2, \cdots, x_n)$, there exists ρ and ϕ such that

$$f(x_1, x_2, \cdots, x_n) = \rho[\sum_{i=1}^{n} \phi(x_i)], \tag{C.1}$$

or $f(x_1, x_2) = \rho(\phi(x_1) + \phi(x_2))$ for $n = 2$. Notice that $\phi(x_i)$ corresponds to the embedding vector $\mathbf{E}_i$, ρ corresponds to the decoder. The addition operator naturally emerges from the commutativity of the operator, not restricting the operator itself to be addition. For example, multiplication of two numbers x_1 and x_2 can be written as $x_1 x_2 = \exp(\ln(x_1) + \ln(x_2))$ where $\rho(x) = \exp(x)$ and $\phi(x) = \ln(x)$.

C.3 An illustrative example

We use a concrete case to illustrate why parallelograms lead to generalization (see Figure C.1). For the purpose of illustration, we exploit a curriculum learning setting, where a neural network is fed with a few new samples each time. We will illustrate that, as we have more samples in the training set, the ideal model $\mathcal{M}^*$ (defined in Section 6.3.2) will arrange the representation $\mathbf{R}^*$ in a more structured way, i.e., more parallelograms are formed, which helps generalization to unseen validation samples. For simplicity we choose $p = 6$.

- $D_1 = (0, 4)$ and $D_2 = (1, 3)$ have the same label, so $(0, 4, 1, 3)$ becomes a parallelogram such that $\mathbf{E}_0 + \mathbf{E}_4 = \mathbf{E}_1 + \mathbf{E}_3 \to \mathbf{E}_3 - \mathbf{E}_0 = \mathbf{E}_4 - \mathbf{E}_1$. $D_3 = (1, 5)$ and $D_4 = (2, 4)$ have the same label, so $(1, 5, 2, 4)$ becomes a parallelogram such that $\mathbf{E}_1 + \mathbf{E}_5 = \mathbf{E}_2 + \mathbf{E}_4 \to \mathbf{E}_4 - \mathbf{E}_1 = \mathbf{E}_5 - \mathbf{E}_2$. We can derive from the first two equations that $\mathbf{E}_5 - \mathbf{E}_2 = \mathbf{E}_3 - \mathbf{E}_0 \to \mathbf{E}_0 + \mathbf{E}_5 = \mathbf{E}_2 + \mathbf{E}_3$, which implies that $(0, 5, 2, 3)$ is also a parallelogram (see Figure C.1(a)). This means if $(0, 5)$ in training set, our model can predict $(2, 3)$ correctly.

- $D_5 = (0, 2)$ and $D_6 = (1, 1)$ have the same label, so $\mathbf{E}_0 + \mathbf{E}_2 = 2\mathbf{E}_1$, i.e., 1 is the middle point of 0 and 2 (see Figure C.1(b)). Now we can derive that $2\mathbf{E}_4 = \mathbf{E}_3 + \mathbf{E}_5$, i.e., 4 is the middle point of 3 and 5. If $(4, 4)$ is in the training data, our model can predict $(3, 5)$ correctly.

- Finally, $D_7 = (2, 4)$ and $D_8 = (3, 3)$ have the same label, so $2\mathbf{E}_3 = \mathbf{E}_2 + \mathbf{E}_4$, i.e., 3 should be placed at the middle point of 2 and 4, ending up Figure C.1(c). This linear structure agrees with the arithmetic structure of $\mathbb{R}$.

In summary, although we have $p(p + 1)/2 = 21$ different training samples for $p = 6$, we only need 8 training samples to uniquely determine the perfect linear structure (up to linear transformation). The punchline is: representations lead to generalization.

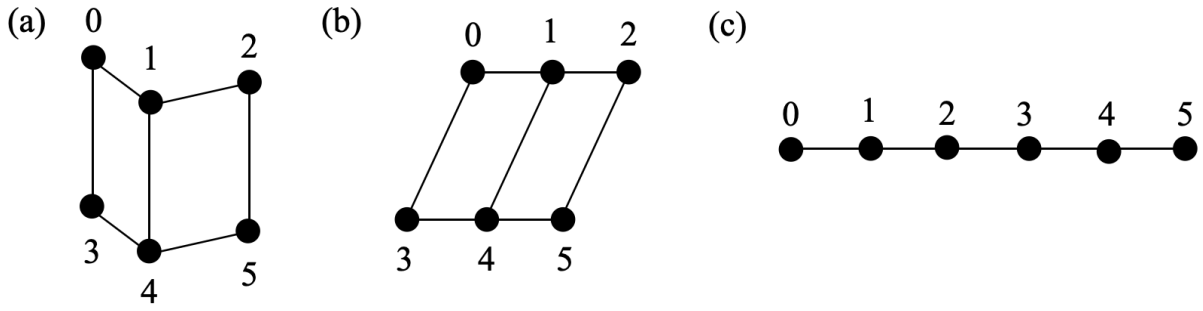


Figure C.1: As we include more data in the training set, the (ideal) model is capable of discovering increasingly structured representations (better RQI), from (a) to (b) to (c).

C.4 Definition of $\widehat{\text{Acc}}$

Given a training set D and a representation $\mathbf{R}$, if (i, j) is a validation sample, can the neural network correctly predict its output, i.e., $\text{Dec}(\mathbf{E}_i + \mathbf{E}_j) = \mathbf{Y}_{i+j}$? Since neural network has never seen (i, j) in the training set, one possible mechanism of induction is through

$$\text{Dec}(\mathbf{E}_i + \mathbf{E}_j) = \text{Dec}(\mathbf{E}_m + \mathbf{E}_n) = \mathbf{Y}_{m+n} (= \mathbf{Y}_{i+j}). \tag{C.2}$$

The first equality $\text{Dec}(\mathbf{E}_i + \mathbf{E}_j) = \text{Dec}(\mathbf{E}_m + \mathbf{E}_n)$ holds only when $\mathbf{E}_i + \mathbf{E}_j = \mathbf{E}_m + \mathbf{E}_n$ (i.e., (i, j, m, n) is a parallelogram). The second equality $\text{Dec}(\mathbf{E}_\text{m} + \mathbf{E}_\text{n}) = \mathbf{Y}_{\text{m+n}}$, holds when (m, n) is in the training set, i.e., $(m, n) \in D$, under the zero training loss assumption. Rigorously, given a training set D and a parallelogram set P (which can be calculated from $\mathbf{R}$), we collect all zero loss samples in an *augmented* training set $\overline{D}$

$$\overline{D}(D, P) = D \bigcup \{(i, j) | \exists (m, n) \in D, (i, j, m, n) \in P\}. \tag{C.3}$$

Keeping D fixed, a larger P would probably produce a larger $\overline{D}$, i.e., if $P_1 \subseteq P_2$, then $\overline{D}(D, P_1) \subseteq \overline{D}(P, P_2)$, which is why in Eq. (6.3) our defined RQI $\propto |P|$ gets its name "representation quality index", because higher RQI normally means better generalization. Finally, the expected accuracy from a dataset D and a parallelogram set P is:

$$\widehat{\text{Acc}} = \frac{|\overline{D}(D, P)|}{|D_0|}, \tag{C.4}$$

which is the estimated accuracy (of the full dataset), and $P = P(\mathbf{R})$ is defined on the representation after training. On the other hand, accuracy Acc can be accessed empirically from trained neural network. We verified $\text{Acc} \approx \widehat{\text{Acc}}$ in a toy setup (addition dataset $p = 10$, 1D embedding space, hard code addition), as shown in Figure 6.3 (c). Figure 6.3 (a)(b) show Acc and $\widehat{\text{Acc}}$ as a function of training set ratio, with each dot corresponding to a different random seed. The dashed red diagonal corresponds to memorization of the training set, and the vertical gap refers to generalization.

Although the agreement is good for 1D embedding vectors, we do not expect such agreement can trivially extend to high dimensional embedding vectors. In high dimensions,

our definition of RQI is too restrictive. For example, suppose we have an embedding space with N dimensions. Although the representation may form a linear structure in the first dimension, the representation can be arbitrary in other $N-1$ dimensions, leading to RQI ≈ 0. However, the model may still generalize well if the decoder learns to keep only the useful dimension and drop all other $N-1$ useless dimensions. It would be interesting to investigate how to define an RQI that takes into account the role of decoder in future works.

C.5 The gap of a realistic model $\mathcal{M}$ and the ideal model $\mathcal{M}^*$

Realistic models $\mathcal{M}$ usually form fewer number of parallelograms than ideal models $\mathcal{M}^*$. In this section, we analyze the properties of ideal models and calculated ideal RQI and ideal accuracy, which set upper bounds for empirical RQI and accuracy. The upper bound relations are verified via numerical experiments in Figure C.2.

Similar to Eq. (C.3) where some validation samples can be derived from training samples, we demonstrate how *implicit parallelograms* can be 'derived' from explicit ones in $P_0(D)$. The so-called derivation follows a simple geometric argument that: if A_1B_1 is equal and parallel to A_2B_2, and A_2B_2 is equal and parallel to A_3B_3, then we can deduce that A_1B_1 is equal and parallel to A_3B_3 (hence (A_1, B_2, A_2, B_1) is a parallelogram).

Recall that a parallelogram (i, j, m, n) is equivalent to $\mathbf{E}_i + \mathbf{E}_j = \mathbf{E}_m + \mathbf{E}_n$ $(*)$. So we are equivalently asking if equation $(*)$ can be expressed as a linear combination of equations in $A(P_0(D))$. If yes, then $(*)$ is dependent on $A(P_0(D))$ (defined in Eq. (6.7)), i.e., $A(P_0(D))$ and $A(P_0(D) \bigcup (i, j, m, n))$ should have the same rank. We augment $P_0(D)$ by adding implicit parallelograms, and denote the augmented parallelogram set as

$$P(D) = P_0(D) \bigcup \{q \equiv (i, j, m, n) | q \in P_0, \text{rank}(A(P_0(D))) = \text{rank}(A(P_0(D) \bigcup q))\}. \quad \text{(C.5)}$$

We need to emphasize that an assumption behind Eq. (C.5) is that we have an ideal model $\mathcal{M}^*$. When the model is not ideal, e.g., when the injectivity of the encoder breaks down, fewer parallelograms are expected to form, i.e.,

$$P(R) \subseteq P(D). \quad \text{(C.6)}$$

The inequality is saying, whenever a parallelogram is formed in the representation after training, the reason is hidden in the training set. This is not a strict argument, but rather a belief that today's neural networks can only copy what datasets (explicitly or implicitly) tell it to do, without any autonomous creativity or intelligence. For simplicity we call this belief *Alexander Principle*. In very rare cases when something lucky happens (e.g., neural networks are initialized at approximate correct weights), Alexander principle may be violated. Alexander principle sets an upper bound for RQI:

$$\text{RQI}(R) \leq \frac{|P(D)|}{|P_0|} \equiv \overline{\text{RQI}}, \quad \text{(C.7)}$$

and sets an upper bound for $\widehat{\text{Acc}}$:

$$\widehat{\text{Acc}} \equiv \widehat{\text{Acc}}(D, P(R)) \leq \widehat{\text{Acc}}(D, P(D)) \equiv \overline{\text{Acc}}. \quad \text{(C.8)}$$

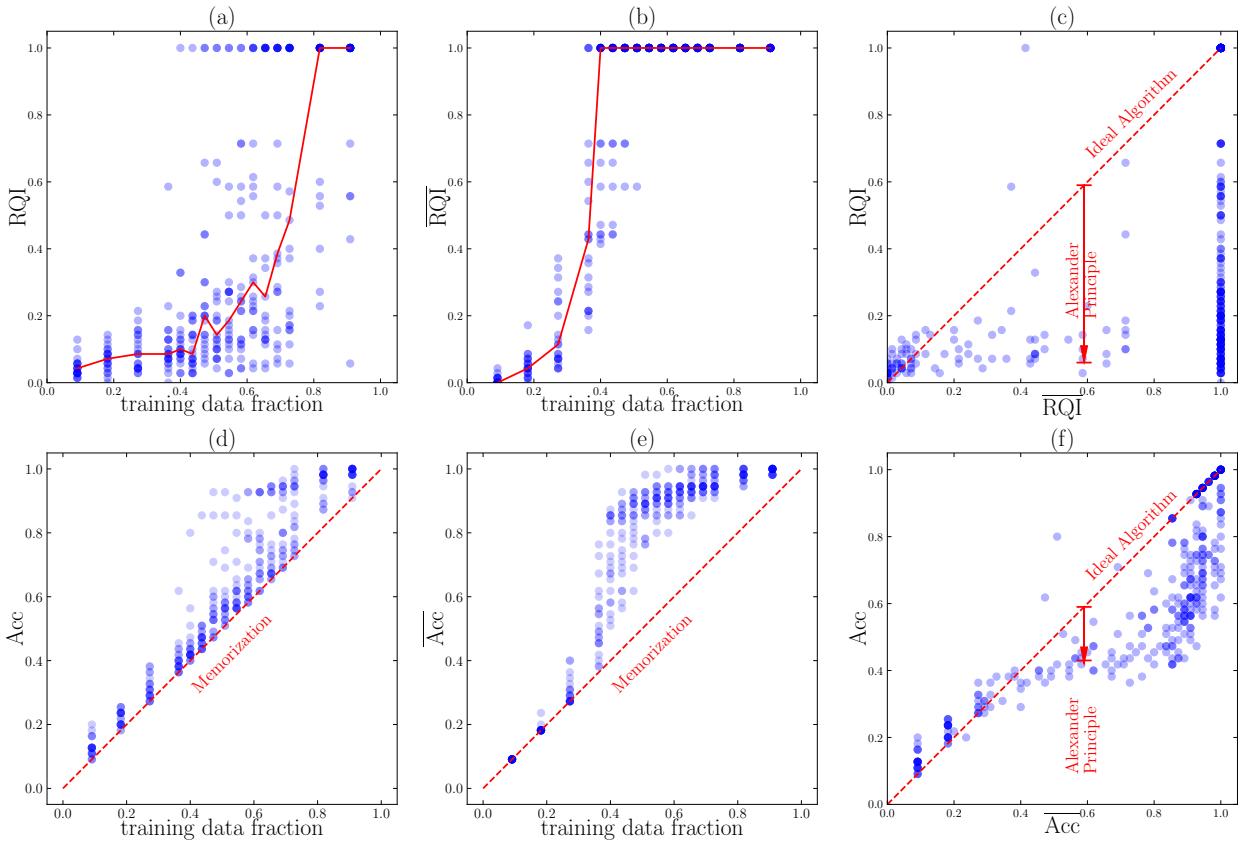


Figure C.2: We compare RQI and Acc for an ideal algorithm (with bar) and a realistic algorithm (without bar). In (a)(b)(d)(e), four quantities (RQI, $\overline{\text{RQI}}$, Acc, $\overline{\text{Acc}}$) as functions of training data fraction are shown. In (c)(f), RQI and Acc of the ideal algorithm sets upper bounds for those of the realistic algorithm.

In Figure C.2 (c)(f), we verify Eq. (C.7) and Eq. (C.8). We choose $\delta = 0.01$ to compute RQI(R,δ). We find the trained models are usually far from being ideal, although we already include a few useful tricks proposed in Section 6.4 to enhance representation learning. It would be an interesting future direction to develop better algorithms so that the gap due to Alexander principle can be reduced or even closed. In Figure C.2 (a)(b)(d)(e), four quantities (RQI, $\overline{\text{RQI}}$, Acc, $\overline{\text{Acc}}$) as functions of the training data fraction are shown, each dot corresponding to one random seed. It is interesting to note that it is possible to have $\overline{\text{RQI}} = 1$ only with $< 40\%$ training data, i.e., $55 \times 0.4 = 22$ samples, agreeing with our observation in Section 6.3.

Realistic representations Suppose an ideal model $\mathcal{M}^*$ and a realistic model $\mathcal{M}$ which train on the training set D give the representation R^* and R, respectively. What is the relationship between R and R_*? Due to the Alexander principle we know $P(R) \subseteq P(D) = P(R^*)$. This means R^* has more parallelograms than R, hence R^* has fewer degrees of freedom than R.

We illustrate with the toy case $p = 4$. The whole dataset contains $p(p+1)/2 = 10$ samples, i.e.,

$$D_0 = \{(0,0),(0,1),(0,2),(0,3),(1,1),(1,2),(1,3),(2,2),(2,3),(3,3)\}. \tag{C.9}$$

The parallelogram set contains only three elements, i.e.,

$$P_0 = \{(0,1,1,2),(0,1,2,3),(1,2,2,3)\}, \tag{C.10}$$

Or equivalently the equation set

$$A_0 = \{\text{A1} : \mathbf{E}_0 + \mathbf{E}_2 = 2\mathbf{E}_1, \text{A2} : \mathbf{E}_0 + \mathbf{E}_3 = \mathbf{E}_1 + \mathbf{E}_2, \text{A3} : \mathbf{E}_1 + \mathbf{E}_3 = 2\mathbf{E}_2\}. \tag{C.11}$$

Pictorially, we can split all possible subsets $\{A | A \subseteq A_0\}$ into different levels, each level defined by $|A|$ (the number of elements). A subset A_1 in the i^{th} level points an direct arrow to another subset A_2 in the $(i+1)^{\text{th}}$ level if $A_2 \subset A_1$, and we say A_2 is a child of A_1, and A_1 is a parent of A_2. Each subset A can determine a representation R with $n(A)$ degrees of freedom. So R should be a descendant of R_*, and $n(R_*) \leq n(R)$. Numerically, $n(A)$ is equal to the dimension of the null space of A.

Suppose we have a training set

$$D = \{(0,2),(1,1),(0,3),(1,2),(1,3),(2,2)\}, \tag{C.12}$$

and correspondingly $P(D) = P_0, A(P) = A_0$. So an ideal model $\mathcal{M}_*$ will have the linear structure $\mathbf{E}_k = \mathbf{a} + k\mathbf{b}$ (see Figure C.3 leftmost). However, a realistic model $\mathcal{M}$ may produce any descendants of the linear structure, depending on various hyperparameters and even random seeds.

In Figure C.4, we show our algorithms actually generates all possible representations. We have two settings: (1) fast decoder $(\eta_1, \eta_2) = (10^{-3}, 10^{-2})$ (Figure C.4 left), and (2) relatively slow decoder $(\eta_1, \eta_2) = (10^{-2}, 10^{-3})$ (Figure C.4) right). The relatively slow decoder produces better representations (in the sense of higher RQI) than a fast decoder, agreeing with our observation in Section 6.4.

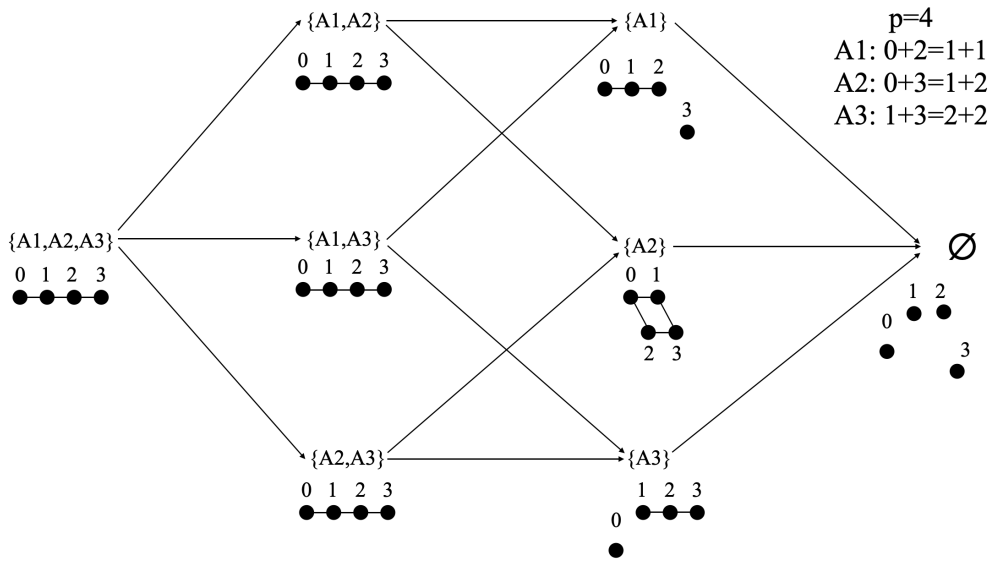


Figure C.3: $p = 4$ case. Equation set A (or geometrically, representation) has a hierarchy: $a \to b$ means a is a parent of b, and b is a child of a. A realistic model can only generate representations that are descendants of the representation generated by an ideal model.

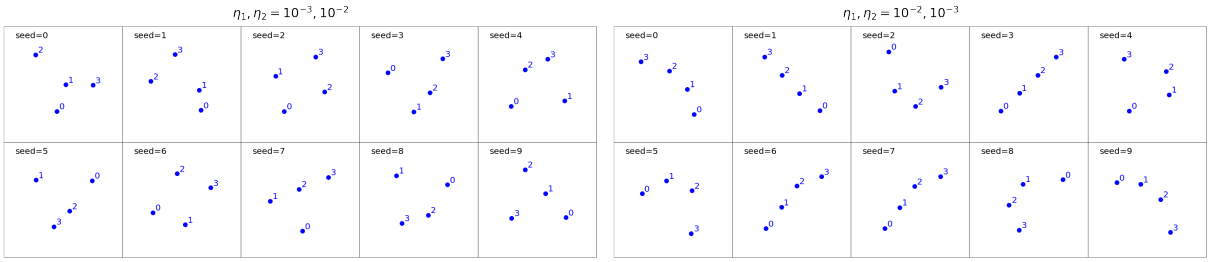


Figure C.4: $p = 4$ case. Representations obtained from training neural networks are displayed. η_1 and η_2 are learning rates of the representation and the decoder, respectively. As described in the main text, $(\eta_1, \eta_2) = (10^{-2}, 10^{-3})$ (right) is more ideal than $(\eta_1, \eta_2) = (10^{-3}, 10^{-2})$ (left), thus producing representations containing more parallelograms.

C.6 Conservation laws of the effective theory

Recall that the effective loss function

$$\ell_{\text{eff}} = \frac{\ell_0}{Z_0}, \quad \ell_0 \equiv \sum_{(i,j,m,n)\in P_0(D)} |\mathbf{E}_i + \mathbf{E}_j - \mathbf{E}_m - \mathbf{E}_n|^2/|P_0(D)|, \quad Z_0 \equiv \sum_k |\mathbf{E}_k|^2 \tag{C.13}$$

where ℓ_0 and Z_0 are both quadratic functions of $R = \{\mathbf{E}_0, \cdots, \mathbf{E}_{p-1}\}$, and $\ell_{\text{eff}} = 0$ remains zero under rescaling and translation $\mathbf{E}'_i = a\mathbf{E}_i + \mathbf{b}$. We will ignore the $1/|P_0(D)|$ factor in ℓ_0 since having it is equivalent to rescaing time, which does not affect conservation laws. The representation vector $\mathbf{E}_i$ evolves according to the gradient descent

$$\frac{d\mathbf{E}_i}{dt} = -\frac{\partial \ell_{\text{eff}}}{\partial \mathbf{E}_i}. \tag{C.14}$$

We will prove the following two quantities are conserved:

$$\mathbf{C} = \sum_k \mathbf{E}_k, \quad Z_0 = \sum_k |\mathbf{E}_k|^2. \tag{C.15}$$

Eq. (C.13) and Eq. (C.14) give

$$\frac{d\mathbf{E}_i}{dt} = -\frac{\ell_{\text{eff}}}{\partial \mathbf{E}_i} = -\frac{\partial(\frac{\ell_0}{Z_0})}{\partial \mathbf{E}_i} = -\frac{1}{Z_0}\frac{\partial \ell_0}{\partial \mathbf{E}_i} + \frac{\ell_0}{Z_0^2}\frac{\partial Z_0}{\partial \mathbf{E}_i}. \tag{C.16}$$

Then

$$\begin{aligned} \frac{dZ_0}{dt} &= 2\sum_i \mathbf{E}_k \cdot \frac{d\mathbf{E}_k}{dt} \\ &= \frac{2}{Z_0^2}\sum_i \mathbf{E}_i \cdot (-Z_0 \frac{\partial \ell_0}{\partial \mathbf{E}_k} + 2\ell_0 \mathbf{E}_k) \\ &= \frac{2}{Z_0}(-\sum_k \frac{\partial \ell_0}{\partial \mathbf{E}_k} \cdot \mathbf{E}_k + 2\ell_0) \\ &= 0. \end{aligned} \tag{C.17}$$

where the last equation uses the fact that

$$\begin{aligned} \sum_k \frac{\partial \ell_0}{\partial \mathbf{E}_k} \cdot \mathbf{E}_k &= 2\sum_k \sum_{(i,j,m,n)\in P_0(D)} (\mathbf{E}_i + \mathbf{E}_j - \mathbf{E}_m - \mathbf{E}_n)(\delta_{ik} + \delta_{jk} - \delta_{mk} - \delta_{nk}) \cdot \mathbf{E}_k \\ &= 2\sum_{(i,j,m,n)\in P_0(D)} (\mathbf{E}_i + \mathbf{E}_j - \mathbf{E}_m - \mathbf{E}_n) \sum_k (\delta_{ik} + \delta_{jk} - \delta_{mk} - \delta_{nk}) \cdot \mathbf{E}_k \\ &= \sum_{(i,j,m,n)\in P_0(D)} (\mathbf{E}_i + \mathbf{E}_j - \mathbf{E}_m - \mathbf{E}_n) \cdot (\mathbf{E}_i + \mathbf{E}_j - \mathbf{E}_m - \mathbf{E}_n) \\ &= 2\ell_0 \end{aligned}$$

The conservation of Z_0 prohibits the representation from collapsing to zero. Now that we have demonstrated that Z_0 is a conserved quantity, we can also show

$$\begin{aligned}\frac{d\mathbf{C}}{dt} &= \sum_k \frac{d\mathbf{E}_k}{dt} \\ &= -\frac{1}{Z_0}\sum_k \frac{\partial \ell_0}{\partial \mathbf{E}_k} \\ &= -\frac{2}{Z_0}\sum_k \sum_{(i,j,m,n)\in P_0(D)} (\mathbf{E}_i + \mathbf{E}_j - \mathbf{E}_m - \mathbf{E}_n)(\delta_{ik} + \delta_{jk} - \delta_{mk} - \delta_{nk}) \\ &= \mathbf{0}.\end{aligned} \tag{C.18}$$

The last equality holds because the two summations can be swapped and $\sum_k(\delta_{ik} + \delta_{jk} - \delta_{mk} - \delta_{nk}) = 0$.

C.7 More phase diagrams of the toy setup

We study another three hyperparameters in the toy setup by showing phase diagrams similar to Figure 6.6. The toy setup is: (1) addition without modulo ($p = 10$); (2) training/validation is split into 45/10; (3) hard code addition; (4) 1D embedding. In the following experiments, the decoder is an MLP with size 1-200-200-30. The representation and the encoder are optimized with AdamW with different hyperparameters. The learning rate of the representation is 10^{-3}. We sweep the learning rate of the decoder in range $[10^{-4}, 10^{-2}]$ as the x axis, and sweep another hyperparameter as the y axis. By default, we use full batch size 45, initialization scale $s = 1$ and zero weight decay of representation.

Batch size controls the amount of noise in the training dynamics. In Figure C.5, the grokking region appears at the top left of the phase diagram (small decoder learning rate and small batch size). However, large batch size (with small learning rate) leads to comprehension, implying that smaller batch size seems harmful. This makes sense since to get crystals (good structures) in experiments, one needs a freezer which gradually decreases temperature, rather than something perturbing the system with noise.

Initialization scale controls distances among embedding vectors at initialization. We initialize components of embedding vectors from independent uniform distribution $U[-s/2, s/2]$ where s is called the initialization scale. Shown in Figure C.6, it is beneficial to use a smaller initialization scale. This agrees with the physical intuition that closer particles are more likely to interact and form structures. For example, the distances among molecules in ice are much smaller than distances in gas.

Representation weight decay controls the magnitude of embedding vectors. Shown in Figure C.7, we see the representation weight decay in general does not affect model performance much.

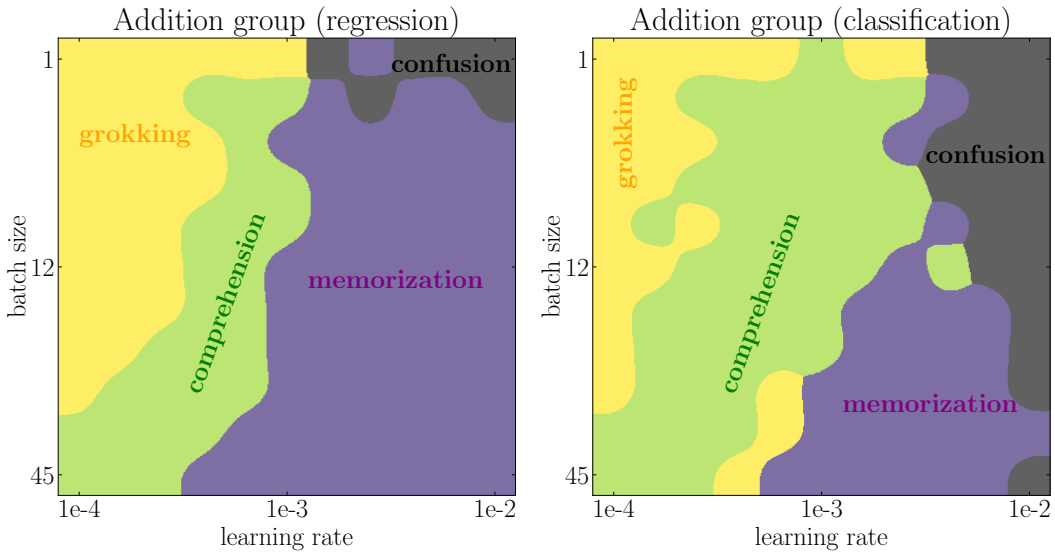


Figure C.5: Phase diagrams of decoder learning rate (x axis) and batch size (y axis) for the addition group (left: regression; right: classification). Small decoder leanrning rate and large batch size (bottom left) lead to comprehension.

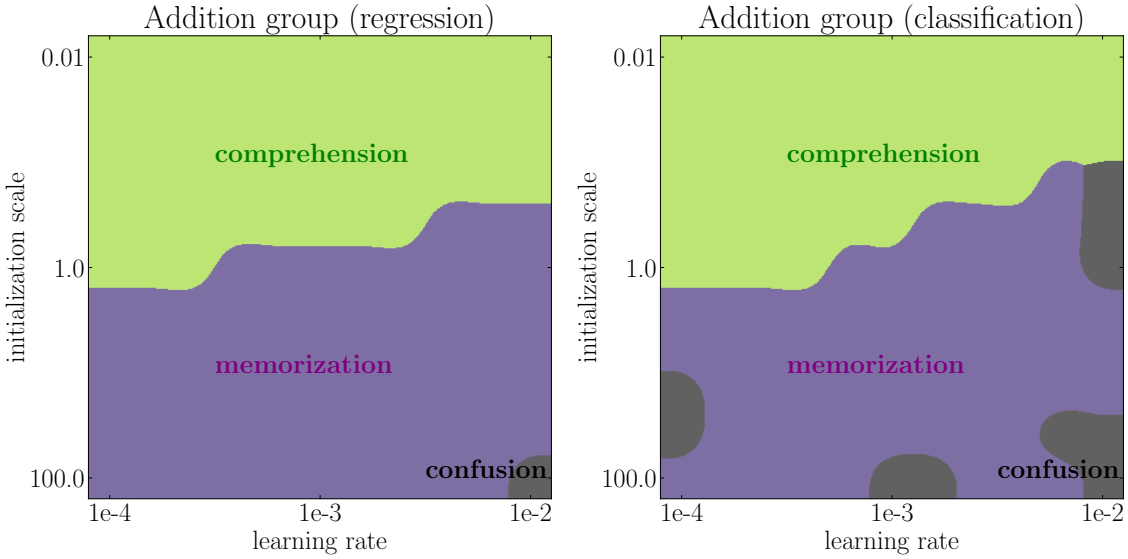


Figure C.6: Phase diagrams of decoder learning rate (x axis) and initialization (y axis) for the addition group (left: regression; right: classification). Small intialization scale (top) leads to comprehension.

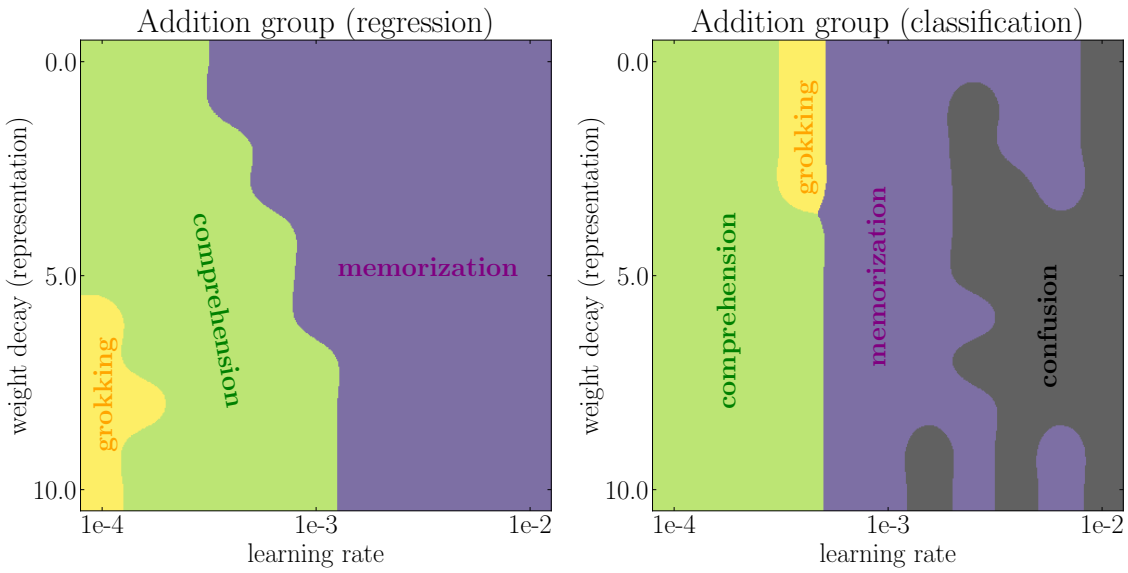


Figure C.7: Phase diagrams of decoder learning rate (x axis) and representation weight decay (y axis) for the addition group (left: regression; right: classification). Representation weight decay does not affect model performance much.

C.8 General groups

C.8.1 Theory

We focused on Abelian groups for the most part of the paper. This is, however, simply due to pedagogical reasons. In this section, we show that it is straight-forward to extend definitions of parallelograms and representation quality index (RQI) to general non-Abelian groups. We will also show that most (if not all) qualitative results for the addition group also apply to the permutation group.

Matrix representation for general groups Let us first review the definition of group representation. A representation of a group G on a vector space V is a group homomorphism from G to $\mathrm{GL}(V)$, the general linear group on V. That is, a representation is a map $\rho : G \to \mathrm{GL}(V)$ such that

$$\rho(g_1 g_2) = \rho(g_1)\rho(g_2), \quad \forall g_1, g_2 \in G. \tag{C.19}$$

In the case V is of finite dimension n, it is common to identify $\mathrm{GL}(V)$ with n by n invertible matrices. The punchline is that: each group element can be represented as a matrix, and the binary operation is represented as matrix multiplication.

A new architecture for general groups Inspired by the matrix representation, we embed each group element a as a learnable matrix $\mathbf{E}_a \in \mathbb{R}^{d\times d}$ (as opposed to a vector), and manually do matrix multiplication before sending the product to the deocder for regression or classification. More concretely, for $a \circ b = c$, our architecture takes as input two embedding matrices $\mathbf{E}_a$ and $\mathbf{E}_b$ and aims to predict $\mathbf{Y}_c$ such that $\mathbf{Y}_c = \mathrm{Dec}(\mathbf{E}_a\mathbf{E}_b)$, where $\mathbf{E}_a\mathbf{E}_b$ means the matrix multiplication of $\mathbf{E}_a$ and $\mathbf{E}_b$. The goal of this simplication is to disentangle learning

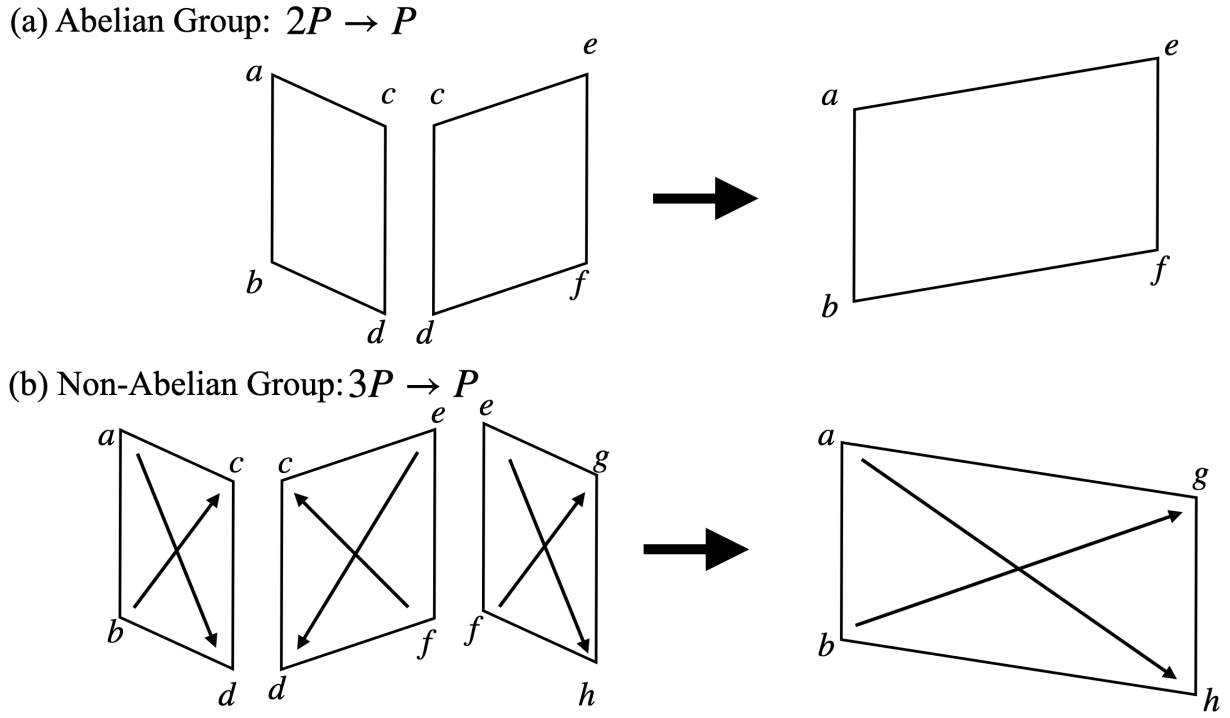


Figure C.8: Deduction of parallelograms

the representation and learning the arithmetic operation (i.e, the matrix multiplication). We will show that, even with this simplification, we are still able to reproduce the characteristic grokking behavior and other rich phenomenon.

Generalized parallelograms we define generalized parallelograms: (a, b, c, d) is a generalized parallelogram in the representation if $||\mathbf{E}_a\mathbf{E}_b - \mathbf{E}_c\mathbf{E}_d||_F^2 \leq \delta$, where $\delta > 0$ is a threshold to tolerate numerical errors. Before presenting the numerical results for the permutation group, we show an intuitive picture about how new parallelograms can be deduced from old ones for general groups, which is the key to generalization.

Deduction of parallelograms We first recall the case of the Abelian group (e.g., addition group). As shown in Figure C.8, when (a, d, b, c) and (c, f, d, e) are two parallelograms, we have

$$\begin{aligned} \mathbf{E}_a + \mathbf{E}_d &= \mathbf{E}_b + \mathbf{E}_c, \\ \mathbf{E}_c + \mathbf{E}_f &= \mathbf{E}_d + \mathbf{E}_d. \end{aligned} \tag{C.20}$$

We can derive that $\mathbf{E}_a + \mathbf{E}_f = \mathbf{E}_b + \mathbf{E}_e$ implying that (a, f, b, e) is also a parallelogram. That is, for Abelian groups, two parallelograms are needed to deduce a new parallelogram.

For the non-Abelian group, if we have only two parallelograms such that

$$\begin{aligned} \mathbf{E}_a\mathbf{E}_d &= \mathbf{E}_b\mathbf{E}_c, \\ \mathbf{E}_f\mathbf{E}_c &= \mathbf{E}_e\mathbf{E}_d, \end{aligned} \tag{C.21}$$

we have $\mathbf{E}_b^{-1}\mathbf{E}_a = \mathbf{E}_c\mathbf{E}_d^{-1} = \mathbf{E}_f^{-1}\mathbf{E}_e$, but this does not lead to something like $\mathbf{E}_f\mathbf{E}_a = \mathbf{E}_e\mathbf{E}_b$, hence useless for generalization. However, if we have a third parallelogram such that

$$\mathbf{E}_e\mathbf{E}_h = \mathbf{E}_f\mathbf{E}_g \tag{C.22}$$

we have $\mathbf{E}_b^{-1}\mathbf{E}_a = \mathbf{E}_c\mathbf{E}_d^{-1} = \mathbf{E}_f^{-1}\mathbf{E}_e = \mathbf{E}_g\mathbf{E}_h^{-1}$, equivalent to $\mathbf{E}_a\mathbf{E}_h = \mathbf{E}_b\mathbf{E}_g$, thus establishing a new parallelogram (a, h, b, g). That is, for non-Abelian groups, three parallelograms are needed to deduce a new parallelogram.

C.8.2 Numerical Results

In this section, we conduct numerical experiments on a simple non-abelian group: the permutation group S_3. The group has 6 group elements, hence the full dataset contains 36 samples. We embed each group element a into a learnable 3×3 embedding matrix $\mathbf{E}_a$. We adopt the new architecture described in the above subsection: we hard code matrix multiplication of two input embedding matrices before feeding to the decoder. After defining the generalized parallelogram in the last subsection, we can continue to define RQI (as in Section 6.3) and predict accuracy $\widehat{\text{Acc}}$ from representation (as in appendix C.4). We also compute the number of steps needed to reach RQI $= 0.95$.

Representation We flatten each embedding matrix into a vector, and apply principal component analysis (PCA) to the vectors. We show the first three principal components of these group elements in Figure C.9. On the plane of PC1 and PC3, the six points are organized as a hexagon.

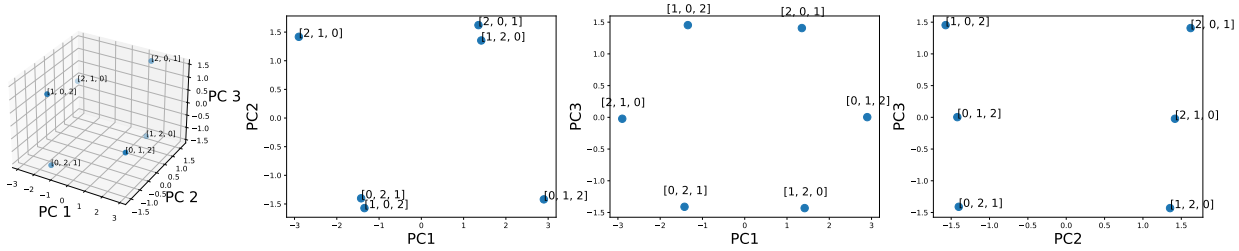


Figure C.9: Permuation group S_3. First three principal components of six embedding matrices $\mathbb{R}^{3\times3}$.

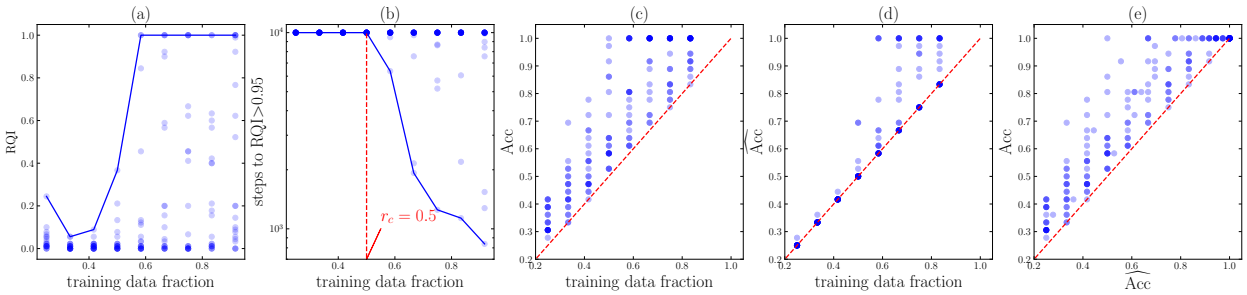


Figure C.10: Permutation group S_3. (a) RQI increases as training set becomes larger. Each scatter point is a random seed, and the blue line is the highest RQI obtained with a fixed training set ratio; (b) steps to reach RQI > 0.95. The blue line is the smallest number of steps required. There is a phase transition around $r_c = 0.5$. (c) real accuracy Acc; (d) predicted accuracy $\widehat{\text{Acc}}$; (e) comparison of Acc and $\widehat{\text{Acc}}$: $\widehat{\text{Acc}}$ serves as a lower bound of Acc.

RQI In Figure C.10 (a), we show RQI as a function of training data fraction. For each training data fraction, we run 11 random seeds (shown as scatter points), and the blue line corresponds to the highest RQI.

Steps to reach RQI$= 0.95$ In Figure C.10 (b), we whow the steps to reach RQI > 0.95 as a function of training data fraction, and find a phase transition at $r = r_c = 0.5$. The blue line corresponds to the best model (smallest number of steps).

Accuracy The real accuracy Acc is shown in Figure C.10 (c), while the predicted accuracy $\widehat{\text{Acc}}$ (calculated from RQI) is shown in Figure C.10 (d). Their comparison is shown in (e): $\widehat{\text{Acc}}$ is a lower bound of Acc, implying that there must be some generalization mechanism beyond RQI.

Phase diagram We investigate how the model performance varies under the change of two knobs: decoder learning rate and decoder weight decay. We calculate the number of steps to training accuracy ≥ 0.9 and validation accuracy ≥ 0.9, respectively, shown in Figure 6.6 (d).

C.9 Effective theory for image classification

In this section, we show our effective theory proposed in Section 6.3.2 can generalize beyond algorithmic datasets. In particular, we will apply the effective theory to image classifications. We find that: (i) The effective theory naturally gives rise to a novel self-supervised learning method, which can provably avoid mode collapse without contrastive pairs. (ii) The effective theory can shed light on the neural collapse phenomenon [283], in which same-class representations collapse to their class-means.

We first describe how the effective theory applies to image classification. The basic idea is again that, similar to algorithmic datasets, neural networks try to develop a structured representation of the inputs based on the relational information between samples (class labels in the case of image classification, sum parallelograms in the case of addition, etc.). The effective theory has two ingredients: (i) samples with the same label are encouraged to have similar representations; (ii) the effective loss function is scale-invariant to avoid all representations collapsing to zero (global collapse). As a result, an effective loss for image classification has the form

$$\ell_{\text{eff}} = \frac{\ell}{Z}, \quad \ell = \sum_{(\mathbf{x},\mathbf{y})\in P} |\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y})|^2, \quad Z = \sum_{\mathbf{x}} |\mathbf{f}(\mathbf{x})|^2 \tag{C.23}$$

where $\mathbf{x}$ is an image, $\mathbf{f}(\mathbf{x})$ is its representation, $(\mathbf{x}, \mathbf{y}) \in P$ refers to unique pairs $\mathbf{x}$ and $\mathbf{y}$ that have the same label. Scale invariance means the loss function ℓ_{eff} does not change under the linear scaling $\mathbf{f}(\mathbf{x}) \to a\mathbf{f}(\mathbf{x})$.

Relation to neural collapse It was observed in [283] that image representations in the penultimate layer of the model have some interesting features: (i) representations of same-class images collapse to their class-means; (ii) class-means of different classes develop into an equiangular tight frame. Our effective theory is able to predict the same-class collapse, but does not necessarily put class-means into equiangular tight frames. We conjecture that little explicit repulsion among different classes can help class-means develop into an equiangular

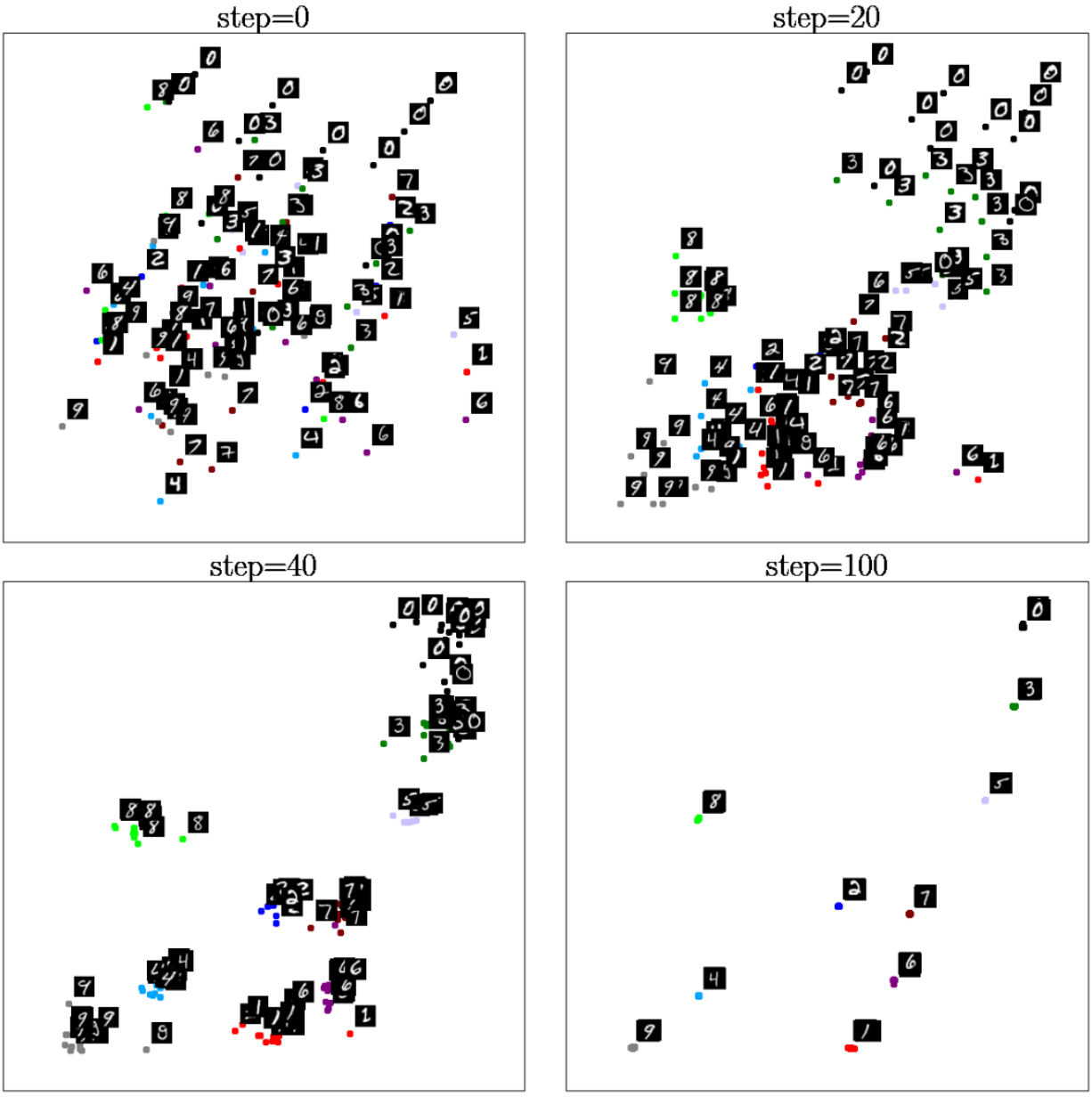


Figure C.11: Our effective theory applies to MNIST image classifications. Same-class images collapse to their class-means, while class-means of different classes stay separable. As such, the effective theory serves as a novel self-supervised learning method, as well as shed some light on neural collapse. Please see texts in Appendix C.9.

tight frame, similar to electrons developing into lattice structures on a sphere under repulsive Coulomb forces (the Thomson problem [284]). We would like to investigate this modification of the effective theory in the future.

Experiment on MNIST We directly apply the effective loss Eq. (C.23) to the MNIST dataset. Firstly, each image $\mathbf{x}$ is randomly encoded to a 2D embedding $\mathbf{f}(\mathbf{x})$ via the same encoder MLP whose weights are randomly initialized. We then train these embeddings by minimizing the effective loss ℓ_{eff} with an Adam optimizer (10^{-3} learning rate) for 100 steps. We show the evolution of these embeddings in Figure C.11. Images of the same class collapse to their class-means, and different class-means do not collapse. This means that our effective theory can give rise to a good representation learning method which only exploits non-contrastive relational information in datasets.

Link to self-supervised learning Note that ℓ itself is vulnerable to global collapse, in the context of Siamese learning without contrastive pairs. Various tricks (e.g., decoder with momentum, stop gradient) [209], [285] have been proposed to avoid global collapse. However, the reasons why these tricks can avoid global collapse are unclear. We argue ℓ fails simply because $\ell \to a^2\ell$ under scaling $\mathbf{f}(\mathbf{x}) \to a\mathbf{f}(\mathbf{x})$ so gradient descent on ℓ encourage $a \to 0$. Based on this picture, our effective theory provides another possible fix: make the loss function ℓ scale-invariant (by the normalized loss ℓ_{eff}), so the gradient flow has no incentive to change representation scales. In fact, we can prove that the gradient flow on ℓ_{eff} preserve Z (variance of representations) so that global collapse is avoided provably:

$$\begin{aligned}
\frac{\partial \ell_{\text{eff}}}{\partial \mathbf{f}(\mathbf{x})} &= \frac{1}{Z}\frac{\partial \ell}{\partial \mathbf{f}(\mathbf{x})} - \frac{l}{Z^2}\frac{\partial Z}{\partial \mathbf{f}(\mathbf{x})} = \frac{2}{Z}\sum_{\mathbf{y}\sim\mathbf{x}}(\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y})) - \frac{2\ell}{Z^2}\mathbf{f}(\mathbf{x}), \\
\frac{dZ}{dt} &= 2\sum_{\mathbf{x}} \mathbf{f}(\mathbf{x}) \cdot \frac{d\mathbf{f}(\mathbf{x})}{dt} = 2\sum_{\mathbf{x}} \mathbf{f}(\mathbf{x}) \cdot \frac{\partial \ell_{\text{eff}}}{\partial \mathbf{f}(\mathbf{x})} \\
&= \frac{4}{Z}\sum_{\mathbf{x}} \mathbf{f}(\mathbf{x}) \cdot (\sum_{\mathbf{y}\sim\mathbf{x}}(\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y})) - \frac{\ell}{Z}\mathbf{f}(\mathbf{x})) \\
&= \frac{4}{Z}\big[\sum_{\mathbf{x}} \mathbf{f}(\mathbf{x}) \cdot \sum_{\mathbf{y}\sim\mathbf{x}}(\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y})) - \sum_{\mathbf{x}} \frac{\ell}{Z}|\mathbf{f}(\mathbf{x})|^2\big] \\
&= 0.
\end{aligned} \tag{C.24}$$

where we use the fact that

$$\sum_{\mathbf{x}} \mathbf{f}(\mathbf{x}) \cdot \sum_{\mathbf{y}\sim\mathbf{x}}(\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y})) = \sum_{(\mathbf{x},\mathbf{y})\in P} (\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y})) \cdot (\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y})) = \ell \tag{C.25}$$

C.10 Grokking on MNIST

To induce grokking on MNIST, we make two nonstandard decisions: (1) we reduce the size of the training set from 50k to 1k samples (by taking a random subset) and (2) we increase the scale of the weight initialization distribution (by multiplying the initial weights, sampled with Kaiming uniform initialization, by a constant > 1).

The choice of large initializations is justified by ([221], [286], [287]) which find large initializations overfit data easily but prone to poor generalization. Relevant to this, initialization scale is found to regulate "kernel" vs "rich" learning regimes in networks [288].

With these modifications to training set size and initialization scale, we train a depth-3 width-200 MLP with ReLU activations with the AdamW optimizer. We use MSE loss with one-hot targets, rather than cross-entropy. With this setup, we find that the network quickly fits the train set, and then much later in training validation accuracy improves, as shown in Figure 6.8a. This closely follows the stereotypical grokking learning, first observed in algorithmic datasets.

With this setup, we also compute a phase diagram over the model weight decay and the last layer learning rate. See Figure 6.8b. While in MLPs it is less clear what parts of the network to consider the "encoder" vs the "decoder", for our purposes here we consider the last layer to be the "decoder" and vary its learning rate relative to the rest of the network. The resulting phase diagram has some similarity to Figure 6.7. We observe a "confusion"phase in the bottom right (high learning rate and high weight decay), a "comprehension" phase bordering it, a "grokking" phase as one decreases weight decay and decoder learning rate, and a "memorization" phase at low weight decay and low learning rate. Instead of an accuracy threshold of 95%, we use a threshold of 60% here for validation accuracy for runs to count as comprehension or grokking. This phase diagram demonstrates that with sufficient regularization, we can again "de-grok" learning.

We also investigate the effect of training set size on time to generalization on MNIST. We find a result similar to what Power et al. [1] observed, namely that generalization time increases rapidly once one drops below a certain amount of training data. See Figure C.12.

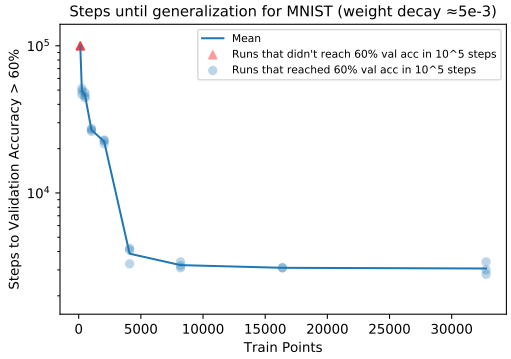


Figure C.12: Time to generalize as a function of training set size, on MNIST.

C.11 Lottery Ticket Hypothesis Connection

In Figure C.13, we show the projection of the learned embeddings after generalization to their first two principal components. Compared to the projection at initialization, structure clearly emerges in embedding space when the neural network is able to generalize ($> 99\%$ validation accuracy). What is intriguing is that the projection of the embeddings at initialization to the principal components of the embeddings at generalization seem to already contain much

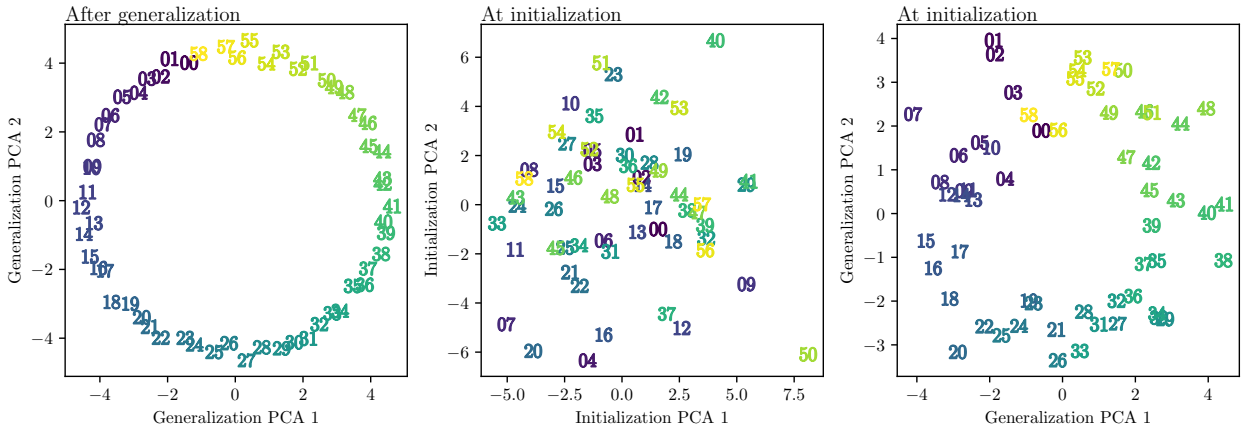


Figure C.13: **(Left)** Input embeddings after generalization projected on their first 2 principal components.**(Center)** Input embeddings at initialization projected on their first 2 principal components. **(Right)** Input embeddings at initialization projected on the first 2 principal components of the embeddings after generalization at the end of training (same PCA as the left figure).

of that structure. In this sense, the structured representation necessary for generalization already existed (partially) at initialization. The training procedure essentially prunes other unnecessary dimensions and forms the required parallelograms for generalization. This is a nonstandard interpretation of the lottery ticket hypothesis where the winning tickets are not weights or subnetworks but instead particular axes or linear combinations of the weights (the learned embeddings).

In Figure C.14, we show the original training curves (dashed lines). In solid lines, we recompute accuracy with models which use embeddings that are projected onto the n principal components of the embeddings at the end of training (and back). Clearly, the first few principal components contain enough information to reach 99% accuracy. The first few PCs explain the most variance by definition, however, we note that this is not necessarily the main reason for why they can generalize so well. In fact, embeddings reconstructed from the PCA at the end of training (solid lines) perform better than current highest variance axes (dotted line). This behavior is consistent across seeds.

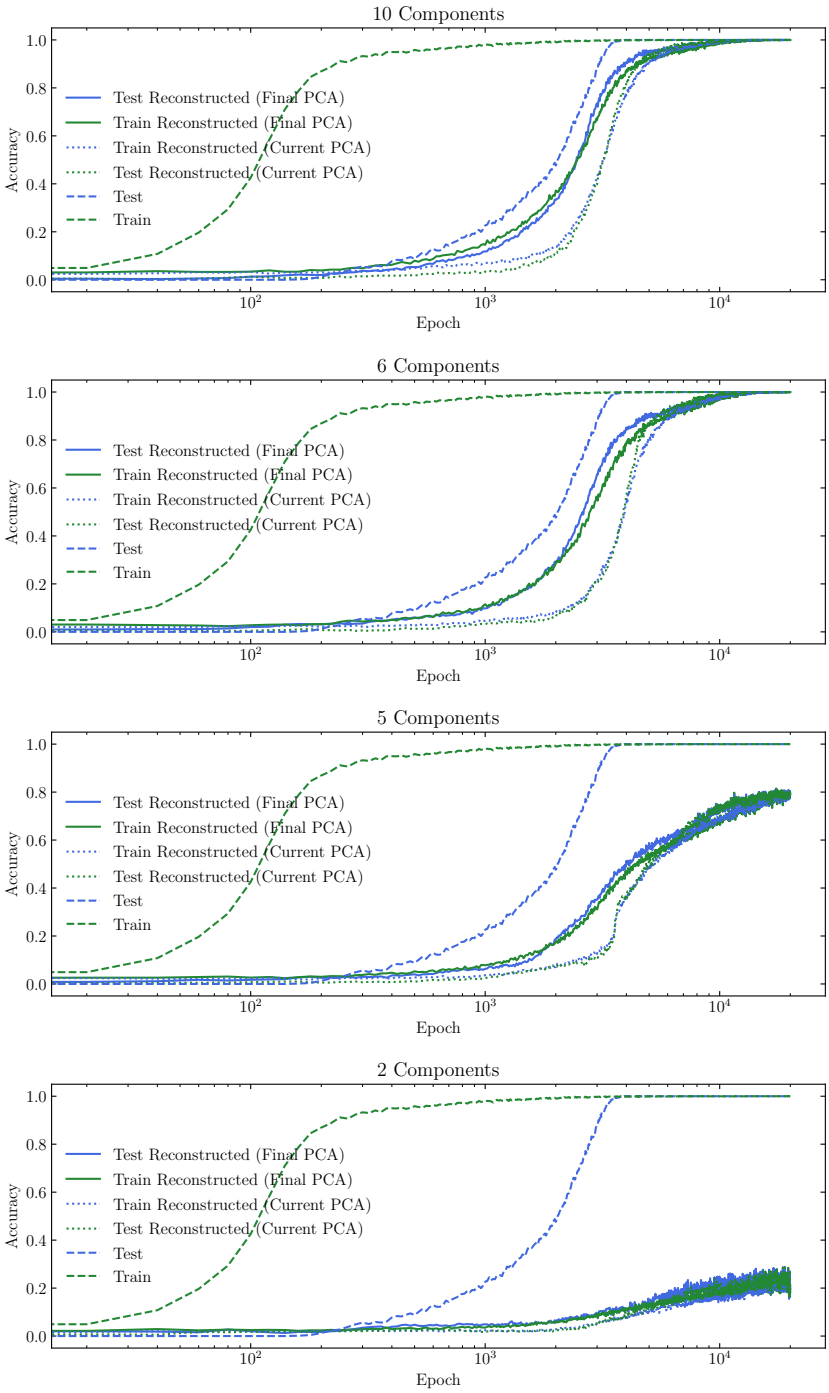


Figure C.14: Train and test accuracy computed while using actual embeddings (dashed line) and embeddings projected onto and reconstructed from their first n principal components (dotted lines) and, finally, using embeddings projected onto and reconstructed from the first n PCs of the embeddings at the end of training (solid lines).

C.12 Derivation of the effective loss

In this section, we will further motivate the use of our effective loss to study the dynamics of representation learning by deriving it from the gradient flow dynamics on the actual MSE loss in linear regression. The loss landscape of a neural network is in general nonlinear, but the linear case may shed some light on how the effective loss can be derived from actual loss. For a sample $\mathbf{r}$ (which is the sum of two embeddings $\mathbf{E}_i$ and $\mathbf{E}_j$), the prediction of the linear network is $D(\mathbf{r}) = \mathbf{A}\mathbf{r} + \mathbf{b}$. The loss function is ($\mathbf{y}$ is its corresponding label):

$$\ell = \underbrace{\frac{1}{2}|\mathbf{A}\mathbf{r} + \mathbf{b} - \mathbf{y}|^2}_{\ell_{\text{pred}}} + \underbrace{\frac{\gamma}{2}||\mathbf{A}||_F^2}_{\ell_{\text{reg}}}, \tag{C.26}$$

where the first and the second term are prediction error and regularization, respectively. Both the model $(\mathbf{A}, \mathbf{b})$ and the input $\mathbf{r}$ are updated via gradient flow, with learning rate η_A and η_x, respectively:

$$\frac{d\mathbf{A}}{dt} = -\eta_A \frac{\partial \ell}{\partial \mathbf{A}}, \frac{d\mathbf{b}}{dt} = -\eta_A \frac{\partial \ell}{\partial \mathbf{b}}, \frac{d\mathbf{r}}{dt} = -\eta_x \frac{\partial \ell}{\partial \mathbf{r}}. \tag{C.27}$$

Inserting ℓ into the above equations, we obtain the gradient flow:

$$\begin{aligned}
\frac{d\mathbf{A}}{dt} &= -\eta_A \frac{\partial \ell}{\partial \mathbf{A}} = -\eta_A[\mathbf{A}(\mathbf{r}\mathbf{r}^T + \gamma) + (\mathbf{b} - \mathbf{y})\mathbf{r}^T], \\
\frac{d\mathbf{b}}{dt} &= -\eta_A \frac{\partial \ell}{\partial \mathbf{b}} = -\eta_A(\mathbf{A}\mathbf{r} + \mathbf{b} - \mathbf{y}) \\
\frac{d\mathbf{r}}{dt} &= -\eta_x \frac{\partial \ell}{\partial \mathbf{r}} = -\eta_x \mathbf{A}^T(\mathbf{A}\mathbf{r} + \mathbf{b} - \mathbf{y}).
\end{aligned} \tag{C.28}$$

For the $d\mathbf{b}/dt$ equation, after ignoring the $\mathbf{A}\mathbf{r}$ term and set the initial condition $\mathbf{b}(0) = \mathbf{0}$, we obtain analytically $\mathbf{b}(t) = (1 - e^{-2\eta_A t})\mathbf{y}$. Inserting this into the first and third equations, we have

$$\begin{aligned}
\frac{d\mathbf{A}}{dt} &= -\eta_A[\mathbf{A}(\mathbf{r}\mathbf{r}^T + \gamma) - e^{-2\eta_A t}\mathbf{y}\mathbf{r}^T], \\
\frac{d\mathbf{r}}{dt} &= \underbrace{-\eta_x \mathbf{A}^T\mathbf{A}\mathbf{r}}_{\text{internal interaction}} + \underbrace{\eta_x e^{-2\eta_A t}\mathbf{A}^T\mathbf{y}}_{\text{external force}}.
\end{aligned} \tag{C.29}$$

For the second equation on the evolution of $d\mathbf{r}/dt$, we can artificially decompose the right hand side into two terms, based on whether they depend on the label $\mathbf{y}$. In this way, we call the first term "internal interaction" since it does not depend on $\mathbf{y}$, while the second term "external force". Note this distinction seems a bit artificial from a mathematical perspective, but it can be conceptually helpful from a physics perspective. We will show below the internal interaction term is important for representations to form. Because we are interested in how two samples interact, we now consider another sample at $\mathbf{r}'$, and the evolution becomes

$$\begin{aligned}
\frac{d\mathbf{A}}{dt} &= -\eta_A[\mathbf{A}(\mathbf{r}\mathbf{r}^T + \mathbf{r}'\mathbf{r}'^T + 2\gamma) - e^{-2\eta_A t}\mathbf{y}(\mathbf{r} + \mathbf{r}')^T], \\
\frac{d\mathbf{r}}{dt} &= -\eta_x \mathbf{A}^T\mathbf{A}\mathbf{r} + \eta_x e^{-2\eta_A t}\mathbf{A}^T\mathbf{y}, \\
\frac{d\mathbf{r}'}{dt} &= -\eta_x \mathbf{A}^T\mathbf{A}\mathbf{r}' + \eta_x e^{-2\eta_A t}\mathbf{A}^T\mathbf{y}.
\end{aligned} \tag{C.30}$$

Subtracting $d\mathbf{r}/dt$ by $d\mathbf{r}'/dt$ and setting $\mathbf{r}' = -\mathbf{r}$, the above equations further simply to

$$\begin{aligned} \frac{d\mathbf{A}}{dt} &= -2\eta_A \mathbf{A}(\mathbf{r}\mathbf{r}^T + \gamma), \\ \frac{d\mathbf{r}}{dt} &= -\eta_x \mathbf{A}^T \mathbf{A}\mathbf{r}. \end{aligned} \tag{C.31}$$

The second equation implies that the pair of samples interact via a quadratic potential $U(\mathbf{r}) = \frac{1}{2}\mathbf{r}^T \mathbf{A}^T \mathbf{A}\mathbf{r}$, leading to a linear attractive force $f(r) \propto r$. We now consider the adiabatic limit where $\eta_A \to 0$.

The adiabatic limit Using the standard initialization (e.g., Xavier initialization) of neural networks, we have $\mathbf{A}_0^T \mathbf{A}_0 \approx \mathbf{I}$. As a result, the quadratic potential becomes $U(\mathbf{r}) = \frac{1}{2}\mathbf{r}^T \mathbf{r}$, which is time-independent because $\eta_A \to 0$. We are now in the position to analyze the addition problem. For two samples $\mathbf{x}^{(1)} = \mathbf{E}_i + \mathbf{E}_j$ and $\mathbf{x}^{(2)} = \mathbf{E}_m + \mathbf{E}_n$ with the same label $(i + j = m + n)$, they contribute to an interaction term

$$U(i, j, m, n) = \frac{1}{2}|\mathbf{E}_i + \mathbf{E}_j - \mathbf{E}_m - \mathbf{E}_n|_2^2. \tag{C.32}$$

Averaging over all possible quadruples in the training dataset D, the total energy of the system is

$$\ell_0 = \sum_{(i,j,m,n) \in P_0(D)} \frac{1}{2}|\mathbf{E}_i + \mathbf{E}_j - \mathbf{E}_m - \mathbf{E}_n|_2^2 / |P_0(D)|, \tag{C.33}$$

where $P_0(D) = \{(i, j, m, n) | i + j = m + n, (i, j) \in D, (m, n) \in D\}$. To make it scale-invariant, we define the normalized Hamiltonian Eq. (C.33) as

$$\ell_{\text{eff}} = \frac{\ell_0}{Z_0}, \quad Z_0 = \sum_i |\mathbf{E}_i|_2^2 \tag{C.34}$$

which is the effective loss we used in Section 6.3.2.

Appendix D

Diffusion Models for Reasoning

D.1 Why does AR w/reverse sequences fail?

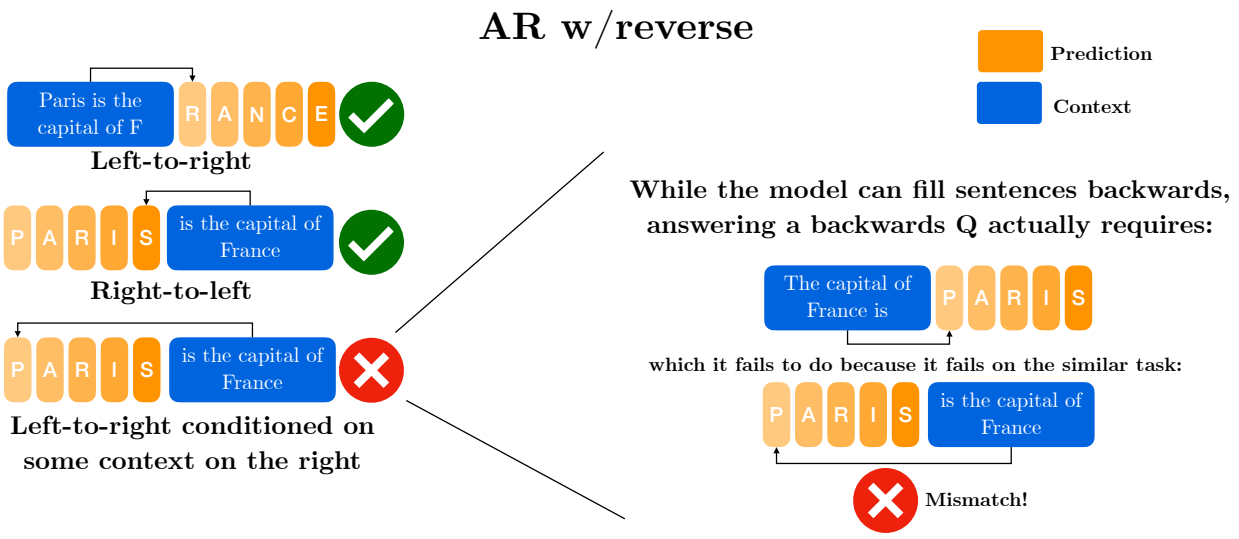


Figure D.1: AR w/reverse cannot predict (left-to-right) entities that appeared on the left during training as it only learned to complete them from right to left. The two sequences in the bottom right indicate that backward retrieval is roughly equivalent to refactorizing the conditionals such that the entity of interest is predicted last conditioned on everything else. This is only approximate because answering a backward QA might require adding new tokens like *"The answer to the question is ..."* but we make a weak assumption that such differences are generally irrelevant compared to the entities and relations of interest.

Table D.1: Summary of qualitative results, formatted as (forward)/(backward). Stargraph only has one direction.

Task	MLM	MLM-$\mathcal{U}$	AR	AR rev.	AR rev. ent.
Retrieval	✓/✓	✓/✓	✓/✗	✓/✓	✓/✓
Relationship	✓/✓	✓/✓	✓/✗	✓/✓	✓/✓
BioS	✗/✗	✓/✓	✓/✗	✓/✗	✓/✓[231]
Wiki	✗/✗	∼/∼	✓/✗	✓/✗	–
Stargraph	✓	✓	✗	✗	✓[248]

D.2 Permutation Language Modeling and Discrete State Diffusion

To illustrate the similarity between the diffusion loss and permutation language modeling, let's continue walking through our $D = 2$ example. Permutation modeling averages over factorizations $p(\boldsymbol{x}) = \frac{1}{2}p(x_2|x_1)p(x_1) + \frac{1}{2}p(x_1|x_2)p(x_2)$ and optimizes a lower bound on the likelihood

$$\log p(\boldsymbol{x}) \geq -\mathcal{L}_P = \frac{1}{2}(\log p(x_2|x_1) + \log p(x_1)) + \frac{1}{2}(\log p(x_1|x_2) + \log p(x_2)). \tag{D.1}$$

Finally, the diffusion model averages over masking rates $\frac{1}{2}(\log p(x_1)+\log p(x_2))+\frac{1}{2}(\log p(x_1|x_2)+\log p(x_2|x_1))$ and optimizes

$$\log p(\boldsymbol{x}) \geq -\mathcal{L}_{\text{MLM}\mathcal{U}} = \frac{1}{2}(\log p(x_1) + \log p(x_2)) + \frac{1}{2}(\log p(x_1|x_2) + \log p(x_2|x_1)). \tag{D.2}$$

This is the same as Equation (D.1). This implies that the permutation language modeling and the absorbing state diffusion objectives are in fact the same. Though practically speaking, they may have very different implications.

D.3 Summary of Tables

Table D.1 shows a qualitative comparison of the optimization objectives explored on the different datasets in this paper. We conclude that MLM with a fixed masking rate mitigates the reversal curse due to its bi-directionality, but lacks generative quality and thus generally fails when having to provide longer answers. Also unsurprisingly, the left to right AR objective works well in the forward retrieval direction but is unable to answer backwards questions and has a hard time reasoning multiple tokens ahead to solve a task like graph traversal without intermediate supervision. Reversing the tokens can aid backwards retrieval for single token lookups, but fail otherwise. Reversing entities intuitively should be able to solve every retrieval task, but finding the right token permutation is a difficult task by itself. MLM-$\mathcal{U}$ averages over all possible prediction tasks that exist for a sequence given a tokenization and prevails in most our experiments. MLM-$\mathcal{U}$ displays the highest backwards retrieval

capabilities in the most realistic Wikireversal benchmark, but the performance is not strong enough to qualitatively state success and we mark it with $\sim$ in Table D.1. We hypothesize that the reason is increased task complexity requiring larger models. Notably, in Table D.5 we show that MLM-$\mathcal{U}$ outperforms all other objectives when it has access to either the forward or backward type question. From there, it can generalize well to the other type.

D.4 Additional Tables

Table D.2: Retrieval Task forward and backward per token accuracy of different training paradigms.

	AR	AR w/reverse	MLM 15%	MLM 40%	MLM 85%	MLM-$\mathcal{U}$	PLM
Forward	**100**	**100**	21	17	27	**100**	**100**
Backward	0	0	22	16	28	**100**	**100**

Table D.3: BioS exact match accuracy for property retrieval in the backward direction (birth date to full name) and in the forward direction (full name to birthdate).

	AR	AR w/reverse	MLM 15%	MLM 40%	MLM 85%	PLM	MLM-$\mathcal{U}$
Forward	**1.00**	**1.00**	0.00	0.08	0.04	**1.00**	**1.00**
Backward	0.00	0.00	0.00	0.08	0.08	**0.72**	0.68

Table D.4: Exact match QA accuracies for relationship tasks. Forward and backward accuracies are calculated normally, but due to the non-reciprocal relationship, a model that swaps the subject and object will make errors (e.g., inferring B is A's child from A being B's child). Entity reversal without a delimiter is marked with a*.

	AR w/reverse (entity)	AR w/reverse (entity)*	MLM 15%	MLM 40%	MLM 85%	MLM-$\mathcal{U}$	PLM
Forward	**100**	54	24	77	2	**100**	**100**
Backward	**100**	53	19	35	1	**100**	**100**
Incorrect Inference	0	**44**	0	1	0	0	0

Table D.5: Wikireversal task exact match QA accuracies. MLM-$\mathcal{U}$, MLM and AR are all 100M parameter models trained from scratch. (Right) uses different seeds for train test splits in forward and backward questions while (Left) uses the same seed. For MLM, we tried 15%, 40% and 85% masking rates and we present only the best models (15%). Details on hyperparameter selection can be found in Appendix D.5.3

	Mistral 7B	MLM	MLM-$\mathcal{U}$	AR	Mistral 7B	MLM	MLM-$\mathcal{U}$	AR
Forward	**21**	3.4	11	14	20	29	**66**	28
Backward	5.2	2.7	**7.9**	4.3	9.0	10	**46**	6.2

Algorithm 2 Dataset Creation

Input: GenWiki Corpus $\mathcal{G} = \{(P, E, T)\}$
Output: QA Dataset $\mathcal{D} = \{(q, a, P)\}$
$\mathcal{D} \leftarrow \emptyset$
for $(P, E, T) \in \mathcal{G}$ **do** ▷ Each GenWiki sample
  for $(e_i, r, e_j) \in T$ **do** ▷ Each relation triple
    if e_i appears before e_j in P **then** ▷ Forward relation
      $q_f \leftarrow F_r(e_i)$ ▷ Forward question
      $q_b \leftarrow B_r(e_j)$ ▷ Backward question
      $\mathcal{D} \leftarrow \mathcal{D} \cup \{(q_f, e_j, P), (q_b, e_i, P)\}$
    end if
  end for
end for
Filter $\mathcal{D}$ to keep unambiguous QA pairs ▷ See filtering in Appendix D.5.1
Filter $\mathcal{D}$ to remove rare QA pairs where relation r appears < 500 times

D.5 WikiReversal

D.5.1 Filtering Ambiguous Samples

To mitigate ambiguity in the generated QA pairs, we filter the dataset to retain only (e_i, r, e_j) triples where the (e_i, r) and (r, e_j) pairs are unique across the entire dataset. This ensures that each question has a single unambiguous answer. Algorithm 2 summarizes the dataset creation process.

D.5.2 Examples from the Wikireversal dataset

Table D.7 shows the relations present in the Wikireversal dataset. Table D.6 shows examples of passages and corresponding forward and backward questions that are trained on. WikiReversal is filtered from GenWiki [243], a dataset based on Wikipedia released under a Creative Commons Attribution 4.0 International License.

D.5.3 Details on Wikireversal training

To measure performance on the Wikireversal dataset, we split the available data into training and validation, where we include all passages and 80% of both forward and backward questions in the training set and 20% of questions in the validation set. We run a hyperparameter grid search over every objective. We sweep over feasible learning rates for all models and weight decay for all except for MLM-$\mathcal{U}$, where we haven't found weight decay to be effective so it is set to 0. We run sweeps for both different (Table D.5 right) and same (Table D.5 left) train test splits in forward and backward questions. GPT and BERT models have 12 layers with 12 heads and 768 embedding dimension for a total of 108 M parameters. The encoder-decoder model has 12 layers with 9 heads and 576 embedding dimension for a total of 109 M parameters. All models except Mistral were trained for 1500 epochs, and Mistral was trained for 200 to alleviate overfitting. The learning rates were warmed up for 1% of the training time in all cases. For Mistral, the LoRA α and r parameters are set to 256 to sum to about 109 M trainable parameters. Learning rates are 5e-5 for MLM-$\mathcal{U}$ in both train split modes, 3e-4 for both BERT (MLM-15%) and GPT (AR) for both modes and 1e-4 for Mistral (AR). The best weight decays are 1e-2 for both BERT and GPT, and no weight decay for LoRA on Mistral. No dropout was used. All models were trained with AdamW with default β parameters.

Table D.5 shows that MLM-$\mathcal{U}$ outperforms other objectives, achieving 66% and 46% accuracy on forward and backward questions, respectively. Using different seeds for train/test splits in forward and backward directions (right section) allows bidirectional models to learn to answer questions from both the passage and the question itself, explaining MLM-$\mathcal{U}$'s significant improvement. AR performs poorly on backward questions due to the "reversal curse". MLM and Mistral 7B show intermediate performance. Although Mistral 7B uses $\sim$ 100M LoRA parameters, fewer than the other models, this setup mimics common fine-tuning recipes. Naturally, models trained from scratch do not learn general language modeling capabilities.

Table D.6: Examples from Wikireversal

Passage	**Forward Q**	**Backward Q**
Agostino Magliani (23 July 1824 – 20 February 1891), Italian financier, was a native of Laurino, near Salerno.	Where was Agostino Magliani born?	Who was born in Laurino?
Zhou Yongkang has two sons, Zhou Bin and Zhou Han, with his first wife, Wang Shuhua, whom he met while working in the oilfields of Liaoning province.	Who is Zhou Yongkang's spouse?	Who is married to Wang Shuhua?
The total area of Mitan-myeon is 109.74 square kilometers, and, as of 2008, the population was 1,881 people.	What is the total area of Mitan-myeon?	Which populated place has a total area of 109.74?
Mohammad Ali Araki was born on 1894 in Arak, Iran. He started his education from Arak Hawza. Grand Ayatollah Haeri allowed him to wear the turban and robe because qualified individuals were limited. Also, Araki studied many years in Yazd Hawza.	What title does Mohammad Ali Araki hold?	Who holds the title of Grand Ayatollah?
Tibor Navracsics (born Veszprém, Hungary, 13 June 1966) is a Hungarian lawyer and politician, who served as Minister of Foreign Affairs and Trade from June to September 2014.	What region is Tibor Navracsics located in?	What or who is located in the Veszprém region?
WWWX (96.9 FM, "96.9 The Fox") is an Alternative rock formatted radio station licensed to Oshkosh, Wisconsin, that serves the Appleton-Oshkosh area.	What is WWWX's alias?	Whose alias is 96.9 The Fox?

Table D.7: Relations in Wikireversal

Attribute	**Count**
birthPlace	6100
birthName	5018
alias	3745
location	2532
deathPlace	2064
title	1923
city	1871
populationTotal	1651
owner	1328
name	1274
spouse	1163
isPartOf	1000
type	920
office	893
associatedBand	762
associatedMusicalArtist	756
synonym	743
knownFor	729
artist	724
PopulatedPlace/areaTotal	719
birthDate	672
ground	670
occupation	665
place	631
address	631
family	589
hometown	559
region	551
developer	541
label	538
writer	517
total count	42479

D.6 Delayed Generalization in Language Modeling

We include accuracy curves for training with MLM-$\mathcal{U}$ for both Bios and WikiReversal in Figure D.2. We see the model is able to gradually learn both the forward and backward questions throughout training. For Bios, unlike the forward questions which saturate much more quickly, the backward accuracy still shows an upward trend after training for 20k optimization steps. We observe a similar trend in the delayed generalization in WikiReversal for both forward and backwards questions even after training for 300k optimization steps. These results empirically demonstrate that the MLM-$\mathcal{U}$ objective, which requires modelling all possible factorizations of an input into context and predictions, is a more challenging task that exhibit delayed generalization relative to standard next-to-prediction training.

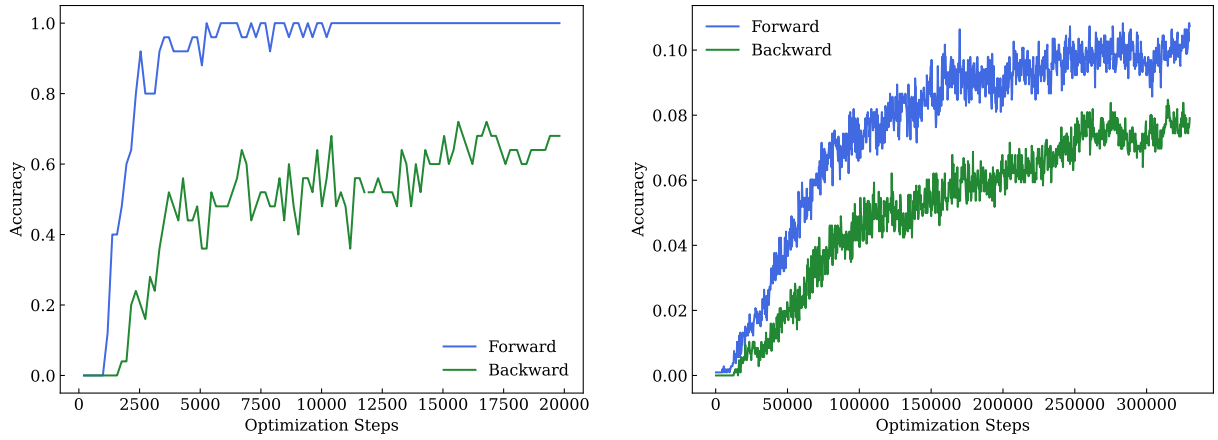


Figure D.2: Accuracy in Forward/Backward Questions on the Bios dataset (left) and the Wikireversal dataset (right)

D.7 Architecture Details

The Encoder-Decoder architecture used to train the MLM-$\mathcal{U}$ objective is modeled with ideas from XLNet [238] in mind in order to support different attention/masking strategies including permutation language modeling. The encoder has GPT-like blocks and works with RoPE as positional bias. The decoder also has GPT-like blocks, but it cross-attends over keys and values from the corresponding encoder layer, also via a RoPE bias. The decoder input contains the same learnable embedding for all tokens, such that only the positional bias defines the initial attention pattern. This idea comes from XLNet's positional attention stream. In left to right AR training mode, both encoder and decoder use a causal attention mask. In MLM-X modes, a fraction of inputs are masked before given to the model and neither decoder nor encoder attend over the masked tokens. All inference is performed in left-to-right AR fashion.

D.8 Compute Requirements

Models were trained on 64 NVidia V100 and A100 GPUs with supporting Intel(R) Xeon(R) Gold 6230 CPUs. From conception to finalization of this paper we trained about 2000 models. The computationally most expensive runs were on the BioS and the Wikireversal dataset. Those comprised about 300 runs with on 8 GPUs for around a day per model. About 30 Mistral models were trained on 32 GPUs for about a day per model.

References

[1] A. Power, Y. Burda, H. Edwards, I. Babuschkin, and V. Misra, "Grokking: Generalization beyond overfitting on small algorithmic datasets," *arXiv preprint arXiv:2201.02177*, 2022.

[2] S. G. BRUSH, "History of the lenz-ising model," *Rev. Mod. Phys.*, vol. 39, pp. 883–893, 4 Oct. 1967. DOI: 10.1103/RevModPhys.39.883. URL: https://link.aps.org/doi/10.1103/RevModPhys.39.883.

[3] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65 6, pp. 386–408, 1958. URL: https://api.semanticscholar.org/CorpusID:12781225.

[4] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. United States of America: Pearson Education, 2010, pp. 16–28, PDF, ISBN: 978-0-13-604259-4.

[5] M. Minsky and S. Papert, *Perceptrons; an Introduction to Computational Geometry*. MIT Press, 1969, ISBN: 9780262630221. URL: https://books.google.com/books?id=Ow1OAQAAIAAJ.

[6] A. G. Ivakhnenko, *Cybernetic Predicting Devices*. CCM Information Corporation, 1973.

[7] A. G. Ivakhnenko and V. G. Lapa, *Cybernetics and forecasting techniques*. American Elsevier Pub. Co., 1967.

[8] S. Amari, "A theory of adaptive pattern classifier," *IEEE Transactions on Electronic Computers*, vol. EC-16, no. 3, pp. 279–307, 1967. DOI: 10.1109/PGEC.1967.264678.

[9] P. Werbos, "Applications of advances in nonlinear sensitivity analysis," in *System modeling and optimization*, PDF available. Archived from the original on 14 April 2016. Retrieved 2 July 2017., Springer, 1982, pp. 762–770. URL: https://web.archive.org/web/20160414080159/http://www.werbos.com/Neural/sensitivity.pdf.

[10] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989. DOI: 10.1162/neco.1989.1.4.541.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[12] Kurzweil, *Interview with juergen schmidhuber on the eight competitions won by his deep learning team 2009–2012*, Kurzweil AI, Archived at https://web.archive.org/web/20180831000000/http://www.kurzweilai.net/how-bio-inspired-deep-learning-keeps-winning-competitions on 31 August 2018, 2012.

[13] Kurzweil, *How bio-inspired deep learning keeps winning competitions*, KurzweilAI, Archived from the original on 31 August 2018. Retrieved 16 June 2017, 2017. URL: https://www.kurzweilai.net/how-bio-inspired-deep-learning-keeps-winning-competitions.

[14] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855–868, May 2009, Archived from the original on 2 January 2014. Retrieved 30 July 2014, ISSN: 0162-8828. DOI: 10.1109/tpami.2008.137.

[15] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Advances in Neural Information Processing Systems 21 (NIPS 2008)*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., Archived from the original on 19 May 2024. Retrieved 3 June 2022, Neural Information Processing Systems (NIPS) Foundation, 2009, pp. 545–552, ISBN: 978-1-60560-949-2. URL: https://papers.nips.cc/paper/2008/file/66368270ffd51418ec58bd793f2d9b1b-Paper.pdf.

[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[17] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, 2022. arXiv: 1312.6114 [stat.ML]. URL: https://arxiv.org/abs/1312.6114.

[18] D. J. Rezende and S. Mohamed, *Variational inference with normalizing flows*, 2016. arXiv: 1505.05770 [stat.ML]. URL: https://arxiv.org/abs/1505.05770.

[19] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[21] A. Radford, J. W. Kim, C. Hallacy, *et al.*, *Learning transferable visual models from natural language supervision*, 2021. arXiv: 2103.00020 [cs.CV]. URL: https://arxiv.org/abs/2103.00020.

[22] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[23] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

[24] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, 2015. arXiv: 1502.03167 [cs.LG]. URL: https://arxiv.org/abs/1502.03167.

[25] J. L. Ba, J. R. Kiros, and G. E. Hinton, *Layer normalization*, 2016. arXiv: 1607.06450 [stat.ML]. URL: https://arxiv.org/abs/1607.06450.

[26] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.

[27] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, *et al.*, "Training compute-optimal large language models," *arXiv preprint arXiv:2203.15556*, 2022.

[28] G. Yang, E. J. Hu, I. Babuschkin, S. Sidor, X. Liu, D. Farhi, N. Ryder, J. Pachocki, W. Chen, and J. Gao, *Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer*, 2022. arXiv: 2203.03466 [cs.LG]. URL: https://arxiv.org/abs/2203.03466.

[29] O. Kitouni, N. Nolte, and M. Williams, "Expressive Monotonic Neural Networks," in *International Conference on Learning Representations (ICLR 2023)*, 2023.

[30] O. Kitouni, N. Nolte, and M. Williams, "Robust and provably monotonic networks," *Machine Learning: Science and Technology*, vol. 4, no. 3, p. 035 020, Aug. 2023, ISSN: 2632-2153. DOI: 10.1088/2632-2153/aced80. URL: http://dx.doi.org/10.1088/2632-2153/aced80.

[31] R. Aaij *et al.*, "Allen: A high level trigger on GPUs for LHCb," *Comput. Softw. Big Sci.*, vol. 4, no. 1, p. 7, 2020. DOI: 10.1007/s41781-020-00039-7. arXiv: 1912.09161 [physics.ins-det].

[32] R. Aaij *et al.*, "The LHCb trigger and its performance in 2011," *JINST*, vol. 8, P04022, 2013. DOI: 10.1088/1748-0221/8/04/P04022. arXiv: 1211.3055 [hep-ex].

[33] R. Aaij *et al.*, "Performance of the LHCb trigger and full real-time reconstruction in Run 2 of the LHC," *JINST*, vol. 14, no. LHCb-DP-2019-001, P04013, 2019. DOI: 10.1088/1748-0221/14/04/P04013. arXiv: 1812.10790 [hep-ex].

[34] X. Liu, X. Han, N. Zhang, and Q. Liu, *Certified monotonic neural networks*, 2020. arXiv: 2011.10219 [cs.LG].

[35] S. You, D. Ding, K. Canini, J. Pfeifer, and M. Gupta, *Deep lattice networks and partial monotonic functions*, 2017. arXiv: 1709.06680 [stat.ML].

[36] J. Sill, "Monotonic networks," in *Advances in Neural Information Processing Systems*, M. Jordan, M. Kearns, and S. Solla, Eds., vol. 10, MIT Press, 1998. URL: https://proceedings.neurips.cc/paper/1997/file/83adc9225e4deb67d7ce42d58fe5157c-Paper.pdf.

[37] J. Behrmann, W. Grathwohl, R. T. Q. Chen, D. Duvenaud, and J.-H. Jacobsen, *Invertible residual networks*, 2019. arXiv: 1811.00995 [cs.LG].

[38] H. Gouk, E. Frank, B. Pfahringer, and M. J. Cree, *Regularisation of neural networks by enforcing lipschitz continuity*, 2020. arXiv: 1804.04368 [stat.ML].

[39] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral Normalization for Generative Adversarial Networks," *arXiv e-prints*, arXiv:1802.05957, arXiv:1802.05957, Feb. 2018. arXiv: 1802.05957 [cs.LG].

[40] T. Huster, C.-Y. J. Chiang, and R. Chadha, *Limitations of the lipschitz constant as a defense against adversarial examples*, 2018. arXiv: 1807.09705 [cs.LG].

[41] C. Anil, J. Lucas, and R. Grosse, "Sorting out Lipschitz function approximation," in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, Jun. 2019, pp. 291–301. URL: http://proceedings.mlr.press/v97/anil19a.html.

[42] L. Béthune, T. Boissin, M. Serrurier, F. Mamalet, C. Friedrich, and A. González-Sanz, *Pay attention to your loss: Understanding misconceptions about 1-lipschitz neural networks*, 2021. DOI: 10.48550/ARXIV.2104.05097. URL: https://arxiv.org/abs/2104.05097.

[43] Y. Amhis *et al.*, "Averages of b-hadron, c-hadron, and τ-lepton properties as of 2021," Jun. 2022. arXiv: 2206.07501 [hep-ex].

[44] S. Gori, M. Williams, *et al.*, "Dark Matter Production at Intensity-Frontier Experiments," in *2021 Snowmass Summer Study*. arXiv: 2209.04671 [hep-ph].

[45] M. Graham, C. Hearty, and M. Williams, "Searches for Dark Photons at Accelerators," *Ann. Rev. Nucl. Part. Sci.*, vol. 71, pp. 37–58, 2021. DOI: 10.1146/annurev-nucl-110320-051823. arXiv: 2104.10280 [hep-ph].

[46] C. Auguste, S. Malory, and I. Smirnov, *A better method to enforce monotonic constraints in regression and classification trees*, 2020. arXiv: 2011.00986 [stat.ML].

[47] V. V. Gligorov and M. Williams, "Efficient, reliable and fast high-level triggering using a bonsai boosted decision tree," *JINST*, vol. 8, P02013, 2013. DOI: 10.1088/1748-0221/8/02/P02013. arXiv: 1210.6861 [physics.ins-det].

[48] T. Likhomanenko *et al.*, "LHCb topological trigger reoptimization," *J. Phys. Conf. Ser.*, vol. 664, p. 082 025, Oct. 2015. DOI: 10.1088/1742-6596/664/8/082025.

[49] G. Ke, Q. Meng, T. Finely, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems 30 (NIP 2017)*, Dec. 2017. URL: https://www.microsoft.com/en-us/research/publication/lightgbm-a-highly-efficient-gradient-boosting-decision-tree/.

[50] O. Kitouni, N. Nolte, and M. Williams, "Finding NEEMo: Geometric Fitting using Neural Estimation of the Energy Mover's Distance," in *Advances in Neural Information Processing Systems (NeurIPS 2022), Machine Learning and the Physical Sciences*, 2022. arXiv: 2209.15624 [stat.ML].

[51] O. Kitouni, B. Nachman, C. Weisser, and M. Williams, "Enhancing searches for resonances with machine learning and moment decomposition," *Journal of High Energy Physics*, vol. 2021, no. 4, Apr. 2021, ISSN: 1029-8479. DOI: 10.1007/jhep04(2021)070. URL: http://dx.doi.org/10.1007/JHEP04(2021)070.

[52] J. Button, G. R. Kalbfleisch, G. R. Lynch, B. C. Maglić, A. H. Rosenfeld, and M. L. Stevenson, "Pion-Pion Interaction in the Reaction $\bar{p} + p \to 2\pi^+ + 2\pi^- + n\pi^0$," *Phys. Rev.*, vol. 126, no. 5, pp. 1858–1863, 1962. DOI: 10.1103/PhysRev.126.1858.

[53] G. Aad *et al.*, "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC," *Phys. Lett. B*, vol. 716, pp. 1–29, 2012. DOI: 10.1016/j.physletb.2012.08.020. arXiv: 1207.7214 [hep-ex].

[54] S. Chatrchyan *et al.*, "Observation of a New Boson at a Mass of 125 GeV with the CMS Experiment at the LHC," *Phys. Lett. B*, vol. 716, pp. 30–61, 2012. DOI: 10.1016/j.physletb.2012.08.021. arXiv: 1207.7235 [hep-ex].

[55] A. M. Sirunyan *et al.*, "Search for high mass dijet resonances with a new background prediction method in proton-proton collisions at $\sqrt{s} = 13$ TeV," *JHEP*, vol. 05, p. 033, 2020. DOI: 10.1007/JHEP05(2020)033. arXiv: 1911.03947 [hep-ex].

[56] G. Aad *et al.*, "Search for new resonances in mass distributions of jet pairs using 139 fb^{-1} of pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector," *JHEP*, vol. 03, p. 145, 2020. DOI: 10.1007/JHEP03(2020)145. arXiv: 1910.08447 [hep-ex].

[57] R. Aaij *et al.*, "Searches for low-mass dimuon resonances," Jul. 2020. arXiv: 2007.03923 [hep-ex].

[58] J. Adam *et al.*, "Pair invariant mass to isolate background in the search for the chiral magnetic effect in Au+Au collisions at $\sqrt{s_{\mathrm{NN}}} = 200$ GeV," Jun. 2020. arXiv: 2006.05035 [nucl-ex].

[59] S. Acharya *et al.*, "J/ψ elliptic and triangular flow in Pb-Pb collisions at $\sqrt{s_{\mathrm{NN}}} = 5.02$ TeV," May 2020. arXiv: 2005.14518 [nucl-ex].

[60] P. Adrian *et al.*, "Search for a dark photon in electroproduced e^+e^- pairs with the Heavy Photon Search experiment at JLab," *Phys. Rev. D*, vol. 98, no. 9, p. 091 101, 2018. DOI: 10.1103/PhysRevD.98.091101. arXiv: 1807.11530 [hep-ex].

[61] M. McCracken *et al.*, "Search for baryon-number and lepton-number violating decays of Λ hyperons using the CLAS detector at Jefferson Laboratory," *Phys. Rev. D*, vol. 92, no. 7, p. 072 002, 2015. DOI: 10.1103/PhysRevD.92.072002. arXiv: 1507.03859 [hep-ex].

[62] M. Ablikim *et al.*, "Observation of the leptonic decay $D^+ \to \tau^+\nu_\tau$," *Phys. Rev. Lett.*, vol. 123, no. 21, p. 211 802, 2019. DOI: 10.1103/PhysRevLett.123.211802. arXiv: 1908.08877 [hep-ex].

[63] "Search for Axion-Like Particles produced in e^+e^- collisions at Belle II," Jul. 2020. arXiv: 2007.13071 [hep-ex].

[64] M. Frate, K. Cranmer, S. Kalia, A. Vandenberg-Rodes, and D. Whiteson, "Modeling Smooth Backgrounds and Generic Localized Signals with Gaussian Processes," Sep. 2017. arXiv: 1709.05681 [physics.data-an].

[65] A. J. Larkoski, I. Moult, and B. Nachman, "Jet Substructure at the Large Hadron Collider: A Review of Recent Advances in Theory and Machine Learning," *Phys. Rept.*, vol. 841, pp. 1–63, 2020. DOI: 10.1016/j.physrep.2019.11.001. arXiv: 1709.04464 [hep-ph].

[66] D. Guest, K. Cranmer, and D. Whiteson, "Deep Learning and its Application to LHC Physics," 2018. arXiv: 1806.11484 [hep-ex].

[67] K. Albertsson *et al.*, "Machine Learning in High Energy Physics Community White Paper," 2018. arXiv: 1807.02876 [physics.comp-ph].

[68] A. Radovic, M. Williams, D. Rousseau, M. Kagan, D. Bonacorsi, A. Himmel, A. Aurisano, K. Terao, and T. Wongjirad, "Machine learning at the energy and intensity frontiers of particle physics," *Nature*, vol. 560, no. 7716, pp. 41–48, 2018. DOI: 10.1038/s41586-018-0361-2.

[69] D. Bourilkov, "Machine and Deep Learning Applications in Particle Physics," *Int. J. Mod. Phys. A*, vol. 34, no. 35, p. 1 930 019, 2020. DOI: 10.1142/S0217751X19300199. arXiv: 1912.08245 [physics.data-an].

[70] M. Aaboud *et al.*, "Performance of top-quark and W-boson tagging with ATLAS in Run 2 of the LHC," *Eur. Phys. J. C*, vol. 79, no. 5, p. 375, 2019. DOI: 10.1140/epjc/s10052-019-6847-8. arXiv: 1808.07858 [hep-ex].

[71] A. M. Sirunyan *et al.*, "Identification of heavy, energetic, hadronically decaying particles using machine-learning techniques," *JINST*, vol. 15, no. 06, P06005, 2020. DOI: 10.1088/1748-0221/15/06/P06005. arXiv: 2004.08262 [hep-ex].

[72] ATLAS Collaboration, "Search for diboson resonances in hadronic final states in 139 fb^{-1} of pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector," *JHEP*, vol. 09, p. 091, 2019, [Erratum: JHEP 06, 042 (2020)]. DOI: 10.1007/JHEP09(2019)091. arXiv: 1906.08589 [hep-ex].

[73] ATLAS Collaboration, "Search for heavy diboson resonances in semileptonic final states in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector," Apr. 2020. arXiv: 2004.14636 [hep-ex].

[74] CMS Collaboration, "A multi-dimensional search for new heavy resonances decaying to boosted WW, WZ, or ZZ boson pairs in the dijet final state at 13 TeV," *Eur. Phys. J. C*, vol. 80, no. 3, p. 237, 2020. DOI: 10.1140/epjc/s10052-020-7773-5. arXiv: 1906.05977 [hep-ex].

[75] CMS Collaboration, "Combination of CMS searches for heavy resonances decaying to pairs of bosons or leptons," *Phys. Lett. B*, vol. 798, p. 134 952, 2019. DOI: 10.1016/j.physletb.2019.134952. arXiv: 1906.00057 [hep-ex].

[76] ATLAS Collaboration, "Search for resonances decaying into a weak vector boson and a Higgs boson in the fully hadronic final state produced in proton−proton collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector," 2020. arXiv: 2007.05293 [hep-ex].

[77] CMS Collaboration, "Search for heavy resonances decaying into two Higgs bosons or into a Higgs boson and a W or Z boson in proton-proton collisions at 13 TeV," *JHEP*, vol. 01, p. 051, 2019. DOI: 10.1007/JHEP01(2019)051. arXiv: 1808.01365 [hep-ex].

[78] ATLAS Collaboration, "Reconstruction and identification of boosted di-τ systems in a search for Higgs boson pairs using 13 TeV proton−proton collision data in ATLAS," 2020. arXiv: 2007.14811 [hep-ex].

[79] CMS Collaboration, "Search for resonances decaying to a pair of Higgs bosons in the $\mathrm{b\bar{b}q\bar{q}'}\ell\nu$ final state in proton-proton collisions at $\sqrt{s} = 13$ TeV," *JHEP*, vol. 10, p. 125, 2019. DOI: 10.1007/JHEP10(2019)125. arXiv: 1904.04193 [hep-ex].

[80] CMS Collaboration, "Search for a massive resonance decaying to a pair of Higgs bosons in the four b quark final state in proton-proton collisions at $\sqrt{s} = 13$ TeV," *Phys. Lett. B*, vol. 781, pp. 244–269, 2018. DOI: 10.1016/j.physletb.2018.03.084. arXiv: 1710.04960 [hep-ex].

[81] G. Aad *et al.*, "Search for Higgs boson decays into a Z boson and a light hadronically decaying resonance using 13 TeV pp collision data from the ATLAS detector," Apr. 2020. arXiv: 2004.01678 [hep-ex].

[82] ATLAS Collaboration, "A search for resonances decaying into a Higgs boson and a new particle X in the $XH \to qqbb$ final state with the ATLAS detector," *Phys. Lett. B*, vol. 779, pp. 24–45, 2018. DOI: 10.1016/j.physletb.2018.01.042. arXiv: 1709.06783 [hep-ex].

[83] G. Aad *et al.*, "Dijet resonance search with weak supervision using $\sqrt{s} = 13$ TeV pp collisions in the ATLAS detector," *Phys. Rev. Lett.*, vol. 125, no. 13, p. 131 801, 2020. DOI: 10.1103/PhysRevLett.125.131801. arXiv: 2005.02983 [hep-ex].

[84] ATLAS Collaboration, "Search for light resonances decaying to boosted quark pairs and produced in association with a photon or a jet in proton-proton collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector," *Phys. Lett. B*, vol. 788, p. 316, 2019. DOI: 10.1016/j.physletb.2018.09.062. arXiv: 1801.08769 [hep-ex].

[85] CMS Collaboration, "Search for Low Mass Vector Resonances Decaying to Quark-Antiquark Pairs in Proton-Proton Collisions at $\sqrt{s} = 13$ TeV," *Phys. Rev. Lett.*, vol. 119, p. 111 802, 2017. DOI: 10.1103/PhysRevLett.119.111802. arXiv: 1705.10532 [hep-ex].

[86] CMS Collaboration, "Search for low-mass resonances decaying into bottom quark-antiquark pairs in proton-proton collisions at $\sqrt{s} = 13$ TeV," *Phys. Rev. D*, vol. 99, p. 012 005, 2019. DOI: 10.1103/PhysRevD.99.012005. arXiv: 1810.11822 [hep-ex].

[87] CMS Collaboration, "Search for Low-Mass Quark-Antiquark Resonances Produced in Association with a Photon at $\sqrt{s}$ =13 TeV," *Phys. Rev. Lett.*, vol. 123, p. 231 803, 2019. DOI: 10.1103/PhysRevLett.123.231803. arXiv: 1905.10331 [hep-ex].

[88] CMS Collaboration, "Search for low mass vector resonances decaying into quark-antiquark pairs in proton-proton collisions at $\sqrt{s} = 13$ TeV," *Phys. Rev. D*, vol. 100, no. 11, p. 112 007, 2019. DOI: 10.1103/PhysRevD.100.112007. arXiv: 1909.04114 [hep-ex].

[89] ATLAS Collaboration, "Search for boosted resonances decaying to two b-quarks and produced in association with a jet at $\sqrt{s} = 13$ TeV with the ATLAS detector," *ATLAS-CONF-2018-052*, 2018. URL: http://cds.cern.ch/record/2649081.

[90] CMS Collaboration, "Inclusive search for a highly boosted Higgs boson decaying to a bottom quark-antiquark pair," *Phys. Rev. Lett.*, vol. 120, p. 071 802, 2018. DOI: 10.1103/PhysRevLett.120.071802. arXiv: 1709.05543 [hep-ex].

[91] G. Louppe, M. Kagan, and K. Cranmer, "Learning to pivot with adversarial networks," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc., 2017, pp. 981–990. eprint: 1611.01046. URL: http://papers.nips.cc/paper/6699-learning-to-pivot-with-adversarial-networks.pdf.

[92] J. Dolen, P. Harris, S. Marzani, S. Rappoccio, and N. Tran, "Thinking outside the ROCs: Designing Decorrelated Taggers (DDT) for jet substructure," *JHEP*, vol. 05, p. 156, 2016. DOI: 10.1007/JHEP05(2016)156. arXiv: 1603.00027 [hep-ph].

[93] I. Moult, B. Nachman, and D. Neill, "Convolved Substructure: Analytically Decorrelating Jet Substructure Observables," *JHEP*, vol. 05, p. 002, 2018. DOI: 10.1007/JHEP05(2018)002. arXiv: 1710.06859 [hep-ph].

[94] J. Stevens and M. Williams, "uBoost: A boosting method for producing uniform selection efficiencies from multivariate classifiers," *JINST*, vol. 8, P12013, 2013. DOI: 10.1088/1748-0221/8/12/P12013. arXiv: 1305.7248 [nucl-ex].

[95] C. Shimmin, P. Sadowski, P. Baldi, E. Weik, D. Whiteson, E. Goul, and A. Søgaard, "Decorrelated Jet Substructure Tagging using Adversarial Neural Networks," 2017. arXiv: 1703.03507 [hep-ex].

[96] L. Bradshaw, R. K. Mishra, A. Mitridate, and B. Ostdiek, "Mass Agnostic Jet Taggers," 2019. arXiv: 1908.08959 [hep-ph].

[97] "Performance of mass-decorrelated jet substructure observables for hadronic two-body decay tagging in ATLAS," CERN, Geneva, Tech. Rep. ATL-PHYS-PUB-2018-014, Jul. 2018. URL: https://cds.cern.ch/record/2630973.

[98] G. Kasieczka and D. Shih, "DisCo Fever: Robust Networks Through Distance Correlation," 2020. arXiv: 2001.05310 [hep-ph].

[99] L.-G. Xia, "QBDT, a new boosting decision tree method with systematical uncertainties into training for High Energy Physics," *Nucl. Instrum. Meth.*, vol. A930, pp. 15–26, 2019. DOI: 10.1016/j.nima.2019.03.088. arXiv: 1810.08387 [physics.data-an].

[100] C. Englert, P. Galler, P. Harris, and M. Spannowsky, "Machine Learning Uncertainties with Adversarial Neural Networks," *Eur. Phys. J.*, vol. C79, no. 1, p. 4, 2019. DOI: 10.1140/epjc/s10052-018-6511-8. arXiv: 1807.08763 [hep-ph].

[101] S. Wunsch, S. Jórger, R. Wolf, and G. Quast, "Reducing the dependence of the neural network function to systematic uncertainties in the input space," 2019. arXiv: 1907.11674 [physics.data-an].

[102] A. Rogozhnikov, A. Bukva, V. Gligorov, A. Ustyuzhanin, and M. Williams, "New approaches for boosting to uniformity," *JINST*, vol. 10, no. 03, T03002, 2015. DOI: 10.1088/1748-0221/10/03/T03002. arXiv: 1410.4140 [hep-ex].

[103] "A deep neural network to search for new long-lived particles decaying to jets," *Machine Learning: Science and Technology*, 2020. DOI: 10.1088/2632-2153/ab9023. eprint: 1912.12238.

[104] J. M. Clavijo, P. Glaysher, and J. M. Katzy, "Adversarial domain adaptation to reduce sample bias of a high energy physics classifier," 2020. arXiv: 2005.00568 [stat.ML].

[105] G. Kasieczka, B. Nachman, M. D. Schwartz, and D. Shih, "ABCDisCo: Automating the ABCD Method with Machine Learning," Jul. 2020. arXiv: 2007.14400 [hep-ph].

[106] S. Chang, T. Cohen, and B. Ostdiek, "What is the Machine Learning?" *Phys. Rev.*, vol. D97, no. 5, p. 056 009, 2018. DOI: 10.1103/PhysRevD.97.056009. arXiv: 1709.10106 [hep-ph].

[107] J. M. Clavijo, P. Glaysher, and J. M. Katzy, "Adversarial domain adaptation to reduce sample bias of a high energy physics classifier," May 2020. arXiv: 2005.00568 [stat.ML].

[108] A. M. Sirunyan *et al.*, "A deep neural network to search for new long-lived particles decaying to jets," Dec. 2019. arXiv: 1912.12238 [hep-ex].

[109] A. M. Sirunyan *et al.*, "Search for low mass vector resonances decaying into quark-antiquark pairs in proton-proton collisions at $\sqrt{s} = 13$ TeV," *JHEP*, vol. 01, p. 097, 2018. DOI: 10.1007/JHEP01(2018)097. arXiv: 1710.00159 [hep-ex].

[110] A. M. Sirunyan *et al.*, "Search for dark matter produced in association with a Higgs boson decaying to a pair of bottom quarks in proton–proton collisions at $\sqrt{s} = 13$ TeV," *Eur. Phys. J. C*, vol. 79, no. 3, p. 280, 2019. DOI: 10.1140/epjc/s10052-019-6730-7. arXiv: 1811.06562 [hep-ex].

[111] A. M. Sirunyan *et al.*, "Measurement and interpretation of differential cross sections for Higgs boson production at $\sqrt{s} = 13$ TeV," *Phys. Lett. B*, vol. 792, pp. 369–396, 2019. DOI: 10.1016/j.physletb.2019.03.059. arXiv: 1812.06504 [hep-ex].

[112] A. M. Sirunyan *et al.*, "Inclusive search for highly boosted Higgs bosons decaying to bottom quark-antiquark pairs in proton-proton collisions at $\sqrt{s} = 13$ TeV," Jun. 2020. arXiv: 2006.13251 [hep-ex].

[113] R. Aaij *et al.*, "Amplitude analysis of the $B^+ \to D^+D^-K^+$ decay," Aug. 2020. arXiv: 2009.00026 [hep-ex].

[114] R. Aaij *et al.*, "A model-independent study of resonant structure in $B^+ \to D^+D^-K^+$ decays," Aug. 2020. arXiv: 2009.00025 [hep-ex].

[115] R. Aaij *et al.*, "Measurement of the CP-violating phase ϕ_s from $B_s^0 \to J/\psi\pi^+\pi^-$ decays in 13 TeV pp collisions," *Phys. Lett. B*, vol. 797, p. 134 789, 2019. DOI: 10.1016/j.physletb.2019.07.036. arXiv: 1903.05530 [hep-ex].

[116] R. Aaij *et al.*, "Search for a dimuon resonance in the Υ mass region," *JHEP*, vol. 09, p. 147, 2018. DOI: 10.1007/JHEP09(2018)147. arXiv: 1805.09820 [hep-ex].

[117] R. Aaij *et al.*, "Search for hidden-sector bosons in $B^0 \to K^{*0}\mu^+\mu^-$ decays," *Phys. Rev. Lett.*, vol. 115, no. 16, p. 161 802, 2015. DOI: 10.1103/PhysRevLett.115.161802. arXiv: 1508.04094 [hep-ex].

[118] R. Aaij *et al.*, "First observation of forward $Z \to b\bar{b}$ production in pp collisions at $\sqrt{s} = 8$ TeV," *Phys. Lett. B*, vol. 776, pp. 430–439, 2018. DOI: 10.1016/j.physletb.2017.11.066. arXiv: 1709.03458 [hep-ex].

[119] R. Aaij *et al.*, "Measurement of forward $t\bar{t}$, $W + b\bar{b}$ and $W + c\bar{c}$ production in pp collisions at $\sqrt{s} = 8$ TeV," *Phys. Lett. B*, vol. 767, pp. 110–120, 2017. DOI: 10.1016/j.physletb.2017.01.044. arXiv: 1610.08142 [hep-ex].

[120] H. Edwards and A. J. Storkey, "Censoring representations with an adversary," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016. eprint: 1511.05897.

[121] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. March, and V. Lempitsky, "Domain-adversarial training of neural networks," *Journal of Machine Learning Research*, vol. 17, no. 59, pp. 1–35, 2016. eprint: 1505.07818. URL: http://jmlr.org/papers/v17/15-239.html.

[122] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," 2019. arXiv: 1908.09635 [cs.LG].

[123] A. Chouldechova and A. Roth, "The frontiers of fairness in machine learning," 2018. arXiv: 1810.08810 [cs.LG].

[124] L. de Oliveira, M. Kagan, L. Mackey, B. Nachman, and A. Schwartzman, "Jet-Images – Deep Learning Edition.," *JHEP*, vol. 07, p. 069, 2016. DOI: 10.1007/JHEP07(2016)069. arXiv: 1511.05190 [hep-ph].

[125] G. J. Székely, M. L. Rizzo, and N. K. Bakirov, "Measuring and testing dependence by correlation of distances," *Ann. Statist.*, vol. 35, no. 6, pp. 2769–2794, 2007. DOI: 10.1214/009053607000000505. URL: https://doi.org/10.1214/009053607000000505.

[126] G. J. Székely and M. L. Rizzo, "Brownian distance covariance," *Ann. Appl. Stat.*, vol. 3, no. 4, pp. 1236–1265, 2009. DOI: 10.1214/09-AOAS312. URL: https://doi.org/10.1214/09-AOAS312.

[127] G. J. Székely and M. L. Rizzo, "The distance correlation t-test of independence in high dimension," *J. Multivar. Anal.*, vol. 117, pp. 193–213, 2013, ISSN: 0047-259X. DOI: 10.1016/j.jmva.2013.02.012. URL: http://dx.doi.org/10.1016/j.jmva.2013.02.012.

[128] G. J. Székely and M. L. Rizzo, "Partial distance correlation with methods for dissimilarities," *Ann. Statist.*, vol. 42, no. 6, pp. 2382–2412, 2014. DOI: 10.1214/14-AOS1255. URL: https://doi.org/10.1214/14-AOS1255.

[129] T. Sjöstrand, S. Mrenna, and P. Z. Skands, "PYTHIA 6.4 Physics and Manual," *JHEP*, vol. 05, p. 026, 2006. DOI: 10.1088/1126-6708/2006/05/026. arXiv: hep-ph/0603175 [hep-ph].

[130] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C. O. Rasmussen, and P. Z. Skands, "An introduction to PYTHIA 8.2," *Comput. Phys. Commun.*, vol. 191, pp. 159–177, 2015. DOI: 10.1016/j.cpc.2015.01.024. arXiv: 1410.3012 [hep-ph].

[131] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaitre, A. Mertens, and M. Selvaggi, "DELPHES 3, A modular framework for fast simulation of a generic collider experiment," *JHEP*, vol. 02, p. 057, 2014. DOI: 10.1007/JHEP02(2014)057. arXiv: 1307.6346 [hep-ex].

[132] A. Mertens, "New features in Delphes 3," *J. Phys. Conf. Ser.*, vol. 608, p. 012 045, 2015. DOI: 10.1088/1742-6596/608/1/012045.

[133] M. Selvaggi, "DELPHES 3: A modular framework for fast-simulation of generic collider experiments," *J. Phys. Conf. Ser.*, vol. 523, p. 012 033, 2014. DOI: 10.1088/1742-6596/523/1/012033.

[134] M. Cacciari, G. P. Salam, and G. Soyez, "The anti-k_t jet clustering algorithm," *JHEP*, vol. 04, p. 063, 2008. DOI: 10.1088/1126-6708/2008/04/063. arXiv: 0802.1189 [hep-ph].

[135] M. Cacciari, G. P. Salam, and G. Soyez, "FastJet User Manual," *Eur. Phys. J.*, vol. C72, p. 1896, 2012. DOI: 10.1140/epjc/s10052-012-1896-2. arXiv: 1111.6097 [hep-ph].

[136] M. Cacciari and G. P. Salam, "Dispelling the N^3 myth for the k_t jet-finder," *Phys. Lett.*, vol. B641, p. 57, 2006. DOI: 10.1016/j.physletb.2006.08.037. arXiv: hep-ph/0512210 [hep-ph].

[137] "Performance of Top Quark and W Boson Tagging in Run 2 with ATLAS," CERN, Geneva, Tech. Rep. ATLAS-CONF-2017-064, Aug. 2017. URL: https://cds.cern.ch/record/2281054.

[138] A. J. Larkoski, I. Moult, and D. Neill, "Power Counting to Better Jet Observables," *JHEP*, vol. 12, p. 009, 2014. DOI: 10.1007/JHEP12(2014)009. arXiv: 1409.6298 [hep-ph].

[139] J. Thaler and K. Van Tilburg, "Identifying Boosted Objects with N-subjettiness," *JHEP*, vol. 03, p. 015, 2011. DOI: 10.1007/JHEP03(2011)015. arXiv: 1011.2268 [hep-ph].

[140] G. C. Fox and S. Wolfram, "Observables for the analysis of event shapes in e^+e^- annihilation and other processes," *Phys. Rev. Lett.*, vol. 41, pp. 1581–1585, 23 Dec. 1978. DOI: 10.1103/PhysRevLett.41.1581. URL: https://link.aps.org/doi/10.1103/PhysRevLett.41.1581.

[141] L. G. Almeida, S. J. Lee, G. Perez, I. Sung, and J. Virzi, "Top Jets at the LHC," *Phys. Rev. D*, vol. 79, p. 074 012, 2009. DOI: 10.1103/PhysRevD.79.074012. arXiv: 0810.0934 [hep-ph].

[142] G. Aad *et al.*, "ATLAS Measurements of the Properties of Jets for Boosted Particle Searches," *Phys. Rev. D*, vol. 86, p. 072 006, 2012. DOI: 10.1103/PhysRevD.86.072006. arXiv: 1206.5369 [hep-ex].

[143] C. Chen, "New approach to identifying boosted hadronically-decaying particle using jet substructure in its center-of-mass frame," *Phys. Rev. D*, vol. 85, p. 034 007, 2012. DOI: 10.1103/PhysRevD.85.034007. arXiv: 1112.2567 [hep-ph].

[144] J. Thaler and L.-T. Wang, "Strategies to Identify Boosted Tops," *JHEP*, vol. 07, p. 092, 2008. DOI: 10.1088/1126-6708/2008/07/092. arXiv: 0806.0023 [hep-ph].

[145] G. Aad *et al.*, "Measurement of k_T splitting scales in $W \to \ell\nu$ events at $\sqrt{s} = 7$ TeV with the ATLAS detector," *Eur. Phys. J. C*, vol. 73, no. 5, p. 2432, 2013. DOI: 10.1140/epjc/s10052-013-2432-8. arXiv: 1302.1415 [hep-ex].

[146] S. Catani, Y. Dokshitzer, M. Seymour, and B. Webber, "Longitudinally-invariant k $\perp$-clustering algorithms for hadron-hadron collisions," *Nuclear Physics B*, vol. 406, no. 1, pp. 187–224, 1993, ISSN: 0550-3213. DOI: https://doi.org/10.1016/0550-3213(93)90166-M. URL: http://www.sciencedirect.com/science/article/pii/055032139390166M.

[147] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2017. arXiv: 1710.05941 [cs.NE].

[148] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. URL: http://arxiv.org/abs/1412.6980.

[149] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," 2018. arXiv: 1708.07120 [cs.LG].

[150] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," 2017. arXiv: 1608.03983 [cs.LG].

[151] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," 2017. arXiv: 1708.07120 [cs.LG].

[152] C. M. Bishop, "Mixture density networks," Birmingham, Technical Report, 1994. URL: http://publications.aston.ac.uk/id/eprint/373/.

[153] G. Kasieczka and D. Shih, *Datasets for boosted w tagging*, version v1, Zenodo, Jan. 2020. DOI: 10.5281/zenodo.3606767. URL: https://doi.org/10.5281/zenodo.3606767.

[154] P. T. Komiske, E. M. Metodiev, and J. Thaler, "Metric space of collider events," *Physical Review Letters*, vol. 123, no. 4, Jul. 2019. DOI: 10.1103/physrevlett.123.041801. URL: https://doi.org/10.1103%2Fphysrevlett.123.041801.

[155] P. T. Komiske, E. M. Metodiev, and J. Thaler, "The Hidden Geometry of Particle Collisions," *JHEP*, vol. 07, p. 006, 2020. DOI: 10.1007/JHEP07(2020)006. arXiv: 2004.04159 [hep-ph].

[156] R. Gambhir, A. Dogra, J. Thaler, D. Ba, and A. Tasissa, "Can you hear the shape of a jet?" 14th International Workshop on Boosted Object Phenomenology, Reconstruction, Measurements, and Searches in HEP, 2022. URL: https://indi.to/rbQ5j.

[157] J. Feydy, T. Séjourné, F.-X. Vialard, S.-i. Amari, A. Trouvé, and G. Peyré, *Interpolating between optimal transport and mmd using sinkhorn divergences*, 2018. DOI: 10.48550/ARXIV.1810.08278. URL: https://arxiv.org/abs/1810.08278.

[158] M. Arratia, Y. Makris, D. Neill, F. Ringer, and N. Sato, "Asymmetric jet clustering in deep-inelastic scattering," *Phys. Rev. D*, vol. 104, no. 3, p. 034005, 2021. DOI: 10.1103/PhysRevD.104.034005. arXiv: 2006.10751 [hep-ph].

[159] O. Kitouni, N. Nolte, V. S. Pérez-Díaz, S. Trifinopoulos, and M. Williams, "From Neurons to Neutrons: A Case Study in Interpretability," in *41st International Conference on Machine Learning*, May 2024. arXiv: 2405.17425 [cs.LG].

[160] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[161] I. Higgins, D. Amos, D. Pfau, S. Racaniere, L. Matthey, D. Rezende, and A. Lerchner, "Towards a definition of disentangled representations," *arXiv preprint arXiv:1812.02230*, 2018.

[162] F. Locatello, S. Bauer, M. Lucic, G. Raetsch, S. Gelly, B. Sch"olkopf, and O. Bachem, "Challenging common assumptions in the unsupervised learning of disentangled representations," in *International Conference on Machine Learning*, PMLR, 2019, pp. 4114–4124.

[163] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, "Understanding disentangling in β-vae," in *NeurIPS Workshop on Learning Disentangled Representations*, 2018.

[164] R. T. Chen, X. Li, R. B. Grosse, and D. K. Duvenaud, "Isolating sources of disentanglement in variational autoencoders," in *Advances in Neural Information Processing Systems*, 2018, pp. 2610–2620.

[165] H. Kim and A. Mnih, "Disentangling by factorising," in *International Conference on Machine Learning*, PMLR, 2018, pp. 2649–2658.

[166] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, *Lora: Low-rank adaptation of large language models*, 2021. arXiv: 2106.09685 [cs.CL].

[167] A. Aghajanyan, S. Gupta, and L. Zettlemoyer, "Intrinsic dimensionality explains the effectiveness of language model fine-tuning," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2021, pp. 7319–7328.

[168] C. Li, H. Farkhoor, R. Liu, and J. Yosinski, "Measuring the intrinsic dimension of objective landscapes," in *International Conference on Learning Representations*, 2018.

[169] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient Finetuning of Quantized LLMs," *arXiv e-prints*, arXiv:2305.14314, arXiv:2305.14314, May 2023. DOI: 10.48550/arXiv.2305.14314. arXiv: 2305.14314 [cs.LG].

[170] Q. Zhang, M. Chen, A. Bukharin, N. Karampatziakis, P. He, Y. Cheng, W. Chen, and T. Zhao, "AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning," *arXiv e-prints*, arXiv:2303.10512, arXiv:2303.10512, Mar. 2023. DOI: 10.48550/arXiv.2303.10512. arXiv: 2303.10512 [cs.CL].

[171] T. Kadir and M. Brady, "Saliency, scale and image description," *International Journal of Computer Vision*, vol. 45, no. 2, pp. 83–105, 2001.

[172] Y. Zhang, P. Tiňo, A. Leonardis, and K. Tang, "A survey on neural network interpretability," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 5, pp. 726–742, 2021. DOI: 10.1109/TETCI.2021.3100641.

[173] N. Elhage, N. Nanda, C. Olsson, *et al.*, "A mathematical framework for transformer circuits," *Transformer Circuits Thread*, 2021, https://transformer-circuits.pub/2021/framework/index.ht

[174] C. Olah, "Mechanistic interpretability, variables, and the importance of interpretable bases," *Transformer Circuits Thread*, 2022, https://transformer-circuits.pub/2022/mech-interp-essay/index.html.

[175] Z. Liu, O. Kitouni, N. S. Nolte, E. Michaud, M. Tegmark, and M. Williams, "Towards understanding grokking: An effective theory of representation learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 34 651–34 663, 2022.

[176] N. Nanda, L. Chan, T. Lieberum, J. Smith, and J. Steinhardt, "Progress measures for grokking via mechanistic interpretability," *arXiv preprint arXiv:2301.05217*, 2023.

[177] C. F. v. Weizsäcker, "Zur theorie der kernmassen," *Zeitschrift für Physik*, vol. 96, no. 7, pp. 431–458, Jul. 1935, ISSN: 0044-3328. DOI: 10.1007/BF01337700. URL: https://doi.org/10.1007/BF01337700.

[178] Z. Zhong, Z. Liu, M. Tegmark, and J. Andreas, "The clock and the pizza: Two stories in mechanistic explanation of neural networks," *arXiv preprint arXiv:2306.17844*, 2023.

[179] M. Hassid, H. Peng, D. Rotem, J. Kasai, I. Montero, N. A. Smith, and R. Schwartz, "How much does attention actually attend? questioning the importance of attention in pretrained transformers," *arXiv preprint arXiv:2211.03495*, 2022.

[180] O. Kitouni, N. Nolte, S. Trifinopoulos, S. Kantamneni, and M. Williams, *Nuclr: Nuclear co-learned representations*, 2023. arXiv: 2306.06099 [nucl-th]. URL: https://arxiv.org/abs/2306.06099.

[181] J. Novembre and M. Stephens, "Interpreting principal component analyses of spatial population genetic variation," *Nature genetics*, vol. 40, no. 5, pp. 646–649, 2008.

[182] J. Antognini and J. Sohl-Dickstein, "Pca of high dimensional random walks with comparison to neural network training," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[183] M. A. Lebedev, A. Ossadtchi, N. A. Mill, N. A. Urpí, M. R. Cervera, and M. A. Nicolelis, "Analysis of neuronal ensemble activity reveals the pitfalls and shortcomings of rotation dynamics," *Scientific Reports*, vol. 9, no. 1, p. 18 978, 2019.

[184] T. Proix, M. G. Perich, and T. Milekovic, "Interpreting dynamics of neural activity after dimensionality reduction," *bioRxiv*, pp. 2022–03, 2022.

[185] S. Ashkboos, M. L. Croci, M. G. do Nascimento, T. Hoefler, and J. Hensman, *Slicegpt: Compress large language models by deleting rows and columns*, 2024. arXiv: 2401.15024 [cs.LG].

[186] M. Shinn, "Phantom oscillations in principal component analysis," *bioRxiv*, pp. 2023–06, 2023.

[187] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[188] W. Pauli, "Über den zusammenhang des abschlusses der elektronengruppen im atom mit der komplexstruktur der spektren," *Zeitschrift für Physik*, vol. 31, no. 1, pp. 765–783, Feb. 1925, ISSN: 0044-3328. DOI: 10.1007/BF02980631. URL: https://doi.org/10.1007/BF02980631.

[189] P. Lemos, N. Jeffrey, M. Cranmer, S. Ho, and P. Battaglia, "Rediscovering orbital mechanics with machine learning," *Machine Learning: Science and Technology*, vol. 4, no. 4, p. 045 002, 2023.

[190] M. Cranmer, "Interpretable machine learning for science with pysr and symbolicregression. jl," *arXiv preprint arXiv:2305.01582*, 2023.

[191] K. Li, A. K. Hopkins, D. Bau, F. Viégas, H. Pfister, and M. Wattenberg, "Emergent World Representations: Exploring a Sequence Model Trained on a Synthetic Task," *arXiv e-prints*, arXiv:2210.13382, arXiv:2210.13382, Oct. 2022. DOI: 10.48550/arXiv.2210.13382. arXiv: 2210.13382 [cs.LG].

[192] Y. Benchekroun, M. Dervishi, M. Ibrahim, J.-B. Gaya, X. Martinet, G. Mialon, T. Scialom, E. Dupoux, D. Hupkes, and P. Vincent, "WorldSense: A Synthetic Benchmark for Grounded Reasoning in Large Language Models," *arXiv e-prints*, arXiv:2311.15930, arXiv:2311.15930, Nov. 2023. DOI: 10.48550/arXiv.2311.15930. arXiv: 2311.15930 [cs.CL].

[193] S. R. Bowman, "Eight Things to Know about Large Language Models," *arXiv e-prints*, arXiv:2304.00612, arXiv:2304.00612, Apr. 2023. DOI: 10.48550/arXiv.2304.00612. arXiv: 2304.00612 [cs.CL].

[194] J. Roberts, T. Lüddecke, S. Das, K. Han, and S. Albanie, "GPT4GEO: How a Language Model Sees the World's Geography," *arXiv e-prints*, arXiv:2306.00020, arXiv:2306.00020, May 2023. DOI: 10.48550/arXiv.2306.00020. arXiv: 2306.00020 [cs.CL].

[195] W. Gurnee and M. Tegmark, "Language Models Represent Space and Time," *arXiv e-prints*, arXiv:2310.02207, arXiv:2310.02207, Oct. 2023. DOI: 10.48550/arXiv.2310.02207. arXiv: 2310.02207 [cs.LG].

[196] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[197] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, Springer, 2014, pp. 818–833.

[198] C. Olah, L. Schubert, and A. Mordvintsev, "Feature visualization," *Distill*, 2017. URL: https://distill.pub/2017/feature-visualization/.

[199] N. Nanda and T. Lieberum, *A mechanistic interpretability analysis of grokking*, 2022. URL: https://www.alignmentforum.org/posts/N6WM6hs7RQMKDhYjB/a-mechanistic-interpretability-analysis-of-grokking.

[200] B. Millidge, *Grokking 'grokking'*, https://beren.io/2022-01-11-Grokking-Grokking/, 2022.

[201] R. Shah, *Alignment Newsletter #159*, https://www.alignmentforum.org/posts/zvWqPmQasssaAWkrj/an-159-building-agents-that-know-how-to-experiment-by#DEEP_LEARNING_, 2021.

[202] Y. Hoshen and S. Peleg, "Visual learning of arithmetic operation," in *AAAI*, 2016.

[203] Y.-H. He, "Machine-learning mathematical structures," *arXiv preprint arXiv:2101.06317*, 2021.

[204] S. Gukov, J. Halverson, F. Ruehle, and P. Sułkowski, "Learning to unknot," *Machine Learning: Science and Technology*, vol. 2, no. 2, p. 025 035, 2021.

[205] A. Davies, P. Veličković, L. Buesing, S. Blackwell, D. Zheng, N. Tomašev, R. Tanburn, P. Battaglia, C. Blundell, A. Juhász, *et al.*, "Advancing mathematics by guiding human intuition with ai," *Nature*, vol. 600, no. 7887, pp. 70–74, 2021.

[206] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever, "Deep double descent: Where bigger models and more data hurt," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2021, no. 12, p. 124 003, 2021.

[207] P. Nakkiran, P. Venkat, S. Kakade, and T. Ma, "Optimal regularization can mitigate double descent," *arXiv preprint arXiv:2003.01897*, 2020.

[208] Y. Ouali, C. Hudelot, and M. Tami, "An overview of deep semi-supervised learning," *arXiv preprint arXiv:2006.05278*, 2020.

[209] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 271–21 284, 2020.

[210] P. H. Le-Khac, G. Healy, and A. F. Smeaton, "Contrastive representation learning: A framework and review," *IEEE Access*, vol. 8, pp. 193 907–193 934, 2020.

[211] J. Halverson, A. Maiti, and K. Stoner, "Neural networks and quantum field theory," *Machine Learning: Science and Technology*, vol. 2, no. 3, p. 035 002, 2021.

[212] D. A. Roberts, S. Yaida, and B. Hanin, "The principles of deep learning theory," *arXiv preprint arXiv:2106.10165*, 2021.

[213] D. Kunin, J. Sagastuy-Brena, S. Ganguli, D. L. Yamins, and H. Tanaka, "Neural mechanics: Symmetry and broken conservation laws in deep learning dynamics," *arXiv preprint arXiv:2012.04728*, 2020.

[214] Y. Gao and P. Chaudhari, "A free-energy principle for representation learning," in *International Conference on Machine Learning*, PMLR, 2020, pp. 3367–3376.

[215] F. Gerace, B. Loureiro, F. Krzakala, M. Mézard, and L. Zdeborová, "Generalisation error in learning with random features and the hidden manifold model," in *International Conference on Machine Learning*, PMLR, 2020, pp. 3452–3462.

[216] M. Pezeshki, A. Mitra, Y. Bengio, and G. Lajoie, "Multi-scale feature learning dynamics: Insights for double descent," in *International Conference on Machine Learning*, PMLR, 2022, pp. 17 669–17 690.

[217] S. Goldt, B. Loureiro, G. Reeves, F. Krzakala, M. Mezard, and L. Zdeborova, "The gaussian equivalence of generative models for learning with shallow neural networks," in *Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference*, J. Bruna, J. Hesthaven, and L. Zdeborova, Eds., ser. Proceedings of Machine Learning Research, vol. 145, PMLR, Aug. 2022, pp. 426–471. URL: https://proceedings.mlr.press/v145/goldt22a.html.

[218] R. Kuhn and S. Bos, "Statistical mechanics for neural networks with continuous-time dynamics," *Journal of Physics A: Mathematical and General*, vol. 26, no. 4, p. 831, 1993.

[219] C. Olsson, N. Elhage, N. Nanda, *et al.*, "In-context learning and induction heads," *Transformer Circuits Thread*, 2022, https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

[220] D. A. Roberts, S. Yaida, and B. Hanin, *The Principles of Deep Learning Theory*. Cambridge University Press, 2022, https://deeplearningtheory.com. arXiv: 2106.10165 [cs.LG].

[221] Z. Liu, E. J. Michaud, and M. Tegmark, *Omnigrok: Grokking beyond algorithmic data*, 2022. arXiv: 2210.01117 [cs.LG].

[222] D. Ganguli, D. Hernandez, L. Lovitt, N. DasSarma, T. Henighan, A. Jones, N. Joseph, J. Kernion, B. Mann, A. Askell, *et al.*, "Predictability and surprise in large generative models," *arXiv preprint arXiv:2202.07785*, 2022.

[223] J. Steinhardt, *Future ML Systems Will Be Qualitatively Different*, https://www.lesswrong.com/s/4aARF2ZoBpFZAhbbe/p/pZaPhGg2hmmPwByHc, 2022.

[224] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, *Deep unsupervised learning using nonequilibrium thermodynamics*, 2015. arXiv: 1503.03585 [cs.LG]. URL: https://arxiv.org/abs/1503.03585.

[225] O. Kitouni, N. Nolte, J. Hensman, and B. Mitra, *Disk: A diffusion model for structured knowledge*, 2024. arXiv: 2312.05253 [cs.LG].

[226] J. Austin, D. D. Johnson, J. Ho, D. Tarlow, and R. van den Berg, *Structured denoising diffusion models in discrete state-spaces*, 2023. arXiv: 2107.03006 [cs.LG].

[227] O. Kitouni, N. Nolte, D. Bouchacourt, A. Williams, M. Rabbat, and M. Ibrahim, *The factorization curse: Which tokens you predict underlie the reversal curse and more*, 2024. arXiv: 2406.05183 [cs.LG]. URL: https://arxiv.org/abs/2406.05183.

[228] M. Dahl, V. Magesh, M. Suzgun, and D. E. Ho, *Hallucinating Law: Legal Mistakes with Large Language Models are Pervasive*, 2024. URL: https://hai.stanford.edu/news/hallucinating-law-legal-mistakes-large-language-models-are-pervasive (visited on 05/21/2024).

[229] L. Berglund, M. Tong, M. Kaufmann, M. Balesni, A. C. Stickland, T. Korbak, and O. Evans, *The reversal curse: Llms trained on "a is b" fail to learn "b is a"*, 2023. arXiv: 2309.12288 [cs.CL].

[230] Z. Allen-Zhu and Y. Li, *Physics of language models: Part 3.2, knowledge manipulation*, 2023.

[231] O. Golovneva, Z. Allen-Zhu, J. Weston, and S. Sukhbaatar, *Reverse training to nurse the reversal curse*, 2024. arXiv: 2403.13799 [cs.CL].

[232] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, *Language models are unsupervised multitask learners*, 2019. URL: https://api.semanticscholar.org/CorpusID:160025533.

[233] H. Touvron, T. Lavril, G. Izacard, *et al.*, *Llama: Open and efficient foundation language models*, 2023. arXiv: 2302.13971 [cs.CL].

[234] H. Touvron, L. Martin, K. Stone, *et al.*, *Llama 2: Open foundation and fine-tuned chat models*, 2023. arXiv: 2307.09288 [cs.CL].

[235] OpenAI, "Gpt-4 technical report," *PREPRINT*, 2023.

[236] Y. Tay, M. Dehghani, V. Q. Tran, *et al.*, "Ul2: Unifying language learning paradigms," in *International Conference on Learning Representations*, 2022. URL: https://api.semanticscholar.org/CorpusID:252780443.

[237] J. Zhang, N. Nolte, R. Sadhukhan, B. Chen, and L. Bottou, *Memory mosaics*, 2024. arXiv: 2405.06394 [cs.LG].

[238] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, *Xlnet: Generalized autoregressive pretraining for language understanding*, 2020. arXiv: 1906.08237 [cs.CL].

[239] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds., Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[240] A. Q. Jiang, A. Sablayrolles, A. Mensch, *et al.*, *Mistral 7b*, 2023. arXiv: 2310.06825 [cs.CL].

[241] Z. A. Zhu and Y. Li, *Physics of language models: Part 3.1, knowledge storage and extraction*, 2023.

[242] S. Mindermann, J. Brauner, M. Razzak, *et al.*, *Prioritized training on points that are learnable, worth learning, and not yet learnt*, 2022. arXiv: 2206.07137 [cs.LG].

[243] Z. Jin, Q. Guo, X. Qiu, and Z. Zhang, "GenWiki: A dataset of 1.3 million content-sharing text and graphs for unsupervised graph-to-text generation," in *Proceedings of the 28th International Conference on Computational Linguistics*, D. Scott, N. Bel, and C. Zong, Eds., Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 2398–2409. DOI: 10.18653/v1/2020.coling-main.217. URL: https://aclanthology.org/2020.coling-main.217.

[244] A. Wettig, T. Gao, Z. Zhong, and D. Chen, *Should you mask 15% in masked language modeling?* 2023. arXiv: 2202.08005 [cs.CL].

[245] N. Dziri, X. Lu, M. Sclar, *et al.*, *Faith and fate: Limits of transformers on compositionality*, 2023. arXiv: 2305.18654 [cs.CL].

[246] Y. LeCun, *Do large language models need sensory ground- ing for meaning and understanding?* University Lecture, 2023.

[247] F. Gloeckle, B. Y. Idrissi, B. Rozière, D. Lopez-Paz, and G. Synnaeve, *Better & faster large language models via multi-token prediction*, 2024.

[248] G. Bachmann and V. Nagarajan, *The pitfalls of next-token prediction*, 2024. arXiv: 2403.06963 [cs.CL].

[249] O. Pfungst and R. Rosenthal, *Clever hans : The horse of mr. von osten*, 1911. URL: https://api.semanticscholar.org/CorpusID:142217369.

[250] O. Glickman, I. Dagan, and M. Koppel, "Web based probabilistic textual entailment," in *Proceedings of the 1st Pascal Challenge Workshop*, 2005, pp. 33–36.

[251] R. Adams, G. Nicolae, C. Nicolae, and S. Harabagiu, "Textual entailment through extended lexical overlap and lexico-semantic matching," in *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, S. Sekine, K. Inui, I. Dagan, B. Dolan, D. Giampiccolo, and B. Magnini, Eds., Prague: Association for Computational Linguistics, Jun. 2007, pp. 119–124. URL: https://aclanthology.org/W07-1420.

[252] I. Dasgupta, D. Guo, A. Stuhlmüller, S. J. Gershman, and N. D. Goodman, "Evaluating compositionality in sentence embeddings," in *Proceedings of the 40th Annual Conference of the Cognitive Science Society*, Madison, WI, 2018, pp. 1596–1601. URL: https://cognitivesciencesociety.org/wp-content/uploads/2019/01/cogsci18_proceedings.pdf.

[253] A. Naik, A. Ravichander, N. Sadeh, C. Rose, and G. Neubig, "Stress test evaluation for natural language inference," in *Proceedings of the 27th International Conference on Computational Linguistics*, E. M. Bender, L. Derczynski, and P. Isabelle, Eds., Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 2340–2353. URL: https://aclanthology.org/C18-1198.

[254] I. Sanchez, J. Mitchell, and S. Riedel, "Behavior analysis of NLI models: Uncovering the influence of three factors on robustness," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, M. Walker, H. Ji, and A. Stent, Eds., New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 1975–1985. DOI: 10.18653/v1/N18-1179. URL: https://aclanthology.org/N18-1179.

[255] T. McCoy, E. Pavlick, and T. Linzen, "Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, A. Korhonen, D. Traum, and L. Màrquez, Eds., Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 3428–3448. DOI: 10.18653/v1/P19-1334. URL: https://aclanthology.org/P19-1334.

[256] S. Rajaee, Y. Yaghoobzadeh, and M. T. Pilehvar, "Looking at the overlooked: An analysis on the word-overlap bias in natural language inference," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Y. Goldberg, Z. Kozareva, and Y. Zhang, Eds., Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 10 605–10 616. DOI: 10.18653/v1/2022.emnlp-main.725. URL: https://aclanthology.org/2022.emnlp-main.725.

[257] A. Williams, T. Thrush, and D. Kiela, "ANLIzing the adversarial natural language inference dataset," in *Proceedings of the Society for Computation in Linguistics 2022*, A. Ettinger, T. Hunter, and B. Prickett, Eds., online: Association for Computational Linguistics, Feb. 2022, pp. 23–54. URL: https://aclanthology.org/2022.scil-1.3.

[258] K. Sinha, S. Sodhani, J. Dong, J. Pineau, and W. L. Hamilton, "CLUTRR: A diagnostic benchmark for inductive reasoning from text," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds., Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 4506–4515. DOI: 10.18653/v1/D19-1458. URL: https://aclanthology.org/D19-1458.

[259] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[260] J. Gauthier and R. Levy, "Linking artificial and human neural representations of language," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds., Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 529–539. DOI: 10.18653/v1/D19-1050. URL: https://aclanthology.org/D19-1050.

[261] D. C. Chiang and H. Lee, "Pre-training a language model without human language," *CoRR*, vol. abs/2012.11995, 2020. arXiv: 2012.11995. URL: https://arxiv.org/abs/2012.11995.

[262] K. Sinha, R. Jia, D. Hupkes, J. Pineau, A. Williams, and D. Kiela, "Masked language modeling and the distributional hypothesis: Order word matters pre-training for little," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, Eds., Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 2888–2913. DOI: 10.18653/v1/2021.emnlp-main.230. URL: https://aclanthology.org/2021.emnlp-main.230.

[263] A. Lv, K. Zhang, S. Xie, Q. Tu, Y. Chen, J.-R. Wen, and R. Yan, *Are we falling in a middle-intelligence trap? an analysis and mitigation of the reversal curse*, 2023. arXiv: 2311.07468 [cs.CL].

[264] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "SpanBERT: Improving Pre-training by Representing and Predicting Spans," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 64–77, Jan. 2020, ISSN: 2307-387X. DOI: 10.1162/tacl_a_00300. eprint: https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00300/1923170/tacl_a_00300.pdf. URL: https://doi.org/10.1162/tacl%5C_a%5C_00300.

[265] A. Chowdhery, S. Narang, J. Devlin, *et al.*, *Palm: Scaling language modeling with pathways*, 2022. arXiv: 2204.02311 [cs.CL].

[266] A. Sivaraman, G. Farnadi, T. Millstein, and G. Van den Broeck, "Counterexample-guided learning of monotonic neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 11 936–11 948, 2020.

[267] S. A. J. Larson and L. Kirchner, *There's software used across the country to predict future criminals. and it's biased against blacks*, 2016.

[268] K. Buza, "Feedback prediction for blogs," in *Data analysis, machine learning and knowledge discovery*, Springer, 2014, pp. 145–152.

[269] Kaggle, "Lending club loan data," in 2015. URL: https://www.kaggle.com/datasets/wordsforthewise/lending-club.

[270] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. Summers, "Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases," in *IEEE CVPR*, vol. 7, 2017. URL: https://www.kaggle.com/datasets/nih-chest-xrays/sample.

[271] D. Dua and C. Graff, *UCI machine learning repository*, 2017. URL: http://archive.ics.uci.edu/ml.

[272] J. H. Gennari, P. Langley, and D. Fisher, "Models of incremental concept formation," *Artificial Intelligence*, vol. 40, no. 1, pp. 11–61, 1989, ISSN: 0004-3702. DOI: https://doi.org/10.1016/0004-3702(89)90046-5. URL: https://www.sciencedirect.com/science/article/pii/0004370289900465.

[273] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[274] T. Huster, C.-Y. J. Chiang, and R. Chadha, "Limitations of the lipschitz constant as a defense against adversarial examples," in *ECML PKDD 2018 Workshops: Nemesis 2018, UrbReas 2018, SoGood 2018, IWAISe 2018, and Green Data Mining 2018, Dublin, Ireland, September 10-14, 2018, Proceedings 18*, Springer, 2019, pp. 16–29.

[275] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning (still) requires rethinking generalization," *Communications of the ACM*, vol. 64, no. 3, pp. 107–115, 2021.

[276] M. Wang, W. J. Huang, F. G. Kondev, G. Audi, and S. Naimi, "The AME 2020 atomic mass evaluation (II). Tables, graphs and references," *Chin. Phys. C*, vol. 45, no. 3, p. 030 003, 2021. DOI: 10.1088/1674-1137/abddaf.

[277] I. Angeli and K. P. Marinova, "Table of experimental nuclear ground state charge radii: An update," *Atomic Data and Nuclear Data Tables*, vol. 99, no. 1, pp. 69–95, Jan. 2013. DOI: 10.1016/j.adt.2011.12.006.

[278] H. A. Bethe and R. F. Bacher, "Nuclear Physics A. Stationary States of Nuclei," *Rev. Mod. Phys.*, vol. 8, pp. 82–229, 1936. DOI: 10.1103/RevModPhys.8.82.

[279] M. W. Kirson, "Mutual influence of terms in a semi-empirical mass formula," *Nucl. Phys. A*, vol. 798, pp. 29–60, 2008. DOI: 10.1016/j.nuclphysa.2007.10.011.

[280] T. Mengel, P. Steffanic, C. Hughes, A. C. O. da Silva, and C. Nattrass, "Interpretable machine learning methods applied to jet background subtraction in heavy ion collisions," *arXiv preprint arXiv:2303.08275*, 2023.

[281] B. L. Davis and Z. Jin, "Discovery of a planar black hole mass scaling relation for spiral galaxies," *The Astrophysical Journal Letters*, vol. 956, no. 1, p. L22, 2023.

[282] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," *Advances in neural information processing systems*, vol. 30, 2017.

[283] V. Papyan, X. Han, and D. L. Donoho, "Prevalence of neural collapse during the terminal phase of deep learning training," *Proceedings of the National Academy of Sciences*, vol. 117, no. 40, pp. 24 652–24 663, 2020.

[284] Wikipedia contributors, *Thomson problem — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/w/index.php?title=Thomson_problem&oldid=1091431454, [Online; accessed 29-July-2022], 2022.

[285] X. Chen and K. He, "Exploring simple siamese representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 750–15 758.

[286] Z.-Q. J. Xu, Y. Zhang, and Y. Xiao, "Training behavior of deep neural network in frequency domain," in *International Conference on Neural Information Processing*, Springer, 2019, pp. 264–274.

[287] Y. Zhang, Z.-Q. J. Xu, T. Luo, and Z. Ma, "A type of generalization error induced by initialization in deep neural networks," in *Mathematical and Scientific Machine Learning*, PMLR, 2020, pp. 144–164.

[288] B. Woodworth, S. Gunasekar, J. D. Lee, E. Moroshko, P. Savarese, I. Golan, D. Soudry, and N. Srebro, "Kernel and rich regimes in overparametrized models," in *Proceedings of Thirty Third Conference on Learning Theory*, J. Abernethy and S. Agarwal, Eds., ser. Proceedings of Machine Learning Research, vol. 125, PMLR, Jul. 2020, pp. 3635–3673. URL: https://proceedings.mlr.press/v125/woodworth20a.html.