UNIVERSITY OF CALIFORNIA

Los Angeles

Aspects of Many-Body Physics and Machine Learning

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Physics

by

Steven Matthew Durr

2022

ABSTRACT OF THE DISSERTATION

Aspects of Many-Body Physics and Machine Learning

by

Steven Matthew Durr
Doctor of Philosophy in Physics
University of California, Los Angeles, 2022
Professor Sudip Chakravarty, Chair

After an introduction which gives a brief overview of the relevant machine learning topics, we begin chapter 1 by introducing the phenomenon of many-body localization (MBL). We emphasize its significance by contrasting many-body localized states with thermal states, and describe efforts to distinguish the two phases. We then move on to describe the phenomenological picture of MBL which makes use of local integrals of motion (LIOMs). This enlightening picture explains many of the features of MBL, and has inspired various renormalization group approaches to understanding the phase. We illustrate the intuition for this picture using one such numerical RG scheme known as the Spectrum Bifurcation Renormalization Group, which allows one to construct iterative diagonalizing unitaries and obtain a LIOM representation of a given Hamiltonian.

After introducing the topic of many-body localization, we describe an application of unsupervised learning to the problem of distinguishing distinct phases within the MBL regime. This begins with a discussion of many-body localized phases, and the difficulty of identifying them. We then give an approach to this problem using a form of unsupervised learning, in which eigenstates are classified by clustering them according to their entanglement spectra. This is compared to the state of the art approach, which uses supervised learning. Despite the comparable results of the two methods, the unsupervised technique introduced relies on no labeled data. Potential applications as well as limitations of the method are then described.

In chapter 2, we recount recent research which applies a physics-based approach in order to understand the training dynamics of a class of machine learning models known as generative adversarial networks (GANs). This begins with an introduction to GANs, and a description of GAN training. Emphasizing the difficulty of training GANs, we then outline a very common problematic phenomenon within GAN training, known as mode collapse, in which outputs fail to become sufficiently diverse.

We then distill the key features necessary to observe this phenomenon of GAN failure, and construct a toy model for GAN training in which we can study it with more clarity. Finally, within this toy model, we are able to identify a phase transition, separating the regime of successful training from the regime of mode collapse. Additionally, we diagnose the source of this transition. We conclude by remarking on potential applications of such a result to a more realistic GAN setting.

The dissertation of Steven Matthew Durr is approved.

Shenshen Wang

Stuart Brown

Yaroslav Tserkovnyak

Sudip Chakravarty, Committee Chair

University of California, Los Angeles

2022

To Laura

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to thank my parents and my sister for their support throughout my life. Thank you for making me who I am. Thank you to Keegan Stoner, my childhood best friend. Since then, we have lived parallel lives, pursuing physics degrees and finding incredible significant others to spend our lives with. Also, many thanks to my fellow graduate students – among them Ian Powell, John Terry, Kevin Chen, and Pratik Sathe – for their friendship and kindness. I am profoundly grateful for the constant support of Laura Kambourian, my girlfriend over these six years.

I have been lucky enough to have a series of excellent mentors throughout middle and high school. I am incredibly grateful for my middle school math teacher, John Tulloch, who encouraged me to be a lifelong philomath. Thank you to Martha Ryan, who taught me science and was incredibly compassionate while doing so. Thanks to Heidi Gleason, who left a lasting mark on my life by guiding me in the pursuit science research as a discipline. I am forever indebted to Tom Ross, who convinced me to take calculus early, taught me physics, and always treated me like a mature adult even when I wasn't one.

I am extremely grateful to my advisor, Sudip Chakravarty, for taking me into his group those many years ago, and giving me the freedom to pursue a broad range of research interests.

Thank you to Shenshen Wang, who has always been so profoundly generous with her time and attention. Finally, I would like to thank my committee members including Stuart Brown and Yaroslav Tserkovnyak.

VITA

2016	Bachelor of Arts in Physics, Mathematics (Double Major), Cornell University.
2017	M.S. (Physics), UCLA, Los Angeles, California.
2016–2022	Teaching Assistant, Department of Physics and Astronomy, UCLA.

PUBLICATIONS

Durr, S., Chakravarty, S. *Unsupervised learning eigenstate phases of matter.*, Phys. Rev. B. **100**(7):075102, Aug 2019

Powell, I. E., Durr, S., Rombes, N. and Chakravarty, S. *Density Wave Mediated Dzyaloshinskii Moriya Interactions.*, Phys. Rev. B. **103**(2):024433, January 2021

Durr, S., Tu, Y., Mroueh, Y. and Wang, S. *A Phase Transition in Model GAN Dynamics.*, *in preparation*

Introduction

The power of machine learning (ML) applications over the past decades has brought the field to prominence [LBD89, KSH12, MSC13, GPM14, MKS15, HZR16, VSP17, BMR20]. Broadly, the purpose of machine learning is to extract patterns present in data, and through advancements in both ML tools and computing power, this has been performed with great success. ML tools are now used regularly to analyze extremely large and convoluted data-sets, far too complex for humans to parse. State-of-the-art ML approaches, such as deep learning [LBH15], have achieved near (or beyond) human-level performance on many tasks previously assumed to be inaccessible to computer algorithms, including translation [VSP17, BCB15], automatic driving [KST21], and strategic games such as Go [SHM16].

The success of machine learning has also encouraged diverse interdisciplinary directions of research. In particular, ML has found relevance within the physical sciences [CCC19, MBW18], where researchers regularly encounter profoundly large and complex data-sets, and are tasked with understanding the structure present within them.

Quantum mechanics, for example, is naturally accompanied by data which might live in an extremely large vector space of exponentially growing dimension. In the past, faced with the problem of understanding the underlying physics from these impossibly complicated data-sets, physicists have turned to techniques such as random matrix theory [Wig55, Wig57]. By applying tools from random matrix theory to the spectra of many-body Hamiltonians, physicists have been able to draw important conclusions concerning, for example, ergodicity.

In recent years, The strength of machine learning when applied to large data-sets has motivated its use by physicist to analyze forms of quantum data. In pursuing this approach, physicists have encountered remarkable success. By using a simple form of neural network called a restricted Boltzmann machine, it was demonstrated that neural networks could efficiently represent many-body ground states using relatively few parameters [CT17, MCC19]. From this starting point, other types of neural networks have been successfully applied to versions of this problem, with later researchers studying neural network wavefunctions built from models typically used for modeling language, or for image recognition [HGH20, LSC19, LDH21, GD17].

Beyond ground state ansatze, machine learning has been applied to the problem of predicting phases of matter [Wet17, Wan16, WS17]. This is particularly useful in cases where distinct order parameters are not present. In this work, we will describe a technique for predicting out-of-equilibrium phases of matter, and an application which yields impressive results compared to the best available alternative.

As we will observe, the strength of machine learning comes at a cost. Although analyses and predictions provided by ML models can be extremely useful, the outputs of these models are often difficult or impossible to rigorously explain. Indeed, the profound success of certain ML tools such as deep learning has preceded by far the understanding of how these tools work[RYH22]. Therefore, machine learning has become not only a tool for researchers, but a topic of study in its own right. In recent years, physicists in particular have applied techniques from their field to study the training dynamics of machine learning models, explain the predictive tendencies of neural networks, and lay the mathematical foundations for the theory behind deep learning [RYH22, MS14, HMS21, FT21].

The application of physics to understand new technologies is not novel. The progress of deep learning in the 21$^{\text{st}}$ century has been fueled by technological advances in computing, rather than theoretical foundations. This fact has been compared to the progress of steam engines in the 19$^{\text{th}}$ century, fueled by industrial advances [RYH22]. Physicists, with the motivation of understanding these engines [CTC], developed thermodynamics and later statistical mechanics.

Along with researchers from other fields, modern day physicists have found themselves in an analogous position: tasked with applying theoretical tools to explain the success of neural networks and deep learning. Consequently, concepts from high-energy theory [HMS21], condensed matter theory [MS14], spin glasses [She93], and statistical physics [FT21, Yai19] have found applications within the study of neural networks and deep learning. In this vein, we will also describe an approach to understanding the training of GANs using the perspective of dynamical systems.

Generally, machine learning refers to the class of algorithms which are able to improve their own performance through exposure to data [Bis96, HTF01]. Such a broad definition can apply to approaches as simple as linear regression, or as complex as deep learning. We therefore provide a brief overview of the topics in ML which are here relevant.

Supervised Learning

In supervised learning, a function f_W with parameters W is trained to produce a desired output, given a corresponding input.

$$f_W : \mathbb{R}^N \to \mathbb{R}^n$$

Training is done by constructing a loss function, $\mathcal{L}$, (which defines the error of the model) based on a set of training data, $\{(x_i, y_i)\}$, comprising inputs, x_i, and corresponding labels y_i.

When inputs (in $\mathbb{R}^N$) are mapped to continuous labels (in $\mathbb{R}^n$), this procedure is known as *regression.* Here, such a loss function might be given by the mean squared error (MSE):

$$\mathcal{L} = \text{MSE} = \frac{1}{D}\sum_i^D (y_i - f_W(x_i))^2$$

Where D is the size of the dataset. During training, parameters, W, are updated to minimize this loss. Once the minimizing parameters, W', are found, we may then use the trained function $f_{W'}$ as a predictor.

In the case of discrete categorical labels, this problem is called *classification.* In classification, the training labels take the form of integers indexing the correct category of their corresponding input: $x_i \in \mathbb{R}^N$, $y_i \in \{1, 2, \ ..., n\}$. Functions used for classification have n-dimensional outputs, where n corresponds to the number of categories. We would like to interpret the output of $f_W(x_i)_j$ to obtain a probability for a data point, $x_i \in R^N$, belonging to category j. To do this, we normalize f_W in such a way that its outputs are mapped to positive numbers which sum to 1. Typically this is done using the softmax function:

$$\text{softmax}(\vec{l})_i = \frac{\exp(l_i)}{\sum_j \exp(l_j)} = \frac{\exp(l_i)}{Z}$$

which we write above using Z, noting the similarity to probabilities within a canonical ensemble. Indeed, the form of this ratio can be justified in both machine learning and physics through maximum entropy arguments (in fact, the concept of a 'partition function' is used throughout the field of ML [GBC16, HTF01]). The inputs to softmax, $l_i \in \mathbb{R}$, are referred to as *logits*.

We may then interpret the output of our function as follows:

$$P(x_i \in \text{Category}_j) = \text{softmax}(f_W(x_i))_j$$

Using a training data set, we find optimal parameters, W', such that $\text{softmax}(f_{W'}(x_i))_{y_i}$ is maximized, while for $j \neq y_i$, $\text{softmax}(f_{W'}(x_i))_j$ is minimized. This is achieved through the minimization of some loss function which characterizes the error of the prediction. The cross-entropy loss function is often used for this purpose:

$$\mathcal{L} = -\frac{1}{D}\sum_i^D \log(\text{softmax}(f_W(x_i))_{y_i})$$

If training is successful, then $\text{softmax} \circ f_{W'}$ can be used as a function which probabilistically categorizes our data into n sets.

Unsupervised Learning

In unsupervised learning, the training data is unlabeled, instead of (data, label) pairs, $\{(x_i, y_i)\}$, we use a data-set $\{x_i\}$, and extremize some objective in order to identify structure within the data.

An extremely simple example of such a problem that of identifying the mean of a dataset, $\{x_i\}$. We define a function: $\mathcal{L}(y) = \frac{1}{D}\sum_i (y - x_i)^2$. When this is extremized, we reveal the structure of the data, and find that y takes on the value of the mean of $\{x_i\}$.

In physics, the variational method can be interpreted as a form of unsupervised learning, as it applies this same approach of extremizing an objective to extract features [SN17]. A variational ansatz (trial wavefunction), $|\psi_\theta\rangle$ is introduced with parameters θ. The energy of this state, $E(\theta) = \langle\psi_\theta|H|\psi_\theta\rangle$ is then minimized. The minimum energy found, $E(\theta')$, represents an upper bound of the ground state energy, while $|\psi_{\theta'}\rangle$ can often be used to approximate the ground state.

Another common application of unsupervised learning is clustering, in which data is separated into groups based on some measure of similarity. Different clustering algorithms exist, with a wide variety of distinct approaches.

With such an open criteria, unsupervised learning comprises an extremely broad range of algorithms.

Neural Networks

Neural networks (also referred to as artificial neural networks) are a class of functions modeled after the structure of biological neurons and their connections within brains.

Roughly, biological neurons function by combining impulses from other neurons. If the combined signal passes a certain threshold, the neuron fires, sending signals to other neurons. This can be abstracted using the following functional form:

$$z = \phi\left(\sum_j w_j x_j + b\right)$$

Where $\phi : \mathbb{R} \to \mathbb{R}$, referred to as an *activation function*, is some nonlinearity which gives the activation of the neuron as a function of inputs. Typically, an activation function is some monotonically increasing function[1] with common choices including[2] ReLU: $\max(0, x)$, sigmoid: $1/(1+e^{-x})$, and hyperbolic tangent [GBC16].

The parameters, w_j, scale the signals of the inputs, and are learned during training. The parameter b represents an overall bias, and is also learned during training. To describe a layer of neurons, with each neuron having an index j, we may write

$$z_i = \phi\left(\sum_j w_{i,j} x_j + b_i\right)$$

In a *feed-forward* neural network, a layer of neurons is used as the input to a subsequent layer. This can be repeated to add *depth* to the neural network. The output of such a fully-connected neural network with parameters $\theta = \{w^l_{i,j},\ b^l_i\}$ and depth d, $f_\theta(x)$, can then be expressed

$$z_i^{(1)} = \phi(w_{i,j}^{(1)} x_j + b_i^{(1)})$$

$$z_i^{(l+1)} = \phi(w_{i,j}^{(l+1)} z_j^{(l)} + b_i^{(l+1)})$$

$$f^i_\theta(x) = z_i^{(d)}$$

The practice of using many layers is known as *deep learning* [LBH15, GBC16].

Different architectures for neural networks exist, but all share the common feature of a nonlinearity applied to a linear combination of inputs, typically done in an iterative way.

Stochastic Gradient Descent

Neural networks (as well as other parametrized models), f_θ, with parameters θ, are trained to minimize a *loss function*, $\mathcal{L}(\theta)$, which characterizes the cost associated with the model. For instance, $\mathcal{L}$ might correspond to the mean squared prediction error of a neural network

[1]Note that non-monotonic activation functions have been used successfully in practice.

[2]ReLU is short for Rectified Linear Unit

on a data-set [HTF01]:

$$\mathcal{L}(\theta) = \frac{1}{D}\sum_{i}^{D}(y_i - f_\theta(x_i))^2$$

During training, we wish to find parameters which minimize the value of this loss function. This could be done through gradient descent, iteratively updating the parameters according to

$$\theta_{t+1} = \theta_t - \eta\frac{d\mathcal{L}(\theta)}{d\theta}|_{\theta=\theta_t}$$

using some step size η.

In practice, however, computing the error over an entire data-set can be computationally infeasible. An alternate approach is to, at each step of training, sample a smaller *mini-batch* of data of size d. The resulting gradient of this loss is now a function of both the parameters, θ, as well as the set of specific data-points selected, which are effectively sampled randomly. Therefore the gradients obtained are stochastic, and the method is known as *stochastic gradient descent.*

Stochastic gradient descent (SGD) has been referred to as the "workhorse of machine learning" [RYH22]. Aside from making training computationally easier, SGD has in fact been shown to actually improve the quality of the loss minima discovered by neural networks compared to those found through gradient descent, aiding in generalization [FT21]. Various modifications of simple SGD exist. Some of which incorporate momentum, some which modify the training rate over the course of training [GBC16]. In general, however, these work through the same basic mechanism.

CHAPTER 1

Unsupervised Learning for Probing Many-Body Localized Phases

This chapter is adapted from the publication:

Durr, S., Chakravarty, S. *Unsupervised learning eigenstate phases of matter.*, Phys. Rev. B. **100**(7):075102, Aug 2019

1.1 Introduction

Thermalization within a closed quantum system can be expressed in terms of the long time expectation of a local observable. In thermalizing systems – even those which are isolated – these expectation values are observed to reach their corresponding thermal value. This phenomenon prompted an explanation in terms of the Hamiltonian eigenstates[Deu91, Sre94], referred to as the eigenstate thermalization hypothesis (ETH). The ETH states that local observables of eigenstates will have an expectation which matches the microcanonical ensemble at that energy:

$$\langle E|\mathcal{O}|E\rangle = \frac{1}{\mathcal{N}(E,\Delta E)} \sum_{|E-E_\alpha|<\Delta E} \mathcal{O}_{\alpha,\alpha}$$

This rather profound statement implies that thermalization occurs within a single eigenstate.

When the ETH is true, local observables are guaranteed to reach their thermal values. Eigenstates obeying the ETH are highly distributed. Under time evolution, sums of these thermal eigenstates dephase, and local information quickly becomes inaccessible as it is distributed throughout the system [ODR08]. The full system is able to act as a bath to its subsystems.

The ETH is believed to apply to all non-integrable systems, with the only robust exception being those which display many body localization (MBL). Many-body localization (MBL) is the generalization of Anderson localization to interacting systems [BAA06, AAB19]. Given sufficiently strong disorder, transport drops to zero, and the system fails to thermalize. Such a system will preserve the initial conditions even after infinite time has passed. With the enticing possibility of preserving quantum information, and encouraged by the development of experimental advancements [MIQ22, AC17], MBL has become a prominent area of research.

The area-law entanglement and localization properties of MBL systems has motivated a phenomenological picture of the phase. Using the existence of an extensive number of local conserved operators, τ_i^z (called LIOMs), the description expresses MBL Hamiltonians in the form [CKV15, AAB19, KCA14]

$$H = \sum_i h_i \tau_i^z + \sum_{i,j} J_{i,j} \tau_i^z \tau_j^z + \sum_{i,j,k} J_{i,j,k} \tau_i^z \tau_j^z \tau_j^z + \cdots$$

where the interactions decay exponentially with distance. Here, the operators, τ_i^z, can be mapped to a Pauli through a sequence of local unitary rotations: $\tau_i^z = U \sigma^z U^\dagger$ [CKV15, Imb16]. This perspective has inspired the use of approximate methods to study properties of MBL [YQX16, SBY17].

One such method which gives considerable insight is the Spectrum Bifurcation Renormalization Group (SBRG) [YQX16]. Using SBRG, it is possible to approximately map a strongly disordered Hamiltonian in the MBL phase to its LIOM representation. An example of such a transformation is shown in Figure 1.1

Although conceptually useful, methods which rely on the properties of MBL may fail near the thermal phase, when resonances arise.

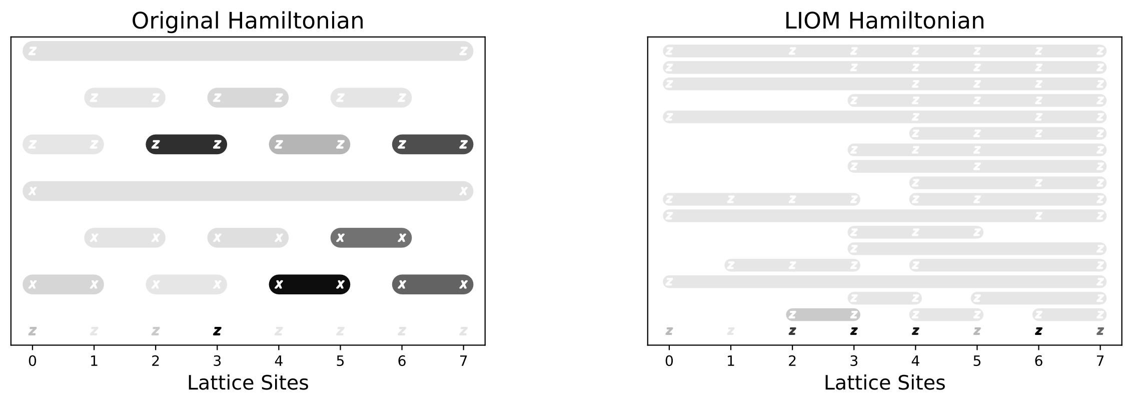


Figure 1.1: Above we depict a strongly disordered Hamiltonian (described in more detailed in the first chapter's appendix (1.6.1.2)). To the left is shown the original Hamiltonian, with darkness indicating the strength of the terms. On the right is shown the version of this Hamiltonian expressed using LIOMs. Note that the weakest (faintest) terms in the LIOM picture are those which are the longest-ranged. This mapping was obtained through the SBRG algorithm [YQX16], which provides a clear connection to the phenomenological MBL Hamiltonian from a strongly disordered starting point.

Another useful vein of research applies exact diagonalization to Hamiltonians in order to understand the features of MBL systems and probe the relevant phase transitions [OH07, PH10a]. In this chapter, with the additional use of tools from machine learning, we will use this approach to study eigenstate phases.

1.2 Unsupervised Learning for Identifying Phases

Recently, machine learning has been applied to the task of identifying phases of matter – particularly in scenarios in which local order parameters are not available [VKK18, Jö18, CM17, SRN17, Wet17, AMH19, RS18, CVN19, JAM18, Wan16, WS17, BAT17, BCM17, ZK17, CT17, LN18, ZMK17, OO17, NLH17, HSS17, HDW18]. Approaches have largely focused on the application of supervised learning. Using this technique, data is sampled from points in parameter space known to belong to a certain phase. Some function (often a neural network) is then trained to predict the phase given input data. If the function is able to effectively generalize, it is then possible to apply it to data coming from points across the parameter space. This allows one to produce a phase diagram, and gain insight into the underlying physics.

Techniques from unsupervised learning have also been shown to be effective at identifying phases [Wet17, RS18, JAM18, Wan16, WS17, BAT17, CVN19, HSS17, HDW18]. When applied to the 2D Ising model, for instance, tools such as autoencoders have been able to extract a local order parameter [Wan16, HSS17]. In addition, clustering techniques have been used to accurately distinguish spin data known to correspond to distinct topological sectors [RS18].

Here, we expand on work applying unsupervised learning towards identifying phases. Using readily available clustering algorithms, we explore the parameter space of a system known to display many-body localization and eigenstate phase transitions. In particular, the problem we study has been effectively treated using supervised learning [VKK18, Jö18], which, when compared to conventional techniques, was able to predict the sharpest phase boundary to date [VKK18].

We therefore use this supervised approach as a starting point from which to compare our unsupervised method, and find that we are able to produce highly similar results. Notably, our method relies on no separate training data, no prior knowledge of the phase space, and even no explicit assumption of the number of phases present.

We conclude with a discussion of what can be learned from this success, as well as the role and usefulness of machine learning algorithms for the task of identifying phases.

1.3 Clustering Many-Body Localized Phases

Generally, an isolated interacting quantum system is said to display many-body localization (MBL) if it fails to thermalize under its own unitary time evolution. On the other hand, a quantum system is said to be thermal if it is able to serve as its own heat bath. In addition, many-body localized phases exhibit area law entanglement entropy scaling, while in the thermal phase entanglement entropy is extensive and obeys a volume law. Different

MBL phases exist, displaying different symmetries and topological order. These phases are difficult to realize, as they require isolation from a thermal environment. However, MBL phases have been produced in experiment in, for instance, the context of one-dimensional strings of ultracold atoms [SHB15]. The transition of a state among thermal and MBL phases represents a dynamic eigenstate phase transition – for which an extensive theoretical description does not currently exist.

Here we use clustering algorithms to analyze eigenstate phase transitions within the transverse-field Ising model in the presence of interactions and disorder:

$$H = -\sum_{i=1}^{L} \left(J_i \sigma_i^z \sigma_{i+1}^z + h_i \sigma_i^x + \lambda(\bar{h} \sigma_i^x \sigma_{i+1}^x + \bar{J} \sigma_i^z \sigma_{i+2}^z) \right)$$

Above, σ_i^α are the Pauli matrices and $\{J_i\}$ and $\{h_i\}$ are log-normal distributions with respective means $\bar{J}$ and $\bar{h}$, and the standard deviations of their logarithms equal to 1. We use a length 12 spin chain with open ends. Note that our Hamiltonian has a global $\mathbb{Z}_2$ symmetry given by $P = \prod_{i=1}^{L} \sigma_i^x$.

We use this model largely because its limits have been previously studied, and are known to exhibit distinct eigenstate phases [HNO13, PRA14a, KBP14, Fis95a]. In particular, for $\overline{h} \gg \overline{L}, \lambda$, the eigenstates are expected to asymptotically correspond to product states in the σ_x basis (e.g. $| \leftarrow\leftarrow\rightarrow\leftarrow \cdots \rangle$). In this limit, the system is said to exhibit a many-body localized paramagnetic phase (MBL PM). In the opposite limit of $\overline{J} \gg \overline{h}, \lambda$, the states resemble global superpositions of spins in the σ_z basis with frozen domain walls. These are so-called Schrodinger 'cat' states, and are of the form $\frac{1}{\sqrt{2}}(| \uparrow\downarrow\uparrow\uparrow \cdots \rangle \pm | \downarrow\uparrow\downarrow\downarrow \cdots \rangle)$. This is the many-body localized spin-glass phase (MBL SG). Finally, in the limit of $\lambda \gg \overline{J} = \overline{h}$ the system is expected to be in a thermal phase, with the entanglement entropy of the states showing volume law scaling, rather than area law scaling.

In addition, the model is self dual about $\overline{\mathrm{Log}(J)} = \overline{\mathrm{Log}(h)}$. Therefore, up to the effects of finite size, we expect to observe symmetry about $\overline{\mathrm{Log}(J)} - \overline{\mathrm{Log}(h)} = 0$ in the phase diagram.

A further reason we use this model is that its phases have been previously analyzed using supervised learning [VKK18, Jö18]. In [VKK18], the authors use a neural network to extract eigenstate phases given the entanglement spectrum. They compare their results to those of the standard approach, in which the standard deviation of the entanglement entropy is calculated across parameter space. The authors find that the supervised method is able to identify phase boundaries with far greater precision than the conventional alternative.

Because supervised learning has been shown to provide an unmatched degree of clarity, this highest-precision approach is our natural point of reference. Therefore, to evaluate the success of our unsupervised analysis, we will compare our results to those found through the use of supervised learning following the procedure of [VKK18].

1.3.1 Producing Data

We vary two parameters: λ and $\Delta_{Jh} := \overline{\text{Log}(J)} - \overline{\text{Log}(h)}$, and obtain data for a grid of points in parameter space where $\lambda \in [0, 2]$ and $\Delta_{Jh} \in [-3, 3]$ [VKK18]. At each point, we obtain the Hamiltonian and find its eigenvectors using exact diagonalization [1]. For each eigenvector, ψ, we can calculate the reduced density matrix as follows:

We consider the system to be split into two parts: region A containing the middle 4 spins, and region B containing the outer 8. We then trace over the states of B to obtain the reduced density matrix:

$$\rho_A = Tr_{\mathcal{H} \setminus \mathcal{H_A}}\big(|\psi\rangle\langle\psi|\big)$$

To calculate this in practice, we used Schmidt decomposition.

The $-$Log of the eigenvalues of ρ_A would give us the entanglement spectrum – known to carry information concerning many body localization [GNR16, PH10a, BN13]. The eigenvalues of ρ_A themselves are probabilities, giving us a vector in $2^4 = 16$ dimensions with elements summing to 1.

There exist established distance metrics motivated by information theory for expressing

[1]We exclude the highest and lowest 10% of the eigenvectors to reduce potential deviation from the trend of a given phase

the similarity of two probability distributions. For this reason, combined with the additional benefit of having the data live in a compact region, we use the probability vectors corresponding to the eigenvalues of the reduced density matrix as our data for clustering. We take our distance metric between two probability vectors to be the Jensen-Shannon distance [ES03], which is a bounded metric expressing similarity between probability distributions.

Therefore, for each disorder realization and each point in parameter space, we calculate an array of lists containing reduced density matrix eigenvalues (one list of eigenvalues for each eigenvector). Here we generate 100 disorder realizations and evaluate Hamiltonians at 1200 points in parameter space.

1.3.2 Clustering Data

We collect $N = 1000$ samples of our data, each with $n = 5$ elements taken at each of the 1200 positions in parameter space (= 6000 elements per sampling).

As a first step, we apply the HDBSCAN clustering algorithm [MHA17] (Hierarchical Density-Based Spatial Clustering of Applications with Noise) to run an exploratory analysis and identify structure within the data set. HDBSCAN is ideal for exploratory clustering due to its lack of hard assumptions about the data. In particular, it does not assume clusters to be convex, nor does it assume a set number of clusters to search for. A more complete discussion of HDBSCAN can be found in the appendix.

We next set HDBSCAN's two main parameters. We set the minimum cluster size (`min_cluster_size`) by using a rough prior concerning the size of the smallest cluster we expect to see. This clearly varies based on application. Here we specify that we are interested in finding clusters which comprise at least 1/10 of the total data set. This gives us a minimum cluster size of 6000/10 = 600 elements.

A suitable value for the `min_samples` parameter can be set by evaluating the density-based validation score [MJC] across a range. The value of `min_samples`, however, is observed to have little effect on the clustering. By identifying cluster labels with their phase space positions, we may then obtain a corresponding phase diagram (Fig. 1.2).

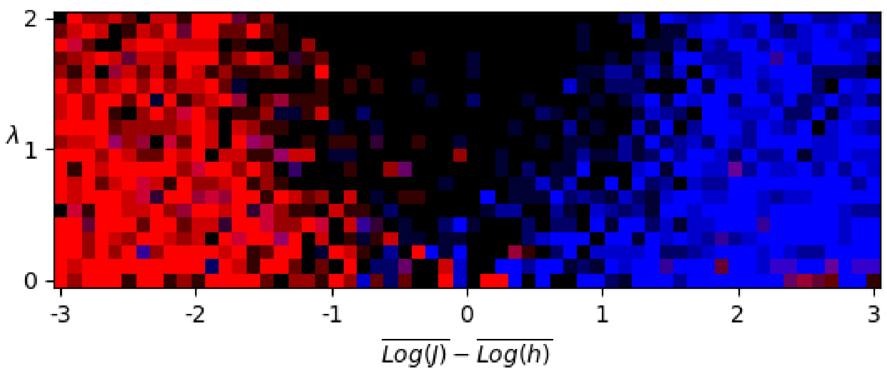


Figure 1.2: A phase diagram found by applying the HDBSCAN clustering algorithm to a single sampling of data from across parameter space. Two clusters are identified, one to the left (red) and one to the right (blue). A darker region in the center corresponds to data classified as low-density noise.

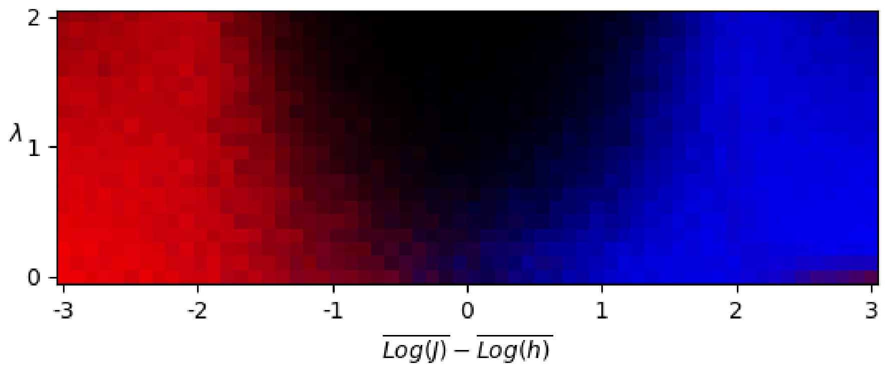


Figure 1.3: A phase diagram obtained by averaging 1000 samples of data from across parameter space, each clustered using HDBSCAN clustering. Clusters from each sample of data were identified with one another using their relative Δ_{Jh} position – the cluster with lower mean value was colored red, and the cluster with the higher value colored blue. Averaging over all diagrams gives us the above result.

By examination, we can see that there are two observed clusters which may be ordered by their average Δ_{Jh} value. We then apply HDBSCAN to all 1000 samples, and note that in over 98% of samples, two such clusters were formed – the remaining $< 2\%$ forming only a single cluster. We discard these, identify clusters by their weighted Δ_{Jh} order, and average over samples to obtain a phase diagram (Fig. 1.3).

By inspection, we can identify three regions of the phase diagram: two to the left and right identified as clusters, and a third in the center considered noise.

HDBSCAN does not attempt to allocate every element into a cluster. Rather, it assumes that data sets may contain sparse noise not belonging to any particular cluster. Here, we can see that the data HDBSCAN considers noise may instead form a third, more sparse cluster.

To investigate this, we apply an algorithm known as spectral clustering [PVG11] (discussed in more detail in the appendix). While still not assuming convexity of clusters, spectral clustering allows us to include assumptions about the number of clusters to form within the data. We repeat the above procedure, taking the number of clusters to form to be 3, and again using Jensen-Shannon distance. Averaging over 1000 samples, we obtain a total phase diagram (Fig. 1.4).

1.4 Analysis of Results

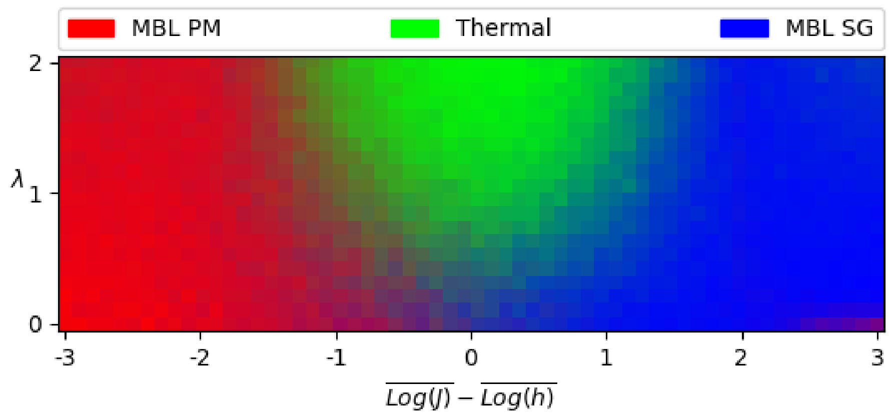


Figure 1.4: **Unsupervised Phase Diagram:** The phase diagram found by applying spectral clustering to data from across parameter space using a total of 100 disorder realizations, and clustering 1000 subsets of data sampled from these realizations. During clustering, we use reduced density matrix eigenvalues as data, and the Jensen-Shannon distance as our metric.

We see that the sparse data considered noise by HDBSCAN was successfully identified as a third cluster. Physically, we do expect three phases to be present: two clusters corresponding to a many-body localized paramagnetic phase and a many-body localized spin-glass phase (referred to as MBL PM and MBL SG, respectively), and one cluster corresponding to a thermal phase which conforms to the eigenstate thermalization hypothesis.

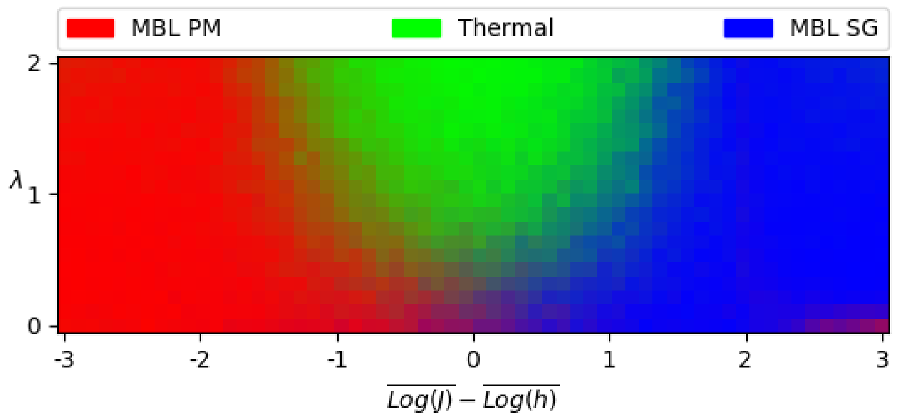


Figure 1.5: **Supervised Phase Diagram:** The phase diagram found by applying a trained neural network to the entanglement spectra from across parameter space using 100 disorder realizations.

Up to the effects of finite system size, we expect our phase diagram to be symmetric under $\Delta_{Jh} \to -\Delta_{Jh}$, and for the phases to exist in the general regions identified within the phase diagram. Therefore, we appear to have formed clusters corresponding to each of these predicted phases.

1.4.1 Comparison to Supervised Learning

Following the procedure outlined in [VKK18] and using the framework of TensorFlow [AAB15], we produce a phase diagram for the system by applying a trained neural network to our data. We can then compare this to the phase diagram found through unsupervised learning.

We train a neural network on a set of simulated data emanating from three points in Δ_{Jh}, λ space:

$$(0.8, 0.2), (-0.8, 0.2), (0.0, 1.0)$$

These correspond to the MBL SG, MBL PM, and Thermal phases, respectively. After training, we apply the neural network to data from across the parameter space and use the resulting phase predictions to produce a phase diagram (Fig. 1.5).

To measure uncertainty in a phase assignment, in [VKK18] the authors define a quantity $\mathcal{C}$. If each point on the final phase diagram has a corresponding probability vector, $\vec{p} =$

(p_1, p_2, p_3), then $\mathcal{C}$ is defined as $\mathcal{C} = 1 - \overline{\mathrm{d}}_{\mathrm{min}}$. Here $\mathrm{d}_{\mathrm{min}} = \min|\vec{p} - \vec{v}| : \vec{v} \in Q$, and Q is the set of points of extremal phase uncertainty: $(1/2, 1/2, 0)$, $(1/2, 0, 1/2)$, $(0, 1/2, 1/2)$, and $(1/3, 1/3, 1/3)$. $\overline{\mathrm{d}}_{\mathrm{min}}$ is then the value of $\mathrm{d}_{\mathrm{min}}$ normalized by its maximum possible value.

We calculate this measure for both the unsupervised (Fig. 1.6) and the supervised (Fig. 1.7) approaches, and compare the two.

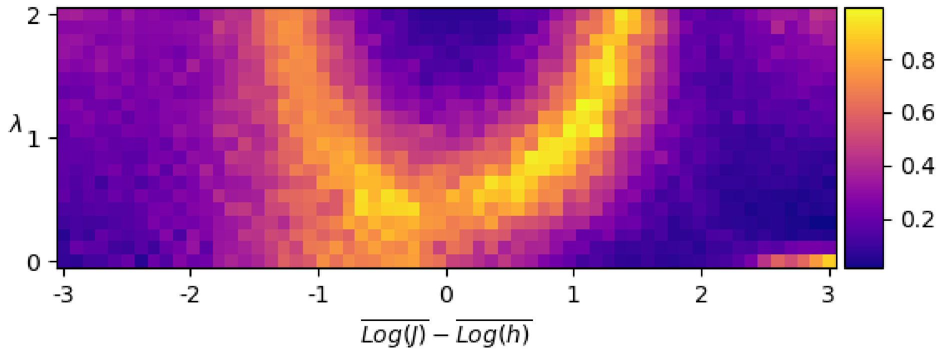


Figure 1.6: **Unsupervised $\mathcal{C}$ Diagram:** The measure of confusion, $\mathcal{C}$, plotted across parameter space obtained through spectral clustering.

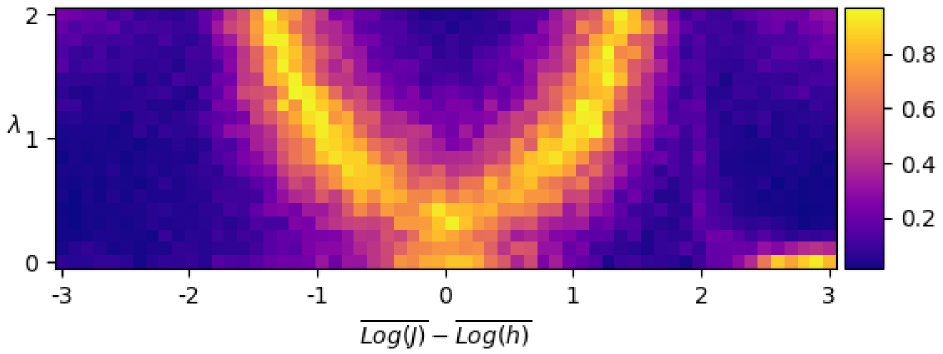


Figure 1.7: **Supervised $\mathcal{C}$ Diagram:** $\mathcal{C}$ plotted across parameter space obtained using a trained neural network.

Qualitatively, the two diagrams are similar. We examine this agreement by taking slices of $\mathcal{C}$ at constant λ. Again, we find agreement at both $\lambda = 1$ and $\lambda = 2$ (Fig. 1.8). Note, however, that the supervised method produces consistently lower values of $\mathcal{C}$ away from phase transitions. We can plot slices of the phases detected at $\lambda = 1$ and $\lambda = 2$ for the supervised (Fig. 1.11) and unsupervised (Fig. 1.10) approaches to again see that these predictions largely match. Slicing along $\Delta_{Jh} = 0$ (Fig. 1.9) also shows similarity between predictions, however with slightly more symmetric predictions resulting from the supervised approach.

We also observe agreement at $\lambda = 0$, where both methods observe the same asymmetry at $\Delta_{Jh} = 3$, present potentially due to finite-size effects (Fig. 1.12).

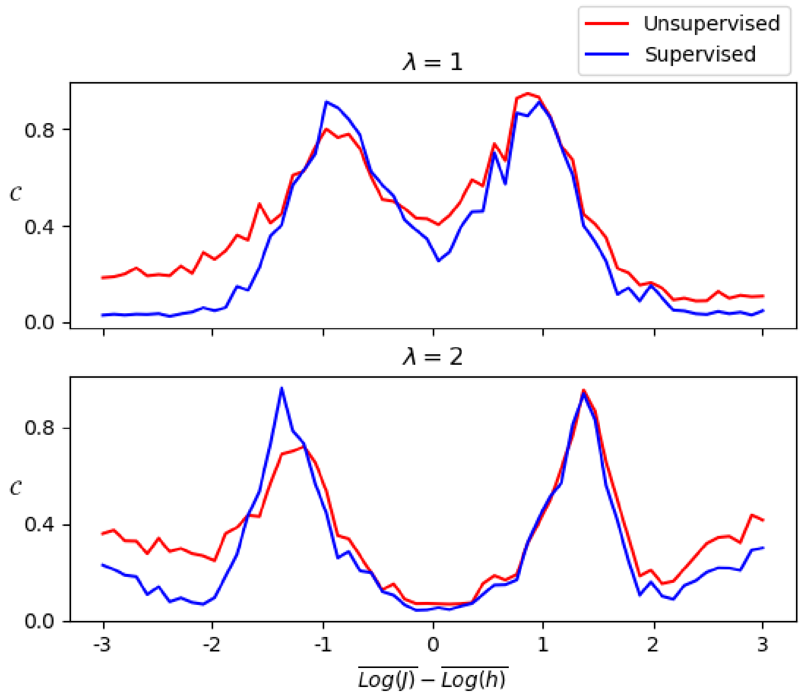


Figure 1.8: Slices of the $\mathcal{C}$ diagrams taken at $\lambda = 1$ and $\lambda = 2$ showing both the supervised and unsupervised results.

1.5 Discussion

Here, we have outlined an unsupervised method to study the phase space of a system demonstrating MBL transitions. Our method applies readily available clustering algorithms to segment phase space into three regions. Specifically, we apply HDBSCAN and spectral clustering to the eigenvalues of reduced density matrices, using Jensen-Shannon distance as a metric for clustering.

When compared to the corresponding result obtained through supervised learning, we find remarkable agreement between the phase boundaries that both methods predict. Both techniques are able to produce these meaningful results using small data sets. Here we

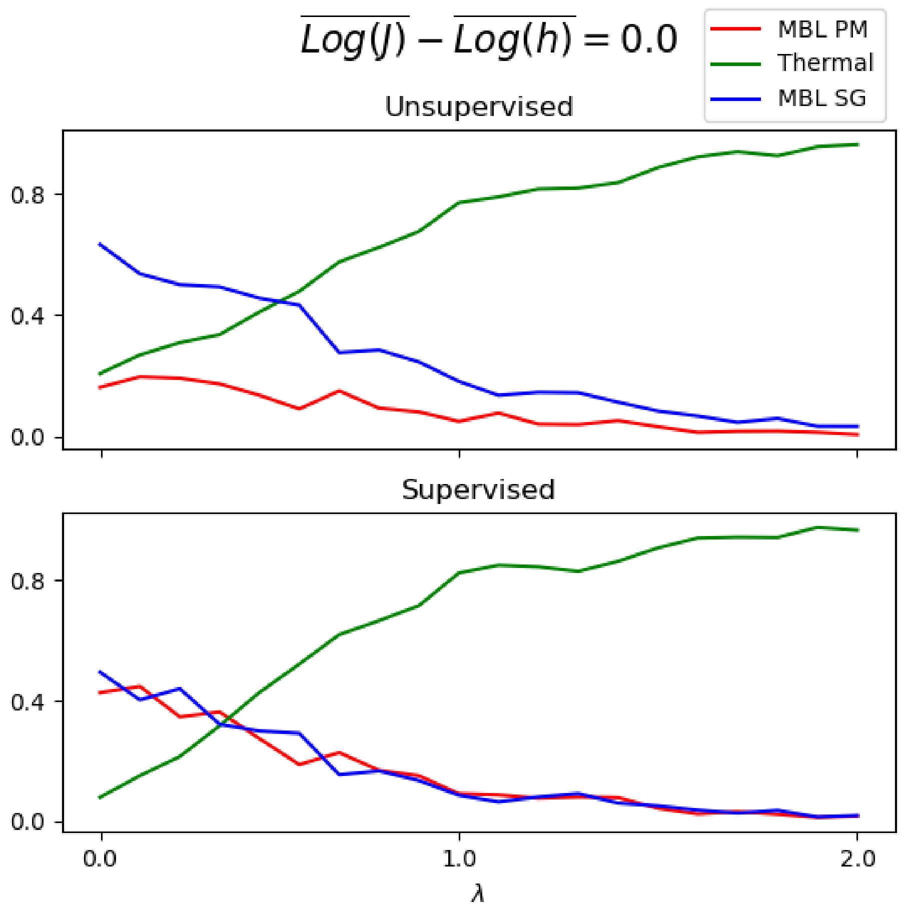


Figure 1.9: Slices of the phase diagrams found using both unsupervised (top) and supervised (bottom) methods, and taken along $\overline{\text{Log}(J)} - \overline{\text{Log}(h)} = 0$. The y-axis indicates the probability of classification into a given phase. Note that both methods predict the similar presence of a thermal phase as λ increases,

relied on a set of only 100 disorder realizations. Our method, however, requires no strict assumptions about the number of clusters present, no labeled training data, and no prior knowledge of the phase diagram.

In the case of supervised learning applied to eigenstate phases, it is not readily apparent which features are being extracted from the data that would indicate the presence of a particular phase. Therefore, this is a black box method – an issue present in many applications of machine learning. In our use of clustering, the same problem exists.

Currently, however, our analysis requires use of this unsupervised clustering step. This is due to two main reasons: First, manual separation of data into 'obvious' clusters would

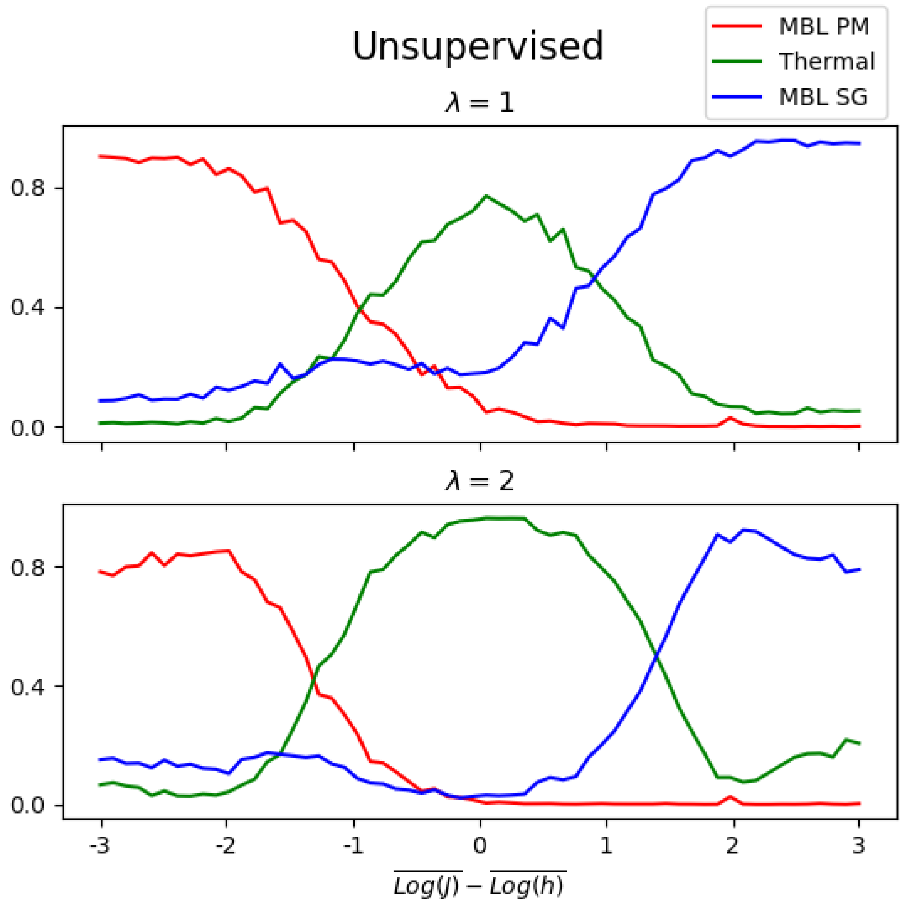


Figure 1.10: Slices of the unsupervised phase diagram taken at $\lambda = 1$ (top) and $\lambda = 2$ (bottom) showing the probability of classification into one of three phases.

require projection into a low dimensional space (two dimensions, for instance). Simple projection using, for example, principal component analysis, would throw away too much information to produce an accurate phase diagram. Second, in problems such as this, we often choose to use a metric other than euclidean distance to express the difference between data points. Therefore, if we did wish to separate data manually, finding an embedding in a low-dimensional flat space which approximately preserved our desired metric would be another machine learning problem of its own.

When applying a clustering algorithm, data is segmented into groups according to each algorithm's implicit conception of what a cluster comprises. Moreover, there does not exist – and cannot exist [Kle03] – a satisfying universal axiomatic approach to define the goals of clustering. Rather, trade-offs between different clustering criteria are intrinsic. These

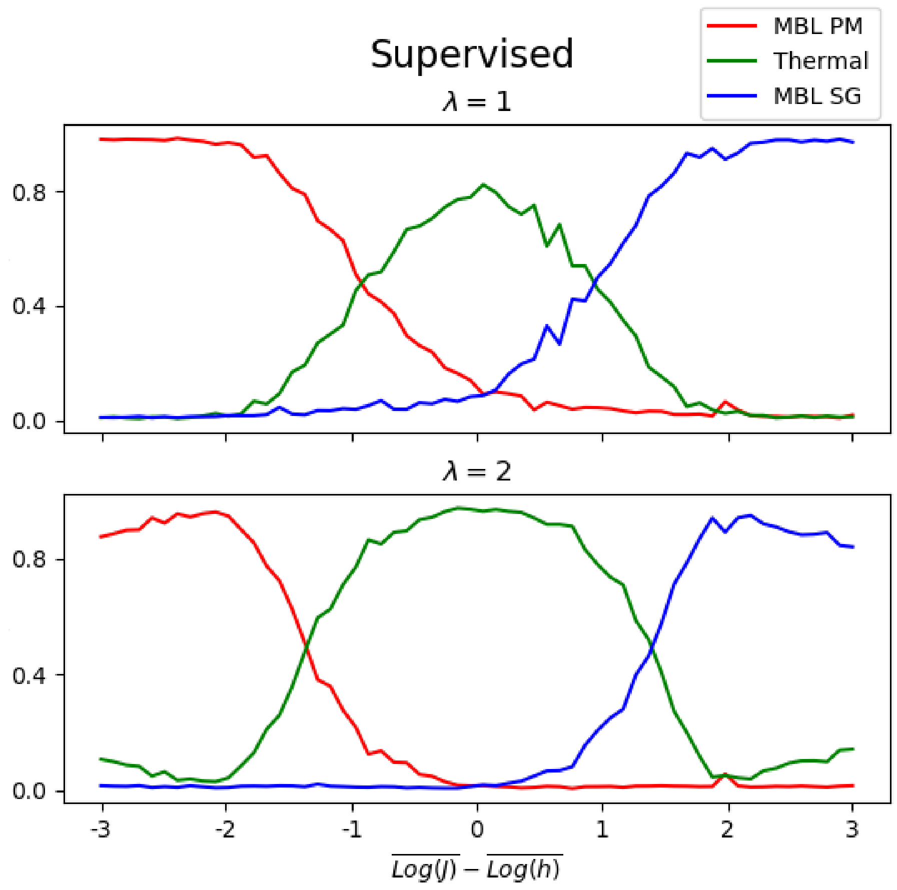


Figure 1.11: Slices of the supervised phase diagram taken at $\lambda = 1$ (top) and $\lambda = 2$ (bottom) showing the predicted probabilities of each phase.

trade-offs can be seen in practice. Clustering algorithms often fail to perform when the ideal clusters are of very different sizes, different densities, and different shapes. Algorithms which perform well in one of these situations may fail in another.

Choosing a function to express the similarity between two elements (i.e. a distance function) also often relies on heuristics. Distance functions can be chosen based on the nature of the data at hand and the goal of the clustering. Other tools from unsupervised learning can be effective here. Autoencoders, for instance, map elements from an original space to a latent space. Spatial separation of two data points in the latent space then corresponds to some meaningful difference between the two initial elements. A set of words, for example, can be mapped to vectors in a latent space. Spatial similarity between vectors in this latent space (e.g. cosine similarity) corresponds to similarity in meanings of the words.

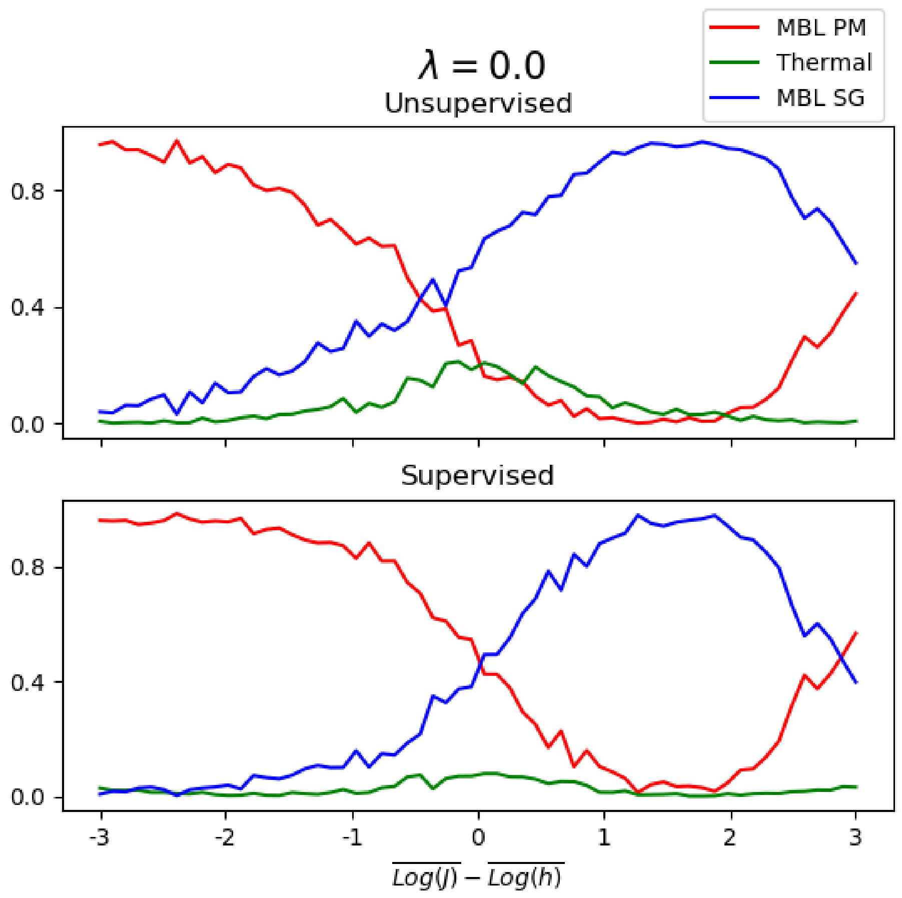


Figure 1.12: Slices of the unsupervised (top) and supervised (bottom) phase diagrams along $\lambda = 0$. Note the asymmetry near $\overline{\text{Log}(J)} - \overline{\text{Log}(h)} = 3$ is found in both procedures.

Forming clusters in this latent space can then yield meaningful groupings.

From the perspective of clustering, there does not necessarily exist an a priori 'correct' partitioning of the data. The optimal clustering of data is instead dependent on the application. HDBSCAN uses a quantified heuristic expressing hierarchical depth within the data to determine the number of clusters to form. Other methods to determine the optimal number of clusters are also available, but all rely on optimizing some conception (either stated explicitly or implied) of what a cluster should be.

With these concerns in mind, when using a clustering algorithm whose optimization criteria is without a direct mapping to physics, one cannot be immediately sure that the resulting partitioning of physical data must usefully correspond to distinct physical categories. Nonetheless, we have demonstrated that the clustering procedure applied here has yielded useful results. Our goal, however, is not to show that clustering techniques accurately and reliably extract MBL phase boundaries. Rather, it is to show that meaningful relationships within physical data can be quickly and cheaply explored by using reasonably applied clustering techniques.

1.6 Appendix

1.6.1 Spectrum Bifurcation Renormalization Group

Many probes of MBL rely on analyzing the eigenvalues and eigenstates. To this end, many studies have used exact diagonalization ([LLA15], [ScA15], [PH10b], [OH07]). However, due to the exponential dimension of the Hilbert space, this approach–incorporating no assumptions about the states under analysis–is limited to small lattices of at most ~ 22 sites.

Due to the properties of many-body localization, MBL eigenstates are not arbitrary. In many ways (e.g. area law entanglement) they are similar to ground states, and for this reason can be described using techniques which rely on low-entanglement, such as tensor network approaches.

The Spectrum Bifurcation Renormalization Group (SBRG) [YQX16] is a technique based on real-space strong-disorder renormalization ([DM80], [BL82], [Fis92], [Fis94], [Fis95b], [PRA14b]), which diagonalizes the Hamiltonian iteratively. SBRG starts at the leading energy term, applies a unitary Clifford rotation[2] to diagonalize this term, Schrieffer-Wolff rotates away off-diagonal terms, and continues on to the next leading-term. The spectrum is therefore bifurcated at each step, hence the name 'Spectrum Bifurcation'. The unitaries are then extracted to construct a Clifford circuit[3] which approximately diagonalizes the Hamiltonian. We are therefore able to produce a tensor network which can be used to efficiently [Got98] express approximate eigenstates of the Hamiltonian, and compute important quantities of the system.

SBRG is uniquely poised to study strongly disordered MBL systems, and in fact naturally reveals the emergent conserved quantities–interpreted as LIOM's [KCA14] in the context of MBL. Using this perspective, SBRG also allows us to directly interpret the obtained Clifford

[2]A Clifford rotation is an element of the Clifford group – the subgroup of unitaries which maps individual elements of the Pauli group to other individual elements of the Pauli group.

[3]A Clifford circuit is a set of sequential Clifford rotations. Since each Clifford rotation maps one Pauli-group element to another Pauli-group element, the complexity of an entire Clifford circuit is sufficiently low to simulate its effect on a product state using a classical computer.

circuit to investigate the localization properties of MBL systems. For completeness, we describe the SBRG routine below (a full treatment can be found within [YQX16]).

1.6.1.1 SBRG Algorithm

SBRG operates on Hamiltonians of the form

$$H = \sum_{\mu} h_\mu \sigma^\mu$$

Here, H acts on N spin sites, μ indicates an vector of N integers ($\mu_i \in \{0, 1, 2, 3\}$). For instance, $\sigma^{[3,0,0]} = \sigma_3 \otimes 1 \otimes 1$. In this way, we can express spin-1/2 Hamiltonians generally.

As mentioned, SBRG works by iteratively diagonalizing the Hamiltonian, starting at the highest magnitude term and proceeding on. The i^{th} step of the procedure proceeds as follows:

1. First, identify the leading order term which has not yet been diagonalized using the algorithm. Take this term to be equal to

   $$H_0 = h_0 \sigma^\mu$$

   If H_0 satisfies that $\mu_i = 3$, and for $j \neq i$, $\mu_j = 0$ (that is $\sigma^\mu = 1_i \otimes \ldots 1_{i-1} \otimes \sigma_i^3 \otimes 1_{i+1} \otimes \ldots \otimes 1_N$), move on to the next step.

   Otherwise, apply a Clifford rotation to H_0 in order to put it into this form. Note that Clifford rotations map products of Pauli operators to products of Pauli operators (rather than sums of Paulis). That is, a Clifford rotation R enacts

   $$R^\dagger \left(\bigotimes_i \sigma_i^{\mu_i} \right) R = \bigotimes_i \sigma_i^{\nu_i}$$

   In this way, Clifford rotations do not grow the size of the Hamiltonian.

2. Next, we enact a Schrieffer-Wolff rotation [SW66] at site i in order to rotate away off-diagonals.

For clarity, take $i = 0$ below. Then we write

$$H_0 = h_0 \sigma^{(3,0,0,\ldots)}$$

All other elements either commute or anticommute with this term. Take Σ to be those terms which anticommute (having σ_1 or σ_2 at position i), and Δ to be those terms which do commute (having σ_0 or σ_3 at position i). We then wish to construct a rotation which rotates away Σ below:

$$H = H_0 + \Sigma + \Delta$$

Writing

$$H' = e^S (H_0 + \Sigma + \Delta) e^{-S}$$

We may expand this in orders of Σ/H_0, where the leading order term is expected to be much greater than the typical off-diagonal terms. To second order in perturbation theory this is accomplished by

$$S = \frac{1}{2h_0^2} H_0 \Sigma$$

Applying this yields (to second order in perturbation theory)

$$e^S (H_0 + \Sigma + \Delta) e^{-S} = H_0 + \Delta + \frac{1}{2h_0^2} H_0 \Sigma^2$$

Where each of the above terms contains only commuting terms at position i, and so we have block-diagonalized successfully.

By repeating this procedure for all terms, it is possible to obtain an approximate diagonalization of the Hamiltonian. Crucially, however, this approximation is reliant on the assumption that the strongest term at each step, H_0, is much greater in magnitude than the typical off-diagonal coupling, Σ. Additionally, much of the strength of this technique comes from the ability to drop SW transformations, valid in the strong disorder limit, which we will explain below.

At the end of the procedure, we have constructed the following unitary which approximately diagonalizes the Hamiltonian:

$$U_{RG} = \prod_{i=1}^{N} S_i R_i$$

Computing quantities with this, however, is still extremely computationally expensive, as the SW transformations produce large numbers of Pauli operators at each step, leading to an exponential number of terms to work with.

By dropping these SW rotations from the procedure, we dramatically reduce the computational cost. What's more, in the strong disorder limit, with the largest N terms dominating the scale of those remaining, this approximation is asymptotically valid, and useful for approximating eigenstates to lowest order [YQX16].

We are then left with a Clifford circuit, with which we are able to compute the entanglement entropy and the localization length of the emergent conserved quantities, among other quantities.

1.6.1.2 Visualization

Using tools to visualize the Hamiltonian, we can see how the algorithm currently functions. Here we use a very small 1-d lattice size (with length 8) with strong disorder. The Hamiltonians we consider are of the form

$$H = -\sum_i J_i \sigma_i^1 \sigma_{i+1}^1 + K_i \sigma_i^3 \sigma_{i+1}^3 + h_i \sigma_i^3$$

Where coefficients are drawn from:

$$P(J)dJ = \frac{1}{\Gamma J} \left(\frac{J}{J_0}\right)^{1/\Gamma}$$

$$P(K)dK = \frac{1}{\Gamma K} \left(\frac{K}{K_0}\right)^{1/\Gamma}$$

$$P(h)dh = \frac{1}{\Gamma h}\left(\frac{h}{h_0}\right)^{1/\Gamma}$$

Note here γ controls the level of randomness, with $\gamma = 0$ corresponding to uniform distributions (far from the strong-disorder limit).

In Figure 1.1, the Hamiltonian took $J_0 = K_0 = h_0 = 1$, and $\Gamma = 5$.

After performing a full diagonalization using SBRG, we are left with a diagonalized, exponentially localized Hamiltonian (see below) in terms of the LIOMs. By comparing the eigenvalues, we can see that such a procedure produces a spectrum closely matching exact diagonalization. Unfortunately, such a method is restricted in its applicability, as it requires Hamiltonians to be strongly disordered, and deep within the MBL regime.

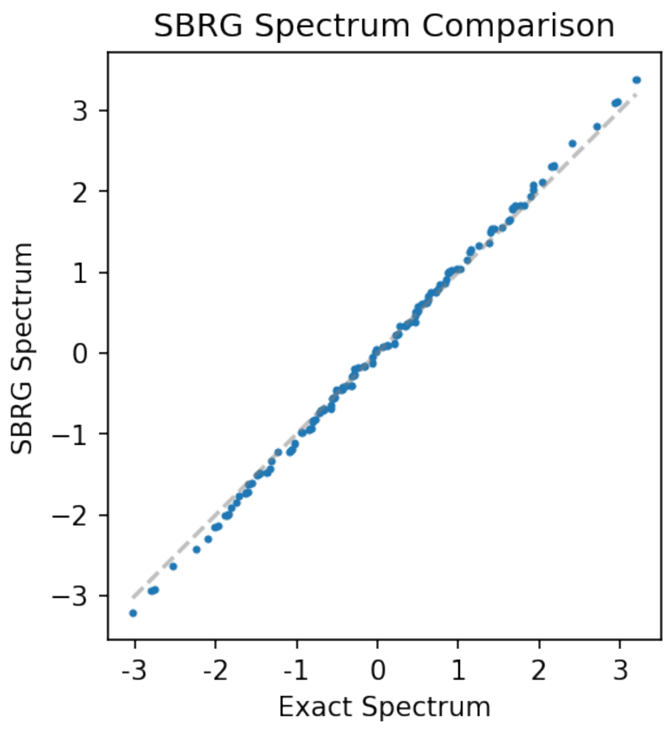


Figure 1.13: A spectrum obtained through SBRG compared to exact diagonalization. The parameters used here are $J_0 = K_0 = h_0 = 1$, and $\Gamma = 5$.

1.6.2 Clustering Algorithms

In unsupervised learning, we do not require data to be labeled. Rather, we follow a procedure to extremize some quantity in order to identify structure present in the data. Clustering is a form of unsupervised learning whose goal is to separate data into groups such that elements within a group are in some sense similar, and elements between groups are different.

Different clustering algorithms use different approaches to group data. Below, we describe some meaningful differences in approaches that clustering algorithms can take.

Parametric vs Density-Based

Parametric clustering algorithms assume (either implicitly or explicitly) that the data takes a certain form. This could be that the clusters are convex (as assumed by algorithms such as k-means) or that the probability density function (pdf) from which the data points are drawn are sums of Gaussians (as assumed by a Gaussian mixture model). Furthermore, these models generally assume knowledge of the number of clusters present in advance.

On the other hand, density-based clustering assumes that the data is generated according to a probability distribution and seeks to identify connected components of level sets of the pdf. In practice, these algorithms separate high density regions of the data from low density regions. Connected components of these high density regions are then considered clusters.

Flat vs Hierarchical

Flat clustering algorithms require us to set a parameter identifying the 'granularity' of the clusters we would like to form. For parametric algorithms, this could correspond to specifying the number of clusters to separate the data into (i.e. the resolution of clustering). For density-based clustering algorithms, this might correspond to choosing which level set of the pdf to use in clustering. Different level sets may then yield different connected components.

Hierarchical clustering algorithms avoid setting a granularity parameter. Rather, they construct a hierarchy of groupings, with similar clusters merging into one another as we decrease the resolution.

HDBSCAN

HDBSCAN is a density-based clustering algorithm which also uses tools from hierarchical clustering. It builds upon DBSCAN, a flat density-based algorithm, and can be ideal for exploratory clustering.

In exploring an unknown data set, we would like our clustering algorithm to make as few unwarranted assumptions about the data as possible. These include assumptions about the number of clusters present, as well as the shape of those clusters.

As the HDBSCAN algorithm is density-based, it does not assume that clusters must be of a specific form. In addition, instead of assuming a specific number of clusters to find, HDBSCAN uses a hierarchical analysis of the data to quantify the 'depth' of potential clusters. Its hierarchical technique allows HDBSCAN to predict which clusters to form, based on how resilient their presence is under variation of the clustering resolution. Furthermore, HDBSCAN does not attempt to segment each point into a cluster. Rather, it assumes that clusters may be surrounded by lower density noise.

HDBSCAN has two parameters: `min_cluster_size`, and `min_samples`. The `min_cluster_size` parameter simply puts a lower bound on the size of clusters to form. `min_samples` is less intuitive. It expresses how conservative or aggressive a given clustering of the data should be. A greater value corresponds to a more conservative clustering and more points being declared as noise. Its value generally does not radically affect the final partitioning. However, some quantitative reference for a suitable value of `min_samples` can be found by applying the density-based validation score to resulting clusters.

Spectral Clustering

Spectral clustering is a density-based clustering algorithm related to manifold learning and the DBSCAN algorithm. It allows the user to include specific assumptions concerning the number of clusters to form. Spectral clustering approaches separating and grouping data as a graph partitioning problem. Given a distance metric, d, the algorithm generates a similarity matrix. Typically this is done using a Gaussian kernel similarity function:

$$S_{i,j} = e^{-d_{i,j}^2/(2\sigma^2)}$$

Where σ corresponds to the size of neighborhoods expected to form within the data.

The algorithm then computes the Laplacian matrix of the resulting graph and collects the first k eigenvectors. Data is then projected to the k dimensional vector space spanned by these eigenvectors and clustered in this space using a more simple algorithm (such as k-means). This process corresponds to moving to a vector space in which position expresses connectivity, and clustering in this space.

CHAPTER 2

A Phase Transition in Model GAN Dynamics

2.1 Introduction

Generative modeling is an application of machine learning in which a probability distribution, p_{model}, (defined implicitly or explicitly) is trained to become 'close' to a target distribution, p_{data} [GBC16]. Typically, p_{data} is taken to represent the implicit distribution of elements from a data-set (which could be, for instance, 28×28 pixel images of hand-written digits). A successfully trained generative model would then produce samples which in some sense 'look like' they could have come from the data-set. One promising framework for constructing generative models is known as a *generative adversarial network*, or GAN [GPM14].

Generative adversarial networks are an approach to building generative model using two neural networks: one called the *generator* and the other called the *discriminator* (also referred to as a *critic*). The end goal of the GAN framework is to train the generator to produce samples which are similar to samples taken from a training data-set. Here, we will denote the parameters of the generator, G, by θ, and those of the discriminator, D, by ϕ. The generator is a function which maps points in the latent space, known as seeds, to points in what we refer to as data-space. the discriminator is a function which determines the similarity between generated samples and samples from the data-set. Here, we take the latent-space dimension to be n, and the dimension of the data-points to be d.

$$\text{Generator: } G_\theta : \mathbb{R}^n \to \mathbb{R}^d \tag{2.1}$$

$$\text{Discriminator: } D_\phi : \mathbb{R}^d \to \mathbb{R} \tag{2.2}$$

During training, the generator takes in randomly selected seeds and returns outputs, effectively sampling from an implicit probability distribution, p_{model}. Meanwhile, the discriminator evaluates these generated samples alongside samples selected from a data-set. The task of the discriminator is to assign high scores to samples from the data-set while assigning low scores to generated samples – distinguishing real data from simulated data. The task of the generator is to increase the score assigned to it by the discriminator. At each training step, the parameters of both functions are incrementally updated in according to their individual goals [GPM14, GBC16, AL18, LCC17].

A useful analogy is to the dynamic between a counterfeiter and a detective [GPM14]. The counterfeiter produces fake bills, which the detective studies. The task of the detective is to detect fake currency, and the task of the counterfeiter is to fool the detective. Eventually, the detective may identify some flaw – a difference between real currency and the fake currency. The counterfeiter then must get back to work improving the faked currency to fix this flaw. Eventually, the two reach an equilibrium in which the detective is unable to distinguish between the fake and real bills.

As in the counterfeiter analogy, the task of the generator is to 'fool' the discriminator; to increase the score assigned to its outputs. Ideally, the combined dynamics of the generator and discriminator find a Nash equilibrium in which the discriminator gives equal scores to both the generated and real samples. At this point, the discriminator can be discarded, and the generator used as a generative model in its own right.

2.1.1 Training

In GANs, the generator is some differentiable function (usually a neural network), G_θ, which is fed random inputs, $z \in \mathbb{R}^n$, selected from some distribution $q(z)$. Typically, $z \sim \mathcal{N}^n(0, 1)$, or z is sampled uniformly from an n-dimensional sphere. Samples $G_\theta(z) \in \mathbb{R}^d$ are then produced according to the pushforward measure, implicitly forming the distribution, p_{model}. Different GAN implementations exist, many with distinct loss functions [NCT16]. Here we consider loss functions of the form [LSZ15, LCC17, AL18, NCT16]:

$$\mathcal{L} \equiv \langle D_\phi(x) \rangle_{x\sim p(x)} - \langle D_\phi(G_\theta(z)) \rangle_{z\sim q(z)} - \lambda R(D_\phi) \tag{2.3}$$

Above, $p(x)$ is the probability distribution of samples in the data-set, while $q(z)$ is the probability distribution from which seeds in the latent space are sampled. Finally, $R(D_\phi)$ represents a regularizer on the discriminator, limiting its magnitude under a norm of interest [1]. $\lambda \geq 0$ controls the magnitude of the regularizer.

Conceptually, the discriminator parameters, ϕ, evolve to maximize this quantity, while the discriminator parameters evolve to minimize it.

$$\dot{\phi} = \alpha_D \frac{d\mathcal{L}}{d\phi} \tag{2.4}$$

$$\dot{\theta} = -\alpha_G \frac{d\mathcal{L}}{d\theta} \tag{2.5}$$

The discriminator and generator evolution each occur at individual learning rates α_D and α_G.

Practically, in neural networks a loss function is defined at each training step using mini-batches of N samples of real data and generated data, both of which are re-sampled for each training step:

$$\mathcal{L}_N \equiv \frac{1}{N}\sum_{i=1}^{N} D_\phi(x_i) - \frac{1}{N}\sum_{i=1}^{N} D_\phi(G_\theta(z_i)) - \lambda R(D_\phi) \tag{2.6}$$

Training is performed in iterations. First, for $n_{disc.}$ steps, the discriminator is updated using stochastic gradient ascent:

$$\phi \leftarrow \phi + \alpha_D \nabla_\phi \mathcal{L}_N \tag{2.7}$$

Then, for $n_{gen.}$ steps the generator is updated analogously:

$$\theta \leftarrow \theta - \alpha_G \nabla_\phi \mathcal{L}_N \tag{2.8}$$

[1]Note, for example, that if $R(f_\phi) = ||f||^2_{\mathcal{H}}$ this becomes an MMD GAN, while if $R(f_\phi) = ||f||^2_K$, we have a Wasserstein GAN [AL18, BSA18]

The resulting stochastic gradient ascent and descent are repeated until convergence, or until training has been halted. In this work we take $n_{gen.} = 1$.

Note that the generator is never exposed to the data directly. Instead, it modifies its weights to maximize the average value that the discriminator assigns to its outputs. Intuitively, this feature means that the generator can learn about only what discriminator has learned already, providing motivation for $n_{disc.} > n_{gen.}$.

Algorithm 1 The GAN training algorithm

```
for iteration number do
    for $n_{disc.}$ do
        • Sample $N$ data-points, $\{x_i\}$
        • Sample $N$ seed points, $\{z_i\}$ from the latent space according to $q(z)$
        • Compute
              $$\mathcal{L}_N = \frac{1}{N}\sum_{i=1}^{N} D_\phi(x_i) - \frac{1}{N}\sum_{i=1}^{N} D_\phi(G_\theta(z_i)) - \lambda R(D_\phi)$$
          and update discriminator parameters by ascending its stochastic gradient
              $$\phi \leftarrow \phi + \alpha_D \nabla_\phi \mathcal{L}_N$$
    end for
    • Sample $N$ seed points, $\{z_i\}$ from the latent space according to $q(z)$
    • Compute
          $$\mathcal{L}_N^{gen.} = -\frac{1}{N}\sum_{i=1}^{N} D_\phi(G_\theta(z_i))$$
      and update generator parameters by descending its stochastic gradient
          $$\theta \leftarrow \theta - \alpha_G \nabla_\theta \mathcal{L}_N^{gen.}$$
end for
```

2.1.2 GAN Failure and Mode Collapse

When GANs train successfully, their results can be extremely impressive. By leveraging the expressive and representational power of neural networks, GANs have been used to produce simulated data which appears real even under intense scrutiny [AMB21, KLA21, TXY20, KGE20].

GANs are, however, notoriously hard to train. The adversarial nature of the dynamics distinguishes a GAN's 'loss', $\mathcal{L}$, from a true loss function – one that is bounded from below and which the training algorithm seeks to minimize. Rather than living at the minimum, the ideal parameter settings here are at the saddle-points of the loss landscape [GPM14]:

$$\theta^* = \arg\min_\theta \max_\phi \mathcal{L}(\phi, \theta) \tag{2.9}$$

Convergence to such a fixed-point is difficult to attain, as it requires the careful balancing of the two competing networks during training.

Indeed, convergence can often not even be guaranteed in the most simple GAN models. A canonical example of such a system is given by the following loss [MGN18, GBC16]:

$$\mathcal{L} = \phi \cdot \theta \tag{2.10}$$

Here, under simultaneous gradient dynamics, $\dot{\phi} = \alpha_D \theta$, $\dot{\theta} = -\alpha_G \phi$, which has the solutions

$$\begin{aligned} \phi(t) &= A\sqrt{\frac{\alpha_D}{\alpha_G}} \sin(\sqrt{\alpha_D \alpha_G}(t - t_0)) \\ \theta(t) &= A\cos(\sqrt{\alpha_D \alpha_G}(t - t_0)) \end{aligned} \tag{2.11}$$

In this case, it is guaranteed that unless we start from precisely $\phi(0) = \theta(0) = 0$, gradient ascent/descent will not converge.

One important form of nonconvergence commonly encountered during GAN training is known as *mode collapse* [SVR17, CLJ17]. Mode collapse occurs when samples from the generator fail to capture the full diversity of the samples in the data-set. For instance, this could take the form of a generator creating images of faces which all look alike. The name "mode collapse" refers to the distinct modes which individual elements from a data-set may exhibit. For instance, an image might depict cars, airplanes, or dogs. During mode collapse, the generator's output 'collapses' and it produces samples from relatively few of the available

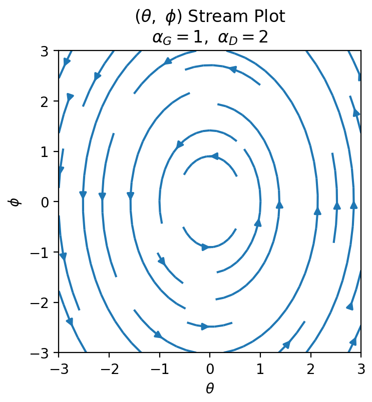


Figure 2.1: A streamplot corresponding to the dynamics of (2.11) for $\alpha_G = 1$, $\alpha_D = 2$. The only stable point lives precisely at $\phi = \theta = 0$.

modes. In other words, the generator's implicit probability distribution, p_{model}, only covers a subset of the data distribution's full support (a simple example of which is shown in figure 2.2).

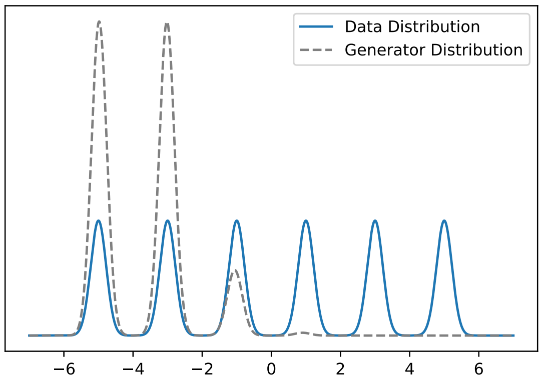


Figure 2.2: Above we depict a toy example of mode collapse. Here, the data distribution is given by 6 Gaussians placed at even increments. The generator has succeeded in replicating the leftmost two modes, however it can produce virtually no samples which would correspond to the modes on the right.

If mode collapse occurs, then during training the generator will focus its distribution on a small subset of the total data-distribution. Eventually, the discriminator learns to identify the concentrated output of the generator, at which point the generator will switch from its current specialization to another [SVR17, CLJ17]. The generator switching from mode to mode, rather than converging on the distribution as a whole is a key symptom of this type of GAN failure.

Relatively little is known about GAN training, however some analytical progress has been made [HRU17, MGN18]. In 2018, [HRU17] proved that GANs could be driven to converge to a local Nash equilibrium when trained using a so-called two time-scale update rule. In the proof, the training rate of the discriminator is generally kept higher than that of the generator. A higher discriminator learning rate makes sense if one recalls that the generator only learns about the data-distribution through feedback provided by the discriminator. Therefore the generator's progress in replicating the data distribution is limited by the progress of the discriminator. Indeed, approaches with faster discriminators have found improved convergence in practice.

2.1.3 The Organization of this Chapter

Here, we attempt to better understand the training dynamics of GANs, with a particular focus on understanding mode collapse.

In section 2.2, we start by experimenting with an extremely simple model for GAN dynamics, and introduce structure until we start to observe the symptoms indicative of mode collapse in true GANs. This leads us to introduce the topic of neural tangent kernels (NTKs) in subsection 2.2.2. We then use this concept in subsection 2.2.3 to enhance mode collapse in our toy model.

We move to a more complex model for GAN training dynamics in section 2.3 – one which still uses the simplifying assumptions of section 2.2, but which comes closer to simulating an actual GAN. We test two limiting cases of this model, one which exhibits fast convergence, and one in which we observe mode collapse.

In section 2.4, we then run experiments in order to probe the boundary between these two 'phases' of GAN training. We are able to characterize the phase-boundary between the two types of dynamics.

This leads to section 2.5, in which we describe the causal mechanism which explain the form of the observed boundary.

Finally, we conclude with a discussion of the relationship between our toy-model GANs, and true GANs.

2.2 Constructing a Minimal GAN Model

To better understand nonconvergence, and specifically mode collapse, we incrementally construct a toy model for a GAN. We begin by studying a variant of the Dirac-GAN in which the data distribution contains multiple modes, and where behavior reminiscent of mode collapse is possible. Drawing lessons from this model, we then proceed to a more realistic toy GAN. We study the parameter settings for this toy model which we lead to convergence, and those which lead to nonconvergence.

2.2.1 Minimal Toy Model

In [MGN18], the authors introduced a toy model for studying GAN training dynamics called a Dirac-GAN. Typically in GANs, the distribution of generated data, p_{model}, is defined implicitly. p_{model} is only accessed by passing random seed inputs to the generator function. Here, however, the distribution[2] is defined explicitly using a single parameter, θ:

$$p_\theta(x) = \delta(x - \theta) \tag{2.12}$$

Note that θ is treated as a parameter of the generator, while it at once describes the one-dimensional output of the generator in data-space [3]. This approach of treating output points as parameters will later be applied more extensively.

[2]Here, we write p_θ instead of p_{model} to reflect the explicit parametrization of the generator's data-distribution

[3]Explicitly, one could take $G_\theta(z) = z + \theta$, where z is trivially selected from the set, $\{0\}$.

The discriminator is taken to be a linear function:

$$D_\phi(x) = \phi \cdot x \tag{2.13}$$

The data distribution is a Dirac-delta centered at zero, $p_{data}(x) = \delta(x)$.

The authors used this model to study the convergence properties of different forms of GAN loss functions in the presence of different regularizers. We are interested in using such a model to reproduce the phenomenon of mode collapse.

In order for a toy model to be capable of demonstrating mode collapse, its data distribution, p_{data}, must contain multiple modes – distinct forms the data may take – and require the generator's distribution to become sufficiently diverse to match it. Incorporating this feature into the Dirac-GAN setup, we define the double Dirac-GAN below.

2.2.1.1 Double Dirac-GAN

In the spirit of the Dirac-GAN, we define the generator for our toy model to have a distribution distribution in a data space (its pushforward measure) of:

$$p_{\theta_1,\theta_2}(x) = (\delta(x - \theta_1) + \delta(x - \theta_2))/2 \tag{2.14}$$

Corresponding to the one-dimensional distribution of two Dirac delta functions parameterized by θ_1 and θ_2. Note θ_1 and θ_2 also give the two possible values of the generator's outputs[4]. Meanwhile, take the data distribution to be:

$$p_{data}(x) = (\delta(x - 1) + \delta(x + 1))/2 \tag{2.15}$$

[4]As will be discussed further, this can be taken to correspond to $G_\theta(z) = \theta_1 + z \cdot (\theta_2 - \theta_1)$, where z is selected from $\{0, 1\}$ with equal probability.

The double Dirac-GAN discriminator is defined as follows:

$$D(x) = b \cdot x + c \cdot x^2 \tag{2.16}$$

Note the additional quadratic term compared to the Dirac-GAN.

Intuitively, during training the discriminator 'tries' to detect the difference between $p_{\theta_1,\theta_2}(x)$ and $p_{data}(x)$, and its parameters, b and c, evolve to increase the difference between D evaluated at ± 1 and D evaluated at θ_1, θ_2. Simultaneously, the generator evolves to reduce this difference, adjusting θ_1 and θ_2 to increase the discriminator's value at these points.

Define our loss function as follows [5]:

$$\mathcal{L} = \langle D(x) \rangle_x - \langle D(G_\theta(z)) \rangle_z - \frac{\lambda}{2} ||D||^2 \tag{2.17}$$

$$= c - \big(b(\theta_1 + \theta_2)/2 + c(\theta_1^2 + \theta_2^2)/2\big) - \lambda(b^2 + c^2)/2 \tag{2.18}$$

Under simultaneous gradient descent/ascent, the dynamics of the generator and discriminator are

$$\dot{\theta}_t = -\frac{d\mathcal{L}}{d\theta_t} = \frac{d}{d\theta_t} \langle D_\phi(G_\theta(z)) \rangle_{z \sim q(z)} \tag{2.19}$$

$$\dot{\phi}_t = \eta \frac{d\mathcal{L}}{d\phi_t} \tag{2.20}$$

Where η is the relative training rate of the discriminator. Here, dropping the time subscript,

[5] Here we take the discriminator to be an element in the reproducing kernel Hilbert space with feature map $\Phi(x) = (x, x^2)$. The discriminator is then $D(x) = \mu \cdot \Phi(x)$, and we define its regularizer using the RKHS norm: $|| \cdot ||_{\mathcal{H}}$. Therefore $||D||^2_{\mathcal{H}} = ||\mu||^2 = b^2 + c^2$.

this corresponds to

$$\dot{\theta}_i = -\frac{d\mathcal{L}}{d\theta_i} \tag{2.21}$$

$$\dot{b} = \eta \frac{d\mathcal{L}}{db} \tag{2.22}$$

$$\dot{c} = \eta \frac{d\mathcal{L}}{dc} \tag{2.23}$$

yielding

$$-\frac{d\mathcal{L}}{d\theta_i} = \frac{d}{d\theta_i}\big(b(\theta_1+\theta_2)/2 + c(\theta_1^2+\theta_2^2)/2\big) = b/2 + c\theta_i \tag{2.24}$$

meanwhile

$$\frac{d\mathcal{L}}{db} = -(\theta_1+\theta_2)/2 - \lambda b \tag{2.25}$$

$$\frac{d\mathcal{L}}{dc} = 1 - (\theta_1^2+\theta_2^2)/2 - \lambda c \tag{2.26}$$

2.2.1.2 Initial Experiments

As a first attempt to probe this system, stability analysis can be applied to study the dynamics close to equilibrium: $(b, c, \theta_1, \theta_2) = (0, 0, \pm 1, \mp 1)$. Define $x_1 = \theta_1 + 1$, $x_2 = \theta_2 - 1$, and scale the speed of the discriminator evolution by η. Then the linearized system is descibed by:

$$\begin{pmatrix} \dot{b} \\ \dot{c} \\ \dot{x_1} \\ \dot{x_2} \end{pmatrix} = \begin{pmatrix} -\eta\lambda & 0 & -\eta/2 & -\eta/2 \\ 0 & -\eta\lambda & -\eta & \eta \\ 1/2 & 1 & 0 & 0 \\ 1/2 & -1 & 0 & 0 \end{pmatrix} \begin{pmatrix} b \\ c \\ x_1 \\ x_2 \end{pmatrix}$$

With the above matrix having eigenvalues

$$-\frac{\eta\lambda}{2} \pm \sqrt{\left(\frac{\eta\lambda}{2}\right)^2 - 2\eta}$$

$$-\frac{\eta\lambda}{2} \pm \sqrt{\left(\frac{\eta\lambda}{2}\right)^2 - \eta/2}$$

Since the discriminator's learning rate, η is always positive, the real part of these eigenvalues will always be negative, implying that near equilibrium the linearized dynamics always implies local convergence. We can also see that for slow discriminators (small η), we expect the dominant behavior to be oscillatory, and convergence to be impeded. This fact is in line with known results on the convergence of GANs, since these typically encourage the use of a discriminator which is trained 'faster' than the generator [HRU17].

We would like to use this double Dirac-GAN to simulate the phenomenon mode collapse: the situation in which the output distribution of a generator replicates only a subset of the full distribution of the data. Within a double Dirac-GAN, this would take the form of both Dirac deltas of p_{θ_1,θ_2} concentrating around one of the modes of the data distribution: either 1 or -1. During GAN training, mode collapse results in the generator changing its output from one mode to another, while failing to distribute its outputs to cover the entire distribution. In our toy model, this would mean that θ_1 and θ_2 stay close to one another while both oscillate between 1 and -1.

The presence of mode collapse would necessarily place our system far from equilibrium[6], and out of the scope of a simple linear stability analysis. We therefore run simulations of the nonlinear dynamics; priming our system to demonstrate mode collapse, and studying the features of mode collapse which present themselves.

In particular, we experiment by placing both θ_1 and θ_2 near the $+1$ mode (setting $\theta_1 = 1$ and $\theta_2 = .99$). The parameters b and c are initially set to zero, and the regularization parameter, λ, is set to 1. To quantify mode collapse for our toy model, we will compare the characteristic time of θ_1 and θ_2 splitting, $\tau_{\text{split.}}$, to the characteristic time of their oscillation[7],

[6]Note that if θ_1 is near θ_2, then we cannot be in the situation of both being close to equilibrium

[7]$\tau_{\text{split.}}$ is computed by fitting the function

$$1 + \tanh\left((t - \tau_{\text{split.}}) \cdot b\right)$$

to the curve produced by $|\theta_1 - \theta_2|$. $\tau_{\text{osc.}}$ is given by the inverse of the dominant Fourier frequency of $(\theta_1+\theta_2)/2$.

$\tau_{\text{osc.}}$. Roughly, we will say that mode collapse corresponds to $\tau_{\text{split.}} \gg \tau_{\text{osc.}}$, since this would mean the outputs oscillate between ± 1 much faster than they split apart to cover both.

We then run experiments, allowing the generator and discriminator to undergo their simultaneous gradient dynamics. Sweeping over a broad range of discriminator learning rates, η, we find that even for very slow discriminators, the system is able to converge rapidly to its equilibrium value of $\theta_1 = \pm 1$, $\theta_2 = \mp 1$, and $b = c = 0$. This can be seen in Figure 2.3, in which a very slow discriminator is used used ($\eta = 0.05$). Despite this, equilibrium is quickly reached.

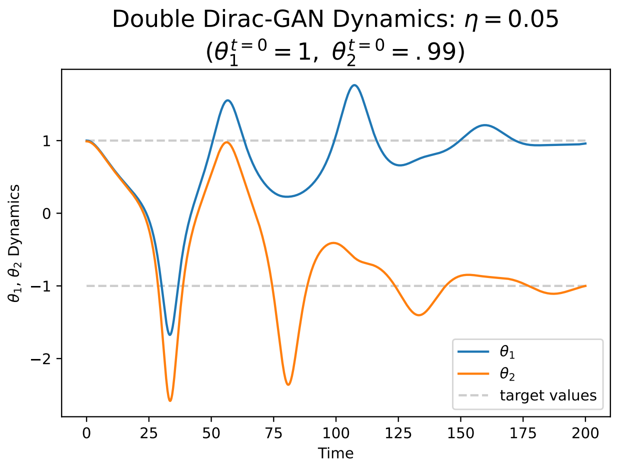


Figure 2.3: Above is shown the dynamics of θ_1 and θ_2 when they are initialized at $\theta_1, \theta_2 = 1, .99$. The discriminator learning rate is taken to be extremely small, with $\eta = 0.05$. Even in this case – primed to display mode collapse – we observe fast splitting of the two modes, which quickly converge to equilibrium at ± 1.

Indeed, the discriminator learning rate must be set to an extremely small value to see any significant signatures of mode collapse. This is shown in Figure 2.4, where a metric for mode collapse, $\log(\tau_{split.}/\tau_{osc.})$, is computed for each η value. In order for this value to be large, η must be extremely small.

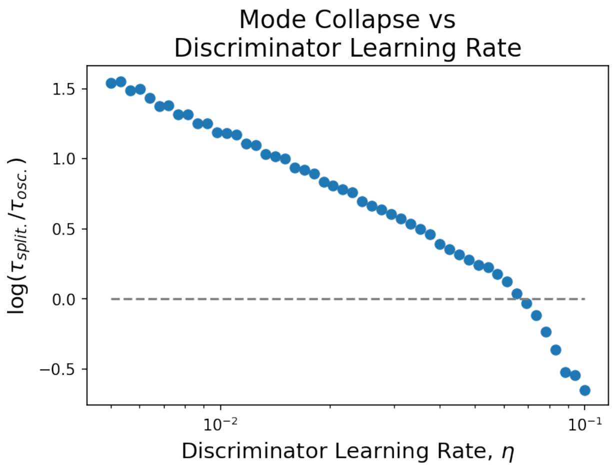


Figure 2.4: Above, our metric for mode collapse, $\log(\tau_{split.}/\tau_{osc.})$, is plotted as a function of the discriminator learning rate. Note that only for extremely (and impractically) small discriminator learning rates do we find an indication of mode collapse.

Therefore, even when both outputs are initialized near one of the two modes, we fail to see strong signatures of mode collapse, and only obtain $\tau_{\text{split.}} \gg \tau_{\text{osc.}}$ for impractically small values of η. This raises the question of which realistic additional feature might be added to our model in order to both make it both a more accurate proxy for neural network dynamics, and allow it to display mode collapse.

2.2.2 The Neural Tangent Kernel

Previously, the generator's outputs evolved independently. In neural networks, however, this is not how points in data-space evolve.

Generically, within a neural network G with parameters θ, $G_\theta : \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out}}$, the neural tangent kernel (NTK) is defined by [JGH18]

$$\Gamma_\theta : \mathbb{R}^{d_{in}} \times \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out} \times d_{out}} \tag{2.27}$$

$$\Gamma_\theta^{i,j}(z, z') = \frac{dG^i(z)}{d\theta_k} \frac{dG^j(z)}{d\theta_k} \tag{2.28}$$

As we will show below, this quantity arises naturally when considering the dynamics of points in data-space.

Taking the parameters to be time dependent, θ_t, we can denote a point in data-space which is mapped to by the seed z at time t by

$$X_t = G_{\theta_t}(z)$$

We can then see that updates in parameter space, $d\theta_t$, are related to updates in data-space, dX_t, by

$$dX_t = \frac{dG_\theta(z)}{d\theta}^T |_{\theta=\theta_t} \frac{d\theta_t}{dt} dt \tag{2.29}$$

Under simultaneous gradient dynamics the generator parameters evolve according to (2.19), and so

$$\frac{d\theta_t}{dt} = \frac{d}{d\theta}\left(\int dz' q(z') D(G_\theta(z'))\right)|_{\theta=\theta_t} = \int dz' q(z') D'(G_{\theta_t}(z')) \frac{dG_\theta(z')}{d\theta}|_{\theta=\theta_t} \tag{2.30}$$

To see the corresponding data-space dynamics, we plug this in to (2.29) find

$$dX = dt \int dz' q(z') D'(G_{\theta_t}(z')) \left(\frac{dG_\theta(z)}{d\theta}^T \frac{dG_\theta(z')}{d\theta}\right)|_{\theta=\theta_t} \tag{2.31}$$

$$dX = dt \int dz' \; \Gamma_{\theta_t}(z, z') D'(G_{\theta_t}(z')) q(z') \tag{2.32}$$

In this way the NTK, Γ_θ, tells us the degree to which the gradients of the loss at other points $X' = G_\theta(z')$ influence the dynamics of the the point $X = G_\theta(z)$. For example, if we take a diagonal NTK, $\Gamma(z, z') = \delta(z - z') \; \mathbb{I}$, then we regain

$$dX \propto dt D'(X) \tag{2.33}$$

corresponding to the original uncorrelated dynamics, in which the velocity of points in data-space is completely determined by the local gradient of the discriminator. As we introduce off-diagonal terms to the NTK, the velocity in data-space becomes correlated, as it is now influenced by the gradient of D evaluated at different points.

NTKs are random at initialization, and in general evolve during training. As we increase the width of the network, however, the degree to which the NTK changes throughout training drops [JGH18, RYH22]. In the full infinite-width limit, the NTK becomes fixed at initialization: $\Gamma_{\theta_t} = \Gamma_{\theta_0}$.

The infinite-width regime is of particular interest, as the performance of neural networks has been observed to improve as their width is increased. Additionally, in this limit it becomes possible to derive analytical results, as certain theoretical aspects of neural networks simplify in the large-width regime. Working from this limit as a starting point has consequently proved fruitful for drawing conclusions about more realistic finite-width networks [RYH22, HN20, HMS21].

For certain architectures, the exact form of the infinite-width NTK can be worked out analytically. Such an exact NTK can be found, for instance, in the case of an infinite width 2-layer ReLU network. Here, ReLU refers to the activation function used within the network, which is defined by

$$\sigma(x) = \max(0, x)$$

and is among the most popular activation functions used in practice.

For example, a single hidden-layer ReLU network of width N without biases, and with $w_i^j,\ a_i \sim \mathcal{N}(0,1)$:

$$f_\theta : \mathbb{R}^d \to \mathbb{R}$$
$$f_\theta(z) = \ a_i \sigma(w_i^j z^j)$$

has an infinite width NTK is given by [BM19]

$$\Gamma_{\text{ReLU}}(z, z') \cdot \frac{2}{N} = |z||z'|\kappa(\frac{z \cdot z'}{|z||z'|}) \tag{2.34}$$

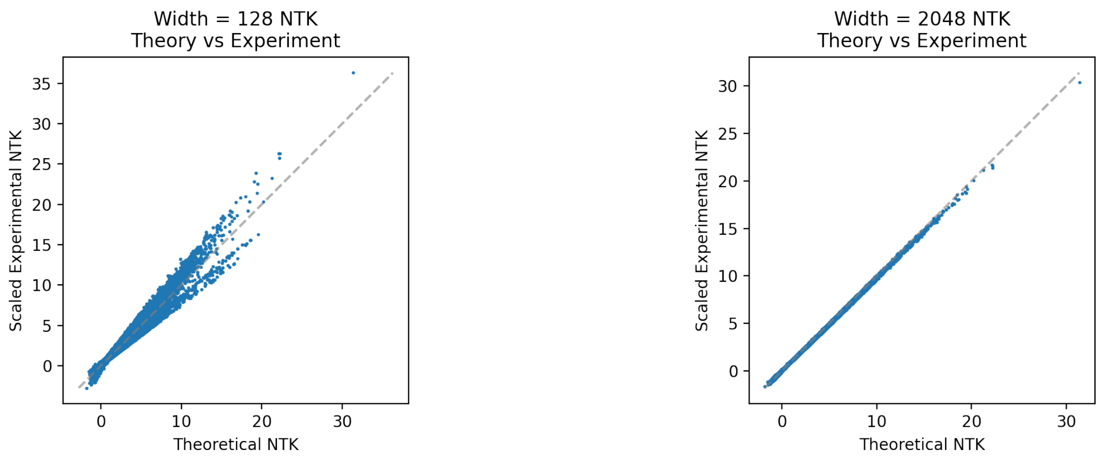


Figure 2.5: Above we compare the theoretical NTK prediction of (2.34) to numerically obtained NTK values on a sample of 500 points selected from a 2-dimensional standard normal. Note that as the width of the neural network is increased, the experimental values approach the theoretically predicted results with high accuracy.

where $2/N$ provides a normalization as $N \to \infty$, and

$$\kappa(x) \equiv x\kappa_0(x) + \kappa_1(x)$$

$$\kappa_0(x) \equiv \frac{1}{\pi}(\pi - \arccos(x)) \quad \kappa_1(x) \equiv \frac{1}{\pi}(x \cdot (\pi - \arccos(x)) + \sqrt{1-x^2})$$

This becomes exact in the large N limit, as can be seen in figure 2.5.

In generative adversarial networks, random seeds are provided to the generator by sampling from a distribution, $q(z)$. As mentioned, z is often selected from $\mathcal{N}^d(0,1)$ or is uniformly selected from the unit sphere in d dimensions, where d is taken to be sufficiently large to encompass variation of the data-set. Since points from $\mathcal{N}^d(0,1)$ are approximately on the unit sphere[8] in d dimensions of radius $\sqrt{d}$, for our purposes it will be sufficient to sample seed points z uniformly from a d-dimensional unit sphere.

If inputs have equal length, as the above $q(z)$ ensures, then $\Gamma_{\text{ReLU}}(z, z')$ becomes a func-

[8]Note that a d-dimensional vector with elements selected from $\mathcal{N}(0,\sigma^2)$ will have an average squared length of $d\sigma^2$, and the relative standard deviation of this estimate will drop as $\sqrt{\frac{2}{d}}$.

tion only of the dot product[9]: $z \cdot z'$. As the value of d grows, the distribution of this inner product can be estimated[10] by

$$\cos(\varphi) \sim \mathcal{N}(0, \sigma^2 = \frac{1}{d})$$

where φ is the angle between z and z'. With this in mind, given a dot product NTK, and a sample of inputs, the distribution of NTK values can be estimated as well. In particular, for distinct points z, z', $\Gamma(z, z')$ will, with high probability, be close to $\Gamma(z, z_\perp)$. In other words, if we express Γ in terms of the angle φ, most values will fall close to $\Gamma(\varphi = \pi/2)$, while for all z in the sample, $\Gamma(z, z) = \Gamma(\varphi = 0)$.

The above observation is reflected in experiments, as almost all distinct points have an NTK value centered near the theoretically predicted value for $\Gamma(\varphi = 0)$. Moreover, since all sampled points are approximately orthogonal (and equidistant), any correlation observed in data-space at initialization between the positions of $X = G_{\theta_0}(z)$ and $X' = G_{\theta_0}(z')$ will fail to be reflected in the corresponding value of $\Gamma_{\theta_0}(z, z')$ as can be seen in Figure 2.6.

Although we have focused on a ReLU NTK, such a discussion applies more generally. For instance the Erf activation function can also be shown analytically to give an NTK which is a function of dot product[LXS20], and has a form similar to that of Tanh, another extremely

[9]This statement still applies if we include biases sampled according to $\mathcal{N}(0, \sigma_b^2)$ in our network, or if we choose different variances for a_i and w_i^j

[10]Take z, z' to be selected from a normal distribution in d dimensions. Without loss of generality, rotate such that

$$z = (|z|, 0, 0, \cdots 0, 0)$$

Then

$$\frac{z \cdot z'}{|z||z'|} = \frac{z'_1}{|z'|}$$

Here, $|z'| \approx \sqrt{d}\sigma$, and $z'_1 \sim \mathcal{N}(0, \sigma^2)$. Therefore in high dimensions

$$\cos(\phi) \sim \mathcal{N}(0, \sigma^2 = \frac{1}{d})$$

And so almost all pairs of points will have

$$\cos(\phi) \approx 0$$

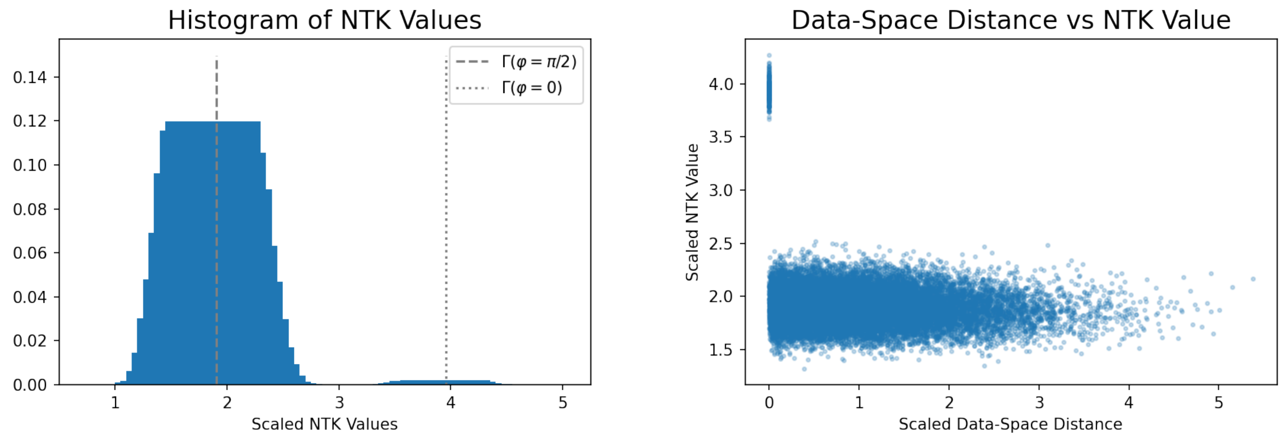


Figure 2.6: Above we show the distribution of NTK values. Two hundred inputs, $\{z_i\}$ are sampled from a unit sphere in 100 dimensions. A ReLU network with width 2048 and one hidden layer is then used to compute NTK values for each unique pair of inputs: $f_\theta(z) = a_i\sigma(w_i^j z^j + b_i)$, taking $a_i,\ w_i^j,\ b_i \sim \mathcal{N}(0,1)$. To the left is shown a histogram of NTK values, reflecting the fact that the vast majority are centered around $\Gamma(\varphi = \pi/2)$, corresponding to orthogonal inputs. Meanwhile, far fewer values make up another peak, corresponding to $\Gamma(z,z) = \Gamma(\varphi = 0)$. To the right we show the relationship between the NTK value and data-space distance. This is done using a scatter plot comparing $|f_\theta(z) - f_\theta(z')|$ to $\Gamma(z,z')$. At initialization, there is no correlation is present for distinct inputs $z \neq z'$, motivating the approximation of (2.35)

popular activation function.

With the above facts about the NTK in mind, in the next sections we will introduce NTKs of the form:

$$\Gamma^{i,j}(z,z') = \delta_{i,j}\left(g_1\delta_{z,z'} + g_2(1-\delta_{z,z'})\right) \tag{2.35}$$

This NTK is static throughout training, corresponding to an infinite-width generator. It also has a single constant value for diagonals, and another constant value for off-diagonals, roughly corresponding to $g_1 \approx \Gamma(\varphi = 0)$ and $g_2 \approx \Gamma(\varphi = \pi/2)$. The $\delta_{i,j}$ out front can be understood as implying a lack of correlation between the gradients of output degrees of freedom of an infinite width neural network[11].

Physically, we will see how adding an NTK such as (2.35) correlates velocities of points in data-space.

2.2.3 Breaking the Toy Model

Our goal is to create a toy model for GAN training dynamics in which the phenomenon of mode collapse is apparent. As an additional feature of our model, we will now add a static NTK (with the form of (2.35)), effectively pretending that the generator outputs come from an infinite-width neural network.

Within the double-Dirac GAN, θ_i were at once taken to be generator parameters, as well as the outputs of the generator (as the distribution (2.14) implies). As noted, we previously

[11]This can be seen by considering a random large-width neural network with two outputs. Writing this as

$$f_\theta(z) = (f_\theta^{(1)}(z), f_\theta^{(2)}(z)) = (\Phi_\vartheta^i(z)\ a_i^{(1)} + b_1, \Phi_\vartheta^i(z)\ a_i^{(2)} + b_2)$$

Where

$$a_i^{(j)} \sim \mathcal{N}(\mu = 0, \sigma^2 = \frac{\sigma_w^2}{n_w})$$

and $\Phi_\vartheta^i(z)$ represents all but the final layers of the network. We can see that

$$\Gamma^{1,2}(z_1,z_2)) = \frac{df^{(1)}(z_1)}{d\theta_a}\frac{df^{(2)}(z_2)}{d\theta_a} = \frac{d\Phi_\vartheta^i(z)}{d\vartheta_k}\frac{d\Phi_\vartheta^j(z)}{d\vartheta_k}a_i^{(1)}a_j^{(2)}$$

which has mean zero and a variance that decays with $1/n_w$.

allowed these generator outputs, θ_1 and θ_2, to independently evolve during training. Each took on a velocity which was a function of the local derivative of the discriminator. Meanwhile, in true neural networks, outputs evolve according to a weighted sum of discriminator gradients across data-space. We will therefore attempt to incorporate this feature of neural networks by modifying the generator parameters by hand, explicitly inserting an NTK into the differential equations which determine the dynamics of θ_i.

For our one-dimensional model, a static NTK can be incorporated as follows: Define

$$q(z=0) = 1/2 \quad q(z=1) = 1/2 \tag{2.36}$$

and

$$G_\theta(0) = \theta_1 \tag{2.37}$$

$$G_\theta(1) = \theta_2 \tag{2.38}$$

Then the addition of an NTK, Γ, modifies the dynamics as follows:

$$d\theta = dt\frac{1}{2}\left(\Gamma(z,0)D'(G_\theta(0)) + \Gamma(z,1)D'(G_\theta(1))\right)$$

Using

$$\Gamma_\alpha(1,1) = \Gamma_\alpha(0,0) \equiv 1-\alpha \tag{2.39}$$

$$\Gamma_\alpha(0,1) = \Gamma_\alpha(1,0) \equiv \alpha \tag{2.40}$$

we can therefore write

$$d\theta_1 = dt\frac{1}{2}\left((1-\alpha)D'(\theta_1) + \alpha D'(\theta_2)\right)$$

$$d\theta_2 = dt\frac{1}{2}\left((1-\alpha)D'(\theta_2) + \alpha D'(\theta_1)\right)$$

Note that for $\alpha = 0$, this produces dynamics proportional to those observed without an NTK. On its face, the NTK here can be seen as correlating the velocities of the outputs.

Indeed, in the trivial limit of $\alpha = 1/2$, $\frac{d\theta_1}{dt} = \frac{d\theta_2}{dt}$, and the two points can never split apart.

Now, when we initialize θ_1 and θ_2 close to one another, and increase α, we fail to see fast convergence. Instead of splitting off to their respective ideal values, the two parameters stay together and travel from one mode to another, displaying $\tau_{\text{split.}} \gg \tau_{\text{osc.}}$.

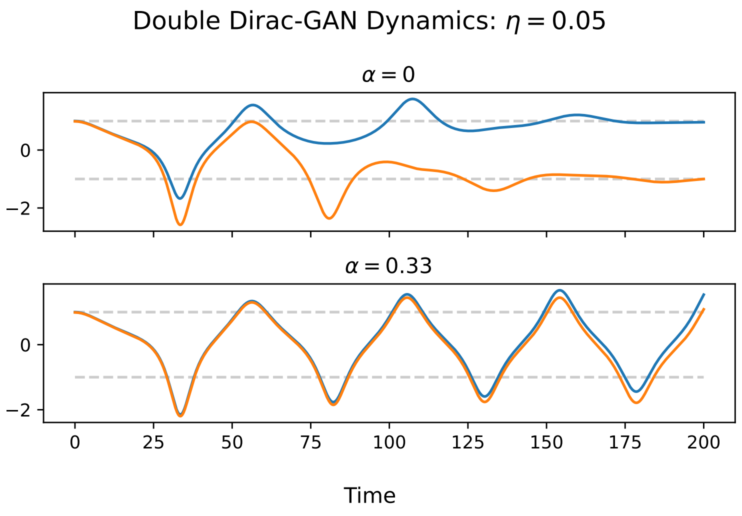


Figure 2.7: Above is shown the dynamics of θ_1 and θ_2 when initialized at $\theta_1, \theta_2 = 1, .99$ and with the discriminator learning rate set to $\eta = 0.05$. We compare the dynamics resulting from the NTK's off-diagonal value, α being set to $\alpha = 0$, with the dynamics for $\alpha = 1/3$. Without an NTK, the system quickly converges to equilibrium (shown in the upper plot). When an NTK is present (shown in the lower plot) we observe strong signatures of mode collapse, with $\tau_{\text{split.}} \gg \tau_{\text{osc.}}$

We can examine this behavior by varying η, the training rate of the discriminator, along with α, the NTK's off-diagonal parameter. Sweeping across parameter space, we can compute a metric indicating mode collapse, $\log(\tau_{\text{split.}}/\tau_{\text{osc.}})$, at each combination. The resulting diagram is shown in Figure 2.8. As the contour lines indicate, as we increase α, mode collapse increases (note the positive slope of the contour lines). This hints at a link between the effect of the generator's NTK and the presence of mode collapse during GAN training.

Using the lessons learned, we will now proceed to a more complex and realistic setting. Again, we will introduce a static NTK to couple the velocities of generator outputs (treated as parameters), and run experiments to more clearly understand the transition to mode collapse.

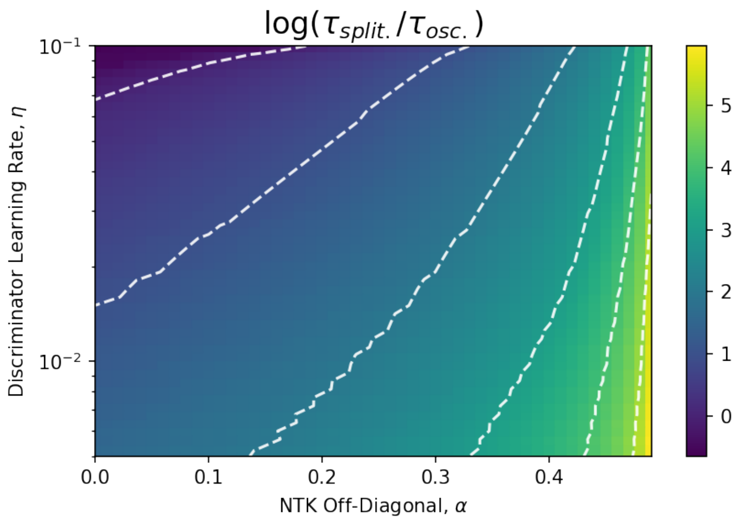


Figure 2.8: The mode collapse (indicated by the value of $\log(\tau_{\text{split.}}/\tau_{\text{osc.}})$) exhibited by the double-Dirac GAN as a function of discriminator learning rate, η, and the NTK parameter α. Whereas before, we were only able to observe $\tau_{\text{split.}} \gg \tau_{\text{osc.}}$ for unrealistically small values of η, with the introduction of the NTK, we can now see mode collapse for much larger and realistic values of η. Note the positive slope of the contours, indicating that α serves to increase the mode collapse metric, $\log(\tau_{\text{split.}}/\tau_{\text{osc.}})$.

2.3 8-Gaussian Toy Model

Here we investigate an analogous toy model in a more complicated setting. We consider a canonical problem of training a GAN on the distribution of 8 Gaussians arranged in a circle of radius 2, each Gaussian having a standard deviation 0.02. This data distribution is used throughout GAN literature as a toy data-set for observing mode collapse [SVR17, CLJ17, MN21], since each Gaussian can be naturally taken to represent a distinct mode. Mode collapse in this context would correspond to a generator whose outputs are focused on one, or a subset, of the eight Gaussians. During training, mode collapse would cause these outputs to oscillate from mode to mode, rather than splitting up to cover the entire distribution.

We replace the double-Dirac GAN generator (composed of two parameterized points, θ_1 and θ_2) with 2000 parameterized points in the plane, initialized as a Gaussian distribution

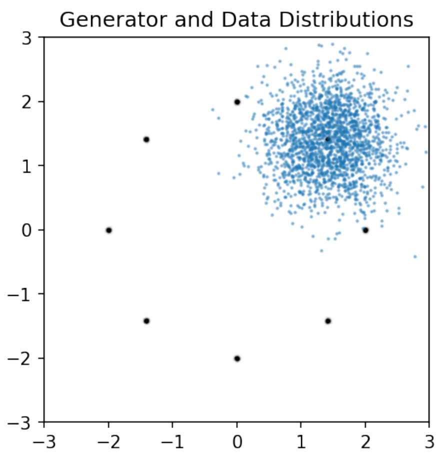


Figure 2.9: A depiction of the distribution of 8 Gaussians arranged in a circle of radius 2 (in black), and the initial distribution of the generator distribution (focused at $(\sqrt{2}, \sqrt{2})$, in blue). The presence of multiple modes within the data distribution (each distinct Gaussian) allows for the possibility of mode collapse during training.

with $\sigma = 0.5$. Whereas previously, the generator distribution was taken to be

$$p_{\theta_1,\theta_2}(x) = \frac{1}{2}(\delta(x - \theta_1) + \delta(x - \theta_2))$$

we now write its distribution as

$$p_X(x) = \frac{1}{N} \sum_i^N \delta(x - X_i) \tag{2.41}$$

Such a 'cloud of points' represents a proxy for the full distribution in data-space of a generator neural network[12]. Note that by working with the cloud itself, and applying the dynamics given by equation (2.32), we can ignore the details of the generator neural network while still learning about its properties[13].

Here, instead of writing the NTK as a function of imagined inputs from a latent space, z, we will now express Γ as a matrix with indices corresponding to points X_i in data-space.

[12]Meaning the pushforward of the distribution $q(z)$ through a generator G.

[13]A similar point-cloud model was previously applied in [MSR19], where it was studied using a diagonal NTK.

Following the notation of (2.35), this takes the form

$$\Gamma^{k,l}(X_i, X_j) = \delta_{k,l}(g_1\delta_{i,j} + g_2(1-\delta_{i,j})) \tag{2.42}$$

By plugging equations (2.41), (2.42) into the generator dynamics given by equation (2.32), we find that X_i evolve according to

$$\frac{dX_i}{dt} \propto \frac{1}{N}\sum_{j}^{N} \Gamma_{i,j} \nabla_x D(X_j) \tag{2.43}$$

with the constant of proportionality dependent on the learning rate of the generator.

The discriminator is a neural network with 4 hidden layers of width 512. During training, it is updated according to gradient ascent, while the generator points, X_i, are updated using gradient descent. The algorithm for this training routine is described in algorithm 2.

Algorithm 2 The model-GAN training algorithm, with a cloud of parameterized points as our simulated generator distribution.

```
for iteration number do
    for $n_{disc.}$ do
        • Sample $N$ data-points, $\{x_i\}$, from the 8-Gaussian distribution.
        • Compute
            $\mathcal{L}_N = \frac{1}{N}\sum_{i=1}^{N} D_\phi(x_i) - \frac{1}{N}\sum_{i=1}^{N} D_\phi(X_i) - \frac{\lambda}{2}\sum_j \phi_j^2$
        and update discriminator parameters by ascending its stochastic gradient
            $\phi \leftarrow \phi + \alpha_D \nabla_\phi \mathcal{L}_N$
    end for
    • update $X_l$ according to equation (2.32)
        $X_l \leftarrow X_l + \alpha_G \frac{1}{N}\sum_{l}^{N} \Gamma_{k,l} \nabla_x D(X_l)$
end for
```

First, we evolve the points using a trivial diagonal NTK, before introducing an NTK with non-trivial off-diagonals and observing the dynamics. The presence of mode collapse will be probed by examination of the distribution of generator points closest to each mode. A uniform distribution of 250 points closest to each mode would be a sign indicating symmetry, and that mode collapse has not occurred. Meanwhile, we can also examine the average minimum distance to the closest mode to characterize how close the distributions have converged to any of the 8 Gaussians.

2.3.1 Trivial NTK

In the first case, the NTK is set to be diagonal, and outputs are allowed to evolve independently. Each data-point, X, evolves according to

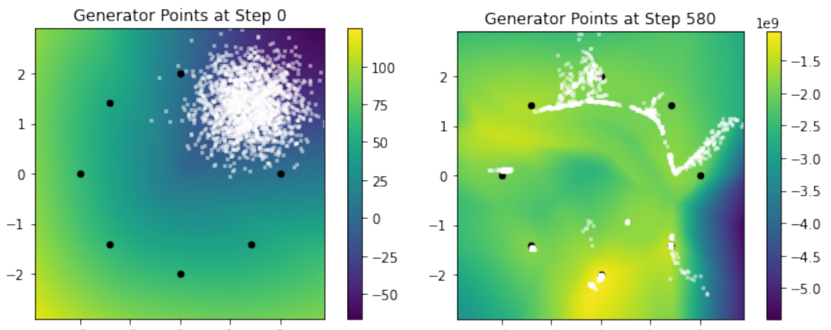
$$dX \propto \nabla_x D(X)$$

To increase the chance of mode collapse, we initialize the generator points using a normal distribution of $\sigma = 0.5$, and with $\mu = (\sqrt{2}, \sqrt{2})$ – centered directly on top of one of the eight modes. Time slices of the dynamics[14] can be found in Figure 2.10.

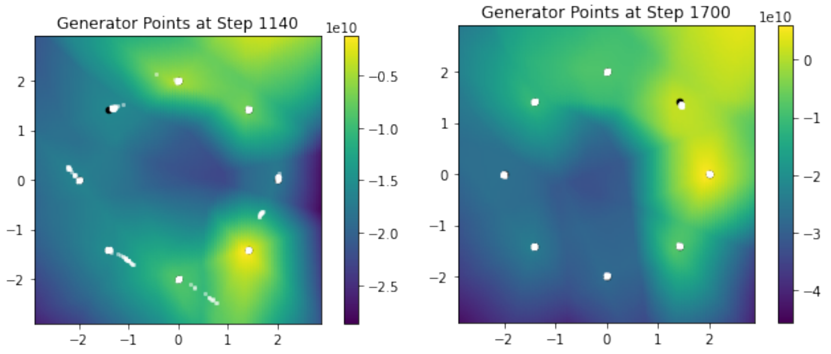
Here, regardless of how the points are initialized, they quickly converge to the ideal distribution. In the above case, the number of points closest to each mode is almost perfectly uniform, with the modes having 243, 255, 250, 249, 249, 251, 251, and 252 nearest points each.

Furthermore, each point is extremely close to its nearest mode, with an average distance of 0.02.

[14]We use a generator learning rate of 0.05, and a discriminator learning rate of 0.1. Following algorithm 1, we set $n_{disc.} = 5$ and $n_{gen.} = 1$. Furthermote, an L_2 regularizer of $\lambda = 10^3$ was used (see algorithm 2 for its role).



(a) Training Step 0 (b) Training Step 580



(c) Training Step 1140 (d) Training Step 1700

Figure 2.10: Here, we display the generator point dynamics over time using a diagonal NTK. Generator points, displayed in white over a color-plot of the discriminator's outputs, are able to converge to cover the eight modes. During training, the discriminator seeks to assign high values (brighter colors) to real data-points (the eight Gaussians), and low values (darker colors) to the generator points. Noting the dynamics described in equation (2.32), we normalize the generator's dynamics by the particle number, setting $g_1 = 2000$, $g_2 = 0$ so that $\frac{1}{N}\sum_{j,k}^{N}\Gamma(X_j, X_k)D'(X_k) = D'(X_k)$.

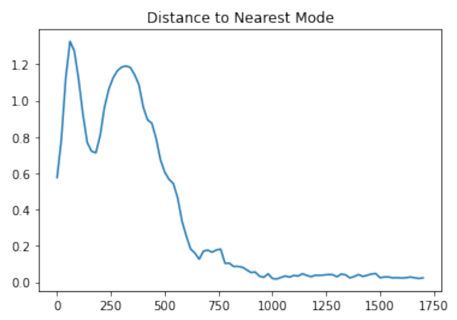


Figure 2.11: The average distance to the nearest mode during training for $g_2/g_1 = 0$. This system quickly converges.

2.3.2 Nontrivial NTK

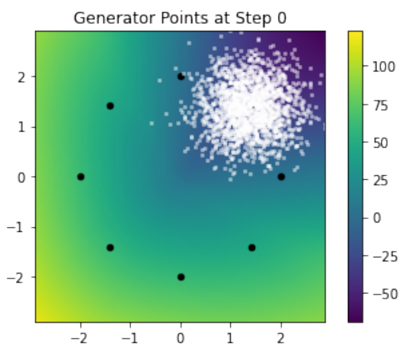
Now we add in an NTK with a significant off-diagonal component, taking the ratio of the off-diagonal to diagonal constants in equation (2.42) to be[15]

$$g_2/g_1 = 1/5$$

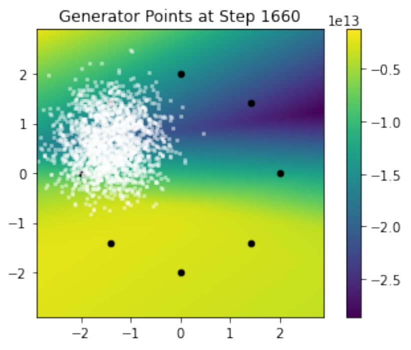
As is clear from Figure 2.12, this places us deep into the mode collapse regime of training.

Now, rather than converging directly to the 8 Gaussians, the generator points oscillate together from mode to mode. As in the simple toy-model, this oscillation strongly resembles the behavior typical during mode collapse, in which generator outputs fail to become diverse, instead staying together and switching from mode to mode. This behavior can also be seen clearly by examining the average distance to the nearest mode during training (shown in Figure 2.13).

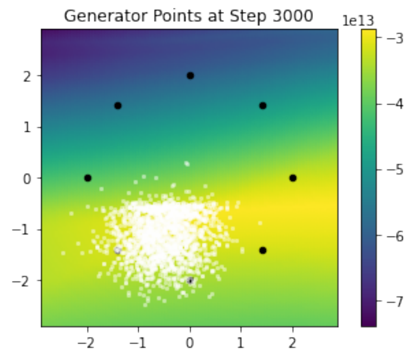
[15]Specifically, noting the form of equation (2.32), we scale these values according to the number of generator points, N. Here we take $g_1 = N$ and $g_2 = N/5$.



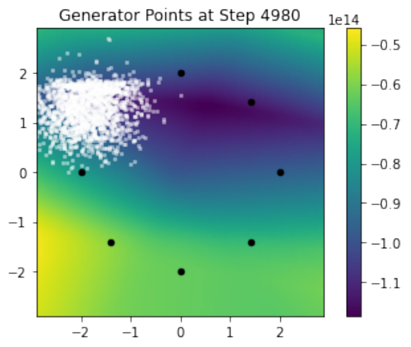
(a) Training Step 0



(b) Training Step 1660



(c) Training Step 3000



(d) Training Step 4980

Figure 2.12: Point dynamics over time with an NTK such that $g_2/g_1 = 1/5$. The generator points, displayed in white over a color-plot of the discriminator's values, now fail to converge. Instead, the switch from mode to mode, a behavior indicative of mode collapse. The discriminator seeks to assign high values (brighter colors) to real datapoints, and low values (darker colors) to generated points, and is unable to 'split' apart the generator points. This indicates that we are deep within the mode-collapse phase

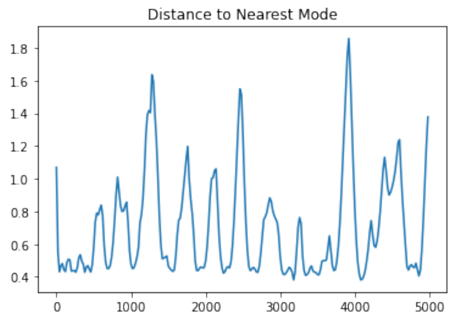


Figure 2.13: The average distance to the nearest mode during training using $g_2/g_1 = 1/5$. Note this system fails to converge, instead switching from mode to mode.

2.3.2.1 Physical Intuition

The physical reason for the onset of mode collapse understood by considering the $g_1 = g_2 = g$ limit. Here, equation (2.43) tells us that

$$\frac{dX_i}{dt} = \frac{1}{N}\sum_j \Gamma(X_i, X_j)D'(X_j) = \gamma\langle D'(X)\rangle$$

Therefore $\frac{dX_i}{dt}$ is the same for all i, and the points are forced to remain together during training, producing mode collapse.

For $g_1 \neq g_2$, the velocity of X_i can be decomposed into two components:

$$\frac{dX_i}{dt} = \frac{1}{N}\sum_j \Gamma(X_i, X_j)D'(X_j) = \left(\frac{g_1 - g_2}{N}\right) D'(X_i) + g_2\langle D'(X)\rangle \qquad (2.44)$$

The first, $\left(\frac{g_1-g_2}{N}\right) D'(X_i)$, contributes a velocity proportional to the local derivative of the discriminator at X_i. The second, $g_2\langle D'(X)\rangle$, contributes a velocity proportional to the average derivative of the discriminator over the entire generator distribution.

In the presence of a diagonal NTK ($g_2 = 0$), two points X, X' might diverge as long as $D'(X)$ opposed $D'(X')$. This could happen whenever the discriminator found a minimum somewhere between X and X'. With $g_2 > 0$, however, the velocity of X and X' would also be influenced by $\langle D'(X)\rangle$.

Although initially, the relative velocity between X and X' would be unaffected by this total velocity contribution, it would cause the generator distribution to be shifted away from the minimum formed by the discriminator. Quickly removing the minimum which existed between X and X', and halting divergence.

In this way, there is competition between the speed with which the discriminator can minimize its value on the generator points, and the speed with which the generator distribution 'slips away' due to the effect of the NTK's off-diagonal components. The necessity for increased discriminator speed is hinted at by the positive slopes of the contours shown in Figure 2.8 – if α was increased, η had to be increased as well to avoid intensifying mode collapse.

To clarify this relationship, we will next perform experiments in which we explore the transition between convergence and mode collapse. We will be able to interpret this boundary to better understand the physical mechanism relating to the transition.

2.4 Phase Transition Experiments

The experiments of section 2.3 exemplified the phenomena of convergence and mode collapse, respectively. The first, with a diagonal NTK, showed rapid convergence to the desired data-distribution. The second, incorporating an NTK with a strong off-diagonal part, completely failed to converge, and instead displayed total mode collapse.

Here we wish to study the transition between these two phases. In order to do this, we use a generator NTK following the form of equation (2.42).

Toy models seemed to indicate that for greater generator off-diagonal NTK values (larger g_2/g_1), discriminators must be trained at a faster relative rate in order for the generator to reach convergence (as is indicated by the contours of Figure 2.8). We will therefore vary the rate at which the discriminator learns, as well as the NTK off-diagonal to diagonal ratio, g_2/g_1.

2.4.1 Experiment Setup

Sweeping Parameters

As we vary g_2/g_1, we must take care to control for the overall effect that the NTK has on total velocity. For example, if all points were initialized at the same location, X, then their velocity would obey

$$\frac{dX_i}{dt} = \alpha_G \frac{1}{N} \sum_j \Gamma(X_i, X_j) D'(X_j) = \alpha_G \frac{D'(X)}{N}(g_1 + g_2(N-1))$$

To control for the above effect, we vary g_2/g_1 while insisting that $g_1 + g_2(N-1) = \text{const.}$ We take $g_1 + g_2(N-1) = N$, so that $g_2/g_1 = 0 \implies g_2 = 0,\ g_1 = N$, and $g_2/g_1 = 1 \implies g_2 = g_1 = 1$.

We vary the learning rate of the discriminator by modifying the value of $n_{disc.}$ used in algorithm 2.

Measuring Mode Collapse

After a number of training steps (here we took 2,000 steps), we evaluate the entropy of the distribution of generator points. Defining P to be the distribution of modes nearest to the generator points[16], we record the quantity

$$\text{Mode Collapse Metric} = \log(8) + \sum_i P_i \log(P_i) \tag{2.45}$$

Zero here corresponds to even convergence to all the modes, while any higher value reveals some lack of convergence.

GAN Setup

To probe this, we train a model GAN with the following specifications:

[16]e.g. $P_i = 1/8$, $i = 1, 2, \cdots 8$ would be total coverage, while $P_1 = 1$, $P_{i>1} = 0$ would correspond to all generator points having the zeroth mode nearest to them

- Generator: A collection of 200 two-dimensional points initialized as a gaussian centered at $(\sqrt{2}, \sqrt{2})$, with standard deviation .1.
- Discriminator: a single hidden layer neural network of width 2048.

$$D(x) = a_i \sigma(w_i^j x^j + b_i) \tag{2.46}$$

We will apply both ReLU and Tanh activation functions. At initialization we take

$$w_i^j \sim \mathcal{N}(0, \sigma^2 = 1) \qquad a_i \sim \mathcal{N}(0, \sigma^2 = \frac{2}{\text{width}}) \qquad b_i \sim \mathcal{N}(0, \sigma^2 = 9)$$

The target data distribution is taken to be the 8 Gaussians used previously (shown in Figure 2.9), with modes set a distance 2 from the center, standard deviation = 0.02.

The loss function is then defined as

$$\mathcal{L} = \langle D(x) \rangle_{\text{target}} - \langle D(x) \rangle_{\text{gen.}} - \frac{1}{\text{width}} \left(\sum_{i,j} (w_i^j)^2 + \sum_i a_i^2 + \sum_i b_i^2 \right)$$

Where $\langle D(x) \rangle_{\text{gen.}}$ is the expectation of the discriminator on the generator distribution, while $\langle D(x) \rangle_{\text{target}}$ is its expectation on the data distribution (the eight Gaussians). The remaining terms represent an L_2 regularizer on the discriminator weights, placing an overall restriction on the discriminator.

An NTK following the form of (2.42) is introduced by hand into the dynamics of the generator by defining the point velocities as:

$$\frac{dX_i}{dt} = \alpha_G \Gamma_{i,j} D'(X_j) = \alpha_G \left(g_1 \frac{1}{N} D'(X_i) + g_2 \frac{1}{N} \sum_{j \neq i} D'(X_j) \right)$$

And the entire system is trained using algorithm 2.

Results

Collecting the mode collapse values (using equation (2.45)) after a number of steps, we plot the values below. Each point corresponds to the mode collapse measured after training the GAN for 2000 steps. Mode collapse values range from 0 (indicating convergence), to $\log 8 \approx 2.08$ (indicating total mode collapse).

On the x axis, 25 values of g_2/g_1 are swept over, from 0.01 to 1. For reasons that will become clear, for the ReLU discriminator these are taken to be values equidistant on a log scale. For a Tanh discriminator, these are spaced linearly in the same range. On the y-axis, we sweep over $n_{disc.}$, the number of discriminator training steps per generator training step. This assumes values from $n_{disc.} = 1$ to $n_{disc.} = 20$.

ReLU Activation

We first plot the mode collapse observed using a ReLU activation function in the discriminator. This is shown in Figure 2.14. Using a log-log plot, we note a linear transition – corresponding to a power law.

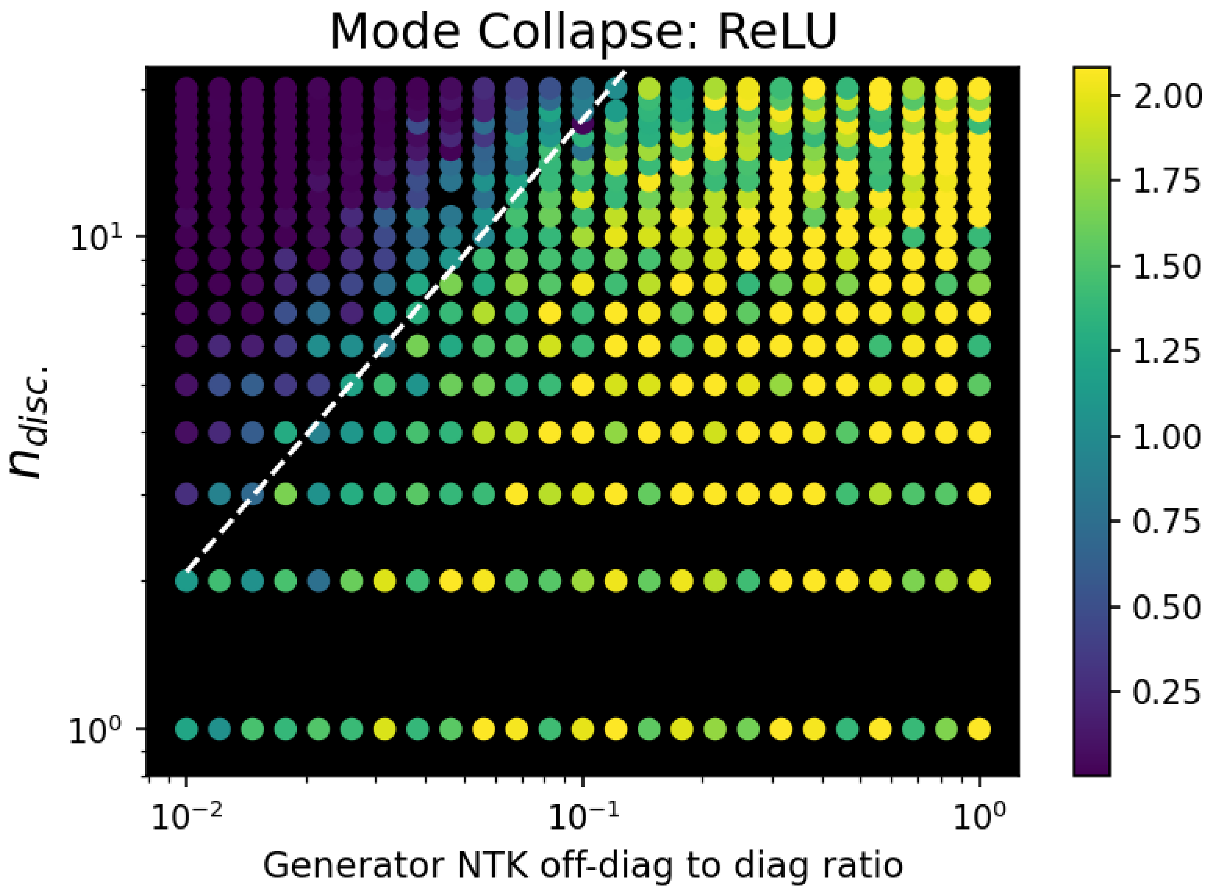


Figure 2.14: Here we use a ReLU discriminator to probe convergence, and train for 3000 steps. Brighter points indicate more mode-collapse, while darker points indicate convergence. The slope of the transition line is approximately 0.92 (see Figure 2.15 for a clearer plot of the transition).

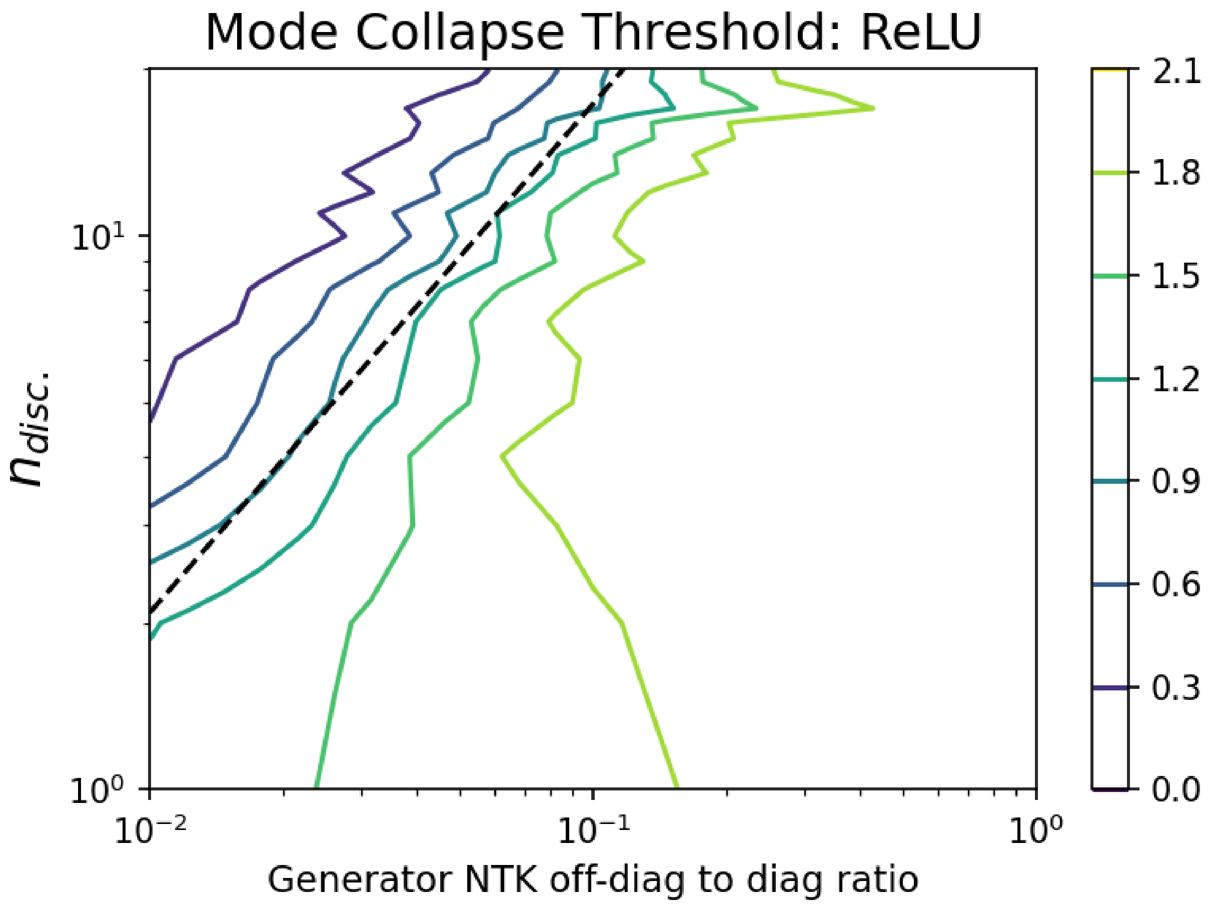


Figure 2.15: Here we obtain a transition by fitting the data using a sigmoid for each $n_{disc.}$ slice, and obtaining a linear fit of the for the halfway points of these sigmoids. This shows a power-law transition on our log-log plot.

Tanh Activation

In Figure 2.16 we plot the same experiment, replacing the activation function of the discriminator with Tanh. Again we can see a clear linear threshold distinguishing the two phases, this time apparent when plotted on a log-linear plot. This corresponds to an exponential transition.

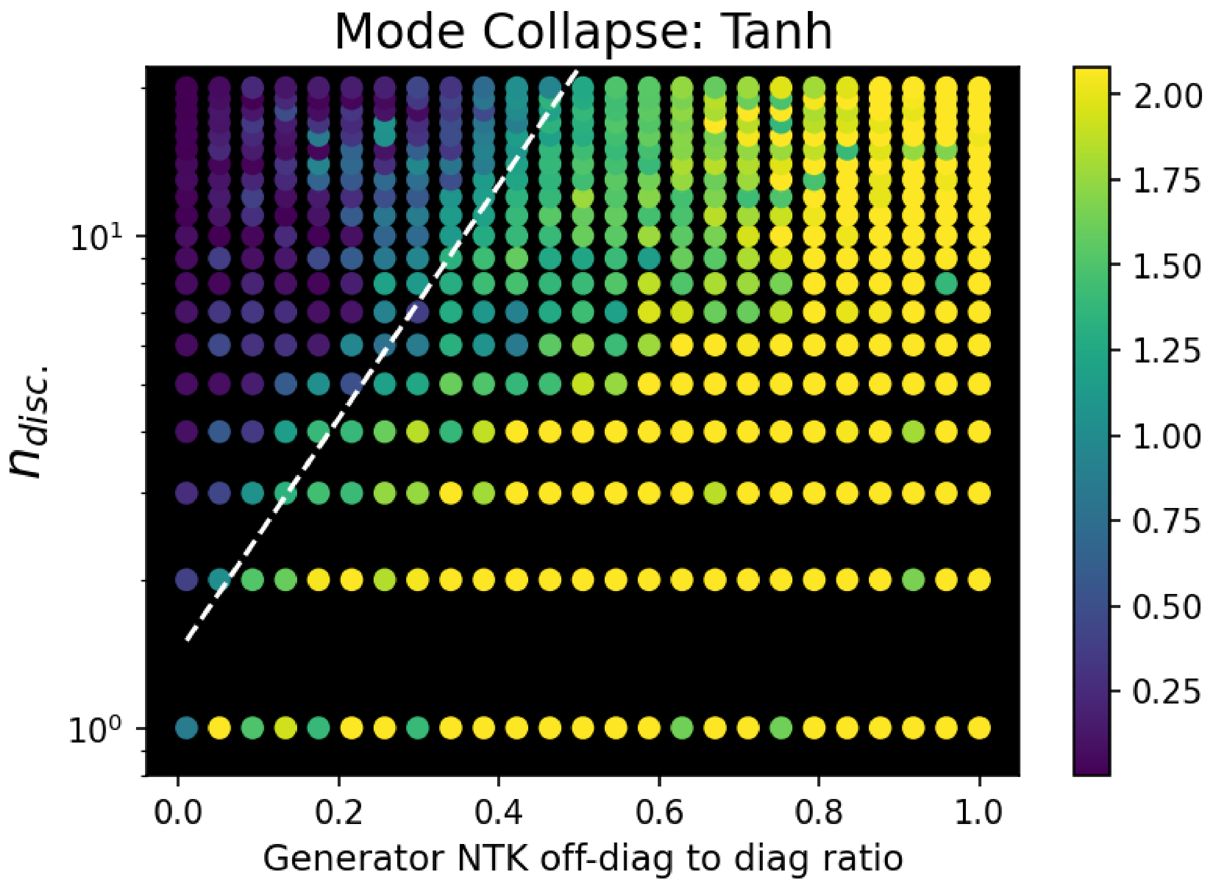


Figure 2.16: Here we show a scatter plot depicting the transition for a Tanh discriminator after 2000 steps. As in Figure 2.14, brighter points indicate mode collapse, and darker points indicate convergence. On this log-linear plot, an exponential transition is found (for details see Figure 2.17).

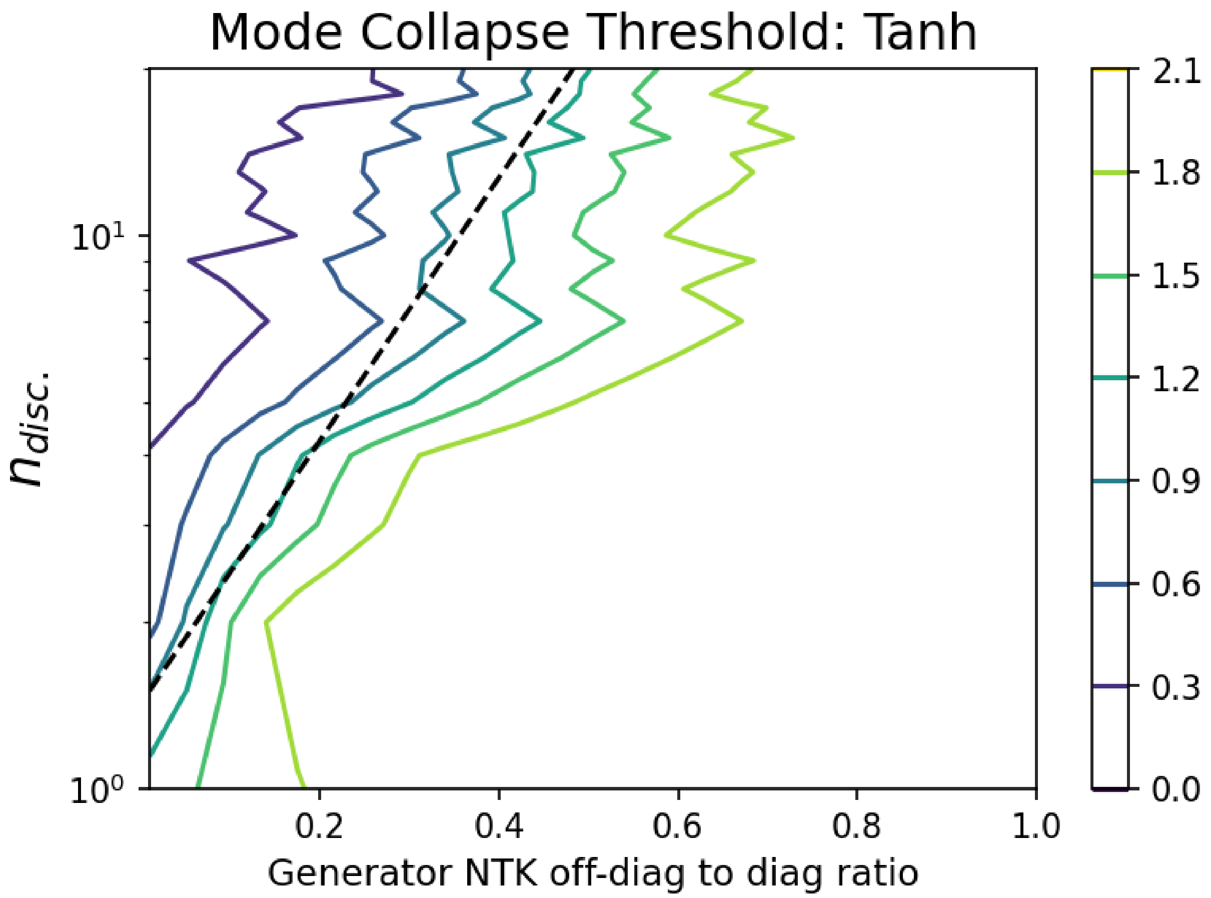


Figure 2.17: Above, a contour plot for the Tanh discriminator transition is obtained by fitting each $n_{disc.}$ slice to a sigmoid. The halfway points of these sigmoids are then used to find a linear fit for the transition. On this log-linear plot, this corresponds to an exponential boundary.

In both cases we see the same relationship we've observed before: the greater the off diagonal component of the generator NTK is, the greater the rate the discriminator needs in order to converge. As we will see, the exponential and power-law transitions are telling, as they help reveal the mechanism of this transition.

2.5 Interpretation and Analysis

In section 2.4, we identified a phase boundary separating GAN convergence from mode collapse using ReLU and Tanh networks. In ReLU networks, this boundary obeyed a power-law, while in Tanh networks, it followed an exponential. This suggestive result can be interpreted using a separate principle observed within neural networks.

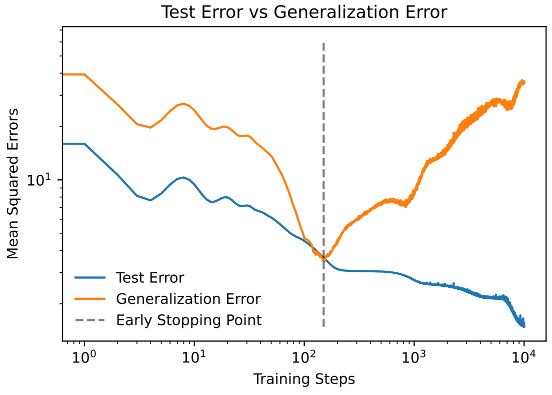
2.5.1 The F-Principle

In practice, neural networks tend to learn 'simple' patterns before learning more complex ones. This feature is useful as it allows practitioners to train neural networks on data containing noise. The network will learn the salient, basic structure of the data, before it overfits, and begins to memorize the complex specifics of the noise. By employing a regularization technique known as early stopping [GBC16], we can halt training before this occurs, improving the network's ability to generalize.

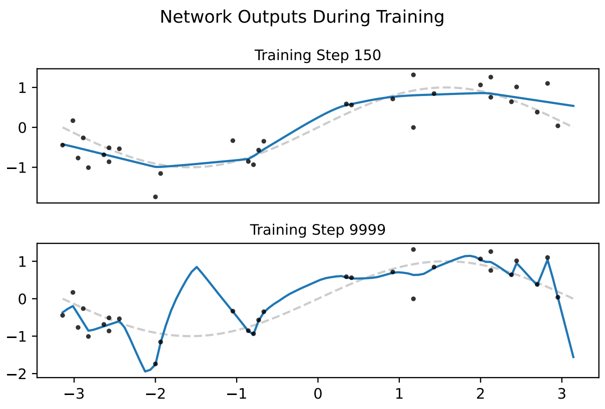
How one should quantify the concept of complexity, however, is not immediately obvious. One tool for characterizing spatial complexity which is ubiquitous in physics is the Fourier transform. Applying this approach, researchers have studied the rates at which neural networks learn data as a function of spatial frequency [XZX19, BJK19, ZLM21]. This direction of research has proved fruitful, as neural networks have been shown to learn Fourier components from lowest to highest, sometimes in analytically predicable ways. This can be characterized by identifying $\gamma(\boldsymbol{k})$, the rate at which a neural network learns a feature of spatial frequency $\boldsymbol{k}$. [17]

In particular, for wide neural networks containing a single hidden-layer, if the activation is ReLU, $\gamma(\boldsymbol{k})$ is expected to obey a *power-law* in $|\boldsymbol{k}|$. For Tanh activations, $\gamma(\boldsymbol{k})$ decays *exponentially* with $|\boldsymbol{k}|$ [ZLM21]. The notable correspondence between the behavior of $\gamma(\boldsymbol{k})$ predicted by the F-Principle, and the phase boundaries observed in Figures 2.15 and 2.17, is

[17] By this I mean that the error of fitting the $\boldsymbol{k}^{th}$ spatial mode of some target function: $f(x) = \sum_{\boldsymbol{k} c_{\boldsymbol{k}} \sin(\boldsymbol{k}\cdot x)}$ decays according to $\exp(-\gamma(\boldsymbol{k})t$

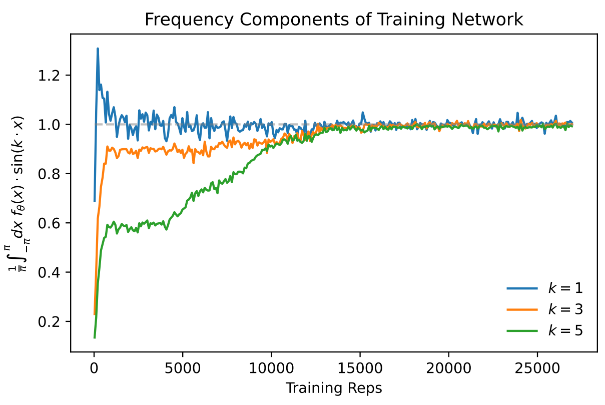


(a)

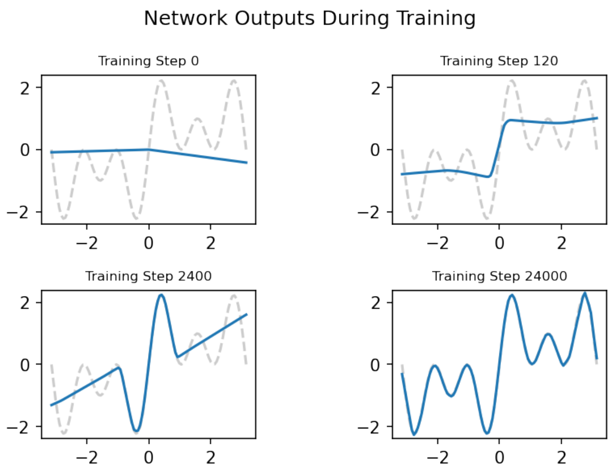


(b)

Figure 2.18: Here, a neural network is trained to fit a data-set which is sampled from a sine curve with noise then added: $\{x, \sin(x)+\mathcal{N}(0, 1/9)\}$. The neural network first learns the more simple, coarse-grained structure of data, at which point it generalizes well. Later, it moves on to memorizing the fine-grained structure of the noise. In figure (a) the mean squared error on the training-data is shown in blue, and the error on noise-less data (corresponding to a sine-curve prediction) is depicted in orange. These decrease together until around step 150, at which point the network begins memorizing the noise. Ideally, training would be halted at this point. Figure (b) shows the neural network at the early stopping point, when the network generalizes well and replicates coarse-grained sinusoidal behavior. Later, the network is shown to fit to the peculiarities of the noise, corresponding to poor generalization.



(a)



(b)

Figure 2.19: A neural network is trained to learn the function $f(x) = \sin(x)+\sin(3\cdot x)+\sin(5\cdot x)$. By evaluating the inner product of the neural network's output with each component (shown in (a)), we can see the learning rate slows as the frequency increases. On the right, in (b), we see the network learns less detailed features first, before eventually learning the full higher-frequency function.

no accident. As we will see, $\gamma(\boldsymbol{k})$ can be related directly to the form of the phase boundaries we have obtained.

2.5.2 Frequency-Based Mechanism

For the generator to cover all modes of the data distribution, the discriminator must be able to 'break apart' the generator's distribution of points. We would therefore like to understand, given a collection of N generator points with an NTK of

$$\Gamma_{i,1} = g_1 \quad \Gamma_{i\neq j} = g_2$$

when a discriminator D can break these apart?

To probe the physics of this, consider an extremely simplified 1d model in which a collection of N points are uniformly distributed within a region of length l, and define extremely simplified discriminator to have the form

$$D(x) = c|x|$$

(depicted in Figure 2.20).

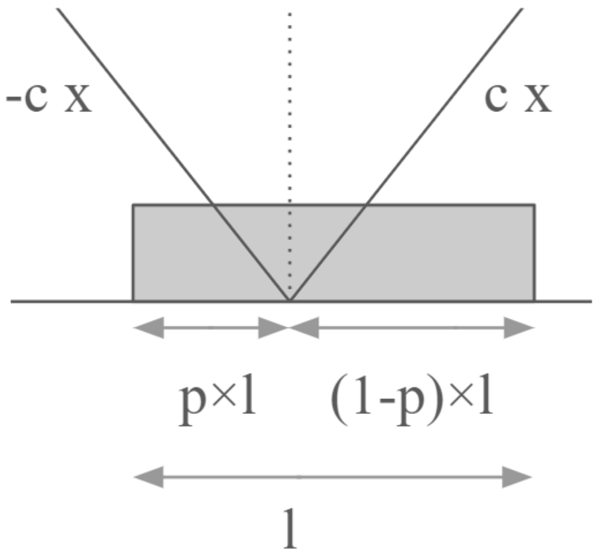


Figure 2.20: The generator's uniform distribution of length l, depicted shifted a distance $pl/2$, $p \in [0,1]$, to the right of the origin. The discriminator of the form $c|x|$ is superimposed.

From this starting point, we would like to answer the following question: **For what values of p do the generator points 'break apart'? – move in opposite directions on $x < 0$ and $x > 0$**

To answer this question, we consider a point $x_L < 0$, $x_L \in [-pl, 0)$, and a point $x_R > 0$, $x_R \in (0, (1-p)l]$. Suppose m of the N particles are to the left ($p = m/N$). Then, using the form of equation (2.44), the velocities of the points to the left and right will be

$$v_L = c \cdot \left(-\frac{(g_1 - g_2)}{N} + g_2(1-2p) \right)$$

$$v_R = c \cdot \left(\frac{(g_1 - g_2)}{N} + g_2(1-2p) \right)$$

These velocities satisfy

$$v_L < 0 \implies p > \frac{1}{2} - \frac{g_1 - g_2}{2g_2 N}$$

$$v_R > 0 \implies p < \frac{1}{2} + \frac{g_1 - g_2}{2g_2 N}$$

Therefore in order to 'split' the points, we require

$$\left(\frac{1}{2} - \frac{g_1}{2Ng_2} + \frac{1}{2N}\right) < p < \left(\frac{1}{2} + \frac{g_1}{2Ng_2} - \frac{1}{2N}\right)$$

This range of p tells us that for the discriminator to be able to split apart the distribution, we require the discriminator's minimum to be near the center of the generator distribution, with a *spatial precision* of order

$$l\left(\frac{g_1}{Ng_2} - N^{-1}\right)$$

Here if we take g_1/g_2 to be large, then the relevant frequency corresponding to this spacial precision is

$$k \sim \frac{Ng_2}{lg_1} \tag{2.47}$$

This implies that to break apart such a distribution, we require the discriminator to learn a spatial frequency proportional to g_2/g_1. Assuming the discriminator has a frequency learning rate of $\gamma(k)$, then the time required by the discriminator to learn such a feature scales like

$$T \sim \gamma(k)^{-1}$$

Plugging in an exponential learning rate, and using (2.47) gives us

$$T \sim \exp(\beta g_2/g_1)$$

Implying that the number of discriminator steps ($n_{disc.}$) necessary to split this distribution should scale like

$$n_{disc.} \sim \exp(\beta g_2/g_1)$$

for some constant β, matching the relationship observed in Figure 2.17.

The same line of argument using a power-law learning rate produces a prediction of

$$n_{disc.} \sim (g_2/g_1)^{\beta}$$

matching the threshold shown in Figure 2.15.

We therefore seem to be able to explain physically the phase transitions observed in experiment.

2.6 Discussion

In this Chapter, we have constructed toy models for simulating the training of GANs, allowing us to understand the convergence of GANs through the dynamics of correlated particles.

We applied concepts from high dimensional probability, and facts concerning the form of NTKs in real neural networks in order to define a simplified NTK for our toy model. By evolving points in data-space using a this NTK (using equation 2.32), we were able to model the learning dynamics of an infinite-width generator neural network *while ignoring the specifics of the network itself.*

Our model GAN allowed us to explore the relationship between discriminator training rate and the generator's NTK on the presence of mode collapse within model-GAN systems. We identified a boundary separating mode collapse from GAN convergence, and used physically motivated reasoning to explain the shape of this boundary.

Mode collapse is among the most common problems faced during GAN training [SVR17, CLJ17]. Understanding the source of this phenomenon, and finding ways to avoid it, is therefore of utmost importance to ML practitioners. Our model-GAN provides a playground in which to experiment with mode collapse, and study methods to improve convergence.

Beyond this, we have identified a connection between the discriminator used, and the shape of the phase boundary. Knowledge of this phase boundary could be used in practice, for instance, to infer the learning rate of a discriminator necessary to guarantee convergence.

Overall, we consider this analysis to be a promising starting point from which to study more realistic GAN models, including those with time-evolving generator NTKs, and more complex discriminators.

2.7 Appendix

2.7.1 The Effective Generator NTK

The generator neural network maps randomly selected seed points from the latent space to the data space, implicitly defining a probability distribution. During training, this distribution evolves.

To simplify this setup, we have used a sample of N points in data-space, $\{X_i\}$, as a proxy for the generator's implied distribution. Writing

$$X_i \equiv G_\theta(z_i)$$

these points evolve according to

$$\dot{X}_i = \frac{1}{N}\sum_{j=1}^{N} \frac{dG_\theta(z_i)}{d\theta}\frac{dG_\theta(z_j)}{d\theta} D'(G_\theta(z_j))$$

Where here, we have computed expectations using the sample of N points.

If, instead of using the previous sample of points to compute this, one uses an average over the distribution in latent space, $q(z)$, an output $X_i \equiv G_\theta(z_i)$ evolves according to

$$\dot{X}_i = \mathcal{N}_d^{-1} \int_{S^{d-1}} dz \frac{dG_\theta(z_i)}{d\theta}\frac{dG_\theta(z)}{d\theta} D'(G_\theta(z)) \tag{2.48}$$

Here we taken the distribution $q(z)$ to be a uniform sample over the unit sphere in d dimensions, and $\mathcal{N}_d$ to be the surface area of this sphere.

Note that if we take the NTK of the generator to be proportional to δ_{z_i,z_j}, then these two velocities are proportional:

$$\frac{1}{N}\frac{dG_\theta(z)}{d\theta}\frac{dG_\theta(z)}{d\theta}D'(G_\theta(z)) \propto \frac{1}{\mathcal{N}_d}\frac{dG_\theta(z)}{d\theta}\frac{dG_\theta(z)}{d\theta}D'(G_\theta(z))$$

However, if the NTK is not proportional, then proportionality is not guaranteed.

For example, take the NTK to be of the form

$$\Gamma(z, z') = \Gamma(\varphi)$$

where φ is the angle between the two seed vectors z and z'. Further, take $\gamma(\varphi)$ to be monotonically decreasing for $\phi \in [0, \pi/2]$.

For our sample of N particles, $1/N$ of the sample satisfied $\Gamma(z_i, z_j) = \Gamma(\varphi = 0)$ (namely, this is satisfied for all $i = j$). Meanwhile, using a sample over the entire latent space (equation (2.48)), this is not the case. As the dimension of the latent space, d, grows, a vanishingly small faction of the points z' on the unit sphere will satisfy $\varphi < \epsilon$.

Therefore by taking a sample of N points as a proxy for the full distribution of the GAN output, we will be overweighting the effect of the diagonal elements of the NTK. If N is taken to infinity, however, this problem disappears since the sample becomes asymptotically equal to the integral.

The above discussion implies some procedure to 'renormalize' the parameters of a network's NTK as we reduce the number of points, N, we take to represent its distribution. Such a renormalization will be the subject of future work.

2.7.2 Supporting the F-Principle Mechanism

Our physically motivated mechanism for the transition (described in section 2.5) makes use of the so-called frequency principle within neural networks to explain the shape of the phase boundary. In light of the stark contrast between the shapes of the ReLU and Tanh boundaries, which match the differences in their respective frequency learning rates, this connection appears very plausible.

We would like, however, to ensure that such a frequency relationship is *sufficient* on its own to create such power-law and exponential phase boundaries, since it is conceivable that some other property of the networks is responsible.

We note that our discriminators are very wide networks with a single hidden layer. In this large-width limit, it is expected to be approximately linear in parameters during training [LXS20]. Additionally, they are known to obey a given frequency principle. We therefore define a new discriminator which has these precise properties alone, and rerun the same experiment to observe the resulting phase boundary. If the same power-law and exponential phase boundaries are found, we can be much more confident in our analysis.

Define

$$D(x) = \sum_k D_k(x) \tag{2.49}$$

$$D_k(x) = w_k^{(1)} \sin(k \cdot x) + w_k^{(1)} \cos(k \cdot x) \tag{2.50}$$

Where $k = (k_1, k_2)$ and k_i are taken from 25 values of equal logarithmic spacing from [.01, 20], as well as the negatives of these values. $w_k^{(i)}$ are the weights of the model.

During training, we follow the routine of algorithm 3, a modification of algorithm 2, in which each $w_k^{(i)}$ is updated with a rate proportional to the value of a function, $\gamma(k)$. We then plug in power-law and exponential $\gamma(k)$ functions by hand, and run the same experiments performed in section 2.4. The power-law and exponential $\gamma(k)$ functions are defined below[18]:

$$\gamma_{\text{pow.}}(k) = \min(10^3, |k|^{-3}) \tag{2.51}$$

$$\gamma_{\text{exp.}}(k) = 668.8 \cdot \exp(-2.05 \cdot |k|) \tag{2.52}$$

[18]These functions were obtained by experimenting with the $\gamma(k)$ functions corresponding to real neural networks

Algorithm 3 The model-GAN training algorithm, with a Fourier-Discriminator and a frequency-dependent learning rate. A cloud of parameterized points is used as our simulated generator distribution.

```
for iteration number do
    for $n_{disc.}$ do
        • Sample $N$ data-points, $\{x_i\}$, from the 8-Gaussian distribution.
        • Compute

            $$\mathcal{L}_N = \frac{1}{N}\sum_{i=1}^{N} D(x_i) - \frac{1}{N}\sum_{i=1}^{N} D(X_i) - \frac{\lambda}{2}\sum_k \left((w_k^{(1)})^2 + (w_k^{(2)})^2\right)$$

        and update discriminator parameters by ascending its stochastic gradient

            $$w_k^{(i)} \leftarrow w_k^{(i)} + \alpha_D\ \gamma(k)\ \nabla_{w_k^{(i)}}\mathcal{L}_N$$

    end for
    • update $X_l$ according to equation (2.32)

        $$X_l \leftarrow X_l + \alpha_G\ \frac{1}{N}\sum_{l}^{N}\Gamma_{k,l}\nabla_x D(X_l)$$

end for
```

Note that our new routine essentializes the properties of the ReLU and Tanh discriminators by being linear in the parameters and explicitly learning frequency k features with a rate $\gamma(k)$.

The results of Figures 2.21 and 2.22 show an extremely clear phase boundary giving precisely the power-law and exponential behavior we had expected. This tells us that a frequency dependant learning rate is sufficient to produce the type of phase boundary we previously observed, and lends credence to the connection drawn between the onset of mode collapse, and the frequency principle of the discriminator network.

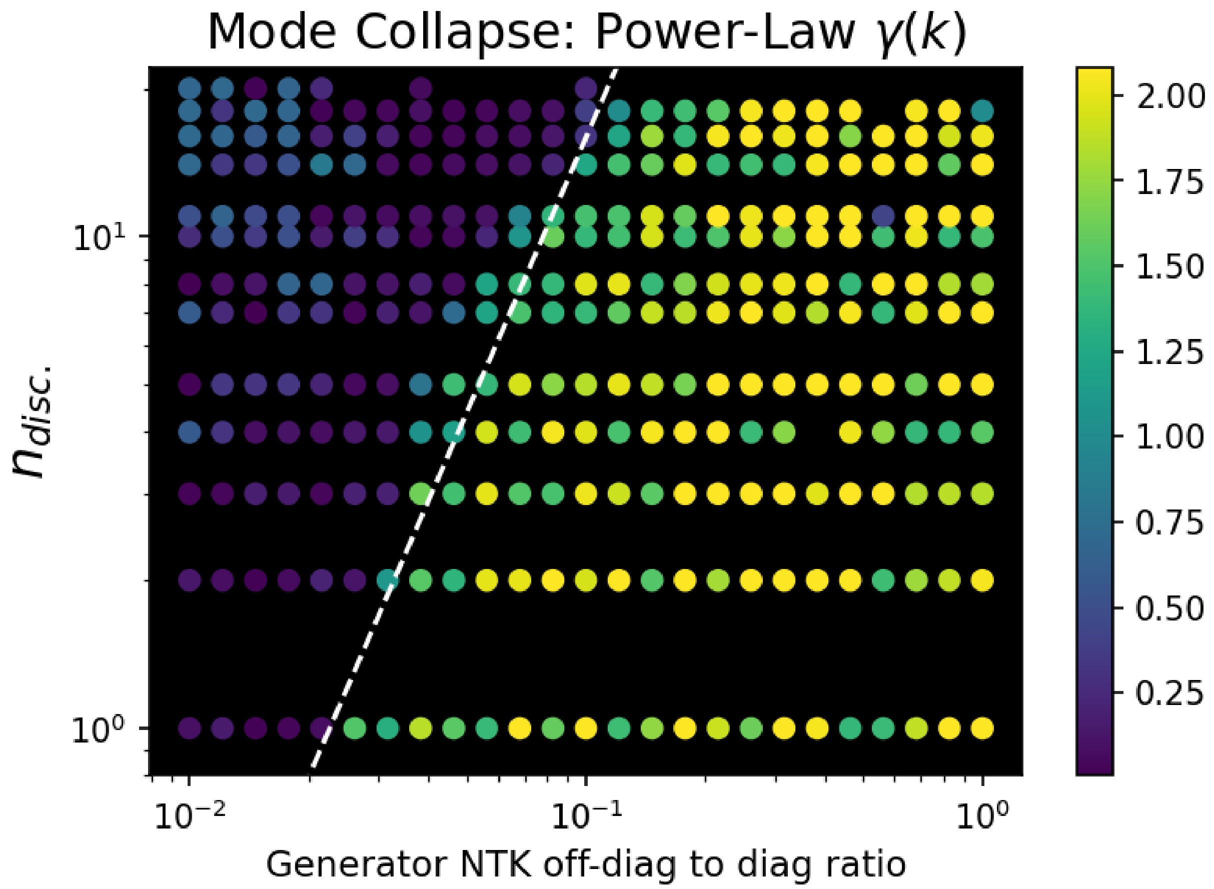


Figure 2.21: Here we show a scatter plot depicting the transition for a power-law $\gamma(k)$ after 3000 steps. Brighter points indicate mode collapse, and darker points indicate convergence. An extremely clear power-law boundary is found here, with a slope of ≈ 4.90.

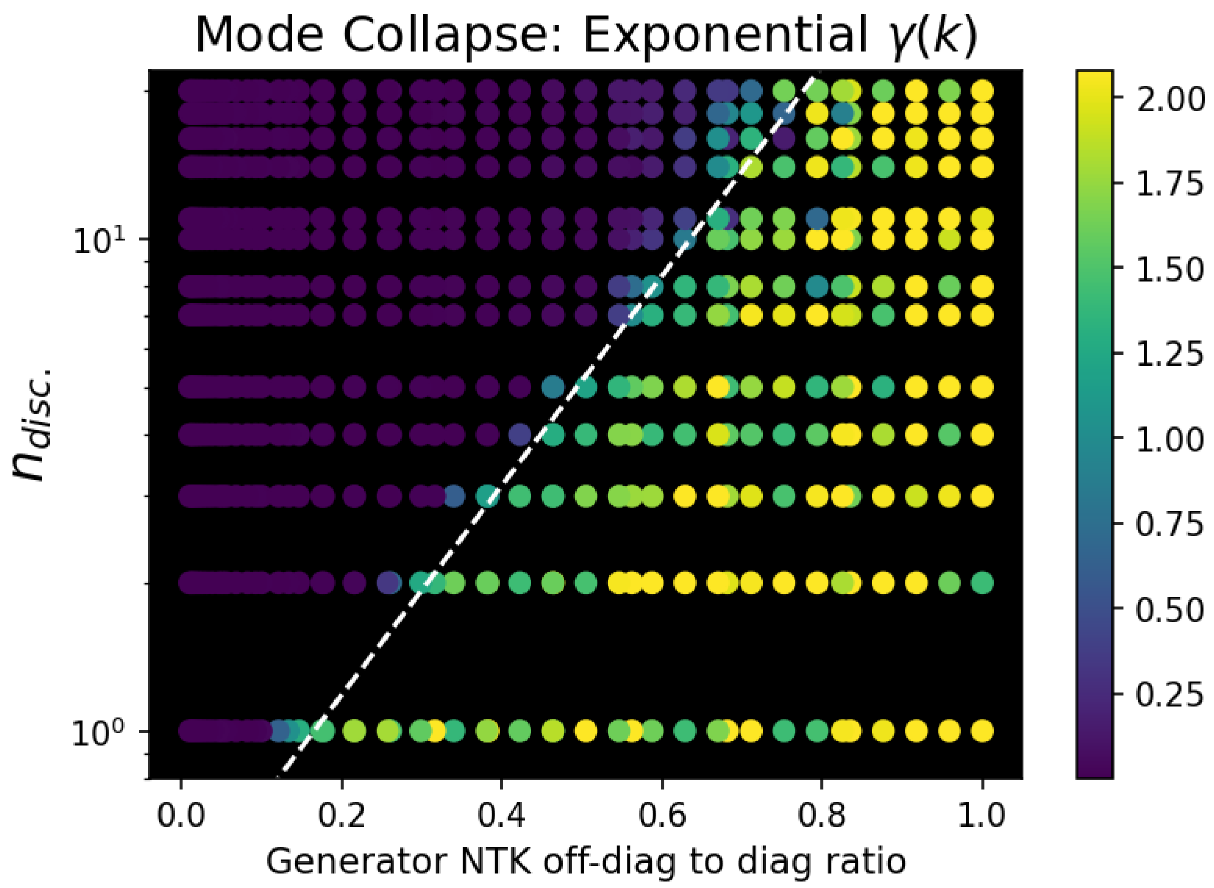
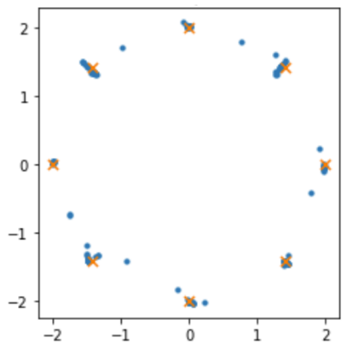


Figure 2.22: This scatter plot shows the transition for an exponential $\gamma(k)$ after 2000 steps. Brighter points indicate mode collapse, and darker points indicate convergence. Again a clear exponential boundary is found here, having a slope of ≈ 1.86.

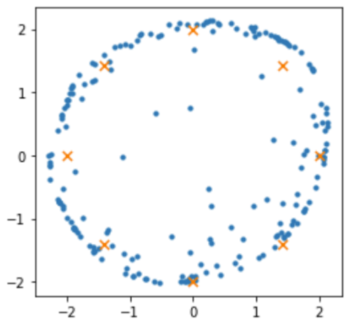
2.7.3 Generator Distributions across the Transition

To visualize the behavior of the generator points through the transition, here we plot the generator distributions for different g_2/g_1 values given a fixed $n_{disc.}$. This uses a ReLU discriminator, and the outputs of the experiment performed in section 2.4.

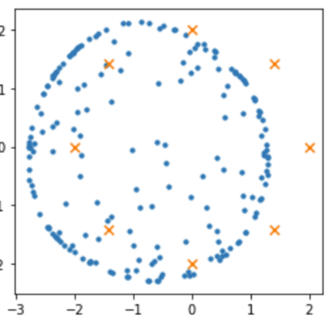
Taking $n_{disc.} = 6$, the transition here occurred roughly at $g_1/g_2 = 0.06$. We therefore show plots from below and above this value.



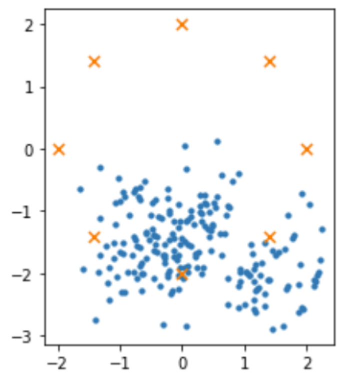
(a) $g_2/g_1 = 0.0001354$



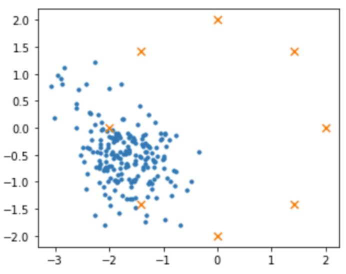
(b) $g_2/g_1 = 0.02610$



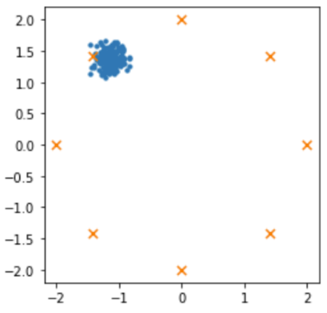
(c) $g_2/g_1 = 0.03831$



(d) $g_2/g_1 = 0.06813$



(e) $g_2/g_1 = 0.08254$



(f) $g_2/g_1 = 0.4642$

Figure 2.23: Generator outputs after 3000 steps. Note the full convergence for small g_2/g_1, while for $g_2/g_1 > 0.06$ the generator fails to converge.

REFERENCES

[AAB15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.", 2015. Software available from tensorflow.org.

[AAB19] Dmitry A. Abanin, Ehud Altman, Immanuel Bloch, and Maksym Serbyn. "Colloquium: Many-body localization, thermalization, and entanglement." *Rev. Mod. Phys.*, **91**:021001, May 2019.

[AC17] Vincenzo Alba and Pasquale Calabrese. "Entanglement and thermodynamics after a quantum quench in integrable systems." *Proceedings of the National Academy of Sciences*, **114**(30):7947–7951, 2017.

[AL18] Jonas Adler and Sebastian Lunz. "Banach Wasserstein GAN." In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[AMB21] Alankrita Aggarwal, Mamta Mittal, and Gopi Battineni. "Generative adversarial network: An overview of theory and applications." *International Journal of Information Management Data Insights*, **1**(1):100004, 2021.

[AMH19] Hiromu Araki, Tomonari Mizoguchi, and Yasuhiro Hatsugai. "Phase diagram of a disordered higher-order topological insulator: A machine learning study." *Phys. Rev. B*, **99**:085406, Feb 2019.

[BAA06] D.M. Basko, I.L. Aleiner, and B.L. Altshuler. "Metal–insulator transition in a weakly interacting many-electron system with localized single-particle states." *Annals of Physics*, **321**(5):1126–1205, 2006.

[BAT17] Peter Broecker, Fakher F. Assaad, and Simon Trebst. "Quantum phase recognition via unsupervised machine learning." *arXiv:1707.00663*, 2017.

[BCB15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate." *CoRR*, **abs/1409.0473**, 2015.

[BCM17] Peter Broecker, Juan Carrasquilla, Roger G. Melko, and Simon Trebst. "Machine learning quantum phases of matter beyond the fermion sign problem." *Scientific Reports*, **7**(1):8823, 2017.

[Bis96] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Inc., USA, 1996.

[BJK19] Ronen Basri, David Jacobs, Yoni Kasten, and Shira Kritchman. *The Convergence Rate of Neural Networks for Learned Functions of Different Frequencies.* Curran Associates Inc., Red Hook, NY, USA, 2019.

[BL82] R. N. Bhatt and P. A. Lee. "Scaling Studies of Highly Disordered Spin-$\frac{1}{2}$ Antiferromagnetic Systems." *Phys. Rev. Lett.*, **48**:344–347, Feb 1982.

[BM19] Alberto Bietti and Julien Mairal. "On the Inductive Bias of Neural Tangent Kernels." In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[BMR20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. "Language Models are Few-Shot Learners." In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020.

[BN13] Bela Bauer and Chetan Nayak. "Area laws in a many-body localized state and its implications for topological order." *Journal of Statistical Mechanics: Theory and Experiment*, **2013**, 06 2013.

[BSA18] Mikołaj Bińkowski, Dougal J. Sutherland, Michael Arbel, and Arthur Gretton. "Demystifying MMD GANs." In *International Conference on Learning Representations*, 2018.

[CCC19] Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. "Machine learning and the physical sciences." *Rev. Mod. Phys.*, **91**:045002, Dec 2019.

[CKV15] Anushya Chandran, Isaac H. Kim, Guifre Vidal, and Dmitry A. Abanin. "Constructing local integrals of motion in the many-body localized phase." *Phys. Rev. B*, **91**:085425, Feb 2015.

[CLJ17] Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. "Mode regularized generative adversarial networks." In *International Conference on Learning Representations (ICLR)*, volume abs/1612.02136, 2017.

[CM17] Juan Carrasquilla and Roger G. Melko. "Machine learning phases of matter." *Nature Physics*, **13**:431, Feb 2017.

[CT17] Giuseppe Carleo and Matthias Troyer. "Solving the quantum many-body problem with artificial neural networks." *Science*, **355**(6325):602–606, 2017.

[CTC] Sadi Carnot, Robert Henry Thurston, Robert Henry Carnot, and Baron Kelvin, William Thomson. *Reflections on the motive power of heat and on machines fitted to develop that power.* New York, J. Wiley, 1890. https://www.biodiversitylibrary.org/bibliography/17778.

[CVN19] C. Casert, T. Vieijra, J. Nys, and J. Ryckebusch. "Interpretable machine learning for inferring the phase boundaries in a nonequilibrium system." *Phys. Rev. E*, **99**:023304, Feb 2019.

[Deu91] J. M. Deutsch. "Quantum statistical mechanics in a closed system." *Phys. Rev. A*, **43**:2046–2049, Feb 1991.

[DM80] Chandan Dasgupta and Shang-keng Ma. "Low-temperature properties of the random Heisenberg antiferromagnetic chain." *Phys. Rev. B*, **22**:1305–1319, Aug 1980.

[ES03] D. M. Endres and J. E. Schindelin. "A new metric for probability distributions." *IEEE Transactions on Information Theory*, **49**(7):1858–1860, July 2003.

[Fis92] Daniel S. Fisher. "Random transverse field Ising spin chains." *Phys. Rev. Lett.*, **69**:534–537, Jul 1992.

[Fis94] Daniel S. Fisher. "Random antiferromagnetic quantum spin chains." *Phys. Rev. B*, **50**:3799–3821, Aug 1994.

[Fis95a] Daniel S. Fisher. "Critical behavior of random transverse-field Ising spin chains." *Phys. Rev. B*, **51**:6411–6461, Mar 1995.

[Fis95b] Daniel S. Fisher. "Critical behavior of random transverse-field Ising spin chains." *Phys. Rev. B*, **51**:6411–6461, Mar 1995.

[FT21] Yu Feng and Yuhai Tu. "The inverse variance&#x2013;flatness relation in stochastic gradient descent is critical for finding flat minima." *Proceedings of the National Academy of Sciences*, **118**(9):e2015617118, 2021.

[GBC16] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, Cambridge, MA, USA, 2016. `http://www.deeplearningbook.org`.

[GD17] Xun Gao and Lu-Ming Duan. "Efficient Representation of Quantum Many-body States with Deep Neural Networks." *Nature Communications*, **8**, 09 2017.

[GNR16] Scott D. Geraedts, Rahul Nandkishore, and Nicolas Regnault. "Many-body localization and thermalization: Insights from the entanglement spectrum." *Phys. Rev. B*, **93**:174202, May 2016.

[Got98] D Gottesman. "The Heisenberg representation of quantum computers." 6 1998.

[GPM14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Nets." In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[HDW18] Patrick Huembeli, Alexandre Dauphin, and Peter Wittek. "Identifying quantum phase transitions with adversarial neural networks." *Phys. Rev. B*, **97**:134109, Apr 2018.

[HGH20] Mohamed Hibat-Allah, Martin Ganahl, Lauren E. Hayward, Roger G. Melko, and Juan Carrasquilla. "Recurrent neural network wave functions." *Phys. Rev. Research*, **2**:023358, Jun 2020.

[HMS21] James Halverson, Anindita Maiti, and Keegan Stoner. "Neural networks and quantum field theory." *Machine Learning: Science and Technology*, **2**(3):035002, apr 2021.

[HN20] Boris Hanin and Mihai Nica. "Finite Depth and Width Corrections to the Neural Tangent Kernel." In *International Conference on Learning Representations*, 2020.

[HNO13] David A. Huse, Rahul Nandkishore, Vadim Oganesyan, Arijeet Pal, and S. L. Sondhi. "Localization-protected quantum order." *Phys. Rev. B*, **88**:014206, Jul 2013.

[HRU17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium." In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, p. 6629–6640, Red Hook, NY, USA, 2017. Curran Associates Inc.

[HSS17] Wenjian Hu, Rajiv R. P. Singh, and Richard T. Scalettar. "Discovering phases, phase transitions, and crossovers through unsupervised machine learning: A critical examination." *Phys. Rev. E*, **95**:062122, Jun 2017.

[HTF01] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning.* Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[HZR16] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[Imb16] John Imbrie. "On Many-Body Localization for Quantum Spin Chains." *Journal of Statistical Physics*, **163**, 06 2016.

[JAM18] Ryan Jadrich, B A. Lindquist, and T M. Truskett. "Unsupervised machine learning for detection of phase transitions in off-lattice systems. I. Foundations." *The Journal of Chemical Physics*, **149**:194109, 11 2018.

[JGH18] Arthur Jacot, Franck Gabriel, and Clément Hongler. "Neural Tangent Kernel: Convergence and Generalization in Neural Networks." In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, p. 8580–8589, Red Hook, NY, USA, 2018. Curran Associates Inc.

[Jö18] Mattias Jönsson. *"Detecting the Many-Body Localization Transition with Machine Learning Techniques."*. Master's thesis, 2018.

[KBP14] Jonas A. Kjäll, Jens H. Bardarson, and Frank Pollmann. "Many-Body Localization in a Disordered Quantum Ising Chain." *Phys. Rev. Lett.*, **113**:107204, Sep 2014.

[KCA14] Isaac H. Kim, Anushya Chandran, and Dmitry A. Abanin. "Local integrals of motion and the logarithmic lightcone in many-body localized systems." *arXiv e-prints*, p. arXiv:1412.3073, December 2014.

[KGE20] Marek Kowalski, Stephan J. Garbin, Virginia Estellers, Tadas Baltrušaitis, Matthew Johnson, and Jamie Shotton. "CONFIG: Controllable Neural Face Image Generation." In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pp. 299–315, Cham, 2020. Springer International Publishing.

[KLA21] T. Karras, S. Laine, and T. Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks." *IEEE Transactions on Pattern Analysis Machine Intelligence*, **43**(12):4217–4228, dec 2021.

[Kle03] Jon M. Kleinberg. "An Impossibility Theorem for Clustering." In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pp. 463–470. MIT Press, 2003.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[KST21] Bangalore Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad Sallab, Senthil Yogamani, and Patrick Perez. "Deep Reinforcement Learning for Autonomous Driving: A Survey." *IEEE Transactions on Intelligent Transportation Systems*, **PP**:1–18, 02 2021.

[LBD89] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, R. Howard, Wayne Hubbard, and Lawrence Jackel. "Handwritten Digit Recognition with a Back-Propagation Network." In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989.

[LBH15] Yann LeCun, Y. Bengio, and Geoffrey Hinton. "Deep Learning." *Nature*, **521**:436–44, 05 2015.

[LCC17] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. "MMD GAN: Towards Deeper Understanding of Moment Matching Network." In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, p. 2200–2210, Red Hook, NY, USA, 2017. Curran Associates Inc.

[LDH21] Xiao Liang, Shao-Jun Dong, and Lixin He. "Hybrid convolutional neural network and projected entangled pair states wave functions for quantum many-particle states." *Phys. Rev. B*, **103**:035138, Jan 2021.

[LLA15] David J. Luitz, Nicolas Laflorencie, and Fabien Alet. "Many-body localization edge in the random-field Heisenberg chain." *Phys. Rev. B*, **91**:081103, Feb 2015.

[LN18] Ye-Hua Liu and Evert P. L. van Nieuwenburg. "Discriminative Cooperative Networks for Detecting Phase Transitions." *Phys. Rev. Lett.*, **120**:176401, Apr 2018.

[LSC19] Yoav Levine, Or Sharir, Nadav Cohen, and Amnon Shashua. "Quantum Entanglement in Deep Learning Architectures." *Physical Review Letters*, **122**, 02 2019.

[LSZ15] Yujia Li, Kevin Swersky, and Richard Zemel. "Generative Moment Matching Networks." In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, p. 1718–1727. JMLR.org, 2015.

[LXS20] Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. "Wide neural networks of any depth evolve as linear models under gradient descent sup$*/sup.'' Journal of Statistical Mechanics : Theory and Experiment, 2020(12) : 124002, dec2020$.

[MBW18] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre Day, Clint Richardson, Charles Fisher, and David Schwab. "A high-bias, low-variance introduction to Machine Learning for physicists." *Physics Reports*, **810**, 03 2018.

[MCC19] Roger Melko, Giuseppe Carleo, Juan Carrasquilla, and J. Cirac. "Restricted Boltzmann machines in quantum physics." *Nature Physics*, **15**, 06 2019.

[MGN18] Lars M. Mescheder, Andreas Geiger, and Sebastian Nowozin. "Which Training Methods for GANs do actually Converge?" In *ICML*, 2018.

[MHA17] Leland McInnes, John Healy, and Steve Astels. "hdbscan: Hierarchical density based clustering." *The Journal of Open Source Software*, **2**(11), mar 2017.

[MIQ22] Xiao Mi, Matteo Ippoliti, Chris Quintana, Ami Greene, Zijun Chen, Jonathan Gross, Frank Arute, Kunal Arya, Juan Atalaya, Ryan Babbush, Joseph C. Bardin, Joao Basso, Andreas Bengtsson, Alexander Bilmes, Alexandre Bourassa, Leon Brill, Michael Broughton, Bob B. Buckley, David A. Buell, Brian Burkett, Nicholas Bushnell, Benjamin Chiaro, Roberto Collins, William Courtney, Dripto Debroy, Sean Demura, Alan R. Derk, Andrew Dunsworth, Daniel Eppens, Catherine Erickson, Edward Farhi, Austin G. Fowler, Brooks Foxen, Craig Gidney, Marissa Giustina, Matthew P. Harrigan, Sean D. Harrington, Jeremy Hilton, Alan Ho, Sabrina Hong, Trent Huang, Ashley Huff, William J. Huggins, L. B. Ioffe, Sergei V. Isakov, Justin Iveland, Evan Jeffrey, Zhang Jiang, Cody Jones, Dvir Kafri, Tanuj Khattar, Seon Kim, Alexei Kitaev, Paul V. Klimov, Alexander N. Korotkov, Fedor Kostritsa, David Landhuis, Pavel Laptev, Joonho Lee, Kenny Lee, Aditya Locharla, Erik Lucero, Orion Martin, Jarrod R. McClean, Trevor McCourt, Matt McEwen, Kevin C. Miao, Masoud Mohseni, Shirin Montazeri, Wojciech Mruczkiewicz, Ofer Naaman, Matthew Neeley, Charles Neill, Michael Newman, Murphy Yuezhen Niu, Thomas E. O'Brien, Alex Opremcak, Eric Ostby, Balint Pato, Andre Petukhov, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vladimir Shvarts, Yuan Su, Doug Strain, Marco Szalay, Matthew D. Trevithick, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Juhwan Yoo, Adam Zalcman, Hartmut Neven, Sergio Boixo, Vadim Smelyanskiy, Anthony Megrant, Julian Kelly, Yu Chen, S. L. Sondhi, Roderich Moessner, Kostyantyn Kechedzhi, Vedika Khemani, and Pedram Roushan. "Time-crystalline eigenstate order on a quantum processor." *Nature*, **601**(7894):531–536, Jan 2022.

[MJC] Davoud Moulavi, Pablo A. Jaskowiak, Ricardo J. G. B. Campello, Arthur Zimek, and Jörg Sander. *Density-Based Clustering Validation*, pp. 839–847.

[MKS15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. "Human-level control through deep reinforcement learning." *Nature*, **518**:529–533, 2015.

[MN21] Youssef Mroueh and Truyen V. Nguyen. "On the Convergence of Gradient Descent in GANs: MMD GAN As a Gradient Flow." In *AISTATS*, 2021.

[MS14] Pankaj Mehta and David J. Schwab. "An exact mapping between the Variational Renormalization Group and Deep Learning.", 2014.

[MSC13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. "Distributed Representations of Words and Phrases and Their Compositionality." In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, p. 3111–3119, Red Hook, NY, USA, 2013. Curran Associates Inc.

[MSR19] Youssef Mroueh, Tom Sercu, and Anant Raj. "Sobolev Descent." In *AISTATS*, 2019.

[NCT16] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. "f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization." In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[NLH17] Evert P. L. van Nieuwenburg, Ye-Hua Liu, and Sebastian D. Huber. "Learning phase transitions by confusion." *Nature Physics*, **13**:435, Feb 2017.

[ODR08] Maxim Olshanii, Vanja Dunjko, and Marcos Rigol. "Thermalization and its mechanism for generic quantum isolated systems." In *APS Division of Atomic, Molecular and Optical Physics Meeting Abstracts*, volume 39 of *APS Meeting Abstracts*, p. OPJ.40, May 2008.

[OH07] Vadim Oganesyan and David A. Huse. "Localization of interacting fermions at high temperature." *Phys. Rev. B*, **75**:155111, Apr 2007.

[OO17] Tomi Ohtsuki and Tomoki Ohtsuki. "Deep Learning the Quantum Phase Transitions in Random Electron Systems: Applications to Three Dimensions." *Journal of the Physical Society of Japan*, **86**(4):044708, 2017.

[PH10a] Arijeet Pal and David A. Huse. "Many-body localization phase transition." *Phys. Rev. B*, **82**:174411, Nov 2010.

[PH10b] Arijeet Pal and David A. Huse. "Many-body localization phase transition." *Phys. Rev. B*, **82**:174411, Nov 2010.

[PRA14a] David Pekker, Gil Refael, Ehud Altman, Eugene Demler, and Vadim Oganesyan. "Hilbert-Glass Transition: New Universality of Temperature-Tuned Many-Body Dynamical Quantum Criticality." *Phys. Rev. X*, **4**:011052, Mar 2014.

[PRA14b] David Pekker, Gil Refael, Ehud Altman, Eugene Demler, and Vadim Oganesyan. "Hilbert-Glass Transition: New Universality of Temperature-Tuned Many-Body Dynamical Quantum Criticality." *Phys. Rev. X*, **4**:011052, Mar 2014.

[PVG11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. “Scikit-learn: Machine Learning in Python.” *Journal of Machine Learning Research*, **12**:2825–2830, 2011.

[RS18] Joaquin F. Rodriguez-Nieva and Mathias S. Scheurer. “Identifying topological order via unsupervised machine learning.” *arXiv:1805.05961*, 2018.

[RYH22] Daniel A. Roberts, Sho Yaida, and Boris Hanin. *The Principles of Deep Learning Theory.* Cambridge University Press, 2022. `https://deeplearningtheory.com`.

[SBY17] Kevin Slagle, Zhen Bi, Yi-Zhuang You, and Cenke Xu. “Out-of-time-order correlation in marginal many-body localized systems.” *Phys. Rev. B*, **95**:165136, Apr 2017.

[ScA15] Maksym Serbyn, Z. Papić, and Dmitry A. Abanin. “Criterion for Many-Body Localization-Delocalization Phase Transition.” *Phys. Rev. X*, **5**:041047, Dec 2015.

[SHB15] Michael Schreiber, Sean S. Hodgman, Pranjal Bordia, Henrik P. Lüschen, Mark H. Fischer, Ronen Vosk, Ehud Altman, Ulrich Schneider, and Immanuel Bloch. “Observation of many-body localization of interacting fermions in a quasirandom optical lattice.” *Science*, **349**(6250):842–845, 2015.

[She93] David Sherrington. “Neural networks: the spin glass approach.” volume 51 of *North-Holland Mathematical Library*, pp. 261–291. Elsevier, 1993.

[SHM16] David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. “Mastering the game of Go with deep neural networks and tree search.” *Nature*, **529**:484–489, 01 2016.

[SN17] J. J. Sakurai and Jim Napolitano. *Modern Quantum Mechanics.* Cambridge University Press, 2 edition, 2017.

[Sre94] Mark Srednicki. “Chaos and quantum thermalization.” *Phys. Rev. E*, **50**:888–901, Aug 1994.

[SRN17] Frank Schindler, Nicolas Regnault, and Titus Neupert. “Probing many-body localization with neural networks.” *Phys. Rev. B*, **95**:245134, Jun 2017.

[SVR17] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U. Gutmann, and Charles Sutton. “VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning.” In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[SW66] J. R. Schrieffer and P. A. Wolff. "Relation between the Anderson and Kondo Hamiltonians." *Phys. Rev.*, **149**:491–492, Sep 1966.

[TXY20] Hao Tang, Dan Xu, Yan Yan, Philip H.S. Torr, and Nicu Sebe. "Local Class-Specific and Global Image-Level Generative Adversarial Networks for Semantic-Guided Scene Generation." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[VKK18] Jordan Venderley, Vedika Khemani, and Eun-Ah Kim. "Machine Learning Out-of-Equilibrium Phases of Matter." *Physical review letters*, **120 25**:257204, 2018.

[VSP17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. "Attention is All you Need." In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[Wan16] Lei Wang. "Discovering phase transitions with unsupervised learning." *Phys. Rev. B*, **94**:195105, Nov 2016.

[Wet17] Sebastian J. Wetzel. "Unsupervised learning of phase transitions: From principal component analysis to variational autoencoders." *Phys. Rev. E*, **96**:022140, Aug 2017.

[Wig55] Eugene P. Wigner. "Characteristic Vectors of Bordered Matrices With Infinite Dimensions." *Annals of Mathematics*, **62**(3):548–564, 1955.

[Wig57] Eugene P. Wigner. "Characteristics Vectors of Bordered Matrices with Infinite Dimensions II." *Annals of Mathematics*, **65**(2):203–207, 1957.

[WS17] Sebastian J. Wetzel and Manuel Scherzer. "Machine learning of explicit order parameters: From the Ising model to SU(2) lattice gauge theory." *Phys. Rev. B*, **96**:184410, Nov 2017.

[XZX19] Zhi-Qin John Xu, Yaoyu Zhang, and Yanyang Xiao. "Training Behavior of Deep Neural Network in Frequency Domain." In Tom Gedeon, Kok Wai Wong, and Minho Lee, editors, *Neural Information Processing*, pp. 264–274, Cham, 2019. Springer International Publishing.

[Yai19] Sho Yaida. "Fluctuation-dissipation relations for stochastic gradient descent." *ArXiv*, **abs/1810.00004**, 2019.

[YQX16] Yi-Zhuang You, Xiao-Liang Qi, and Cenke Xu. "Entanglement holographic mapping of many-body localized system by spectrum bifurcation renormalization group." *Phys. Rev. B*, **93**:104205, Mar 2016.

[ZK17] Yi Zhang and Eun-Ah Kim. "Quantum Loop Topography for Machine Learning." *Phys. Rev. Lett.*, **118**:216401, May 2017.

[ZLM21] Yaoyu Zhang, Tao Luo, Zheng Ma, and Zhi-Qin John Xu. "A Linear Frequency Principle Model to Understand the Absence of Overfitting in Neural Networks." *Chinese Physics Letters*, **38**(3):038701, mar 2021.

[ZMK17] Yi Zhang, Roger G. Melko, and Eun-Ah Kim. "Machine learning $\mathbb{Z}_2$ quantum spin liquids with quasiparticle statistics." *Phys. Rev. B*, **96**:245119, Dec 2017.