Bridging the Neural Divide: Geometrical Insights and Practical Approaches to Machine Learning Interpretability

Mehrab Hamidi

School of Computer Science

McGill University, Montreal

December 2, 2024

A thesis submitted to McGill University in partial fulfillment

of the requirements of the degree of

Master of Computer Science

Abstract

This thesis explores the critical importance of interpretability in machine learning, bridging the gap between theoretical understanding and practical application. We address the current lack of theoretical work on the geometrical analysis of neural networks, emphasizing the need for deeper insights into their topological and geometrical structures. Chapter 3 introduces a novel approach to reverse engineering deep ReLU networks through an optimization-based method, providing a practical means of reconstructing these networks from limited information and enhancing our theoretical understanding of their internal mechanisms. Chapter 4 studies the geometric transformations that data undergoes within neural networks, focusing on aspects such as boundedness, angle degeneracy, and data manifold distortion. This chapter offers detailed mathematical analyses and proofs, contributing to the robustness, discriminative power, and optimization of neural network architectures. Chapter 5 applies these interpretability techniques to a real-life model in reinforcement learning, specifically the VPT agent in Minecraft. By visualizing attention head weights, feature visualizations, and manipulating input streams, we gain valuable insights into the decision-making processes of the agent. This thesis underscores the necessity of both theoretical and practical interpretability, aiming to enhance the transparency, reliability, and performance of machine learning models, and facilitating their broader adoption across various domains.

Abrégé

Cette thèse explore l'importance cruciale de l'interprétabilité dans l'apprentissage automatique, comblant le fossé entre la compréhension théorique et l'application pratique. Nous abordons le manque actuel de travaux théoriques sur l'analyse géométrique des réseaux de neurones, en soulignant la nécessité d'approfondir nos connaissances sur leurs structures topologiques et géométriques. Le chapitre 3 introduit une nouvelle approche pour l'ingénierie inverse des réseaux ReLU profonds à travers une méthode basée sur l'optimisation, fournissant un moyen pratique de reconstruire ces réseaux à partir d'informations limitées et améliorant notre compréhension théorique de leurs mécanismes internes. Le chapitre 4 étudie les transformations géométriques que les données subissent au sein des réseaux de neurones, en se concentrant sur des aspects tels que la délimitation, la dégénérescence angulaire et la distorsion du manifold de données. Ce chapitre offre des analyses et des preuves mathématiques détaillées, contribuant à la robustesse, au pouvoir discriminatif et à l'optimisation des architectures de réseaux de neurones. Le chapitre 5 applique ces techniques d'interprétabilité à un modèle réel en apprentissage par renforcement, spécifiquement l'agent VPT dans Minecraft. En visualisant les poids des têtes d'attention, les visualisations de caractéristiques et en manipulant les flux d'entrée, nous obtenons des informations précieuses sur les processus de prise de décision de l'agent. Cette thèse souligne la nécessité d'une interprétabilité à la fois théorique et pratique, visant à améliorer la transparence, la fiabilité et la performance des modèles d'apprentissage automatique, facilitant ainsi leur adoption plus large dans divers domaines.

Acknowledgements

I would like to express my deepest gratitude to my amazing supervisor and advisor, David Rolnick, for his unwavering support and guidance over the past two years. His expertise and encouragement have been invaluable to my growth and success, and I am profoundly thankful to him.

I also want to extend my heartfelt thanks to my wonderful collaborators: Kathryn Lindsey, Elisenda Grigsby, and Karolis Ramanauskas. Their insights and contributions have greatly enriched this work, and I am grateful for their partnership.

A special thank you goes to my Mom and Dad, Nazee and Shahram. Despite the distance between us, their constant support and the sacrifices they have made have been a source of strength and motivation for me. I could not have achieved this milestone without them.

Lastly, I want to thank my lovely and amazing wife Nilly for her unwavering support during these past few challenging months. Her presence in my life has been a beacon of hope and encouragement. Without her, I would not have been able to complete this work, and I am incredibly thankful to her.

Table of Contents

List of Figures

Contribution of Authors

This thesis presents a comprehensive study of mechanistic interpretability in neural networks, combining theoretical developments with practical applications. The main contributions of this thesis are as follows:

In Chapter 3, we propose a novel approach to reverse engineering deep ReLU networks through an optimization-based method. This chapter is solely written by the author. The preliminary results and proofs presented, which have not been previously cited, are new contributions by the author.

In Chapter 4, we present both theoretical and experimental analyses of neural network geometry. The experimental work in this chapter is solely conducted by the author. The theoretical results are primarily contributed by collaborators Kathryn Lindsey and Julia Elisenda Grigsby.

Chapter 5 focuses on practical interpretability by applying interpretability techniques to a real-world model: the VPT agent in reinforcement learning for Minecraft. The introduction and related work sections on attention are from a published work. The attention visualization part has been conducted by collaborator Karolis Ramanauskas, while the remaining sections are solely authored by the author.

Chapter 1

Introduction

The importance of interpretability in machine learning cannot be overstated, serving as a critical bridge between theoretical understanding and practical application. Dr. David M. Kaufman [8] underscores the role of reflection in deepening one's grasp of concepts, emphasizing that mere theoretical knowledge is insufficient for true comprehension. Mechanical interpretability, which facilitates the connection between new theoretical information and an individual's existing knowledge base, is essential for achieving this deeper understanding. Adding to this, David S. Watson [61] highlights a major conceptual challenge in interpretable machine learning (IML): the tendency to focus on static explanatory products rather than dynamic, iterative processes of explanation. This approach may hinder IML progress by overlooking the need for adaptable explanations that respond to varying inputs. Theoretical interpretability addresses this by ensuring explanations are not only accurate but also consistent across different contexts, fostering robust understanding.

In recent years, machine learning, and particularly deep learning, has achieved remarkable success across a wide range of applications, from image recognition and natural language processing to game playing and autonomous systems. These advancements have been driven by increasingly complex models, such as deep neural networks, which are capable of capturing intricate patterns in large datasets. However, as these models be-

come more complex, understanding their internal workings becomes more challenging, leading to a pressing need for interpretability.

Interpretability in machine learning is crucial for several reasons. It enhances trust in model predictions, facilitates debugging and improvement of models, and is essential for ensuring ethical and responsible use of AI systems, especially in critical applications like healthcare, finance, and autonomous vehicles. Interpretability serves as a bridge between theoretical understanding and practical application, enabling practitioners to comprehend, explain, and justify the decisions made by machine learning models.

Despite the importance of interpretability, much of the existing work focuses on developing practical tools and methods to explain model predictions, often in a post-hoc manner. These include techniques like saliency maps, feature importance measures, and example-based explanations. While valuable, these methods sometimes lack a solid theoretical foundation, and their explanations may not fully capture the underlying mechanisms of the models.

There is a notable gap in theoretical understanding of the geometric and topological properties of neural networks, particularly regarding how data representations transform through the layers of a network. This gap limits our ability to rigorously analyze and interpret the behavior of complex models, and to develop principled methods for improving their interpretability.

In this thesis, we aim to bridge this gap by exploring both theoretical and practical aspects of interpretability in neural networks. We focus on understanding the geometric transformations that data undergoes within neural networks, particularly those employing ReLU activations, and on applying interpretability techniques to real-world models in practical settings.

1.1 Contributions of this work

The main contributions of this thesis are as follows:

- We develop theoretical frameworks for analyzing the geometric and topological transformations induced by deep ReLU networks. By using notions such as convex hull dimensions and max cell dimensions, we provide mathematical tools for quantifying the complexity and structural properties of neural networks at different layers.

- We propose an optimization-based method for reverse engineering deep ReLU networks from limited information. This approach advances the theoretical understanding of deep ReLU networks and offers practical methods for their reconstruction, shedding light on their internal workings.

- We analyze the geometric transformations that data undergoes as it passes through neural networks, focusing on aspects such as boundedness of data representations, angle degeneracy, and data manifold distortion. Through detailed mathematical proofs, we aim to enhance network robustness and optimize data transformations.

- We apply interpretability techniques to a real-world agentic model, specifically the VPT agent in the Minecraft environment. By visualizing attention mechanisms, feature visualizations, and manipulating inputs, we gain practical insights into how the agent makes decisions in complex, dynamic settings.

- Throughout the thesis, we highlight the gap between theoretical explainability and empirical interpretability, emphasizing the need for integrated approaches that advance both theoretical understanding and practical tools for interpretability.

1.2 Thesis Organization

The thesis is organized as follows:

- **Chapter 3** introduces our optimization-based method for reverse engineering deep ReLU networks. We build upon existing research on the complexity of linear regions

and activation patterns in deep ReLU networks, providing theoretical insights and an algorithm for model reconstruction.

- **Chapter 4** follows the previous chapter, and delves into the geometric transformations within neural networks. We examine how data manifolds are distorted through layers of ReLU activations, and we provide mathematical analyses on boundedness, angle degeneracy, and distortion metrics. This chapter aims to bridge theoretical understanding with practical implications for network design and performance.

- **Chapter 5** transitions to practical interpretability, focusing on the VPT agent in the Minecraft environment. We employ various interpretability techniques to understand the agent's decision-making processes, highlighting the challenges and limitations of applying theoretical concepts to complex, real-world models.

- **Chapter 6** concludes the thesis by summarizing our findings, discussing the implications of our work, and suggesting directions for future research in bridging the gap between theoretical and practical interpretability.

Chapter 2

Background and Some Preliminary Results

2.1 Polyhedral Complex

Definition 2.1.1. *A polyhedral set P is defined as a set in $\mathbb{R}^n$ that can be represented as the intersection of a finite number of half-spaces. Each half-space can be mathematically described by a linear inequality, which in general form is:*

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq b,$$

where $a_1, a_2, \ldots, a_n$ and $b \in \mathbb{R}$, and $x_1, x_2, \ldots, x_n \in \mathbb{R}^n$. We denote these closed affine half spaces by $H_1^+, \ldots, H_m^+ \subseteq \mathbb{R}^n$.

Definition 2.1.2. *A convex polytope is defined as the convex hull of a finite set of points in $\mathbb{R}^n$, known as the vertices [26]. Mathematically, it can be expressed as:*

$$P = conv\{v_1, v_2, \ldots, v_k\},$$

Alternatively, a convex polytope can be defined as a bounded polyhedral sets [13]. This duality in definition relates to the polytope being representable either by its vertices (vertex representation)

or by its defining half-spaces (facet representation):

$$P = \bigcap_{i=1}^{m} \{x \in \mathbb{R}^n : a_i^\top x \leq b_i\},$$

where $a_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$ for each i.

Lemma 2.1.3. *Polyhedral sets are convex.*

Proof. First, we prove the convexity of half spaces. Consider any two points $x, y \in H_i$ for a particular half-space H_i. We need to show that the point $z = tx + (1-t)y$ also belongs to H_i for any $t \in [0, 1]$. Since $x, y \in H_i$, we have:

$$a_i^\top x \leq b_i \quad \text{and} \quad a_i^\top y \leq b_i.$$

By the linearity of the dot product and properties of scalar multiplication and addition in inequalities:

$$a_i^\top z = a_i^\top (tx + (1-t)y) = t(a_i^\top x) + (1-t)(a_i^\top y) \leq tb_i + (1-t)b_i = b_i.$$

Therefore, $z \in H_i$.

We already know that the intersection of convex sets is convex. If $x, y \in P$, then $x, y \in H_i$ for all i. For any $t \in [0, 1]$, since $tx + (1-t)y \in H_i$ for each i (as shown above), it follows that:

$$tx + (1-t)y \in \bigcap_{i=1}^{m} H_i = P.$$

□

Definition 2.1.4. *A hyperplane H in $\mathbb{R}^n$ is defined as a **supporting hyperplane** [26] for a polyhedral set P if it meets the following conditions:*

- *H does not intersect the interior of P.*
- *The intersection of H and P is non-empty, i.e., $H \cap P \neq \emptyset$.*

This means that the hyperplane touches the polyhedral set but does not pass through its interior, effectively supporting P along one of its faces.

Notation 2.1.5. A hyperplane H is considered a **cutting hyperplane** [26] for a polyhedral set P under the following condition:

- The hyperplane intersects the interior of the polyhedral set, i.e., $H \cap \text{interior}(P) \neq \emptyset$.

This indicates that the hyperplane cuts through P, dividing it into separate non-empty parts.

Definition 2.1.6. *The affine hull of a set S in $\mathbb{R}^n$, is defined as the smallest affine subspace containing S. It is given by:*

$$\textit{aff}(S) = \left\{ \sum_{i=1}^{k} \lambda_i x_i \mid \sum_{i=1}^{k} \lambda_i = 1, \lambda_i \in \mathbb{R} \right\},$$

where $x_1, x_2, \ldots, x_k$ are points in S. This set includes all affine combinations of the points in S. The dimension of a polyhedral set P with respect to its affine hull is the dimension of the affine hull, aff(P).

$$\dim(P) = \dim(\textit{aff}(P)).$$

Definition 2.1.7. *A face F [27] of a polyhedral set P is a subset of P that can be represented as:*

$$F = P \cap H,$$

where H is a supporting hyperplane. Alternatively F can be described as

$$F = \{x \in P : a^T x = b\}$$

where $a^\top x \leq b$ is the equation of the supporting hyperplane H.

Some take from this definition:

- $P, \emptyset$ are called improper faces of P, while other faces are called proper.

- The dimension of a face is given by the dimension of its affine hull.
- A **k-face** of P is a face of P that has dimension k.
- A **facet** of P is a $(dim(P) - 1)$-face of P and a **vertex** of P is a 0-face of P and an edge of P is a 1-face of P.
- Every face of a face of P is also a face of P.
- Faces constitute parts of the boundary of P, with each face having its own relative interior that does not intersect with the interior of any other distinct face.

Definition 2.1.8. *Let P be a polyhedron in $\mathbb{R}^n$. The interior of P, denoted as $\mathrm{int}(P)$, is defined with respect to its affine hull, $\mathrm{aff}(P)$. The affine hull of P is the smallest affine space containing P.*

The interior of P is defined as:

$$\mathrm{int}(P) = \{x \in P : \textit{there exists an open set } U \textit{ in } \mathrm{aff}(P) \textit{ such that } x \in U \subseteq P\}.$$

This definition implies that the interior of P is considered under the topology induced by its affine hull, $\mathrm{aff}(P)$.

The affine hull 2.1.6 of a polyhedron $\mathrm{aff}(P)$ represents the smallest affine space that can completely encompass P. It is defined by a linear combination of points in P, adjusted by adding a constant vector.

For example, consider a polyhedron formed by a cube in $\mathbb{R}^3$. The interior of this cube includes all points inside the cube but not on the faces, edges, or vertices. If P reduces to a single point or a line segment, the interior under traditional $\mathbb{R}^n$ topology might be empty, but within their respective affine hulls, these structures can have non-empty interiors as described.

Definition 2.1.9. *A **polyhedral complex** [28] $\mathcal{C}$ in $\mathbb{R}^n$ is defined as a **finite** set that satisfies the following criteria:*

1. *Each element in $\mathcal{C}$ is a convex polytopes, called **cells of** $\mathcal{C}$.*

2. *For any $P, Q \in \mathcal{C}$, their intersection $P \cap Q$ is a face of both P and Q. This intersection may be an empty set.*

3. *For every $P \in \mathcal{C}$ every face of P must also be included in $\mathcal{C}$.*

The dimension of $\mathcal{C}$ is defined as the maximum dimension of any polyhedron within $\mathcal{C}$. The union of all polyhedra in $\mathcal{C}$ can form a complex structure, serving various applications from tiling spaces to forming bases for complex geometrical models. [27] refers to these as (geometric) topological cell complexes, but we rather use the polyhedral complex.

Notation 2.1.10. The domain or underlying set $|\mathcal{C}|$ of a polyhedral complex $\mathcal{C}$ is the union of its cells. If $|\mathcal{C}| = \mathbb{R}^n$, we call $\mathcal{C}$ polyhedral decomposition of $\mathbb{R}^n$ [25].

2.2 Hyperplane Algebra

In this section, we discuss the algebraic properties of hyperplanes and their intersections, which are fundamental in understanding the geometric structures induced by neural networks, particularly those using the ReLU activation function.

Definition 2.2.1 (Hyperplane)**.** *A* hyperplane *in $\mathbb{R}^n$ is an affine subspace of dimension $n-1$ (codimension 1), defined as*

$$H = \left\{ x \in \mathbb{R}^n \mid b^\top x = \beta \right\},$$

where $b \in \mathbb{R}^n$ is a non-zero vector (the normal vector to the hyperplane) and $\beta \in \mathbb{R}$.

The intersection of two hyperplanes in $\mathbb{R}^n$, when they are not parallel, is an affine subspace of dimension $n-2$. We formalize this concept and provide a method to compute the intersection explicitly.

Proposition 2.2.2. *Let H_1 and H_2 be two hyperplanes in $\mathbb{R}^n$, defined by*

$$H_1: \quad a_1^\top x = b_1,$$
$$H_2: \quad a_2^\top x = b_2,$$

where $a_1, a_2 \in \mathbb{R}^n$ are non-zero, non-colinear vectors (i.e., a_1 and a_2 are not scalar multiples of each other), and $b_1, b_2 \in \mathbb{R}$. Then, the intersection $H_1 \cap H_2$ is an affine subspace of dimension $n-2$, given by

$$H_1 \cap H_2 = x_{part} + \mathcal{N}(A),$$

where x_{part} is a particular solution to the system $Ax = b$, $A \in \mathbb{R}^{2\times n}$ is the matrix with rows $a_1^\top$ and $a_2^\top$, $b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$, and $\mathcal{N}(A)$ is the null space of A.

Proof. The intersection $H_1 \cap H_2$ consists of all $x \in \mathbb{R}^n$ satisfying both equations:

$$\begin{bmatrix} a_1^\top \\ a_2^\top \end{bmatrix} x = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}.$$

Let $A = \begin{bmatrix} a_1^\top \\ a_2^\top \end{bmatrix} \in \mathbb{R}^{2\times n}$ and $b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$. Since a_1 and a_2 are not colinear, A has rank 2.

The general solution to the system $Ax = b$ is given by

$$x = x_{\text{part}} + \mathcal{N}(A),$$

where x_{part} is any particular solution to $Ax = b$, and $\mathcal{N}(A)$ is the null space of A, which has dimension $n-2$. □

To find a particular solution x_{part}, we can use the method of least squares or find the solution with minimal Euclidean norm (the least-norm solution). Assuming that a_1 and a_2 are normalized such that $\|a_1\| = \|a_2\| = 1$, we can compute x_{part} as follows.

Proposition 2.2.3. *The least-norm solution to the system $Ax = b$ is given by*

$$x_{LN} = A^\top \left(AA^\top\right)^{-1} b.$$

Proof. Since A has full rank (rank 2), the matrix $AA^\top \in \mathbb{R}^{2\times 2}$ is positive definite and invertible. The least-norm solution minimizes $\|x\|$ subject to $Ax = b$, and is given by the pseudoinverse:

$$x_{\text{LN}} = A^\top \left(AA^\top\right)^{-1} b.$$

□

Given that $\|a_1\| = \|a_2\| = 1$ and denoting $\theta = \arccos(a_1^\top a_2)$, we can express $AA^\top$ and its inverse explicitly:

$$AA^\top = \begin{bmatrix} 1 & \cos\theta \\ \cos\theta & 1 \end{bmatrix}, \quad \left(AA^\top\right)^{-1} = \frac{1}{\sin^2\theta} \begin{bmatrix} 1 & -\cos\theta \\ -\cos\theta & 1 \end{bmatrix}.$$

Therefore,

$$\begin{aligned} x_{\text{LN}} &= A^\top \left(AA^\top\right)^{-1} b \\ &= \frac{1}{\sin^2\theta} \left(b_1 - b_2\cos\theta\right) a_1 + \frac{1}{\sin^2\theta} \left(b_2 - b_1\cos\theta\right) a_2. \end{aligned}$$

The null space $\mathcal{N}(A)$ consists of all vectors $x \in \mathbb{R}^n$ satisfying $Ax = 0$. Since A has rank 2, $\dim \mathcal{N}(A) = n - 2$. We can find a basis for $\mathcal{N}(A)$ as follows.

Let $P \in \mathbb{R}^{n\times n}$ be a permutation matrix that reorders the columns of A such that the leading 2×2 submatrix is invertible. Then, we can write $AP^\top = \begin{bmatrix} I_2 & F \end{bmatrix}$, where $F \in \mathbb{R}^{2\times(n-2)}$. The null space of A is then

$$\mathcal{N}(A) = \left\{ P \begin{bmatrix} -F^\top \\ I_{n-2} \end{bmatrix} \tau \;\middle|\; \tau \in \mathbb{R}^{n-2} \right\}.$$

Combining the particular solution and the null space, the intersection $H_1 \cap H_2$ is given by

$$H_1 \cap H_2 = \left\{ x_{\text{LN}} + P \begin{bmatrix} -F^\top \\ I_{n-2} \end{bmatrix} \tau \;\middle|\; \tau \in \mathbb{R}^{n-2} \right\}.$$

2.3 Neural Networks

Definition 2.3.1 (Rectified Linear Unit)**.** *The rectified linear unit (ReLU) function is defined as $ReLU : \mathbb{R} \to \mathbb{R}$, where $ReLU(x) = \max\{0, x\}$. For any $n \in \mathbb{N}$, we denote by $\sigma : \mathbb{R}^n \to \mathbb{R}^n$ the nonlinearity that applies ReLU to each coordinate.*

Definition 2.3.2 (Feedforward ReLU Neural Network)**.** *A feedforward ReLU neural network of architecture $(n_0, n_1, \ldots, n_m) \in \mathbb{N}^{m+1}$ is an ordered collection of affine maps $A^1, A^2, \ldots, A^m$, where each map $A^i : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$ for $i = 1, \ldots, m$. Each affine map A^i can be represented by a weight matrix $W^i \in \mathbb{R}^{n_i \times n_{i-1}}$ and a bias vector $b^i \in \mathbb{R}^{n_i}$, such that for $x \in \mathbb{R}^{n_{i-1}}$,*

$$A^i(x) = W^i x + b^i.$$

Alternatively, using augmented matrices, we have

$$A^i(x) = [W^i \; b^i] \begin{bmatrix} x \\ 1 \end{bmatrix}.$$

The neural network defines a function $F : \mathbb{R}^{n_0} \to \mathbb{R}^{n_m}$ given by

$$F(x) = (\sigma \circ A^m) \circ (\sigma \circ A^{m-1}) \circ \cdots \circ (\sigma \circ A^1)(x),$$

where σ is the ReLU activation function applied coordinate-wise.

Definition 2.3.3 (Parameter Space)**.** *The parameter space of neural networks with architecture* $(n_0, n_1, \ldots, n_m)$ *is the Euclidean space*

$$\mathcal{P}(n_0, n_1, \ldots, n_m) = \mathbb{R}^{D(n_0, n_1, \ldots, n_m)},$$

where the total dimension $D(n_0, n_1, \ldots, n_m)$ *is given by*

$$D(n_0, n_1, \ldots, n_m) = \sum_{i=1}^{m} n_i(n_{i-1} + 1).$$

Consider a piecewise linear function from $\mathbb{R}^{n_0}$ to $\mathbb{R}^{n_m}$. Let $PL(\mathbb{R}^{n_0} \to \mathbb{R}^{n_m})$ denote the set of all such piecewise linear functions. From previous work [2] we know that the class of feedforward ReLU neural network functions coincides with the class of finite piecewise linear (PL) functions. Hence, we define this relationship as a realization map as follows:

Definition 2.3.4 (Realization Map)**.** *The realization map is a function*

$$\phi : \mathcal{P}(n_0, n_1, \ldots, n_m) \to PL(\mathbb{R}^{n_0} \to \mathbb{R}^{n_m}),$$

which maps the network parameters to the corresponding piecewise linear function defined by the neural network.

Definition 2.3.5 (Bounded Piecewise Linear Functions)**.** *Let* $M, T > 0$ *be given constants. Define the set of bounded piecewise linear functions*

$$F_T = \left\{ f \in PL(\mathbb{R}^{n_0} \to \mathbb{R}^{n_m}) \;\middle|\; \|f(x)\| \leq M \text{ for all } x \in B_T(0) \right\},$$

where $B_T(0)$ *denotes the ball of radius* T *centered at the origin in* $\mathbb{R}^{n_0}$*.*

Definition 2.3.6 (Function Distance)**.** *For functions* $f, g : \mathbb{R}^{n_0} \to \mathbb{R}^{n_m}$*, the* L^p *distance is defined as*

$$d_p(f, g) = \left(\int_{\mathbb{R}^{n_0}} \|f(x) - g(x)\|^p dx \right)^{1/p},$$

where $\|\cdot\|$ denotes the Euclidean norm in $\mathbb{R}^{n_m}$.

We now aim to define the realization map

$$\phi : \mathbb{R}^D \to F_T,$$

which maps the parameter space $\mathbb{R}^D$ to the set F_T of bounded piecewise linear functions.

Proposition 2.3.7. *Under the conditions outlined in [19], the pair (F_T, d_p) is a compact metric space without the need for additional assumptions on the parameter space.*

Proof. We consider the results from [19], where the authors prove that for sufficiently large m (the width of the neural network layer) with respect to the Gaussian mean width of a data set contained in the closed unit ball $B_1(0) \subseteq \mathbb{R}^n$, and a randomly initialized layer map F_ℓ, with high probability, the squared distance between the images of two input data points $x, y \in \mathbb{R}^n$ is well approximated by

$$\|F_\ell(x) - F_\ell(y)\|^2 \approx \frac{1}{2}\|x-y\|^2 + \|x\|\|y\|\psi(x,y),$$

where

$$\psi(x,y) := \frac{1}{\pi}\left(\sin\theta - \theta\cos\theta\right),$$

and $\theta \in [0,\pi]$ is the angular distance between x and y.

Since $x, y \in B_1(0)$, we have $\|x\|, \|y\| \leq 1$. Moreover, as shown in Figure 2 of [19], the function $\psi(x,y)$ satisfies $0 \leq \psi(x,y) \leq 1$ when $\theta \in [0,\pi]$. Therefore, with high probability,

$$\frac{1}{2}\|x-y\|^2 \leq \|F_\ell(x) - F_\ell(y)\|^2 \leq \frac{1}{2}\|x-y\|^2 + \|x\|\|y\| \leq \frac{1}{2}\|x-y\|^2 + 1.$$

The lower bound is achieved when $\theta = 0$, and the upper bound is achieved when $\theta = \pi$.

The key insight is that for sufficiently wide networks, the distance between points in the unit ball with small angular distance is decreased, and the distance between points with large angular distance is preserved.

Now, we proceed to formalize this result to show that the functions in F_T have uniformly bounded Lipschitz constants.

The ReLU activation function σ is 1-Lipschitz, i.e.,

$$|\sigma(u) - \sigma(v)| \leq |u - v|, \quad \forall u, v \in \mathbb{R}.$$

This property extends to vector-valued inputs:

$$\|\sigma(a) - \sigma(b)\| \leq \|a - b\|, \quad \forall a, b \in \mathbb{R}^m,$$

where $\|\cdot\|$ denotes the Euclidean norm.

Consider an affine transformation $A^i(x) = W^i x + b^i$, where $W^i \in \mathbb{R}^{m \times n}$ and $b^i \in \mathbb{R}^m$. For any $x, y \in \mathbb{R}^n$, we have

$$\|A^i(x) - A^i(y)\| = \|W^i x + b^i - (W^i y + b^i)\| = \|W^i(x - y)\|.$$

Applying the ReLU activation, we obtain

$$\|\sigma(A^i(x)) - \sigma(A^i(y))\| \leq \|A^i(x) - A^i(y)\| = \|W^i(x - y)\|.$$

Assume that W^i is a random matrix whose entries are independent standard normal random variables, i.e., $w^i_{jk} \sim \mathcal{N}(0, 1)$. Then, for any fixed vector $v \in \mathbb{R}^n$, the random vector $W^i v$ has entries that are independent Gaussian random variables with mean zero and variance $\|v\|^2$. Specifically,

$$\mathbb{E}\left[\|W^i v\|^2\right] = m\|v\|^2.$$

By the Johnson–Lindenstrauss lemma, for any $\epsilon \in (0,1)$ and for m sufficiently large (e.g., $m = O(\log N/\epsilon^2)$ for N data points), the following holds with high probability:

$$(1-\epsilon)\|x-y\|^2 \leq \frac{1}{m}\|W^i(x-y)\|^2 \leq (1+\epsilon)\|x-y\|^2.$$

Thus, the random projection approximately preserves distances up to a scaling factor.

After applying the ReLU activation function, since it sets negative components to zero, approximately half of the components of $W^i(x-y)$ will be zeroed out (assuming Gaussianity). Therefore,

$$\mathbb{E}\left[\|\sigma(W^i(x-y))\|^2\right] = \frac{1}{2}\mathbb{E}\left[\|W^i(x-y)\|^2\right] = \frac{m}{2}\|x-y\|^2.$$

This implies that

$$\|\sigma(W^i x + b^i) - \sigma(W^i y + b^i)\| \leq \|W^i(x-y)\| \leq \sqrt{(1+\epsilon)m}\|x-y\|,$$

and

$$\|\sigma(W^i x + b^i) - \sigma(W^i y + b^i)\| \geq \sqrt{\frac{(1-\epsilon)m}{2}}\|x-y\|,$$

with high probability. Combining the above inequalities, we find that the mapping F_ℓ satisfies

$$L_1\|x-y\| \leq \|F_\ell(x) - F_\ell(y)\| \leq L_2\|x-y\|,$$

where

$$L_1 = \sqrt{\frac{(1-\epsilon)m}{2}}, \quad L_2 = \sqrt{(1+\epsilon)m}.$$

Therefore, the Lipschitz constant of F_ℓ is bounded above by L_2.

Since the Lipschitz constants of the functions F_ℓ are uniformly bounded due to the properties of the random weights, the set of functions $\{F_\ell\}$ is equicontinuous. Along with the uniform boundedness (since $\|F_\ell(x)\| \leq M$ for some $M > 0$), we can apply the

Arzelà–Ascoli theorem to conclude that the set F_T is relatively compact in the space of continuous functions on $B_T(0)$ with the uniform norm.

As convergence in the uniform norm implies convergence in the L^p norm on compact domains, F_T is also relatively compact in $(L^p(B_T(0), \mathbb{R}^{n_m}), d_p)$. Since L^p spaces are complete, the closure of F_T is compact. Therefore, the pair (F_T, d_p) is a compact metric space without the need for additional assumptions on the parameter space.

□

Definition 2.3.8 (Network Isomorphism)**.** *Two neural networks $\mathcal{N}$ and $\mathcal{N}'$ with the same input space $\mathbb{R}^{n_0}$ are said to be functionally equivalent if they produce the same output for all inputs, i.e.,*

$$\mathcal{N}(x) = \mathcal{N}'(x), \quad \forall x \in \mathbb{R}^{n_0}.$$

Definition 2.3.9. *The affine solution set arrangement [25] associated to a layer map $\sigma \circ A : \mathbb{R}^i \to \mathbb{R}^j$ of a feedforward ReLU neural network is the finite ordered set $\{S_1, \ldots, S_j\}$, where each $S_k, 1 \leq k \leq j$, is the solution set defined by:*

$$S_k := \{\mathbf{x} \in \mathbb{R}^i : [A]_k(\mathbf{x}|1)^T = 0\},$$

where $[A]_k$ is the k-th row of the matrix $[A]$. Such an affine solution set arrangement $S = \{S_1, \ldots, S_j\}$ is said to be generic if for all subsets

$$\{S_{i_1}, \ldots, S_{i_p}\} \subseteq S,$$

*it is the case that $S_{i_1} \cap \ldots \cap S_{i_p}$ is an affine-linear subspace of $\mathbb{R}^i$ of dimension $i-p$, where a negative-dimensional intersection is understood to be empty. In the particular case that every solution set $S_k, 1 \leq k \leq j$, is a hyperplane (i.e., has codimension 1) in $\mathbb{R}^i$, we call $S = \{S_1, \ldots, S_j\}$ the **hyperplane arrangement** [25] associated to that layer map. A hyperplane arrangement in $\mathbb{R}^i$ induces a polyhedral decomposition of $\mathbb{R}^i$.*

When each S_k is a hyperplane, the collection of all S_k forms a hyperplane arrangement. This arrangement divides the entire input space $\mathbb{R}^i$ into regions (polyhedral sets), each corresponding to different activation patterns across the neurons in the layer. This polyhedral decomposition is central to understanding how different input vectors are processed and classified by the network. The concept of hyperplane arrangements and affine solution sets is crucial for visualizing and analyzing the behavior of neural networks, particularly how inputs are segregated by the network layers. This geometric perspective helps in understanding the decision boundaries formed by the network and can be useful in tasks like feature visualization, model simplification, and theoretical analysis of network capacities and limitations.

Definition 2.3.10. *[25] Given a feedforward ReLU neural network defined by a sequence of affine maps $A_1, A_2, \ldots, A_m$, where each $A_i : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$ for $i = 1, \ldots, m$, and the ReLU activation function σ applied component-wise, the **canonical polyhedral complex** [23] $C(s)$ associated with the network function F represented by the network parameters s is constructed as follows:*

For each layer i from 1 to m, let R_i denote the polyhedral complex on $\mathbb{R}^{n_{i-1}}$ induced by the hyperplane arrangement associated with the layer map $F_i = \sigma \circ A_i$. This arrangement partitions $\mathbb{R}^{n_{i-1}}$ into polyhedral sets each defined by the activation patterns of the neurons in layer i.

The canonical polyhedral complex $C(s)$, or $C(F)$ where F is the neural network function from $\mathbb{R}^{n_0}$ to $\mathbb{R}^{n_m}$, is then defined by inductively constructing polyhedral complexes $C(F_1), C(F_2 \circ F_1), \ldots, C(F_m \circ \ldots \circ F_1)$ on $\mathbb{R}^{n_0}$ as follows:

- *Set $C(F_1) = R_1$.*
- *For $i = 2, \ldots, m$, define $C(F_i \circ \ldots \circ F_1)$ as the set*

$$\{S \cap (F_{i-1} \circ \ldots \circ F_1)^{-1}(Y) : S \in C(F_{i-1} \circ \ldots \circ F_1), Y \in R_i\}.$$

Here, S is a cell in the polyhedral complex $C(F_{i-1} \circ \ldots \circ F_1)$ on $\mathbb{R}^{n_0}$. Y is a cell in R_i, which is a partition of $\mathbb{R}^{n_{i-1}}$.

Finally, set $C(s) = C(F_m \circ \ldots \circ F_1)$, which represents the polyhedral decomposition of $\mathbb{R}^{n_0}$ induced by the entire network mapped through all its layers.

[43] proposed another way to construct canonical polyhedral complex, which they call it backward construction, as folllow:

1. *Begin with the output layer of the network. Let $C(F(m)) = R(m)$, where $R(m)$ denotes the region in the output space determined by the last layer's activations.*

2. *For each preceding layer $k = m, m-1, \ldots, 1$, construct $C(F(k-1))$ based on $C(F(k))$ by considering the preimages of regions under the affine map F_k followed by the ReLU activation. Specifically, define:*

$$C(F(k-1)) = \{R \cap F_{k-1}^{-1}(C) : R \in R(k-1), C \in C(F(k))\} \tag{2.1}$$

   where $R(k-1)$ represents the polyhedral complex induced by the $(k-1)$-th layer, considering the intersections of hyperplanes that describe the zeros of the layer's activation functions.

3. *This construction proceeds iteratively until the input layer is reached, culminating in $C(F(1))$, which fully describes the canonical polyhedral complex of the network.*

The complexity of the canonical polyhedral complex, such as the number and arrangement of its polyhedral cells, can give insights into the expressiveness and capacity of the neural network. A more complex arrangement suggests a higher capacity to discriminate between different types of input data, reflecting potentially higher complexity and variance the network can handle. The boundaries between different polyhedral cells in the complex represent the decision boundaries of the network. These are the thresholds at which slight changes in input values can lead to different neuron activations and, consequently, different outputs. Studying these boundaries can help in understanding the sensitivity and specificity of the network to various inputs.

Lemma 2.3.11. *Forward and backward construction of a canonical polyhedral complex are indeed equivalent.*

Proof. Let F denote a feedforward ReLU neural network consisting of a sequence of layers, each defined by an affine transformation followed by a ReLU activation. For layer i, the affine transformation is represented by $A_i : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$, and the overall network function is denoted as $F : \mathbb{R}^{n_0} \to \mathbb{R}^{n_m}$.

The forward construction $C_{\text{forward}}(F)$ starts from the input and partitions the input space based on the activation patterns determined by the affine transformations and ReLU functions of each layer. The backward construction $C_{\text{backward}}(F)$ starts from the output, considering preimages of these regions under the mappings of preceding layers, tracing how outputs map back to input configurations.

Define a mapping $\phi : C_{\text{forward}}(F) \to C_{\text{backward}}(F)$ that aligns regions defined by activation sequences in both constructions. We utilize the distributivity of function preimages over intersections:

$$F^{-1}\left(\bigcap_{i=1}^{k} R_i\right) = \bigcap_{i=1}^{k} F^{-1}(R_i),$$

where R_i are regions defined at any layer in the network. Distinct sequences of activations define distinct intersections of regions in $C_{\text{forward}}(F)$, thus mapping to unique regions in $C_{\text{backward}}(F)$ via ϕ, hence it's injective. Every region in $C_{\text{backward}}(F)$ corresponds to a unique sequence of activations from $C_{\text{forward}}(F)$, ensuring that every backward region is accounted for in the forward regions, hence it's surjective.

The bijective nature of ϕ confirms that both $C_{\text{forward}}(F)$ and $C_{\text{backward}}(F)$ partition the input space identically, establishing their equivalence. □

Definition 2.3.12. ***Ternary Labeling:*** *[25] The ternary labeling of a point $x \in \mathbb{R}^{n_0}$ relative to the canonical polyhedral complex $C(\overline{\rho}(s))$ is defined as a sequence of ternary tuples*

$$\theta_x = (\theta_x^1, \ldots, \theta_x^m) \in \{-1, 0, 1\}^{n_1 + \ldots + n_m}$$

indicating the sign of the pre-activation output of each neuron of $\overline{\rho}(s)$ at x. Explicitly, for any input vector $x = x_0 \in \mathbb{R}^{n_0}$,

$$x_\ell = (F_\ell \circ F_{\ell-1} \circ \ldots \circ F_1)(x) \in \mathbb{R}^{n_\ell}$$

denotes the output of the first ℓ layer maps, where

$$y_\ell = (A_\ell \circ F_{\ell-1} \circ \ldots \circ F_1)(x) \in \mathbb{R}^{n_\ell}$$

denotes the pre-activation output of the first ℓ layer maps. The components of θ_x^ℓ are defined by

$$\theta_{x,i}^\ell = \operatorname{sgn}(y_{\ell,i})$$

using the convention. ***An activation region*** *(for a parameter $s \in P_{n_0,\ldots,n_m}$) is a maximal connected component of the set of input vectors $x \in \mathbb{R}^{n_0}$ at which the ternary labeling is constant and has no zeros.*

Authors in [54] provided another definition of activation regions which [33] expand this definition as follows:

Definition 2.3.13. *For a network $\mathcal{F}$ with a given input dimension n_{in}, an activation pattern A is defined as:*

$$A := \{a_z | z \text{ is a neuron in } \mathcal{F}\} \in \{-1, 0, 1\}^{\text{number of neurons}},$$

where a_z indicates the sign of the activation of neuron z (either active if $a_z = 1$ or inactive if $a_z = -1$).

The activation region corresponding to A is the subset of the input space where all neurons have the activations specified by A. This region is formally defined as:

$$R(A; \theta) := \{x \in \mathbb{R}^{n_{in}} \mid (-1)^{a_z} (z(x; \theta) - b_z) > 0, \text{ for all neurons } z \text{ in } N\}, \tag{2.2}$$

where $z(x;\theta)$ represents the pre-activation output of neuron z (i.e., before applying the ReLU function), and b_z is the bias term of neuron z. The condition $(-1)^{a_z}\,(z(x;\theta)-b_z) > 0$ ensures that the neuron z's activation state matches the state specified in A.

The set of all such non-empty regions $R(A,\theta)$ for all possible activation patterns A partitions the input space into distinct regions where the network behaves linearly according to the specified pattern.

Notation 2.3.14. Let's denote hyperplane arrangement in $\mathbb{R}^{n_k-1}$ for the layer k as follows:

$$\mathcal{A}^k = \{H_1^{(k)}, \ldots, H_{n_{i_k}}^{(k)}\}$$

A **bent hyperplane** is the pre-image in $\mathbb{R}^{n_0}$ of any hyperplane $H_i^{(k)} \in \mathbb{R}^{n_k-1}$. We will denote by $\mathcal{B}_{\mathcal{F}}^{(k)}$ the union of the bent hyperplanes associated to the k-th layer of $\mathcal{F}$.

The union of these bent hyperplanes $\mathcal{B}_{\mathcal{F}} = \cup_{k=1}^{m} \mathcal{B}_{\mathcal{F}}^{(k)}$, refereed as **bent hyperplane arrangement** is the same hyperplanes arrangement as introduced by canonical polyhedral complex, we will prove this in the following theorem. We can also define this with definition from previous section, before stating the theorems we states these definitions (these definitions will only be used for 2):

Definition 2.3.15 (Bent Hyperplane)**.** *Let $z : \mathbb{R}^{n_0} \to \mathbb{R}$ be the pre-activation function (an affine function) for a neuron in a neural network. The* bent hyperplane *B_z associated with neuron z is defined as*

$$B_z = \{x \in \mathbb{R}^{n_0} \mid z(x) = 0\}\,.$$

The set of all bent hyperplanes in the network is denoted by

$$B = \bigcup_z B_z,$$

where the union is taken over all neurons in the network.

Definition 2.3.16 (Linear Regions)**.** *The* linear regions *of the neural network are the connected components of the complement of the union of bent hyperplanes in the input space $\mathbb{R}^{n_0}$:*

$$\textit{Linear Regions} = \textit{Connected Components of } \mathbb{R}^{n_0} \setminus B.$$

Within each linear region, the activation patterns of the neurons are constant, and the neural network function is affine.

Theorem 1. *Let F be a ReLU neural network. For each i, $C(F_i \circ \ldots \circ F_1)$ is a polyhedral decomposition of $\mathbb{R}^{n_0}$ satisfying the following conditions:*

(i) $F_i \circ \ldots \circ F_1$ is affine-linear on the cells of $C(F_i \circ \ldots \circ F_1)$,

(ii) $\bigcup_{k=1}^{i} B_F^{(k)}$ is the domain of a polyhedral subcomplex of $C(F_i \circ \ldots \circ F_1)$.

Proof. We use induction to prove that each composition $F_i \circ \ldots \circ F_1$ is affine-linear on the cells of $C(F_i \circ \ldots \circ F_1)$. Base Case ($i = 1$):
For $i = 1$, the function $F_1(x) = \sigma(W_1 x + b_1)$ splits the input space into regions defined by hyperplanes $z_1 = W_1 x + b_1$. In each region, F_1 is either the identity function (for $z_{1,k} > 0$) or zero (for $z_{1,k} \leq 0$), both of which are affine transformations. Thus, F_1 is affine-linear on each cell of $C(F_1)$.

Inductive Step:
Assume that for $i = n$, the function $F_n \circ \ldots \circ F_1$ is affine-linear on the cells of $C(F_n \circ \ldots \circ F_1)$. We need to show that this property holds for $i = n + 1$.
Consider the layer $F_{n+1}(x) = \sigma(W_{n+1} x + b_{n+1})$. This layer introduces a set of hyperplanes defined by $z_{n+1} = W_{n+1} x + b_{n+1}$. The intersection of these hyperplanes with the existing cells partitions the input space further into finer cells. Each new cell is defined by a constant activation pattern of neurons, both from F_{n+1} and from earlier layers. Within each cell, F_{n+1} is a composition of affine functions (from $F_n \circ \ldots \circ F_1$) followed by either an identity or a zero function (from σ at layer $n + 1$), maintaining affine-linearity. Thus, $F_{n+1} \circ \ldots \circ F_1$ remains affine-linear within each newly formed cell, proving the inductive step, in other word level set complexes are polyhedral complexes.

By induction, for each i, $F_i \circ \ldots \circ F_1$ is affine-linear on the cells of $C(F_i \circ \ldots \circ F_1)$, as required by the theorem.

Now for part (ii), we use induction to demonstrate that the union of boundaries from all layers up to i forms a polyhedral subcomplex.

For base Case ($i = 1$), the first layer $F_1(x) = \sigma(W_1 x + b_1)$ defines boundaries $B_F^{(1)}$ based on hyperplanes $W_1 x + b_1 = 0$. These boundaries delineate regions of different activation patterns in $C(F_1)$ and form a polyhedral subcomplex as they represent finite intersections of half-spaces, defining polyhedral cells.

Assume that for $i = n$, $\bigcup_{k=1}^{n} B_F^{(k)}$ forms a polyhedral subcomplex of $C(F_n \circ \ldots \circ F_1)$. To show for $i = n + 1$. The $(n + 1)$-st layer F_{n+1} adds new boundaries $B_F^{(n+1)}$, determined by the hyperplanes $W_{n+1} x + b_{n+1} = 0$. Now we consider its interaction with existing decomposition. These hyperplanes intersect existing cells, refining the polyhedral decomposition by subdividing existing cells into smaller polyhedral regions. Each new boundary aligns with or cuts through the current cells. Formally The introduction of $B_F^{(n+1)}$ does not disrupt the existing polyhedral complex structure. Instead, it refines it by adding more faces and edges, each of which is defined by linear constraints and thus preserves the polyhedral nature, hence preserving the polyhedral nature. Now the formation of polyhedral subcomplex is via the union $\bigcup_{k=1}^{n+1} B_F^{(k)}$ incorporates all previous and new boundaries, maintaining the structure of a polyhedral subcomplex. This is because all new intersections are between linear constraints (hyperplanes), which continue to define polyhedral cells (considering the backward construction of canonical polyhedral complex).

By induction, for each i, $\bigcup_{k=1}^{i} B_F^{(k)}$ constitutes the domain of a polyhedral subcomplex in $C(F_i \circ \ldots \circ F_1)$. Each layer's contribution of boundaries, via the introduction of new hyperplanes, consistently integrates into the existing framework to maintain a comprehensive and well-defined polyhedral subcomplex, capturing all transitional activations and their geometric intersections across the network. $\square$

Now that we proved the canonical representation of hyperplane arrangement is equivalent to the later representation; namely constructions via 2.2 and 2.1 will result in same hyperplane arrangement; we now can easily claim that each region $R(A;\theta)$ 2.2 is convex because of the convex nature of polyhedrons. However [33] provide another proof for convexity.

Theorem 2. *Let N be a network with non-linearity and let A be any activation pattern. Then, for any vector θ of trainable parameters for N, the region $R(A;\theta)$ is convex.*

Proof. We will prove that the activation region $R(A;\theta)$ is convex by induction on the depth d of the network N. For ($d = 1$), for the first layer, the activation region $R_z(A;\theta)$ for a neuron z is given by:

$$R_z(A;\theta) = \{x \in \mathbb{R}^{n_{\text{in}}} \mid z(x) - b_z \in (\xi_{a_z}, \xi_{a_z+1})\}$$

where ξ_{a_z} and ξ_{a_z+1} are thresholds defining the activation interval for z, and b_z is the bias. Each set $R_z(A;\theta)$ is convex as it is defined by linear inequalities, and describes a half-space or an interval.

Assume the activation regions defined by up to d layers are convex. Consider the activation region for the network with $d+1$ layers. The activation region for the $(d+1)$-th layer neuron z can be expressed as:

$$R_z(A;\theta) = \{x \in \mathbb{R}^{n_{\text{in}}} \mid z(x) - b_z \in (\xi_{a_z}, \xi_{a_z+1})\}$$

The intersection of convex sets is convex, hence:

$$R(A;\theta) = \bigcap_{\ell=1}^{d+1} \bigcap_{\text{neurons } z \text{ in layer } \ell} R_z(A;\theta)$$

is convex.

Thus, by induction, $R(A;\theta)$ is convex for any depth d, completing the proof.

□

Lemma 2.3.17. *Consider a topological space X and a set of continuous functions $f_j : X \to \mathbb{R}$, for $j = 1, \ldots, J$. On every connected component of $X \setminus \bigcup_j f_j^{-1}(0)$, the sign of each f_j is constant.*

Proof. Each f_j is non-zero on X_α by construction, where X_α is a connected component of $X \setminus \bigcup_j f_j^{-1}(0)$. The image of a connected set under a continuous function is connected, which implies that $f_j(X_\alpha) \subseteq (-\infty, 0) \cup (0, \infty)$. Hence, $f_j(X_\alpha)$ must entirely lie within one of these intervals, confirming the constancy of the sign of f_j on X_α.

□

In the context of neural networks with ReLU activation functions, *bent hyperplanes* are the hyperplanes where the pre-activation input to a neuron is zero. These hyperplanes partition the input space into regions where the neuron is either active (outputting a positive value) or inactive (outputting zero).

Theorem 3. *For any ReLU net N and any vector θ of trainable parameters, the non-empty activation regions $R(A; \theta)$ are exactly the connected components of the input space minus the hyperplanes defined by zero activations of the neurons:*

$$\mathbb{R}^{n_{in}} \setminus \bigcup_{\text{neurons } z} H_z(\theta)$$

where $H_z(\theta) = \{x \in \mathbb{R}^{n_{in}} : z(x; \theta) = 0\}$ for each neuron z.

Proof. For a ReLU neural network $\mathcal{N}$ with input dimension n_{in} and a specific parameter set θ, define each neuron's pre-activation as $z(x; \theta) = W_z x + b_z$, and the corresponding zero-activation hyperplane as:

$$H_z(\theta) = \{x \in \mathbb{R}^{n_{\text{in}}} : z(x; \theta) = b_z\}.$$

Each connected component of the space $\mathbb{R}^{n_{\text{in}}} \setminus \bigcup_{\text{neurons } z} H_z(\theta)$ is such that within any such component, the sign of $z(x) - b_z$ for each neuron z is fixed (either always positive

or always non-positive). Thus, each connected component corresponds uniquely to an activation pattern, where the pattern specifies which neurons are activated and which are not.

By 2.3.17 The connected components of $\mathbb{R}^{n_{\text{in}}} \setminus \bigcup_{\text{neurons } z} H_z(\theta)$ encompass regions where the activation pattern does not change. Each component is, therefore, contained within some activation region $R(A;\theta)$, defined by a particular activation pattern A. Conversely, each activation region $R(A;\theta)$ is a subset of the complement of the union of the hyperplanes, and is convex and hence connected. Therefore, it coincides with the connected component we started with.

□

Chapter 3

Reverse Engineering Deep Relu Nets, an Optimization Based Method

3.1 Introduction

Deep learning has established itself as a powerful tool across various domains, including computer vision, natural language processing, and reinforcement learning. Among different types of deep neural networks, Rectified Linear Unit (ReLU) networks have gained particular popularity due to their simplicity and strong expressive capabilities. These networks employ piece-wise linear activation functions, allowing them to efficiently approximate complex functions and provide a robust learning mechanism. Despite the widespread success of deep ReLU networks, understanding their properties, expressive power, and the feasibility of reverse-engineering their structure remains a significant challenge in the field.

Reverse engineering deep ReLU networks can serve multiple purposes. First, it can facilitate a better understanding of the inner workings of these models, enabling researchers to devise improved architectures, training methods, and optimization techniques. Second, it can promote interpretability and robustness, helping to identify and mitigate potential vulnerabilities and security risks associated with deep learning models. However, reverse

engineering deep ReLU networks presents significant challenges, as it involves determining the optimal weights and biases for the networks given limited information. Hence, Reverse-engineering deep ReLU networks is a critical task, as it can offer insights into the model's inner workings, contribute to the development of more efficient and accurate models, and uncover potential vulnerabilities associated with the model's architecture and weights. In this context, numerous research efforts have sought to investigate the properties of deep ReLU networks, their expressive power, and their reverse-engineering potential.

In this work, we propose a novel approach to reverse engineering deep ReLU networks by formulating an optimization problem and employing gradient descent to solve it. We build upon insights from previous works on the complexity of linear regions and activation patterns in deep ReLU networks, the expressive power of deep neural networks, and methods for model reconstruction from model explanations. Our approach aims to advance the understanding of deep ReLU networks and provide a practical method for reconstructing them from limited information.

3.2 Related Work

Our work builds upon a rich body of research in the areas of deep learning, neural networks, and their properties. [16] is probably the first one, who explored the problem of reconstructing a neural net from its output, showing that under certain conditions, it is possible to recover the network's structure and parameters from its input-output mapping. This work provided an early theoretical foundation for the study of reverse-engineering neural networks and highlighted the challenges and limitations of this task, such as the non-uniqueness of solutions and the need for regularization techniques to stabilize the reconstruction process. [56] focused on reverse-engineering deep ReLU networks by leveraging the piecewise-linear property of ReLU activations. Their approach required querying the input-output mapping of the model, and their algorithm recov-

ered the network layer by layer. Their method assumed that the weights are integer valued and the network is noiseless. [9] investigated the extraction of neural network models in a cryptanalytic setting, demonstrating the feasibility of recovering model parameters by querying the model's outputs. Their method used a black-box query oracle and was based on solving a system of linear equations. The approach made minimal assumptions about the network architecture and activation functions. However, it required a substantial number of queries, and the exact architecture of the network was not recovered. Both [9] and [56] propose algorithms for reverse-engineering deep ReLU networks, with [9] assuming known architecture and [56] being more efficient in terms of time complexity despite needing much more queries. [47] explored the related problem of model reconstruction from model explanations, showing that it is possible to recover a model from explanations such as feature importance scores or model predictions. They leveraged model explanations like LIME and SHAP to recover the target model's structure and parameters. The method's accuracy depended on the quality of the explanations and the choice of the explanation method. They did not focus specifically on ReLU networks, and the method was applicable to various types of models. However, Their work is connected to our problem of reverse engineering deep ReLU networks, as both problems involve reconstructing model parameters from limited information. [59] investigated the reverse engineering of the Neural Tangent Kernel (NTK), a widely used tool in the analysis of deep learning dynamics. They showed that the NTK can be recovered from a model's input-output pairs, providing a novel approach to understanding the inner workings of deep networks. This work contributes to the growing body of research on reverse-engineering deep learning models and offers a new perspective on the relationship between the NTK and the model's underlying structure. The algorithms for reconstruction of networks have limitations, for example, [56] in terms of scalability and handling complex architectures or [9]'s algorithm assumes that the architecture of the network is known, which is not always the case in practical applications but it requires less queries from the input-output and as a result it has a more efficient algorithm in sense of

space-complexity whilst [56] is a more efficient algorithm in the sense of time complexity. In contrast to the above methods, other works in the related literature focused on understanding the expressive power and properties of deep ReLU networks. Despite recent works [64] [42] [32] [34] investigate the expressive power of deep neural networks, there are lots of unknown and open problems on that regards. [34] performed an analysis of the activation patterns of deep ReLU networks, showing that these networks exhibit surprisingly few activation patterns relative to the number of neurons. Their findings offered insights into the expressive power of deep ReLU networks and the complexity of their input-output mappings, revealing an inherent simplicity that may contribute to the effectiveness of these models. They also provided bounds on the number of activation patterns, which can be useful for understanding the behavior of deep ReLU networks, similarly [32] investigated the complexity of linear regions in deep networks, shedding light on the geometry of the decision boundaries formed by ReLU networks. Their work provided a rigorous understanding of the underlying mathematical structure of these models and the factors that contribute to their expressive power. They showed that the number of linear regions grows exponentially with the depth of the network, highlighting the role of depth in enabling complex decision boundaries. These insights of [34] [32] inform our understanding of the expressive power of deep ReLU networks and motivate our approach to reverse engineering them. In another work [54] investigated the expressive power of deep neural networks, providing insights into the capacity of deep networks to learn complex functions and demonstrating that certain network architectures are better suited for learning specific types of functions. Their work provided insights into the role of depth, width, and architecture in determining the expressiveness of deep learning models. They developed a framework for understanding the expressiveness of networks in terms of their ability to represent functions and transformations. Similarly, [30] prove that for any continuous function on a compact set, there exists a deep neural network with a bounded width and ReLU activation functions that can approximate the function to within any desired degree of accuracy and highlights potential of these kind of net-

works for efficient and accurate representation of complex functions. Hence these works helps us understand the inherent expressive power of deep ReLU networks and guides our choice of network architecture for the reverse engineering problem. One other similar work of [57] examined the power of deeper networks for expressing natural functions, showing that deeper networks can learn a broader class of functions compared to shallow networks. This work further supports the idea that deep ReLU networks are capable of expressing complex functions and serves as a motivation for our study on reverse engineering such networks. [7] examined the functional versus parametric equivalence of ReLU networks, exploring the relationship between the architecture of a network and its ability to represent functions. They showed that functionally equivalent networks can have vastly different architectures and parameters, underscoring the importance of studying functional properties rather than specific parameter settings. Their work also provided a framework for understanding the space of functionally equivalent networks, which can inform the design of more efficient and robust models.

In summary, the related work presented here highlights the rich landscape of research efforts aimed at understanding and reverse-engineering deep ReLU networks. These studies have provided valuable insights into the properties, expressive power, and vulnerabilities of these models, paving the way for further advancements in the field. By building upon these foundational works, including Fefferman's early exploration of the problem in 1994, we hope to inspire new research directions and contribute to the ongoing quest to unravel the mysteries of deep learning.

3.3 Reverse-Engineering the Network Function

The primary goal of our methodology is to reconstruct the piecewise linear function defined by a ReLU neural network without direct access to its internal parameters. Instead, we aim to approximate the network's function by estimating the hyperplanes corresponding to its linear regions using sampled input-output data; our approach specifically aims

to reconstruct the hyperplanes corresponding to its linear regions. We begin by sampling points from the input space $\mathbb{R}^{n_0}$, obtaining a set of points $v_1, v_2, \ldots, v_N \in \mathbb{R}^{n_0}$. hese points can be chosen randomly or based on specific criteria to cover regions of interest within the input domain. Each point serves as a reference for constructing a local approximation of the network's function.

At each sampled point, the neural network function f behaves as an affine function within a small neighborhood due to the piecewise linear nature of ReLU networks. Assuming f is differentiable at point v_i, we can compute the gradient, which provides the coefficients of the linear approximation at that point. This results in a local linear function $g_i(x)$that approximates the network near v_i.

To ensure that each local approximation is only valid near its corresponding point we define modified functions $\tilde{g}$ that are zero outside a specified neighborhood (e.g., within a ball of radius c_i centered at v_i). This localization helps prevent the overlap of different local approximations in regions where they may not be accurate.

Finally, We construct a global approximation $h(x)$ of the neural network function by combining the localized linear functions $\tilde{g}$ using weighting coefficients. The weights determine the contribution of each local approximation to the global function. By appropriately choosing these weights, we aim to closely match to the true network function across the input domain.

To find the optimal weights, we formulate an optimization problem. For the optimization problem to be tractable and have a unique solution, it is desirable for it to be convex. Convexity ensures that any local minimum is also a global minimum. We analyze the Hessian matrix of the loss function with respect to the weights and derive conditions under which the Hessian is positive semidefinite. By carefully designing the localized functions $\tilde{g}$ and their supports, we can ensure that these conditions are satisfied, making the optimization problem convex.

3.3.1 Estimating Hyperplanes via Gradients

For a function $f : \mathbb{R}^{n_0} \to \mathbb{R}$, the gradient $\nabla f(x)$ provides the coefficients of the linear approximation at point x. Specifically, we have:

$$\nabla f(x) = \begin{pmatrix} \dfrac{\partial f(x)}{\partial x_1} \\ \dfrac{\partial f(x)}{\partial x_2} \\ \vdots \\ \dfrac{\partial f(x)}{\partial x_{n_0}} \end{pmatrix}.$$

The directional derivative of f at x in the direction v is given by:

$$\nabla_v f(x) = \frac{d}{dt} f(x + tv)\bigg|_{t=0} = \nabla f(x)^\top v = \sum_{i=1}^{n_0} v_i \frac{\partial f(x)}{\partial x_i}.$$

Assuming f is differentiable at each sampled point v_i, we compute the gradient g_i at v_i:

$$g_i = \nabla f(v_i).$$

Since f is affine in a neighborhood around v_i, we can express f locally as:

$$f(x) \approx f(v_i) + g_i^\top (x - v_i).$$

We define local linear functions $g_i(x)$ to approximate f in the vicinity of each point v_i:

$$g_i(x) = g_i^\top x + c_i,$$

where $c_i = f(v_i) - g_i^\top v_i$.

To localize these approximations, we introduce modified functions $\tilde{g}_i(x)$ defined as:

$$\tilde{g}_i(x) = \begin{cases} g_i(x), & \text{if } x \in B_{r_i}(v_i), \\ 0, & \text{otherwise,} \end{cases}$$

where $B_{r_i}(v_i)$ denotes the ball of radius r_i centered at v_i.

We approximate the network function f by combining the local approximations $\tilde{g}_i(x)$ using weighting functions $w_i(x)$:

$$h(x) = \sum_{i=1}^{N} w_i(x)\tilde{g}_i(x).$$

The weights $w_i(x)$ are non-negative and sum to one:

$$w_i(x) \geq 0, \quad \sum_{i=1}^{N} w_i(x) = 1.$$

One possible choice for the weighting functions is inverse distance weighting:

$$w_i(x) = \frac{\dfrac{1}{d(x, v_i)^p}}{\displaystyle\sum_{j=1}^{N} \frac{1}{d(x, v_j)^p}},$$

where $d(x, v_i)$ is the distance between x and v_i, and $p > 0$ is a parameter controlling the influence of nearby points.

To find the optimal weights $w_i(x)$, we define an objective function measuring the discrepancy between $h(x)$ and $f(x)$ over a domain $D \subseteq \mathbb{R}^{n_0}$:

$$L(h) = \int_D \left(h(x) - f(x)\right)^2 dx.$$

In practice, we may approximate $L(h)$ by evaluating the integral over a finite sample of points or over a bounded domain $B_T(0)$.

However, for computational simplicity, we consider the weights w_i to be constants (i.e., independent of x). Our goal is then to find constant weights w_i that minimize $L(h)$:

$$L(h) = \int_D \left(\sum_{i=1}^{N} w_i \tilde{g}_i(x) - f(x) \right)^2 dx.$$

To ensure that the optimization problem is convex, we examine the Hessian matrix of $L(h)$ with respect to the weights $w = (w_1, w_2, \ldots, w_N)^\top$. The Hessian is given by:

$$\nabla_w^2 L(h) = 2 \int_D \tilde{g}(x) \tilde{g}(x)^\top dx,$$

where $\tilde{g}(x) = \begin{pmatrix} \tilde{g}_1(x) \\ \tilde{g}_2(x) \\ \vdots \\ \tilde{g}_N(x) \end{pmatrix}$.

Each element of the Hessian matrix is:

$$\left(\nabla_w^2 L(h)\right)_{ij} = 2 \int_D \tilde{g}_i(x) \tilde{g}_j(x) dx.$$

For the Hessian to be positive semidefinite, it suffices that $\nabla_w^2 L(h)$ is symmetric and all its eigenvalues are non-negative.

3.3.2 Ensuring Positive Semidefiniteness

We apply the Gershgorin Circle Theorem to provide sufficient conditions for the Hessian to be positive semidefinite. According to the theorem, all eigenvalues of $\nabla_w^2 L(h)$ lie within the union of disks centered at the diagonal entries with radii equal to the sum of the absolute values of the non-diagonal entries in the corresponding rows.

For each i, we require:

$$\left(\nabla_w^2 L(h)\right)_{ii} \geq \sum_{j \neq i} \left| \left(\nabla_w^2 L(h)\right)_{ij} \right|.$$

Substituting the expressions for the Hessian entries, this condition becomes:

$$2 \int_D \tilde{g}_i^2(x) dx \geq 2 \sum_{j \neq i} \left| \int_D \tilde{g}_i(x) \tilde{g}_j(x) dx \right|.$$

Simplifying, we obtain:

$$\int_D \tilde{g}_i^2(x) dx \geq \sum_{j \neq i} \left| \int_D \tilde{g}_i(x) \tilde{g}_j(x) dx \right|.$$

Since we have control over the definitions of $\tilde{g}_i(x)$, specifically their domains $B_{r_i}(v_i)$ and the constants c_i, we can adjust them to satisfy the above condition.

By choosing the radii r_i of the balls $B_{r_i}(v_i)$ appropriately and ensuring that the supports of $\tilde{g}_i(x)$ and $\tilde{g}_j(x)$ have minimal overlap for $i \neq j$, we can make the cross terms $\int_D \tilde{g}_i(x) \tilde{g}_j(x) dx$ sufficiently small.

This approach guarantees that the Hessian $\nabla_w^2 L(h)$ is diagonally dominant and thus positive semidefinite, ensuring that the optimization problem is convex.

3.3.3 Hessian Calculations

To analyze the convexity of the optimization problem for minimizing $L(h)$ with respect to the weights w_i, we compute the Hessian matrix of $L(h)$ with respect to w_i.

Recall that the objective function is:

$$L(h) = \int_{B_T(0)} \|h(x) - f(x)\|^2 dx,$$

where

$$h(x) = \sum_{i=1}^{N} w_i g_i'(x),$$

and w_i are constants, and $g_i'(x)$ are localized linear functions defined as:

$$g_i'(x) = \begin{cases} g_i(x), & \text{if } x \in B_{r_i}(v_i), \\ 0, & \text{otherwise}, \end{cases}$$

with $g_i(x)$ being linear approximations of f near v_i.

First Derivative

The first derivative of $L(h)$ with respect to w_i is:

$$\frac{\partial L(h)}{\partial w_i} = \frac{\partial}{\partial w_i} \left(\int_{B_T(0)} \|h(x) - f(x)\|^2 dx \right).$$

Since $f(x)$ does not depend on w_i, we have:

$$\frac{\partial L(h)}{\partial w_i} = \int_{B_T(0)} 2(h(x) - f(x)) \frac{\partial h(x)}{\partial w_i} dx.$$

Given that

$$\frac{\partial h(x)}{\partial w_i} = g_i'(x),$$

we obtain:

$$\frac{\partial L(h)}{\partial w_i} = \int_{B_T(0)} 2(h(x) - f(x)) g_i'(x) dx.$$

Second Derivative

The second derivative is:

$$\frac{\partial^2 L(h)}{\partial w_i \partial w_j} = \frac{\partial}{\partial w_j} \left(\frac{\partial L(h)}{\partial w_i} \right).$$

Using the result from the first derivative:

$$\frac{\partial L(h)}{\partial w_i} = \int_{B_T(0)} 2(h(x) - f(x))g_i'(x)dx,$$

and noting that $g_i'(x)$ does not depend on w_j, we have:

$$\frac{\partial^2 L(h)}{\partial w_i \partial w_j} = \int_{B_T(0)} 2\left(\frac{\partial h(x)}{\partial w_j}\right) g_i'(x)dx.$$

Since

$$\frac{\partial h(x)}{\partial w_j} = g_j'(x),$$

it follows that:

$$\frac{\partial^2 L(h)}{\partial w_i \partial w_j} = 2\int_{B_T(0)} g_i'(x)g_j'(x)dx.$$

Thus, the Hessian matrix $\nabla_w^2 L(h)$ has entries:

$$H_{ij} = 2\int_{B_T(0)} g_i'(x)g_j'(x)dx.$$

Convexity Guarantee

For the optimization problem to be convex, the Hessian matrix must be positive semidefinite. Using the Gershgorin Circle Theorem, we require that for each i:

$$H_{ii} - \sum_{j\neq i} |H_{ij}| \geq 0.$$

Substituting the expressions for H_{ij}:

$$2\int_{B_T(0)} \left(g_i'(x)\right)^2 dx - \sum_{j\neq i} 2\left|\int_{B_T(0)} g_i'(x)g_j'(x)dx\right| \geq 0.$$

Simplifying:

$$\int_{B_T(0)} \left(g_i'(x)\right)^2 dx \geq \sum_{j\neq i} \left|\int_{B_T(0)} g_i'(x)g_j'(x)dx\right|, \quad \forall i.$$

Assuming that each $g_i'(x)$ is supported on $B_{r_i}(v_i)$ and that the overlaps between supports are minimal, we can compute the integrals explicitly.

Consider $g_i(x)$ defined as:

$$g_i(x) = w_i^\top (x - v_i) + b_i.$$

Shifting coordinates to $x' = x - v_i$, we have:

$$g_i(x) = w_i^\top x' + b_i.$$

The integral of $(g_i(x))^2$ over $B_{r_i}(v_i)$ becomes:

$$I_{ii} = \int_{B_{r_i}(0)} \left(w_i^\top x' + b_i\right)^2 dx'.$$

Expanding:

$$I_{ii} = \int_{B_{r_i}(0)} \left(\left(w_i^\top x'\right)^2 + 2b_i w_i^\top x' + b_i^2\right) dx'.$$

Due to symmetry of the ball centered at the origin:

$$\int_{B_{r_i}(0)} w_i^\top x' dx' = 0.$$

Therefore:

$$I_{ii} = \int_{B_{r_i}(0)} \left(w_i^\top x'\right)^2 dx' + b_i^2 V_n r_i^n,$$

where $V_n r_i^n$ is the volume of the n-dimensional ball.

Similarly, for $i \neq j$, the cross-term integrals are:

$$I_{ij} = \int_{B_{r_i}(0) \cap B_{r_j}(v_j - v_i)} \left(w_i^\top x' + b_i\right) \left(w_j^\top (x' + v_i - v_j) + b_j\right) dx'.$$

If the overlap between $B_{r_i}(v_i)$ and $B_{r_j}(v_j)$ is negligible, I_{ij} is small.

By selecting the radii r_i and positions v_i such that the overlaps are minimal, the off-diagonal terms I_{ij} become negligible compared to the diagonal terms I_{ii}. This ensures that:

$$\int_{B_T(0)} \left(g_i'(x)\right)^2 dx \gg \sum_{j \neq i} \left| \int_{B_T(0)} g_i'(x) g_j'(x) dx \right|,$$

and the Hessian matrix $\nabla_w^2 L(h)$ is diagonally dominant and thus positive semidefinite.

3.3.4 Extending the Methodology

In our approach to estimating the neural network function, we have thus far considered only first-order approximations based on the functions $g_i'(x)$. To enhance our estimation, we propose incorporating interaction terms between these approximations by including inner products of the hyperplanes. Specifically, we define the inner product between two hyperplanes as the intersection area of their corresponding regions.

By including these inner product functions, we can reformulate the approximation of $f(x)$ as:

$$h'(x) = \sum_{i=1}^{N} w_i(x) g_i'(x) + \sum_{i,j} w_{ij}' \langle g_i'(x), g_j'(x) \rangle,$$

where $\langle g_i'(x), g_j'(x) \rangle$ represents the inner product of the functions $g_i'(x)$ and $g_j'(x)$, and w_{ij}' are additional weights to be determined.

We then define a new optimization problem $L_{w,w'}(h')$ aimed at minimizing the discrepancy between $h'(x)$ and the true network function $f(x)$:

$$L_{w,w'}(h') = \int_D \left(h'(x) - f(x)\right)^2 dx,$$

where D is the domain of interest.

Incorporating these inner product terms allows us to capture higher-order interactions between hyperplanes, potentially revealing more intricate details about the network's structure. However, this extension introduces additional complexity to the optimization

problem. We must establish and satisfy new constraints to ensure the convexity of the problem, which is crucial for guaranteeing the existence of a unique global minimum.

At this stage, it is not immediately clear whether convexity can be maintained with the inclusion of the inner product terms. Further theoretical analysis is required to determine the conditions under which the extended optimization problem remains convex. If convexity can be assured, this approach may provide a more accurate estimation of the network's function and offer deeper insights into the relationships between the network's components.

3.3.5 Discussion

In this work, we have proposed a method for reverse-engineering deep ReLU networks by sampling points in the input space, estimating local hyperplanes around these points, and solving a convex optimization problem to approximate the true function represented by the network. The central idea is to exploit the piecewise linear nature of ReLU networks to reconstruct the function without access to the network's internal parameters.

Our approach relies on several key steps:

- **Sampling Input Points:** We select points in the input space and query the black-box model to obtain corresponding output values and gradients.
- **Estimating Local Hyperplanes:** Using the sampled data, we estimate local linear approximations of the network function around each point.
- **Formulating the Optimization Problem:** We define an objective function that measures the discrepancy between our approximation and the true function, and we seek to minimize this function subject to certain constraints.

One of the main advantages of our method is that the optimization problem is convex, ensuring a unique global minimum and allowing for efficient computation using well-established optimization techniques. However, this convexity is contingent upon

the specific formulation of the problem and may not hold if additional terms, such as inner product functions, are included without careful consideration.

Despite the potential benefits, our method has several limitations:

- **Dependence on Sampling:** The accuracy of the approximation heavily depends on the sampling strategy. In high-dimensional input spaces or for functions with complex behavior, sampling may not capture all relevant features, leading to suboptimal approximations.
- **Assumption of Piecewise Linearity:** Our method assumes that the network is composed of ReLU activations, resulting in a piecewise linear function. This assumption limits the applicability of the method to networks with other activation functions or architectures.
- **Computational Complexity:** Calculating gradients and solving the optimization problem can be computationally expensive, especially for large-scale networks or when a large number of samples is required.

Regarding our conjecture about the relationship between the optimized weights and the network's architecture, we hypothesize that adding an ℓ_1 regularization term (the Lasso regularizer) to the optimization problem may encourage sparsity in the weights w_i. Specifically, we posit that:

1. By adding the regularizer $\|w\|_1$ to $L_w(h)$, the number of non-zero weights w_i may correspond to the number of neurons in the first layer of the network.
2. Extending the optimization problem to include inner product terms and corresponding weights w'_{ij}, and adding the regularizer $\|w'\|_1$, may further reveal the number of activation regions in the input space.

However, these conjectures require rigorous proof and validation. The inclusion of regularization terms can affect the convexity of the optimization problem, and ensuring that convexity is preserved is non-trivial. Additionally, the relationship between the

sparsity induced by the regularizer and the network's architecture may not be straightforward.

Chapter 4

Spagettification and Geometrical Properties of Relu Nets

4.1 Introduction

The utilization of ReLU neural networks marks a significant advancement in deep learning, with applications ranging from image recognition to autonomous driving. These networks, understood not just through their computational efficacy but also through their geometric properties. Specifically, we investigate the geometric structures of deep ReLU networks and how these structures dictate the transformation of input data across layers. Our exploration initiates with foundational geometric concepts such as polyhedral sets and convex polytopes. These concepts are crucial for understanding the more intricate structures known as polyhedral complexes, which we discuss subsequently. We elucidate on these complexes to set the groundwork for their application in neural network analysis, illustrating how they can encapsulate the behavior of neural activations within a network. In transitioning to neural networks, we define feedforward ReLU neural networks formally as sequences of affine transformations capped by ReLU activations. This section introduces canonical polyhedral complexes that model the neuron activations geometrically across the network's architecture. These complexes not only help in geo-

metrical insight of the network's behavior but also serve as a tool for understanding the dynamics of data transformation through the network. This section delves into the properties of these complexes, focusing particularly on the evolution of data images under the influence of each network layer.

Understanding the geometric transformations that data undergoes as it passes through neural networks is pivotal for enhancing both the theoretical foundation and practical performance of these models. Neural networks, particularly those employing ReLU activations, introduce complex and non-linear changes to the data manifold. These changes significantly impact the network's ability to generalize, differentiate between data points, and ultimately, perform different tasks There is a growing need to dissect and comprehend how these geometric properties evolve. For example, the boundedness of data representations, the degeneration of angles between data vectors, the overall distortion of the data manifold, and the expected distortions during data transformation are critical aspects that influence the network's performance. Yet, these aspects are often not given due attention in conventional studies focused on empirical results. By meticulously analyzing these geometric transformations, we can uncover insights that are crucial for several reasons:

Enhancing Network Robustness, Understanding how data becomes bounded as it propagates through layers can help in designing networks that are more stable and less prone to issues such as exploding or vanishing gradients.

Preserving Discriminative Power, Analyzing angle degeneracy helps in maintaining the network's ability to distinguish between different data points, which is vital for classification tasks.

Optimizing Data Transformations, Examining the geometry of distortion provides insights into how the network processes data, leading to more effective network architectures that can better capture the underlying structure of the data.

Comparing Network Architectures Quantifying the expected value of distortion allows for a rigorous comparison between different network designs, aiding in the selection of

architectures that offer optimal performance for specific tasks. These motivations drive our in-depth study of the geometric properties of neural networks, aiming to bridge the gap between theoretical understanding and practical application in deep learning.

Boundedness of Images We present detailed mathematical proofs to demonstrate how the images of data points transition through a neural network, emphasizing the network's capacity and structural constraints. A significant portion of our analysis concentrates on the boundedness of these images. We rigorously argue that as data progresses through successive layers of the network, its geometrical representation tends towards boundedness, ultimately reaching a state of complete boundedness with probability one. This finding is crucial as it provides a theoretical foundation for understanding the stability and limitations of neural network transformations.

Angle Degeneracy Angle degeneracy in neural networks refers to the phenomenon where the angles between vectors representing data points become less distinct as they traverse the network's layers. This degeneration can lead to a loss of discriminative power, adversely affecting the network's performance. We analyze the evolution of angle degeneracy within ReLU networks, providing a detailed examination of how it impacts the network's ability to differentiate between distinct data points. Our analysis highlights the importance of maintaining angular distinctions to preserve the network's discriminative capabilities.

Geometry of Distortion The geometry of distortion investigates how the data manifold is distorted by the transformations within a neural network. By examining these distortions, we gain critical insights into the network's processing and transformation of input data. Our study reveals how various network configurations and layer mappings contribute to the distortion of the data manifold, which is essential for understanding and improving network performance. This exploration helps in identifying the factors that influence the network's ability to generalize and adapt to new data.

Expected Value of Distortion The expected value of distortion quantifies the average distortion that a data point undergoes as it passes through the network. This metric is crucial for assessing the overall impact of the network's transformations on the data. By providing a basis for comparing different network architectures and configurations, the expected value of distortion allows us to evaluate the effectiveness of various design choices.

4.2 Related Work

[43] addresses the combinatorial intricacies inherent in ReLU networks by proposing an algorithmic framework to determine the structure of linear regions partitioned by ReLU activations. The study leverages combinatorial optimization and computational geometry to develop algorithms that map out these regions efficiently. The framework's effectiveness is demonstrated through both theoretical analysis and empirical validation, illustrating how the combinatorial structure influences network expressiveness and learning capacity. This work provides a valuable computational tool for dissecting the functional complexity of ReLU networks, offering insights into how architectural choices and network configurations affect their combinatorial landscape. Same authos in [23] explore the concept of transversality in the context of bent hyperplane arrangements formed by ReLU activations. The authors investigate how these arrangements impact the topological expressiveness of ReLU networks. By employing advanced techniques from algebraic topology and differential geometry, they demonstrate that the transversality properties of hyperplanes play a crucial role in determining the network's ability to represent complex topological structures. The findings highlight the importance of geometric considerations in network design, suggesting that ensuring proper transversality can enhance the network's expressive power and facilitate more efficient learning processes. In their comprehensive study, [24] introduce measures for evaluating both local and global topological complexity of ReLU neural network functions. By combining topological data analysis

with neural network theory, the authors propose metrics that quantify the intricacies of network-induced functions. These measures provide a deeper understanding of how local structural features aggregate to form globally complex behaviors. The study underscores the significance of topological considerations in neural network analysis, offering tools to assess and compare the topological properties of different network architectures and their implications for learning dynamics. The paper [25] introduces functional dimension for the first time; they explore the functional dimension as a measure of expressive power of these networks by examining the functional space they can represent. They provide a rigorous mathematical investigation into the functional dimension of feedforward ReLU networks.The authors characterize the functional space of such networks. They demonstrate that the piecewise linear nature of ReLU activation induces a partitioning of the input space into polyhedral regions, with each region corresponding to a distinct linear function. The functional dimension is thus tied to the number and arrangement of these regions, which in turn depends on the network's architecture, including depth, width, and connectivity patterns. This work is pivotal as it lays the foundation for understanding the expressiveness and limitations of ReLU networks from a geometric perspective, providing insights into how network parameters influence the complexity and capacity of the functional space. In a follow-up study, [22] extends their exploration of ReLU networks by uncovering hidden symmetries inherent in these models. The authors identify and characterize symmetry groups that preserve the functional equivalence of networks despite alterations in weights and biases. They show that these symmetries can lead to multiple, functionally identical network configurations, implying redundancy in network parameterizations. This work reveals deeper structural insights, indicating that optimization landscapes for ReLU networks may possess symmetric basins of attraction. Such symmetries can be exploited to design more efficient training algorithms and to understand the convergence properties of gradient-based methods. In another study, [56] examines the linear regions and decision boundaries, thus providing insight into how these networks encode and process information.

[38] addresses a critical issue in deep ReLU networks: **depth degeneracy**. They observe that as network depth increases, the angles between randomly initialized weight vectors in successive layers tend to vanish, leading to degenerate geometries and poor gradient flow. The authors quantify this phenomenon and its impact on training dynamics. They propose that initialization schemes must account for this degeneracy to maintain diversity in weight orientations and ensure effective gradient propagation. This work emphasizes the necessity of robust initialization strategies, particularly for very deep networks, to mitigate issues related to vanishing gradients and to enhance training stability. [20] explores the theoretical underpinnings of deep networks with random Gaussian weights, proposing that such networks can act as universal classifiers. The authors draw on the Johnson-Lindenstrauss lemma and random matrix theory to argue that high-dimensional input data can be effectively projected into a lower-dimensional space while preserving pairwise distances. This dimensionality reduction facilitates linear separability, enabling robust classification. They provide theoretical guarantees and empirical evidence supporting this claim, highlighting the power of random projections in the context of deep learning. This study bridges the gap between random matrix theory and neural network performance, suggesting that certain random architectures may inherently possess strong classification capabilities.

[15] examines the impact of permutation invariance on the linear mode connectivity in neural networks. They present a theoretical analysis demonstrating that neural networks exhibit linear mode connectivity when parameters are permutation invariant. This property allows for smooth interpolation between different sets of weights, facilitating robust optimization and generalization. The study offers insights into the structural properties that contribute to effective training regimes. The findings highlight the critical role of permutation invariance in achieving stable and efficient neural network optimization. [3] investigates the role of symmetry-invariant optimization in enhancing the performance of deep networks. They propose optimization techniques that leverage the inherent symmetries within neural architectures to improve training efficiency and convergence rates. By

utilizing group theory and differential geometry, the authors develop methods that maintain symmetry invariance throughout the optimization process, leading to more robust and generalizable models. The empirical results demonstrate significant improvements in training speed and model performance, underscoring the practical benefits of incorporating symmetry considerations into optimization algorithms. The concept of neural collapse has garnered significant attention in recent studies, providing a unified framework to understand the geometric properties and convergence behavior of deep neural networks during the final stages of training. Several key papers have contributed to this understanding, focusing on various aspects such as feature space behavior, geometric structures, and theoretical implications. [65] and [48] both explore neural collapse in the context of unconstrained features. These works provide a geometric analysis of neural collapse, highlighting how, during the terminal phase of training, the feature vectors corresponding to samples from the same class converge to their class means. These class means, in turn, become equidistant from each other, forming a simplex ETF (Equiangular Tight Frame) structure. This phenomenon indicates a highly symmetric and organized arrangement in the feature space, which simplifies the classification task. Similarly, [52] extends this analysis by empirically demonstrating the prevalence of neural collapse across various architectures and datasets. They show that neural collapse is not only a theoretical construct but also a pervasive empirical pattern observed during the late stages of deep learning training. Their work confirms that neural collapse leads to a deterministic structure in the activation space, characterized by a collapse of within-class variability and an alignment of class means.

The study by [10] on the activation space of ReLU-equipped deep neural networks aligns with the aforementioned findings by investigating how ReLU activations influence the geometry of the feature space. They analyze how ReLU activations partition the input space into linear regions, contributing to the overall geometric structure observed in neural collapse. This partitioning effect is crucial for understanding how neural networks manage to achieve such organized feature representations. [63] extends the analysis of

neural collapse to normalized features, examining how normalization techniques impact the geometric structure of neural networks. By conducting a geometric analysis over the Riemannian manifold, they demonstrate that normalization helps in achieving a more uniform distribution of features, facilitating the convergence to a simplex ETF configuration. This study underscores the importance of normalization in enhancing the geometric properties that underlie neural collapse.

4.3 Theorical Resluts

4.3.1 Rays and Images under Layers Maps

Notation 4.3.1. Consider a full connected, feedforward ReLU neural network of architecture $(n_0, \ldots, n_d)$ given by

$$F : \mathbb{R}^{n_0} \xrightarrow{F^1=\sigma\circ A^1} \mathbb{R}^{n_1} \xrightarrow{F^2=\sigma\circ A^2} \ldots \xrightarrow{F^m=\sigma\circ A^d} \mathbb{R}^{n_d}$$

where σ denotes coordinatewise application of ReLU and $A^i : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$ is an affine-linear map.

- We will denote the composition of the first k layer maps of F by $G_{(k)}$, i.e.

$$G_{(k)} \coloneqq F^k \circ \ldots F^1.$$

- For $1 \leq k \leq m$, define the *kth image* to be the set

$$\text{Im}_{(k)} \coloneqq G_{(k)}(\mathbb{R}^{n_0})$$

and define $\text{Im}_{(0)} = \mathbb{R}^{n_0}$.

- For each i, let A^{i*} denote the *linear* map $\mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$ obtained from A^i by omitting the bias. Let $F^{i*} \coloneqq \sigma \circ A^{i*}$, and let

$$G^*_{(k)} \coloneqq F^{k*} \circ \dots F^{1*}.$$

Definition 4.3.2. *Let $0 \leq k \leq d$ be an integer.*

1. *A* geometric ray *in $\mathbb{R}^{n_k}$ is a set of the form*

$$R_{\vec{x},\vec{v}} \coloneqq \{\vec{x} + t\vec{v} \mid t \geq 0\}$$

for some $\vec{x}, \vec{v} \in \mathbb{R}^{n_k}$. We will call $\vec{v}$ the slope *and $\vec{x}$ the basepoint of the geometric ray $R_{\vec{x},\vec{v}}$.*

2. *Define the k*th ray space *to be the set*

$$\mathit{Ray}_{(k)} \coloneqq \left\{\vec{v} \in \mathbb{R}^{n_k} \mid \exists\, \vec{x} \in \mathbb{R}^{n_k} \text{ such that } R_{\vec{x},\vec{v}} \subseteq \mathit{Im}_{(k)}\right\}$$

It is immediate that $\text{Ray}_{(k)}$ is a cone over $\vec{0}$.

Notation 4.3.3. Let $1 \leq k \leq d$ be an integer. Define $\mathcal{D}_{(k)}$ to be the collection of images (in $\mathbb{R}^{n_k}$) under $G_{(k)}$ of cells (in $\mathbb{R}^{n_0}$) of the canonical polyhedral complex $\mathcal{C}(F)$, i.e

$$\mathcal{D}_{(k)} \coloneqq \left\{G_{(k)}(C) : C \in \mathcal{C}(F)\right\}.$$

Lemma 4.3.4. *For each k, $\mathcal{D}_{(k)}$ is a finite collection of polyhedral sets.*

Proof. The canonical polyhedral complex $\mathcal{C}(F)$ is a finite collection of polyhedral sets. The restriction of $G_{(k)}$ to each cell $C \in \mathcal{C}(F)$ is affine linear, and the image of a polyhedral set under an affine-linear map is a polyhedral set. □

Lemma 4.3.5. *$\mathit{Im}_{(k)}$ is a bounded set if and only if $\mathit{Ray}_{(k)} = \{\vec{0}\}$.*

Proof. The forward direction is immediate. For the backwards direction, it suffices to prove that if $\text{Im}_{(k)}$ is unbounded, then $\text{Ray}_{(k)} = \{\vec{0}\}$ contains a nonzero element. So suppose $\text{Im}_{(k)}$ is unbounded. So there exists a sequence of points $\{x_i\}_{i\in\mathbb{N}}$ in $\text{Im}_{(k)}$ such that $|x_i| \to \infty$ as $i \to \infty$. Since $\mathcal{D}$ is finite, without loss of generality (by passing to a subsequence), we may assume there exists a single polyhedral set $D \in \mathcal{D}_{(k)}$ such that $x_i \in D$ for all i. Then, since D is convex, for each i the line segment $\overline{x_1 x_i}$ is contained in D. Moreover, the lengths of these line segments $\to \infty$ as $i \to \infty$. Let v_i be the unit vector in the direction from x_1 to x_i. Since the unit sphere is compact, we may assume without loss of generality (again, by passing to a subsequence) that $v_i \to v$ as $i \to \infty$, with $|v| = 1$. For each $t \geq 0$, the sequence $x_1 + tv_i$ converges to $x_1 + tv$, and since D is closed, $x + tv$ is in D. Hence the geometric ray $\{x_1 + tv : t \geq 0\} \subset D \subset\in \text{Im}_{(k)}$, so $v \in \text{Ray}_{(k)}$. □

The next lemma says that we can calculate how rays, and ray space, evolves as we pass through the layers of the network by looking at the image of the associated bias-free network.

Lemma 4.3.6. *Let $0 < k \leq d$.*

1. *The image under F^k of a geometric ray with slope v in Im_{k-1} is a geometric ray with slope $F^{k*}(v)$.*
2. $Ray_{(k)} = F^{(k-1)*}(Ray_{(k-1)})$.
3. $Ray_{(k)} = G^*_{(k)}(\mathbb{R}^{n_0})$

Note that the slope vector of a geometry ray may be 0 (in which case the "ray" is really a point).

Proof. Here's the proof for each part:

1. 1 Consider a geometric ray in $\text{Im}_{(k-1)}$ given by

$$R_{\vec{x},\vec{v}} = \{\vec{x} + t\vec{v} \mid t \geq 0\}$$

for some base point $\vec{x}$ and slope $\vec{v}$. The image under the affine-linear map A^k of this ray is:

$$A^k(R_{\vec{x},\vec{v}}) = \{A^k(\vec{x}) + tA^{k*}(\vec{v}) \mid t \geq 0\}$$

This transformation retains its linear nature but with the transformed slope $A^{k*}(\vec{v})$. Applying σ coordinate-wise retains the linear structure when $tA^{k*}(\vec{v})$ is non-negative, thereby giving the slope after F^k as $F^{k*}(\vec{v})$.

2. By definition, $\text{Ray}_{(k-1)}$ consists of all vectors v such that there exists a base point x with $R_{x,v} \subseteq \text{Im}_{(k-1)}$. From Part 1, every ray $R_{x,v}$ in $\text{Im}_{(k-1)}$ has its image under F^k as a ray in $\text{Im}_{(k)}$ with slope $F^{k*}(v)$. Thus, every slope $v \in \text{Ray}_{(k-1)}$ maps to $F^{k*}(v) \in \text{Ray}_{(k)}$, indicating that $\text{Ray}_{(k)} = F^{(k-1)*}(\text{Ray}_{(k-1)})$.

3. $G^*_{(k)} = F^{k*} \circ \ldots \circ F^{1*}$ represents the composite of all linear transformations applied to any vector from the input space. For every vector $\vec{v}$ in $\mathbb{R}^{n_0}$, the set $\{\vec{v}_k = G^*_{(k)}(\vec{v}) \mid t \geq 0\}$ forms a ray in $\mathbb{R}^{n_k}$, belonging to $\text{Ray}_{(k)}$. Conversely, any ray in $\text{Ray}_{(k)}$ originates from a vector in $\mathbb{R}^{n_0}$ transformed through $G^*_{(k)}$, showing $\text{Ray}_{(k)} = G^*_{(k)}(\mathbb{R}^{n_0})$.

□

A first consequence of Lemma 4.3.6 is that the image of a neural network is unlikely to be bounded, unless the depth is quite large compared to the width.

Theorem 4. *The probability that $\text{Im}_{(k)}$ is unbounded (equivalently, the probability that $\text{Ray}_{(k)} \neq \{0\}$) is $\geq \prod_{i=1}^{k}(1 - \frac{1}{2^{n_i}})$.*

Proof. Let $v \in \text{Ray}_{(i)}$, for $0 \leq i < m$. By Lemma 4.3.6 part 1, the probability that a geometric ray with slope v is mapped by layer map F^{i+1} to a geometric ray with nonzero slope (equivalently, the ray is unbounded) is the likelihood that $F^{(i+1)*}(v) \neq \vec{0}$. There is a $\frac{1}{2}$ chance that each of the n_{i+1} coordinates of $F^{(i+1)*}(v)$ is 0. Hence the probability that $F^{(i+1)*}(v) \neq \vec{0}$ is $1 - 2^{-n_{i+1}}$. Then induction on the layer index i gives that the probability that $\text{Im}_{(k)}$ is unbounded is $\geq \prod_{i=1}^{k}(1 - \frac{1}{2^{n_i}})$. □

However, there is a nonzero likelihood that the image of the network is bounded, and this likelihood increases with depth. In fact, unless the widths of the hidden layers grows extremely rapidly (approaching ∞), as the depth $\to \infty$ there is a a 100% probability that the image will be bounded.

Theorem 5. *Let $k \geq 2$. The probability that $Im_{(k)}$ is bounded (equivalently, the probability that $Ray_{(k)} = \{0\}$) is at least*

$$1 - \prod_{i=2}^{k}(1 - \frac{1}{2^{n_{i-1}n_i}}).$$

In particular, the probability that $Im_{(k)}$ is bounded $\to 1$ as $k \to \infty$.

Proof. Consider any non-final layer index $i \geq 2$. For each of the n_i neurons of F^{i*}, there is a $1/2^{n_{i-1}}$ chance that the positive half-space in $\mathbb{R}^{n_{i-1}}$ associated to that neuron has empty intersection with the positive orthant. Hence, the probability that F^{i*} acts on Im_{i-1} (which is contained in the positive orthant) as the 0-map is at least $\frac{1}{2^{n_{i-1}n_i}}$.

Then, the probability that there exists $2 \leq i \leq k$ such that F^{i*} acts as the 0 map on Im_{i-1} is at least

$$1 - \prod_{i=2}^{k}(\text{prob. } F^{i*} \text{ does NOT act as 0}) = 1 - \prod_{i=2}^{k}(1 - \text{prob. } F^{i*} \text{ acts as 0})$$
$$\geq 1 - \prod_{i=2}^{k}(1 - \frac{1}{2^{n_{i-1}n_i}}).$$

□

4.3.2 Angle degeneracy and distance variability

As a first pass at understanding why ReLU neural networks distort the geometry of data, suppose $\mathcal{P}$ is the uniform distribution on the unit sphere in the input space, $\mathbb{R}^n$, and consider sampling points from $\mathcal{P}$. By symmetry, the expected value of the angular distance between a pair of such points x, x' is $\frac{\pi}{2}$, since (viewing x as the north pole) x' has equal probability of being in the northern and southern hemispheres. On the other hand, the

expected angular distance between the images, $F_\ell(x), F_\ell(x') \in \mathbb{R}^m$, of any two points is strictly less than $\frac{\pi}{2}$, because the image of any ReLU layer map is contained in the non-negative orthant,

$$\mathbb{R}^{\geq 0}_m := \{x \in \mathbb{R}^m \ : \ x_i \geq 0 \ \forall \ i\}.$$

Moreover, the *maximum* angular distance between points in the non-negative orthant is $\frac{\pi}{2}$, and there is a nonzero probability that there exist pairs of points with angular distance less than $\frac{\pi}{2}$. The rate at which the expected angular distance degenerates per layer for infinitely wide networks has been established in [37]. For finite width networks, there is uncertainty in the error that grows as the width decreases.

In a different direction, it was proved in [31] that when weights and biases are sampled using the standard He normalization, the expected value of the norm of data points is preserved by ReLU layer maps. Note that this result is an expected value calculation for any *fixed data point*, averaging over the *parameters*.

Do angles between vectors get squished by ReLU?

Given any two vectors $u, v \in \mathbb{R}^n$, denote the angle (in $[0, \pi]$) between them by $\angle(u, v)$.

Lemma 4.3.7. *Fix any two vectors $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$ in $\mathbb{R}^n$. Let $a' = (a_1, \ldots, a_{n-1}, 0)$ and $b' = (b_1, \ldots, b_{n-1}, 0)$, and assume $|a'| = |b'| = 1$. Then*

$$\frac{a_n b_n}{|a_n| + |b_n| + |a_n b_n|} \geq a' \cdot b' \tag{4.1}$$

implies $\angle(a, b) \leq \angle(a', b')$.

Proof. First, recall that for any two vectors u, v, $\cos(\angle(u, v)) = \frac{u \cdot v}{|u||v|}$ and $\cos$ is monotone decreasing on $[0, \pi]$. Thus, it suffices to prove

$$\frac{a \cdot b}{|a||b|} \geq \frac{a' \cdot b'}{|a'||b'|}.$$

The triangle inequality and the assumption $|a'| = |b'| = 1$ give

$$\frac{a \cdot b}{|a||b|} = \frac{a' \cdot b' + a_n b_n}{|a' + (0, \ldots, 0, a_n)||b' + (0, \ldots, 0, b_n)|}$$
$$\geq \frac{a' \cdot b' + a_n b_n}{(|a'| + |a_n|)(|b'| + |b_n|)} = \frac{a' \cdot b' + a_n b_n}{|a'||b'| + |a_n| + |b_n| + |a_n b_n|)}$$

Basic algebraic manipulation gives that for any real numbers x, y, e, f,

$$\frac{x+e}{y+f} \geq \frac{x}{y} \iff \frac{e}{f} \geq \frac{x}{y}.$$

Hence the assumption (4.1) implies

$$\frac{a' \cdot b' + a_n b_n}{|a'||b'| + |a_n| + |b_n| + |a_n b_n|)} \geq \frac{a' \cdot b'}{|a'||b'|}.$$

□

Remark 4.3.8. We lose some sharpness in the proof above when we use the triangle inequality. We can do better, obtaining and if and only if characterization:

$$\left(\frac{a \cdot b}{|a||b|}\right)^2 = \frac{(a' \cdot b' + a_n b_n)^2}{|a'|^2|b'|^2 + a_n^2|b'|^2 + b_n^2|a'|^2 + a_n^2 b_n^2} = \frac{(a' \cdot b')^2 + 2a_n b_n (a' \cdot b') + (a_n b_n)^2}{|a'|^2|b'|^2 + a_n^2|b'|^2 + b_n^2|a'|^2 + a_n^2 b_n^2}.$$

Hence

$$\left(\frac{a \cdot b}{|a||b|}\right)^2 \geq \left(\frac{a' \cdot b'}{|a'||b'|}\right)^2$$

if and only if

$$\frac{2a_n b_n (a' \cdot b') + a_n^2 b_n^2}{a_n^2 + b_n^2 + a_n^2 b_n^2} \geq (a' \cdot b')^2$$

4.3.3 Geometry of distortion of a single layer map

We begin with a fundamental result showing that the average squared magnitude of the ReLU-transformed vectors on the unit sphere is half of the original:

Lemma 4.3.9. *For all $n \in \mathbb{N}$,*

$$\int_{S^{n-1}} |\mathit{Relu}(\vec{x})|^2 \, dA(\vec{x}) = \frac{1}{2}.$$

Proof.

$$\int_{S^{n-1}} |\text{Relu}(\vec{x})|^2 \, dA(\vec{x}) = \int_{S^{n-1}} \sum_{i=1}^{n} x_i^2 1_{x_i>0} \, dA(\vec{x}) = \sum_{i=1}^{n} \int_{S^{n-1}} x_i^2 1_{x_i>0} \, dA(\vec{x}).$$

But

$$\int_{S^{n-1}} x_i^2 1_{x_i>0} \, dA(\vec{x}) = \frac{1}{2} \int_{S^{n-1}} x_i^2 \, dA(\vec{x}) = \frac{1}{2n},$$

where the first equality is due to symmetry and the second to equation (4.2). The result follows. □

Extending the previous result, we show that for any $n \times n$ matrix M with singular values $\{\sigma_i\}_{i=1}^n$, the average squared magnitude of the ReLU-transformed vectors is related to the singular values of M. This proposition illustrates how the transformation by M affects the overall magnitude of the data vectors.

Proposition 4.3.10 (Average distortion of magnitudes)**.** *Let M be a $n \times n$ matrix and denote its singular values $\{\sigma_i\}_{i=1}^m$. Then*

$$\int_{S^{n-1}} |\mathit{Relu} \circ M(\vec{x})|^2 \, dA(\vec{x}) = \frac{1}{2} \int_{S^{n-1}} |M(\vec{x})|^2 \, dA(\vec{x}) = \frac{1}{2n} \sum_{i=1}^{n} \sigma_i^2.$$

Proof. Let $\vec{M}_i$ denote the ith row of the matrix M. Then for any $\vec{x} \in \mathbb{R}^n$,

$$\text{Relu} \circ M(\vec{x}) = ((\vec{M}_1 \cdot \vec{x}) 1_{\vec{M}_1 \cdot \vec{x} > 0}, \ldots, (\vec{M}_n \cdot \vec{x}) 1_{\vec{M}_1 \cdot \vec{x} > 0})^T,$$

so

$$|\text{Relu} \circ M(\vec{x})|^2 = \sum_{i=1}^{n} (\vec{M}_i \cdot \vec{x})^2 1_{\vec{M}_i \cdot \vec{x} > 0}.$$

Hence, using symmetry for the last equality in the expression below, we have

$$\begin{aligned}\int_{S^{n-1}} |\text{Relu} \circ M(\vec{x})|^2 \, d(A)(\vec{x}) &= \int_{S^{n-1}} \sum_{i=1}^{n} (\vec{M}_i \cdot \vec{x})^2 1_{\vec{M}_i \cdot \vec{x} > 0} \, dA(\vec{x}) \\ &= \sum_{i=1}^{n} \int_{S^{n-1}} (\vec{M}_i \cdot \vec{x})^2 1_{\vec{M}_i \cdot \vec{x} > 0} \, dA(\vec{x}) = \sum_{i=1}^{n} \frac{1}{2} \int_{S^{n-1}} (\vec{M}_i \cdot \vec{x})^2 \, dA(\vec{x}) \\ &= \frac{1}{2} \int_{S^{n-1}} \sum_{i=1}^{n} (\vec{M}_i \cdot \vec{x})^2 \, dA(\vec{x}) = \frac{1}{2} \int_{S^{n-1}} \sum_{i=1}^{n} (\vec{M}_i \cdot \vec{x})^2 \, dA(\vec{x}) \\ &= \frac{1}{2} \int_{S^{n-1}} |M(\vec{x})|^2 \, dA(\vec{x}).\end{aligned}$$

This proves the first equality.

For the second equality, by symmetry, we may assume M to be a diagonal matrix with entries $\{\sigma_i\}_{i=1}^n$. Then

$$\int_{S^{n-1}} |M(\vec{x})|^2 \, dA(\vec{x}) = \int_{S^{n-1}} \sum_{i=1}^{n} \sigma_i^2 x_i^2 \, dA(\vec{x}) = \sum_{i=1}^{n} \sigma_i^2 \int_{S^{n-1}} x_i^2 \, dA(\vec{x}).$$

But again by symmetry,

$$\int_{S^{n-1}} x_i^2 \, dA(\vec{x}) = \frac{1}{n} \int_{S^{n-1}} \sum_{i=1}^{n} x_i^2 \, dA(\vec{x}) = \frac{1}{n} \int_{S^{n-1}} dA(\vec{x}) = \frac{1}{n}. \tag{4.2}$$

□

We analyze how the distance between pairs of points is distorted by ReLU transformations. Our results show that the average squared distance between ReLU-transformed points is a quarter of the original distance:

$$\int_{S^{n-1}} \int_{S^{n-1}} |\text{Relu}(x) - \text{Relu}(y)|^2 \, dA(x) dA(y) = \frac{1}{4} \int_{S^{n-1}} \int_{S^{n-1}} |x - y|^2 \, dA(x) dA(y).$$

Furthermore, when considering a matrix M, we show that the average distortion of distances between transformed points is closely related to the singular values of M:

$$\int_{S^{n-1}}\int_{S^{n-1}} |\text{Relu}\circ M(x)-\text{Relu}\circ M(y)|^2\, dA(x)dA(y) = \left(\frac{1}{4n}\sum_{i=1}^{n}\sigma_i^2\right)\int_{S^{n-1}}\int_{S^{n-1}} |x-y|\, dA(x)dA(y).$$

We formalize these results in the following lemmas and propositions

Lemma 4.3.11.

$$\int_{S^{n-1}}\int_{S^{n-1}} |\mathit{Relu}(x)-\mathit{Relu}(y)|^2\, dA(x)dA(y) = \frac{1}{4}\int_{S^{n-1}}\int_{S^{n-1}} |x-y|^2\, dA(x)dA(y)$$

Proposition 4.3.12 (Average distortion of distance between points)**.** *Let M be an $n\times n$ matrix and denote its singular values $\{\sigma_i\}_{i=1}^n$. Then*

$$\int_{S^{n-1}}\int_{S^{n-1}} |\mathit{Relu}\circ M(x)-\mathit{Relu}\circ M(y)|^2\, dA(x)dA(y) = \frac{1}{4}\int_{S^{n-1}}\int_{S^{n-1}} |M(\vec{x})-M(y)|^2\, dA(x)\, dA(y) = \left(\frac{1}{4n}\sum_{i=1}^{n}\sigma_i^2\right)\int_{S^{n-1}}\int_{S^{n-1}} |\vec{x}-\vec{y}|^2\, dA(\vec{x})dA(\vec{y}).$$

Lemma 4.3.13. *Let M be an $n\times n$ matrix and denote its singular values $\{\sigma_i\}_{i=1}^n$.*

$$\int_{S^{n-1}}\int_{S^{n-1}} |M(\vec{x})-M(\vec{y})|^2\, dA(\vec{x})\, dA(\vec{y}) = \left(\frac{1}{n}\sum_{i=1}^{n}\sigma_i^2\right)\int_{S^{n-1}}\int_{S^{n-1}} |\vec{x}-\vec{y}|\, dA(\vec{x})dA(\vec{y}).$$

Proof. By symmetry, we may assume M to be a diagonal matrix with entries $\{\sigma_i\}_{i=1}^n$.

Then

$$\int_{S^{n-1}} \int_{S^{n-1}} |M(\vec{x}) - M(y)|^2 \; dA(\vec{x}) dA(\vec{y}) = \int_{S^{n-1}} \int_{S^{n-1}} \sum_{i=1}^{n} (\sigma_i x_i - \sigma_i y_i)^2 \, dA(\vec{x}) dA(\vec{y}) = \sum_{i=1}^{n} \sigma_i^2 \int_{S^{n-1}} \int_{S^{n-1}} (x_i - y_i)^2 \, dA(\vec{x}) dA(\vec{y})$$

Also symmetry,

$$\int_{S^{n-1}} \int_{S^{n-1}} (x_i - y_i)^2 \, dA(\vec{x}) dA(\vec{y}) = \frac{1}{n} \int_{S^{n-1}} \int_{S^{n-1}} \sum_{i=1}^{n} (x_i - y_i)^2 \, dA(\vec{x}) dA(\vec{y}) = \frac{1}{n} \int_{S^{n-1}} \int_{S^{n-1}} |\vec{x} - \vec{y}| \, dA(\vec{x}) dA(\vec{y}).$$

Combining these two equations yields

$$\int_{S^{n-1}} \int_{S^{n-1}} |M(\vec{x}) - M(y)|^2 \; dA(\vec{x}) dA(\vec{y}) = \sum_{i=1}^{n} \frac{\sigma_i^2}{n} \int_{S^{n-1}} \int_{S^{n-1}} |\vec{x} - \vec{y}| \, dA(\vec{x}) dA(\vec{y}).$$

□

4.3.4 Expected value of distortion by a single layer map

The *Frobenius norm* of a matrix $M = [m_{j,k}]$ is defined as

$$||M||_F := \sqrt{\sum_{j,k} m_{j,k}^2}.$$

The *variance* of a random variable X is

$$\text{Var}(X) = \mathbf{E}((X - \text{mean}(X))^2).$$

This norm is instrumental in our analysis of the expected distortions caused by ReLU transformations.

The following two lemmas are standard; proofs are included for completeness.

Lemma 4.3.14. *Let M be a $n \times n$ real matrix and denote its singular values $\{\sigma_i\}_{i=1}^n$. Then*

$$\sum_{i=1}^{n} \sigma_i^2 = ||M||_F^2$$

.

Proof. Write the singular value decomposition of M as $M = U\Sigma V^T$. Then using the facts that i) $\sum \sigma_i^2 = \text{tr}(\Sigma\Sigma^T)$, and ii) $\text{tr}(AB) = \text{tr}(BA)$, we have

$$||M||_F^2 = \text{tr}(MM^T) = \text{tr}(U\Sigma V^T V\Sigma^T U^T) = \text{tr}(\Sigma\Sigma^T) = \sum \sigma_i^2.$$

□

Lemma 4.3.15. *Let M be a $n \times n$ matrix with entries drawn from a Gaussian distribution on $\mathbb{R}$ with mean 0 and variance σ^2. Then*

$$\mathbf{E}_M \left(\sum_{i=1}^{n} \sigma_i^2 \right) = \mathbf{E}_M \left(||M||_F^2 \right) = n^2 \cdot \sigma^2.$$

Proof. The first equality is Lemma 4.3.14.

Since each M_{ij} is Gaussian with mean 0 and variance σ^2, the expected value of M_{ij}^2 is:

$$\mathbf{E}_M \left(M_{ij}^2 \right) = \sigma^2$$

There are $n \times n = n^2$ entries in the matrix M. Hence:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \mathbf{E}_M\left(M_{ij}^2\right) = n^2 \cdot \sigma^2$$

Therefore, we have:

$$\mathbf{E}_M\left(||M||_F^2\right) = n^2 \cdot \sigma^2$$

□

We can now combine these standard facts with the geometry results from the previous section. For a matrix M with entries drawn from a Gaussian distribution on $\mathbb{R}$ with mean 0 and variance σ^2, we demonstrate that the expected average squared magnitude of the transformed vectors is $n \cdot \sigma^2$ and applying Relu would result in the expected value of $\frac{n}{2} \cdot \sigma^2$.

Proposition 4.3.16. *Let M be a $n \times n$ matrix with entries drawn randomly from a Gaussian distribution on $\mathbb{R}$ with mean 0 and variance σ^2. Then*

$$\mathbf{E}_M\left(\int_{S^{n-1}} |M(x)|^2 \, dA(x)\right) = n \cdot \sigma^2$$

and

$$\mathbf{E}_M\left(\int_{S^{n-1}} |\mathit{Relu} \circ M(x)|^2 \, dA(x)\right) = \frac{n}{2} \cdot \sigma^2.$$

Proof. By the second equality of Proposition 4.3.10 and Lemma 4.3.14, for any $n \times n$ matrix M with singular values $\{\sigma_i\}_{i=1}^n$, we have

$$\int_{S^{n-1}} |M(x)|^2 \, dA(x) = \frac{1}{n}\sum_{i=1}^{n} \sigma_i^2 = \frac{1}{n}||M||_F^2.$$

Therefore Lemma 4.3.15 gives

$$\mathbf{E}_M\left(\int_{S^{n-1}} |M(x)|^2 \, dA(x)\right) = \mathbf{E}_M\left(\frac{1}{n}||M||_F^2\right) = \frac{1}{n}n^2\sigma^2 = n \cdot \sigma^2.$$

The proof of the second equation is the same, but with an additional factor of $\frac{1}{2}$. □

Then we analyze the expected distance distortion between pairs of points after ReLU transformations. For matrices M with Gaussian-distributed entries, the expected distortion of distances is $n\sigma^2$

Proposition 4.3.17. *Let M be a $n \times n$ matrix with entries drawn randomly from a Gaussian distribution on $\mathbb{R}$ with mean 0 and variance σ^2. Then*

$$\frac{\mathbf{E}_M\left(\int_{S^{n-1}}\int_{S^{n-1}} |\mathit{Relu} \circ M(x) - \mathit{Relu} \circ M(y)|^2 \, dA(x)dA(y)\right)}{\int_{S^{n-1}}\int_{S^{n-1}} |x-y|^2 \, dA(x)dA(y)} = n\sigma^2$$

and

$$\frac{\mathbf{E}_M\left(\int_{S^{n-1}}\int_{S^{n-1}} |M(x) - M(y)|^2 \, dA(x)dA(y)\right)}{\int_{S^{n-1}}\int_{S^{n-1}} |x-y|^2 \, dA(x)dA(y)} = \frac{n\sigma^2}{4}$$

Proof. Combining Proposition 4.3.12 with Lemma 4.3.15 yields

$$\frac{\mathbf{E}_M\left(\int_{S^{n-1}}\int_{S^{n-1}} |M(x) - M(y)|^2 \, dA(x)dA(y)\right)}{\int_{S^{n-1}}\int_{S^{n-1}} |x-y|^2 \, dA(x)dA(y)} = \frac{1}{n}\mathbf{E}_M\left(\sum_{i=1}^{n} \sigma_i^2\right) = n\sigma^2.$$

The proof of the second inequality is the same, with an additional factor of $\frac{1}{4}$. □

Remark 4.3.18. The He normalization uses variance $\sigma^2 = 2/n$. So, for the He normalization, Proposition 4.3.16 gives that for a single layer map, average magnitude is preserved

$$\mathbf{E}_M\left(\int_{S^{n-1}} |\text{Relu} \circ M(x)|^2 \, dA(x)\right) = 1$$

but Proposition 4.3.17 gives that average distance between points is halved

$$\frac{\mathbf{E}_M\left(\int_{S^{n-1}}\int_{S^{n-1}} |\text{Relu} \circ M(x) - \text{Relu} \circ M(y)|^2 \, dA(x)dA(y)\right)}{\int_{S^{n-1}}\int_{S^{n-1}} |x-y|^2 \, dA(x)dA(y)} = \frac{1}{2}.$$

4.4 Experimental Results

In this section, we investigate the activity patterns of neurons in ReLU networks under various configurations and depths. Our primary aim is to understand the conditions un-

der which neurons are ”stably active” (the ReLU is on for all observed inputs) or ”stably inactive” (the ReLU is off for all observed inputs). It is important to emphasize that these properties are empirical and depend on the observed data rather than the entire input domain.

We conducted experiments using different data distributions and network configurations to explore how these factors influence neuron activity in ReLU networks. The focus was on two main setups: a network with layers of constant width 100 (see Figure 4.1), and another with layers of constant width 15 (see Figure 4.2). These configurations were chosen to observe how varying the network architecture affects the stability and activity of neurons across different layers. The data distribution used in these experiments is standard normal, and the networks are fully connected ReLU networks with layers of constant width as specified. Our objective was to calculate the activity of neurons, defined as the fraction of input points for which the ReLU is active, and to plot the activity as histograms.

We define the ”activity” of a neuron as the proportion of the input points for which the ReLU activation function outputs a non-zero value (i.e., the pre-activation input is positive). By calculating the activity for all neurons and plotting the results as histograms, we aim to identify patterns and assess the fractions of neurons that are stably inactive or stably active.

4.4.1 Depth Analysis

Given the random initialization of weights in ReLU networks, one might initially conceptualize each neuron as having an equal probability of being active or inactive for a given input distribution. Under this assumption, we hypothesized that neuron activations would predominantly cluster around intermediate activity levels, rather than at the extremes of 0 or 1. This would suggest a balanced system where neurons are neither consistently active nor inactive across the inputs.

Our initial observations seemed to support this hypothesis, particularly in the earlier layers of the networks, such as hidden layer 4 in Figures 4.1 and 4.2. In these layers, the distribution of neuron activity appears more centered, with fewer neurons being stably active or inactive.

However, as the network depth increases, a notable shift in the activity distribution emerges. Contrary to our initial expectation, the histograms show an increasing accumulation of neuron activations at the lowest and highest bins. This trend indicates that neurons in deeper layers are more likely to be either consistently active or consistently inactive, leading to a more polarized activity distribution. This phenomenon suggests that, as the depth of the network increases, the behavior of neurons tends to become more extreme, with a significant portion of neurons consistently remaining in an on or off state.

These observations highlight the impact of network depth on neural stability and activity, diverging from the balanced activity distribution initially anticipated. It underscores the importance of considering network depth and configuration when analyzing neuron activity patterns, as they significantly influence the overall behavior of ReLU networks.

The key observations from the histograms of neuron activity are:

- The activity histograms exhibit a range of neuron activities with varying frequencies.
- A notable fraction of neurons fall into the lowest bin, indicating a significant number of neurons are stably inactive.
- Similarly, there is a considerable fraction in the highest bin, suggesting that many neurons are stably active.
- The histogram patterns indicate that while some neurons consistently remain inactive or active, a substantial number fluctuate between these states depending on the input.

As we analyze how the histograms change with network depth, we observe that they become increasingly skewed towards the extremes. There is a higher likelihood of neurons being either stably active or stably inactive as depth increases. This observation supports the hypothesis that deeper layers in the network exhibit more polarized neuron activities. The deeper the layer, the more pronounced the stable activity or inactivity of neurons becomes.

Comparing networks of different widths, we find that in the network with width 100, the activity of neurons shows a clear trend towards extremes as depth increases. This indicates that deeper layers tend to have neurons that are either mostly active or mostly inactive for the given input data. In contrast, for the network with width 15, the effect of increasing depth on neuron activity is less pronounced. While there is still some polarization, neurons in deeper layers maintain a more balanced activity distribution compared to the wider network.

These observations suggest that layer width also plays a role in neuron activity patterns. Wider layers (width 100) tend to push neuron activity towards the extremes as depth increases, leading to higher neural stability in terms of consistent activation patterns. In contrast, narrower layers (width 15) maintain a more balanced activity distribution, resulting in a more dynamic network behavior where neurons are more likely to fluctuate between active and inactive states.

4.4.2 Width Analysis

In this section, we analyze the activity of neurons in ReLU networks with varying layer widths while keeping the network depth constant. The objective is to understand how the width of the layers affects neuron stability and activation patterns. We considered three different network configurations with constant depths of 3 layers (see Figure 4.3), 12 layers (see Figure 4.4), and 64 layers (see Figure 4.5). For each configuration, the width of the layers was varied, and histograms were generated to depict the fraction of neurons active for different input sizes.

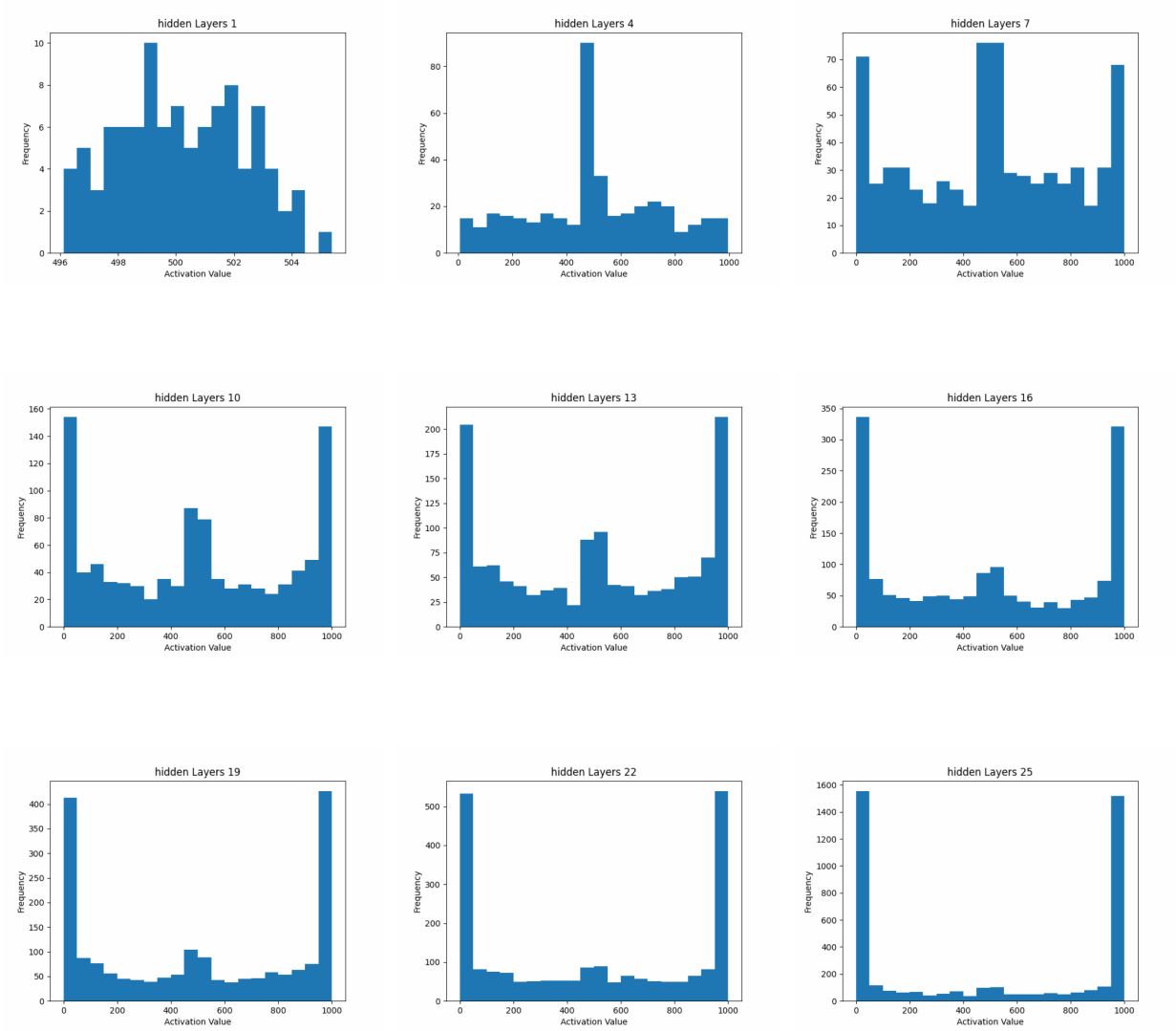

Figure 4.1: This figure shows the histograms of neuron activity for the network with a uniform distribution and constant width of 100 across all layers. Each subplot represents the activity distribution at different layers, illustrating how neuron activity becomes more polarized (i.e., more stably active or inactive) as the depth increases.

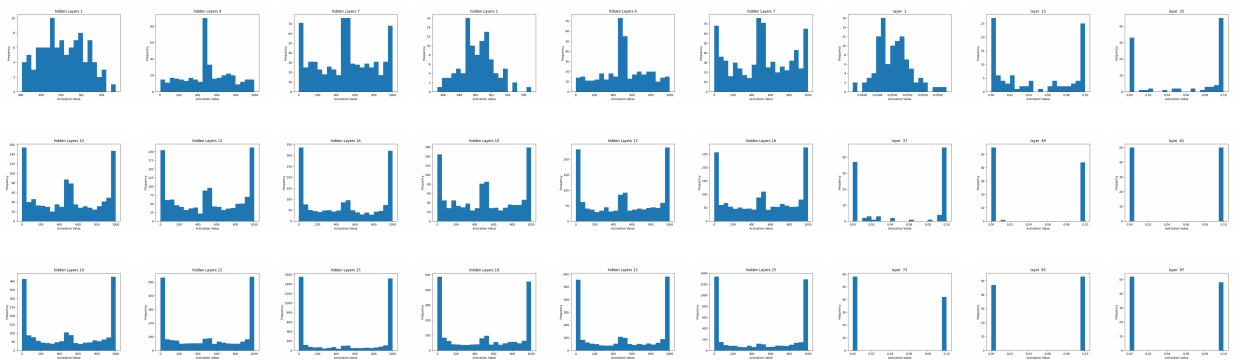

Figure 4.2: This figure presents the histograms of neuron activity for the network with a constant width of 15 across all layers. Similar to the previous setup, the activity distribution at various layers is shown, providing insights into how a smaller layer width influences neuron activity and stability.

In the network with a constant depth of 3 layers (Figure 4.3), as the layer width increases, the neuron activity distribution becomes more balanced. There is a moderate number of neurons that are stably active (activity close to 1) and stably inactive (activity close to 0), but a significant number of neurons exhibit intermediate activity levels. This balanced distribution suggests that in shallow networks, neurons exhibit a diverse range of activation patterns, with only moderate tendencies towards stable activity or inactivity as the width increases.

For the network with a constant depth of 12 layers (Figure 4.4), we observe a more pronounced shift towards polarization compared to the depth 3 network. As the width increases, a greater fraction of neurons become stably active or inactive. The middle bins, representing neurons with intermediate activity levels, are less populated compared to the depth 3 network. This indicates a stronger tendency towards extreme activation values.

In the network with a constant depth of 64 layers (Figure 4.5), a significant fraction of neurons are either fully active or fully inactive, with very few neurons exhibiting intermediate activity levels. This extreme polarization is consistent across all layer widths. The deep network enforces a high level of neuron stability, with neurons predominantly

adopting stable activation or inactivity states. This behavior suggests that network depth has a more significant influence on neuron stability than layer width.

- Shallow Network (Depth 3):
  - Exhibits a balanced distribution of neuron activities across varying widths.
  - Maintains a diverse range of activations with moderate polarization towards extreme values.
- Moderate Depth Network (Depth 12):
  - Shows increased polarization compared to the shallow network.
  - Neurons are more likely to adopt stable activation states as the width increases, reducing the diversity of activation patterns.
- Deep Network (Depth 64):
  - Demonstrates strong polarization towards extreme activation values.
  - Neurons predominantly adopt stable activation or inactivity states, with minimal intermediate activity.

These observations highlight the interplay between network depth and width in influencing neuron activity patterns. While increased width tends to polarize neuron activations, the effect is more pronounced in deeper networks, leading to higher neuron stability and more deterministic activation patterns.

4.4.3 Layer Analysis

We further analyze the activation histograms for different layers within ReLU networks of varying architectures. The aim is to understand the general patterns in activation distributions and how they evolve as we move deeper into the network. We considered

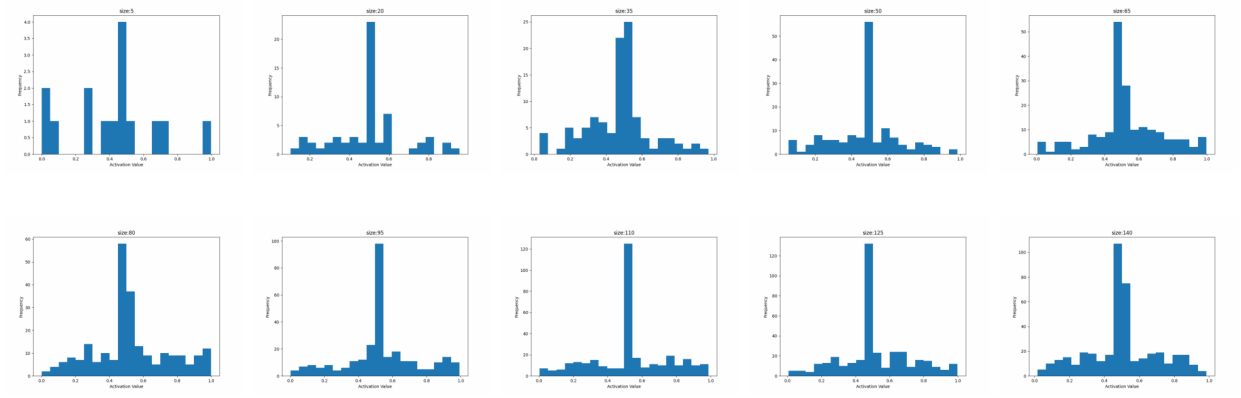

Figure 4.3: Histograms showing the activity of neurons across different input sizes in a ReLU network with a constant depth of 3 layers. The distribution remains relatively balanced, with moderate polarization towards extreme activation values.

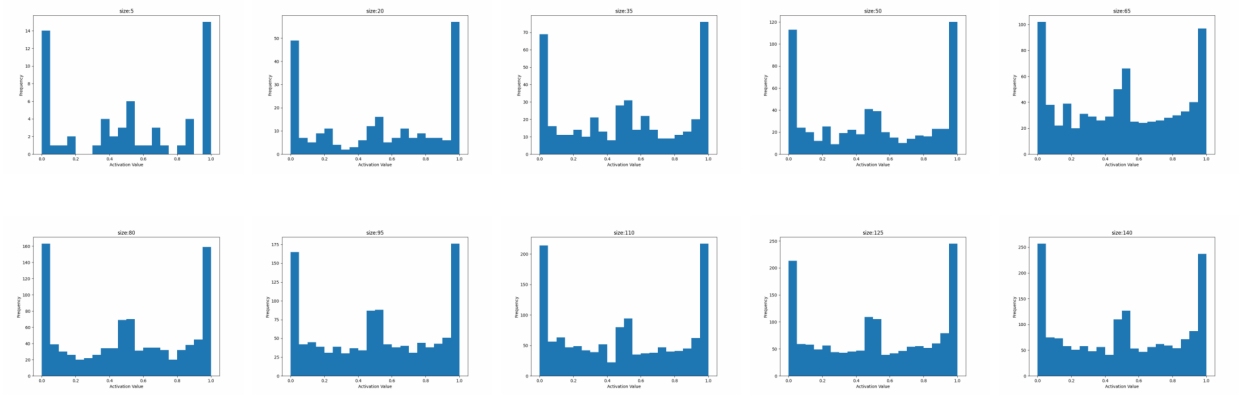

Figure 4.4: Histograms showing the activity of neurons across different input sizes in a ReLU network with a constant depth of 12 layers. The distribution shows increased polarization towards extreme activation values compared to the depth 3 network.

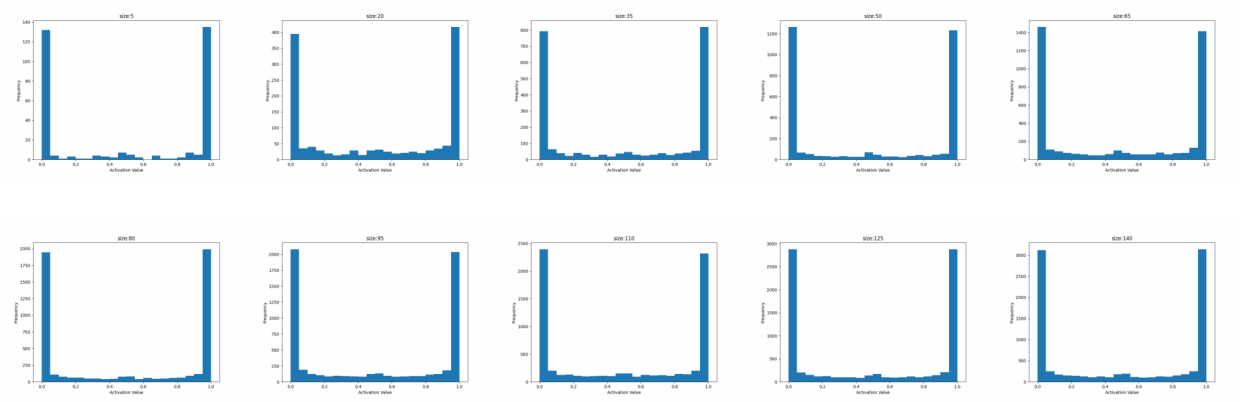

Figure 4.5: Histograms showing the activity of neurons across different input sizes in a ReLU network with a constant depth of 64 layers. The distribution exhibits strong polarization towards extreme activation values, indicating a high level of neuron stability.

several architectures with different layer configurations. For each architecture, we plotted the histograms of neuron activations at various layers to observe the distribution and stability of activations.

In the initial layers (e.g., layer 1), the activation histograms generally show a balanced distribution of activation values. Most neurons have activity levels around the middle of the spectrum, indicating a diverse range of activations. This balanced distribution is consistent across all architectures, suggesting that early layers in ReLU networks tend to maintain a variety of activation states without significant polarization (see Figure 4.6).

As we move to intermediate layers (e.g., layers 3 to 7), the histograms begin to show signs of polarization. There is a noticeable shift where some neurons start accumulating at the extreme bins (close to 0 or 1), although the middle bins still retain a considerable number of neurons. This transition phase indicates that as the network depth increases, neurons start to stabilize their activations, either becoming more consistently active or inactive.

In the deeper layers (e.g., layers 10 to 21), the histograms converge towards extreme bin accumulation. Most neurons now have activity values close to 0 or 1, with very few neurons exhibiting intermediate activity levels. This convergence towards stable acti-

vation states suggests that deeper layers in ReLU networks enforce more deterministic activation patterns, with neurons predominantly settling into either an active or inactive state.

These findings align with our previous observations on neuron stability and polarization in ReLU networks. Earlier, we noted that increased network depth tends to push neuron activations towards more extreme values. The current analysis further supports this, showing that regardless of the specific architecture, there is a common trend of activation convergence in deeper layers.

This convergence is particularly pronounced between layers 7 to 11. Beyond this point, the majority of neurons in the network exhibit extreme activations, either fully active or inactive. This behavior indicates that the network's ability to maintain diverse activation states diminishes as depth increases, leading to a more deterministic and stable neuron activation pattern.

Visualization of Data Transformation through Layers

To gain further insight into how data is transformed as it propagates through the network layers, we performed a Principal Component Analysis (PCA) of the activations at each layer. PCA is a statistical technique that transforms data into a set of linearly uncorrelated variables called principal components, ordered by the amount of variance they capture.

The steps involved in applying PCA to our data are as follows:

1. **Data Normalization:** Normalize the data to have zero mean. Given data matrix X, the normalized data X_{norm} is:

$$X_{\text{norm}} = X - \mu,$$

   where μ is the mean of X.

2. **Covariance Matrix Calculation:** Compute the covariance matrix Σ:

$$\Sigma = \frac{1}{n} X_{\text{norm}}^{\top} X_{\text{norm}},$$

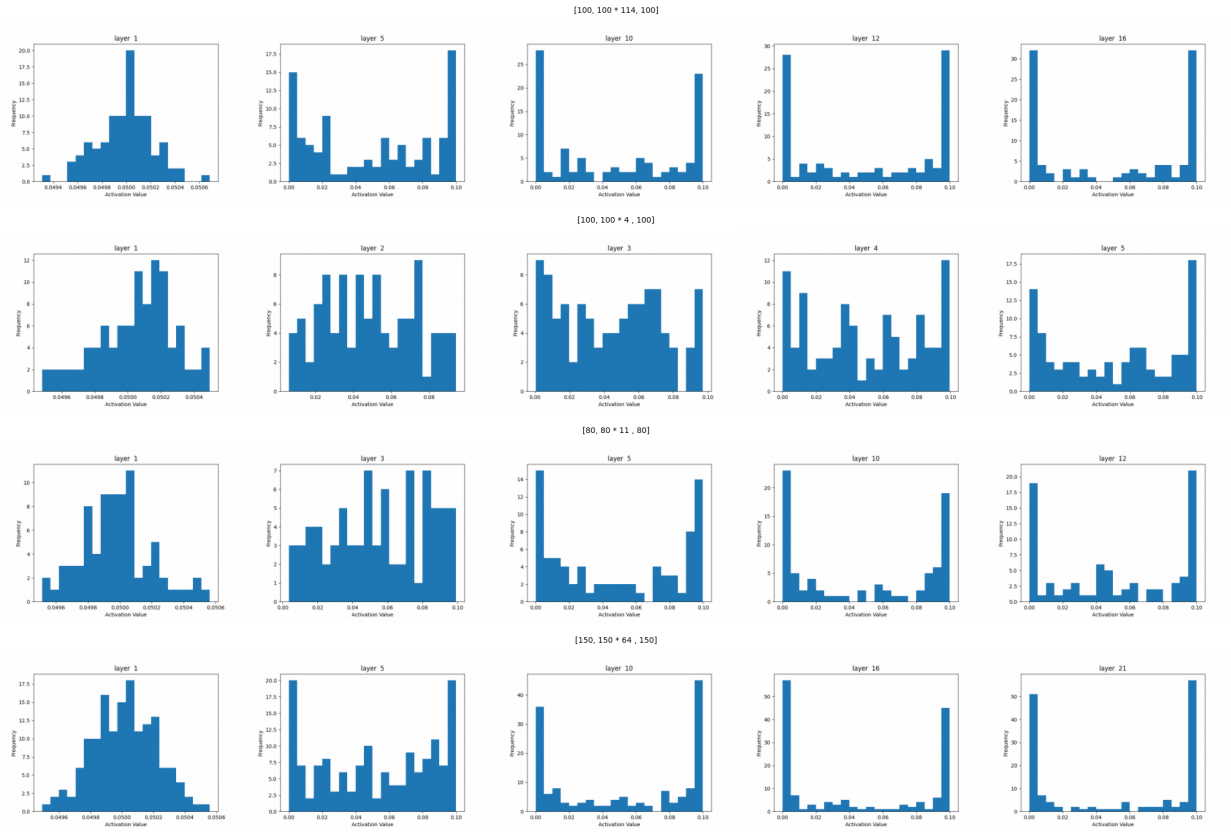

Figure 4.6: Histograms showing the activity of neurons across different input sizes in a ReLU network with a constant depth of 64 layers. The distribution exhibits strong polarization towards extreme activation values, indicating a high level of neuron stability.

where n is the number of samples.

3. **Eigenvalue Decomposition:** Perform eigenvalue decomposition on the covariance matrix:

$$\Sigma v = \lambda v,$$

where λ are the eigenvalues and v are the corresponding eigenvectors.

4. **Projection:** Project the data onto the eigenvectors corresponding to the largest eigenvalues to obtain a lower-dimensional representation:

$$Y = X_{\text{norm}} V,$$

where V is the matrix of selected eigenvectors.

For each layer of the network, we performed these steps to visualize how the data distribution changes through the layers.

In the initial layers, the data points are relatively spread out, indicating that the data retains much of its original variance, with no significant transformation yet. As we move to intermediate layers, the data begins to transform significantly. For instance, in Figure 4.8, the distribution starts to skew, indicating the network's feature extraction process. The data points project more distinctly along the principal components, showing the network's ability to separate different features.

In the deeper layers, the data points converge more closely, forming tighter clusters or lines. This reflects the network's increased capability to capture and compress essential features. In some cases, the data ultimately forms a single line, indicating extreme compression and feature extraction. This convergence is a common pattern, highlighting the network's capacity to distill the input data into the most significant features for final classification or output.

Key observations from this analysis include:

- **Effect of Network Size:** Smaller networks (e.g., with constant width of 100) compress the data into a line earlier compared to larger networks. This suggests that smaller networks may be more aggressive in feature compression, potentially leading to loss of information.

- **Preservation of Feature Diversity:** Larger networks with increasing layer sizes maintain more complex structures for a longer duration before converging. This indicates that larger networks may be better at preserving feature diversity across layers.

- **Convergence to Line:** Regardless of the network size, all networks eventually compress the data into a lower-dimensional structure as they reach the final layers. This

convergence illustrates the network's role in extracting and focusing on the most salient features necessary for the task.

These observations provide insight into how network architecture affects data transformation and feature extraction within ReLU networks. They suggest that both network depth and width play crucial roles in determining how information is processed and compressed through the layers.

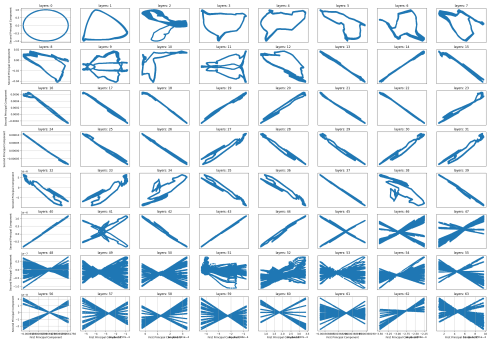

Figure 4.7: PCA visualization of the image cloud at each layer for a ReLU network with points on a sphere of sphere (mean 0, std 1). The network architecture is [20, 20 *63]. The visualizations show the transformation of data as it propagates through the network layers, highlighting the feature extraction and compression capabilities.

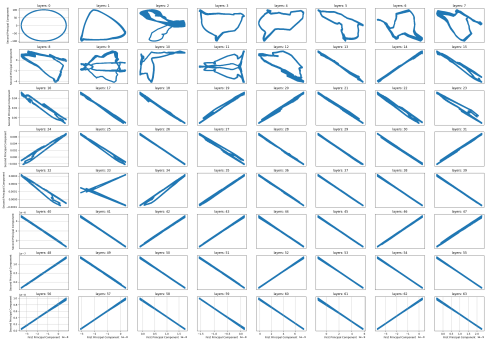

Figure 4.8: PCA visualization of the image cloud at each layer for a zero ReLU network with points on a sphere of sphere (mean 0, std 100). The network architecture is [20, 20 *63]. The visualizations show the transformation of data as it propagates through the network layers, highlighting the feature extraction and compression capabilities.

4.5 Notions of Dimension and Analysis of Network Metrics

In this section, we introduce various notions of dimension relevant to the analysis of ReLU networks and their associated polyhedral complexes. These definitions aim to

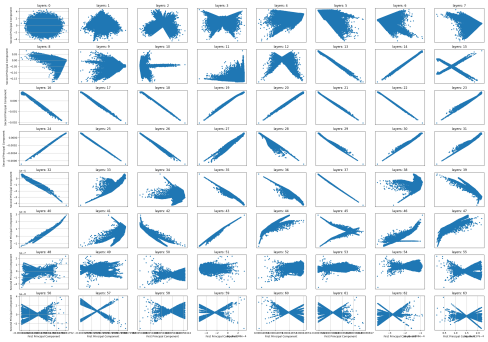

Figure 4.9: PCA visualization of the image cloud at each layer for a ReLU network with normal distribution input (mean 0, std 20) and constant layer width of 20 neurons. The network architecture is [100, 100 * 63]. The visualizations show the transformation of data as it propagates through the network layers, highlighting the feature extraction and compression capabilities.

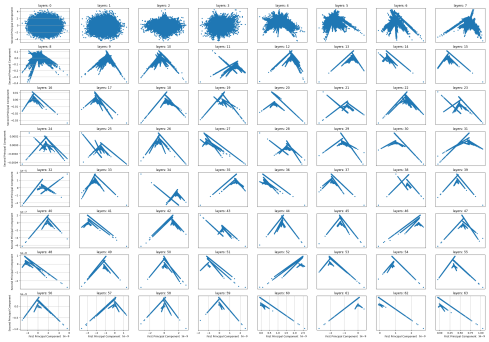

Figure 4.10: PCA visualization of the image cloud at each layer for a zero biased ReLU network with normal distribution input (mean -20, std 100) and constant layer width of 100 neurons. The network architecture is [100, 100 * 63]. The visualizations show the transformation of data as it propagates through the network layers, highlighting the feature extraction and compression capabilities.

quantify the complexity and structural properties of the network at different layers. Additionally, we analyze various network metrics to understand the behavior and transformations occurring within the network, supporting our observations with experimental results.

4.5.1 Definitions

Definition 4.5.1 (Convex Hull Dimension). *Let $1 \leq k \leq d$ be an integer, where d is the depth of the network.*

1. *Define the kth* convex hull dimension, *denoted* $\mathrm{CHdim}_{(k)}$, *to be the dimension (as a polyhedral set) of* $\mathrm{conv}(\mathrm{Im}_{(k)})$, *the convex hull of* $\mathrm{Im}_{(k)}$, *where* $\mathrm{Im}_{(k)}$ *denotes the image of the network up to layer k.*

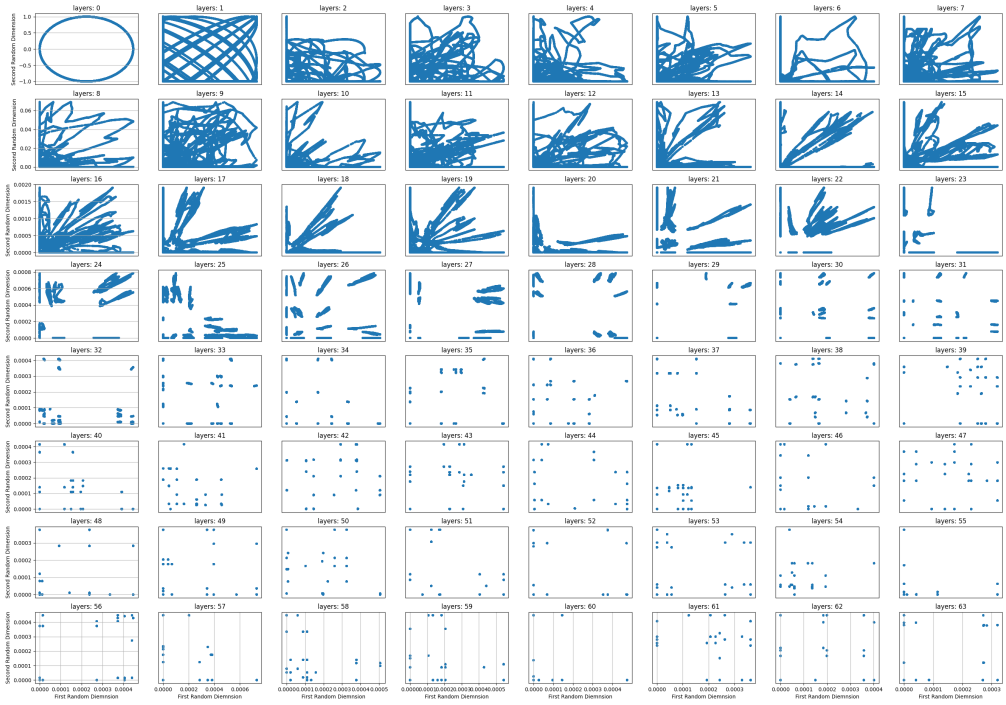

Figure 4.11: Random dimension visualization of the image cloud at each layer for a ReLU network with normal distribution input (mean 0, std 100) and constant layer width of 20 neurons. The network architecture is [20, 20 * 63]. The visualizations show the transformation of data as it propagates through the network layers, highlighting the feature extraction and compression capabilities. The dots at the later layers suggest the one or zero dimension of data cloud.

2. *Define the kth* unbounded convex hull dimension, *denoted* $\mathrm{UCHdim}_{(k)}$, *to be the dimension (as a polyhedral set) of* $\mathrm{conv}(\mathrm{Ray}_{(k)})$, *where* $\mathrm{Ray}_{(k)}$ *denotes the set of rays (directions of unboundedness) at layer* k.

Definition 4.5.2 (Max Cell Dimension). 1. *For any cell $C \in \mathcal{C}(F)$, define its kth cell dimension to be*

$$\mathrm{celldim}_{(k)}(C) = \dim(G_{(k)}(C)),$$

where $G_{(k)}$ is the mapping corresponding to the network up to layer k.

2. *Define the kth* max cell dimension *of F to be*

$$\mathrm{MCdim}_{(k)} \coloneqq \max\{\mathrm{celldim}_{(k)}(C) : C \in \mathcal{C}(F)\} = \max\{\dim(D) : D \in \mathcal{D}_{(k)}\},$$

where $\mathcal{D}_{(k)}$ denotes the set of images of the cells at level k.

Definition 4.5.3 (0/1 Sign Vectors)**.** *For $x \in \mathbb{R}^{n_0}$ and $1 \leq k \leq d$, define $\chi^k = (\chi_1^k, \ldots, \chi_{n_k}^k) \in \{0,1\}^{n_k}$ by*

$$\chi_j^k = \begin{cases} 1 & \textit{if the } j\textit{th coordinate of } A^k \circ G_{k-1}(x) \textit{ is strictly positive}, \\ 0 & \textit{otherwise}, \end{cases}$$

where A^k is the affine transformation at layer k, and G_{k-1} is the mapping up to layer k.

Define $\chi(x) = (\chi^1(x), \ldots, \chi^d(x))$.

It is immediate that χ is constant on each cell of $\mathcal{C}(F)$, since the activation patterns (which ReLUs are active) remain constant within each cell. Thus, given a cell $C \in \mathcal{C}(F)$, we define $\chi(C)$ to be $\chi(x)$ for any point $x \in C$.

Proposition 4.5.4. *Fix a cell $C \in \mathcal{C}(F)$. Then, for any integer $1 \leq k \leq d$,*

$$\mathrm{celldim}_{(k)}(C) \leq \min\left\{ \sum_{i=1}^{n_\ell} \chi_i^\ell(C) \mid 1 \leq \ell \leq k \right\},$$

where n_ℓ is the number of neurons at layer ℓ.

Proof. We proceed by induction on k. For $k = 1$, consider the map $\sigma \circ A^1 : \mathbb{R}^{n_0} \to \mathbb{R}^{n_1}$. The dimension of the image of C under $G_{(1)} = \sigma \circ A^1$ satisfies

$$\dim(G_{(1)}(C)) \leq \min\left\{ \dim(C), \sum_{i=1}^{n_1} \chi_i^1(C) \right\}.$$

This is because the affine map A^1 cannot increase the dimension of C, and the ReLU activation σ further restricts the output to be zero in coordinates where the pre-activation input is non-positive.

Assuming the proposition holds for $k-1$, we consider k. The mapping up to layer k is $G_{(k)} = \sigma \circ A^k \circ G_{(k-1)}$. Then,

$$\dim(G_{(k)}(C)) \leq \min \left\{ \dim(G_{(k-1)}(C)), \sum_{i=1}^{n_k} \chi_i^k(C) \right\}.$$

By the induction hypothesis,

$$\dim(G_{(k-1)}(C)) \leq \min \left\{ \dim(C), \sum_{i=1}^{n_\ell} \chi_i^\ell(C) \mid 1 \leq \ell \leq k-1 \right\}.$$

Therefore,

$$\dim(G_{(k)}(C)) \leq \min \left\{ \dim(C), \sum_{i=1}^{n_\ell} \chi_i^\ell(C) \mid 1 \leq \ell \leq k \right\}.$$

This completes the proof. □

The proposition indicates that the dimension of the cell's image at layer k cannot exceed the minimum number of active neurons up to layer k, nor the original dimension of the cell. This reflects how the ReLU activations can reduce the effective dimension of the data as it propagates through the network.

4.5.2 Analysis of Network Metrics

To better understand the behavior and properties of ReLU networks across layers, we analyze various metrics that provide insights into the transformations occurring within the network. We present the definitions of these metrics, followed by observations from experimental results, and include figures that illustrate these metrics for specific network configurations.

Histogram of Neuron Activation

Definition The histogram displays the distribution of neuron activation percentages for a given dataset across all layers. It shows how many neurons are active (i.e., their ReLU output is non-zero) versus inactive (i.e., their ReLU output is zero). Analyzing this histogram can reveal tendencies toward sparsity or consistent activation in the network.

Activation Patterns The histogram and neuron activation plots reveal how the network's layers handle sparsity and activation distribution. Layers closer to the input might have a more spread-out activation, while deeper layers tend to have more defined patterns.

For the network with a normal distribution and width 100 (Figure 4.13), the activation percentages tend to spread more evenly compared to narrower networks. This suggests that wider networks may maintain a balance in neuron activation across layers. In contrast, the network with width 20 and depth 50 (Figure 4.14) shows significant fluctuations in activation percentages, indicating instability in activation patterns. This suggests that narrower networks may exhibit more pronounced variations in neuron activation, potentially leading to sparsity or inconsistent activation across layers.

Mean Distance from Origin

Definition This metric shows the average Euclidean norm of data points at each layer, providing an understanding of how the data's spread evolves through the layers of the network:

$$\text{Mean Distance} = \frac{1}{N}\sum_{i=1}^{N}\|x_i\|,$$

where N is the number of data points and $\|x_i\|$ is the Euclidean norm of the i-th data point at a given layer.

Data Norm and Spread The mean distance from origin provides insights into how the data's spread changes through the network. Variations in this metric can indicate whether the network is compressing or expanding the data's feature space.

For the normal distribution network with width 100 (Figure 4.13), the mean distance remains relatively stable with a slight increase, suggesting that the data retains its spread through the layers.

In contrast, the network with width 20 and depth 50 (Figure 4.14) shows significant fluctuations in mean distance, indicating more pronounced transformations and possibly instability in the data's spread.

Stable Rank

Definition The stable rank is a measure of the effective rank of a matrix, offering insights into the data's complexity and structure. It is defined as:

$$\text{Stable Rank} = \frac{\|X\|_F^2}{\|X\|_2^2} = \frac{\sum_{i=1}^{r} \sigma_i^2}{\sigma_1^2},$$

where X is the data matrix at a given layer, $\|X\|_F$ is the Frobenius norm, $\|X\|_2$ is the spectral norm (largest singular value), σ_i are the singular values of X, and r is the rank of X. A higher stable rank indicates greater complexity.

Intrinsic Dimensionality The stable rank offers perspectives on the data's intrinsic complexity and how the network layers transform it.

For the normal distribution network with width 100 (Figure 4.13), the stable rank remains relatively high across layers, suggesting that the network maintains a high level of complexity.

In contrast, the network with width 20 and depth 50 (Figure 4.14) shows a rapid decline in stable rank, indicating a reduction in complexity. This suggests that the network is compressing the data into a lower-dimensional space more aggressively.

Spherical Mean Width

Definition This metric shows the average width of data projections along random directions, providing insights into the data's spread in different orientations:

$$\text{Spherical Mean Width} = \mathbb{E}_{\mathbf{d}\sim\mathcal{S}^{n-1}}\left[\max_i \mathbf{d}^\top x_i - \min_i \mathbf{d}^\top x_i\right],$$

where $\mathbf{d}$ is a random unit vector drawn from the unit sphere $\mathcal{S}^{n-1}$.

Data Spread Evolution Changes in spherical mean width indicate how the data's shape evolves through the network layers.

In the normal distribution network with width 100 (Figure 4.13), the spherical mean width remains relatively high, indicating a wide spread of data.

For the network with width 20 and depth 50 (Figure 4.14), there is a more pronounced decrease in spherical mean width, indicating tighter data clustering and potential loss of variance.

Number of Non-Zero Eigenvalues of the Covariance Matrix

Definition This metric indicates the intrinsic dimensionality of the data at each layer by counting the number of non-zero eigenvalues (singular values) of the data's covariance matrix. A decrease in this number suggests dimensionality reduction.

Dimensionality Reduction The number of non-zero eigenvalues provides insights into how the network compresses or expands the data's dimensionality.

In the normal distribution network with width 100 (Figure 4.13), the number of non-zero eigenvalues decreases steadily, showing gradual dimensionality reduction.

In contrast, the network with width 20 and depth 50 (Figure 4.14) shows a rapid decrease in the number of non-zero eigenvalues, indicating a more aggressive compression and possible loss of information.

Activation of Neurons Across Layers

Definition This visualization shows the activation status of neurons across layers, often represented as a heatmap. Darker colors indicate inactive neurons, while lighter colors indicate active neurons.

Feature Representation The activation patterns help understand how the network's layers handle sparsity and activation distribution.

In the wider network (Figure 4.13), the activation patterns are more balanced, with a moderate number of neurons active across layers.

In the narrower network (Figure 4.14), there is significant sparsity, with many neurons inactive in deeper layers.

Cell Dimensions

Definition This metric shows the number of active neurons (dimensions) at each layer, indicating how many neurons are contributing to the representation:

$$\text{Cell Dimensions} = \sum_{i=1}^{n} \mathbb{I}[a_i > 0],$$

where a_i are the activations of neurons at a given layer, and $\mathbb{I}[\cdot]$ is the indicator function.

Dynamic Sparsity Changes in cell dimensions across layers help understand the network's dynamic sparsity and its impact on feature representation.

In the normal distribution network with width 100 (Figure 4.13), the number of active dimensions decreases moderately, balancing sparsity and dimensionality.

In the network with width 20 and depth 50 (Figure 4.14), there is a rapid decrease in cell dimensions, indicating significant sparsity and reduced feature representation.

Combined Observations

- The network with a normal distribution and wider layers (Figure 4.13) tends to have a more balanced activation pattern, whereas the narrower network (Figure 4.14) shows more significant fluctuations, indicating instability in activation patterns.
- The wider network maintains a broader data spread, while the narrower network shows tighter clustering and more aggressive compression.
- The reduction in the number of non-zero eigenvalues and the stable rank indicates how the networks compress the data's intrinsic dimensionality, with the narrower network showing the most significant reduction.
- Overall, wider networks with moderate depth may preserve essential features while reducing dimensionality, whereas narrower or deeper networks may exhibit increased sparsity and aggressive dimensionality reduction, potentially leading to loss of information or instability in activation patterns.

4.6 Discussion

In this work, we have explored the transformations that occur within deep ReLU networks from both theoretical and empirical perspectives. One of the key observations is the manner in which input data undergoes complex manipulations as it propagates through the layers of the network. This process, sometimes referred to metaphorically as "spaghettification," involves the stretching and folding of data manifolds, which is crucial for the network's ability to capture intricate patterns and make accurate predictions.

The formalization of feedforward ReLU neural networks as sequences of affine transformations followed by ReLU activations has provided a rigorous framework for analyzing neuron activations. By using properties of canonical polyhedral complexes and notions of dimension, we have attempted to capture the geometric and topological properties of data transformations within the network. These theoretical constructs aim to

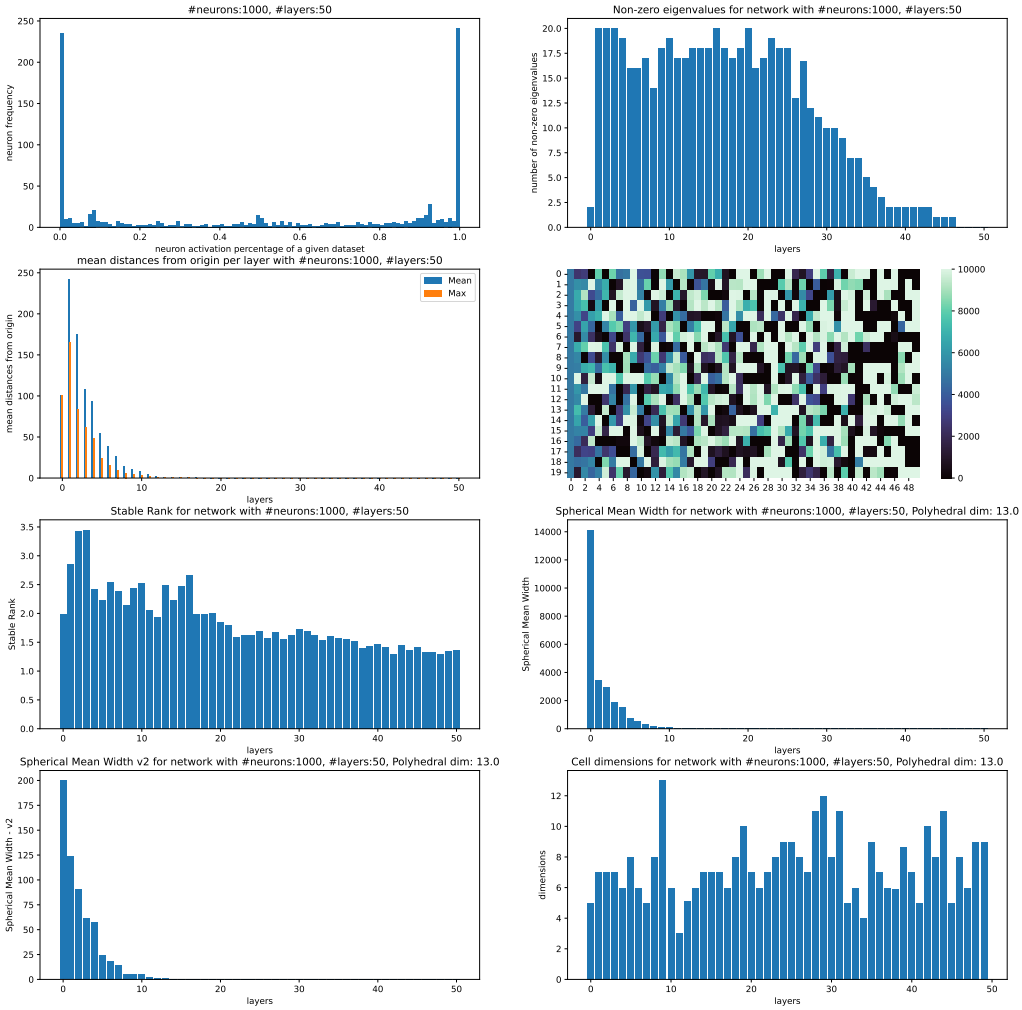

Figure 4.12: Analysis of a ReLU network with input dimension 2, hidden layer size 20, and 50 hidden layers. The plots include (first column) histogram of neuron activations, mean distance from origin, stable rank, spherical mean width v2, and (second column) number of non-zero eigenvalues, neuron activation patterns, spherical mean width v1, and cell dimensions. These metrics provide comprehensive insights into the network's behavior and data transformations across layers.

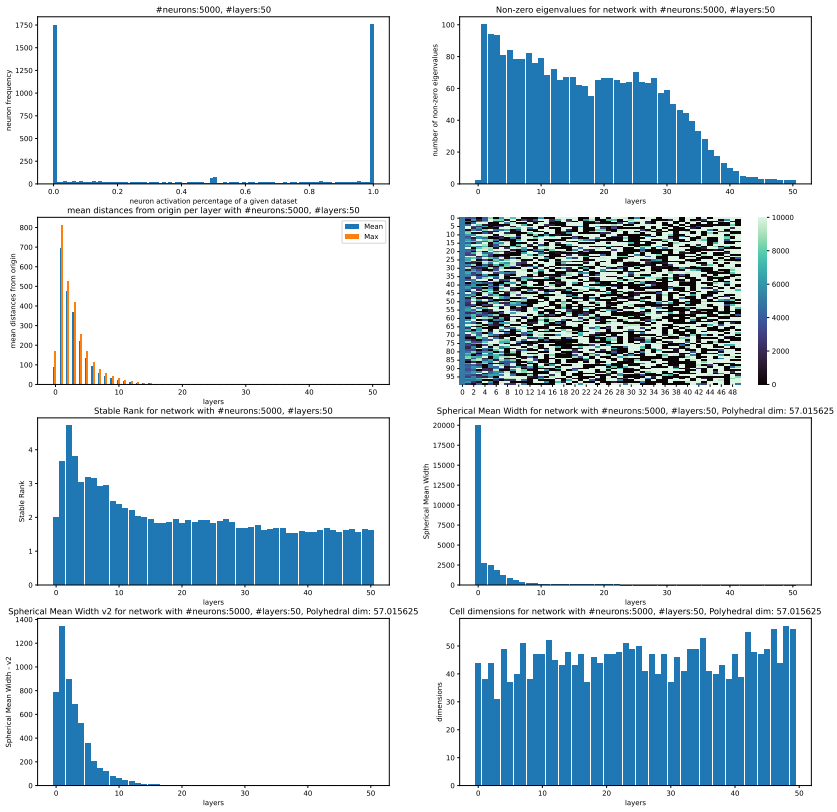

Figure 4.13: Normal Distribution with Mean 0 and Std 100, Width 100, Depth 50

shed light on the non-linear dynamics of data flow and underscore the influence of network architecture on these dynamics.

Empirically, we conducted experiments to investigate how various network configurations, such as depth and width, affect neuron activation patterns and data transformations. The analysis of metrics like neuron activation histograms, mean distance from the origin, stable rank, and spherical mean width revealed trends that align with our theoretical considerations. For instance, deeper networks tend to exhibit increased polarization in neuron activations, leading to more pronounced sparsity or stability in activation patterns. Similarly, variations in network width influence the balance between data compression and the preservation of feature diversity.

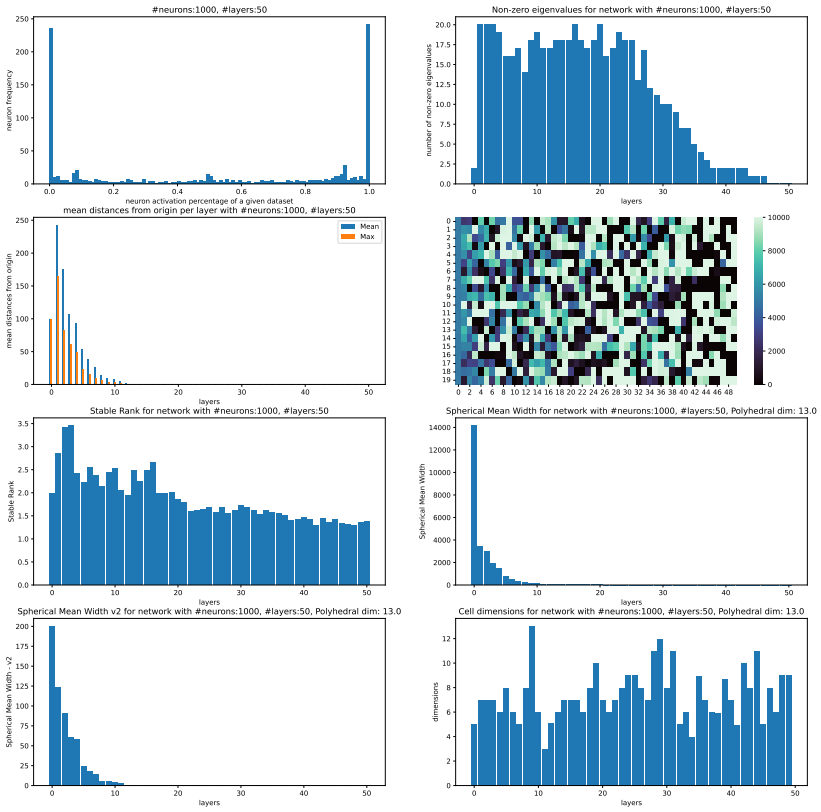

Figure 4.14: Points on Sphere, Width 20, Depth 50

However, it is important to approach these findings with a critical perspective. The theoretical constructs we introduced, while mathematically grounded, may not fully capture the complexities of real-world neural networks, especially when trained on diverse datasets or employing different activation functions. The empirical results, though indicative of certain trends, are based on specific network architectures and input distributions. Generalizing these observations requires further investigation and validation across a broader range of settings.

Moreover, the assumptions made in our theoretical analysis, such as the reliance on ReLU activations and specific sampling strategies, might limit the applicability of our conclusions. Neural networks in practice often incorporate various types of activations,

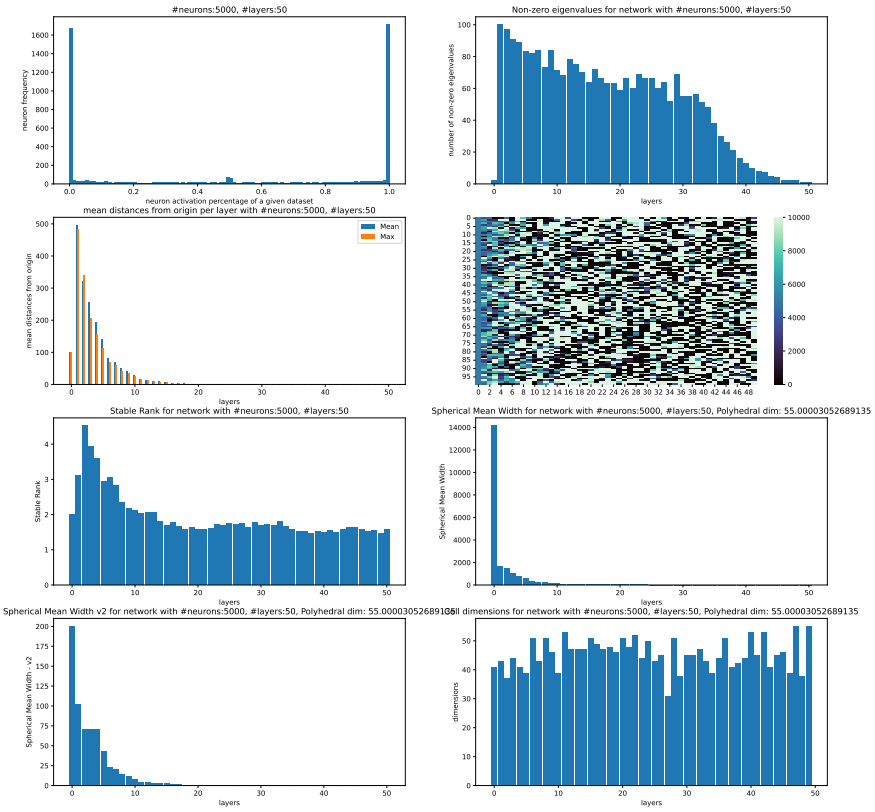

Figure 4.15: Points on Sphere, Width 100, Depth 20

regularization techniques, and architectural innovations that could significantly alter neuron activation behavior and data transformations.

From a practical standpoint, understanding the geometric properties of ReLU activations could inform the design of more effective network architectures and training strategies. Insights into how network depth and width affect activation patterns and data dimensionality might guide the selection of hyperparameters to achieve a desired balance between model capacity and computational efficiency. However, caution must be exercised in translating these theoretical insights into practice without thorough empirical validation.

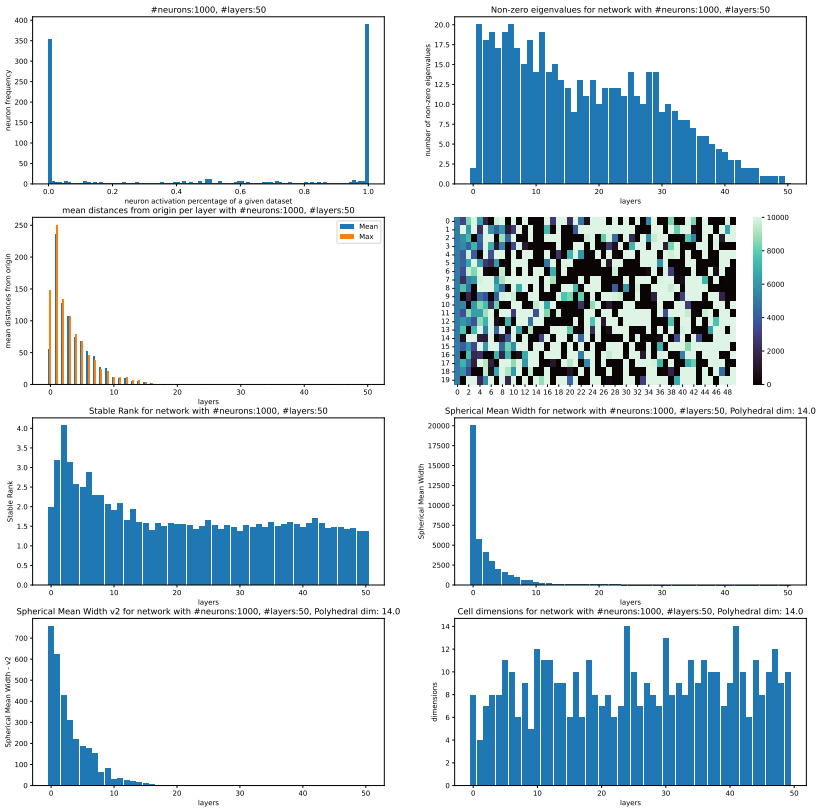

Figure 4.16: Points on Sphere, Width 100, Depth 20

Future work could focus on extending the theoretical framework to encompass other types of activation functions and network architectures. Investigating how elements such as batch normalization, dropout, or residual connections affect the geometric and topological properties of networks may provide a more comprehensive understanding of neural network behavior. Additionally, integrating geometric insights with other analytical tools, such as information theory or optimization theory, could yield deeper explanations for phenomena observed in deep learning, such as generalization capabilities and optimization landscapes.

In conclusion, while our study offers initial insights into the transformations occurring within deep ReLU networks, there remains much to explore. The interplay between

network architecture, activation patterns, and data transformations is complex and multi-faceted. A deeper theoretical and empirical investigation is necessary to fully understand these relationships and to harness this understanding for practical advancements in neural network design and training.

Chapter 5

Interpretability in Action, A Practical Approach

5.1 introduction

Large transformer-based models have achieved significant success in autoregressively predicting the next token across various modalities, including language, images, and audio. Recently, these models have started to be used as agents in online settings. Given the advancements in large-scale AI-based agents interacting with real and simulated worlds [1, 17], it is crucial to understand their decision-making processes both in and out of distribution. VPT [4] is one of the first large-scale transformer-based agents (250 million parameters). The techniques that have been developed in the field of mechanistic interpretability have the potential to be useful for better understanding VPT and other agents like it. We chose VPT as the model organism for our studies because it is one of the largest and most capable open-source embodied agentic models that act in a 3D environment. It has also been used as a backbone for developing other agents [41, 45]. There are more advanced agents, such as SIMA [1]; however, they are not openly available for study.

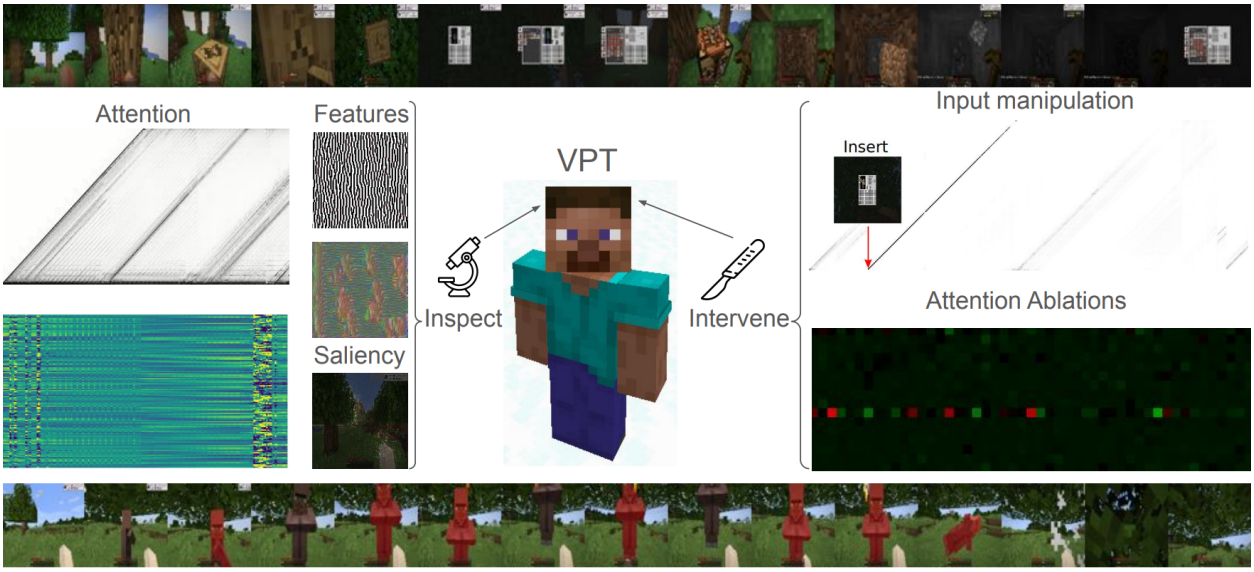

Figure 5.1: We use various interpretability techniques on the Minecraft playing agent VPT to better understand how it makes decisions. These include visualizing attention head weights and outputs, feature visualization, saliency maps, ablating attention head outputs, manipulating the input stream, and others. (Top) a part of a regular episode. (Bottom) an episode with a "villager-tree". See Video01.

Mechanistic interpretability. Mechanistic interpretability seeks to reverse engineer neural networks into the algorithms that the network weights have implemented throughout training. The field has focused on circuit-based methods [14, 60] and understanding attention heads [44, 51]. However, most of this work has been on large language models (LLMs), with limited investigation into other modalities. We propose that applying mechanistic interpretability techniques that have been designed and tested on language to other modalities is essential, not only to validate the method but also to establish a foundation for the nuances of the various new modalities. In addition, VPT has been fine-tuned with reinforcement learning (RL) and vision interpretability techniques often do not work for vision-based RL agents [36].

Mechanistic interpretability on agents. Understanding the behavior of agentic models mechanistically poses more challenges than for traditional LLMs for several reasons. First, isolating an agent's behavior requires replicating the environment and the actions

leading to that behavior, which is difficult for agentic models. For instance, [35] identified a circuit in GPT-2 that computes greater than by prompting the model with sentences like, "The war lasted from the year 1732 to the year 17" and analyzing its activations. In contrast, for stochastic agentic models in dynamic environments, it is challenging to recreate slight variations of similar situations to assess if the model has generalized knowledge about a scenario. While LLMs and their language outputs are stochastic, LLM interpretability researchers benefit from being able to carefully craft their prompts to elicit certain behaviors from the model.

Additionally, computing rollouts for large-scale agentic models is expensive, and comparing actions across different rollouts is complex. Comparing action probabilities and analyzing rollouts proved most useful in assessing the effect of ablations on the overall behavior of VPT. Rewards can sometimes provide a metric to compare the success of rollouts, but it does not capture the detail needed to understand if an agent's behavior has been altered and often can only show if the agent remains functional. Finally, agentic models are trained to maximize long-term rewards, adding another layer of complexity to their decision-making processes compared to LLMs optimized for next-token prediction.

Environment—MineRL. VPT was trained in MineRL [29], which turns Minecraft into an environment. Minecraft [49] is a 3D embodied sandbox video game known for its blocky graphics and open-world gameplay. In survival mode, the player's goal is to gather resources, craft tools, and build shelters to survive against monsters. In MineRL, each rollout starts in a new procedurally generated world, with the agent acting in it from a first-person perspective. The observations are the images of the game view. The action space is the same as what a human would use—keyboard buttons and mouse. One second of in-game time is equal to 20 time steps in the environment. The tasks and rewards can be specified manually. The most popular task has been obtaining a diamond.

Agent—VPT. VPT is a 250 million parameter model that was pre-trained using 70,000 hours of pseudo-labeled human Minecraft play videos and fine-tuned using reinforce-

ment learning for over 230,000 in-game hours [4]. The architecture of the network is a combination of a residual convolutional neural network and a transformer. The task the agent was fine-tuned on was obtaining a diamond pickaxe—one extra step beyond obtaining diamonds.

VPT's behavior is very consistent. At the start of an episode, it goes towards the nearest tree, chops four logs, and proceeds to craft and mine its way through the subtasks. In about 80% of episodes, it gets an iron pickaxe, the item required to mine diamonds. It gets diamonds 20% of the time and the diamond pickaxe 2% of the time.

Agent—Steve-1. Steve-1 is a Minecraft agent that extends the capabilities of VPT by introducing instruction tuning, enabling it to follow natural language commands [41]. It uses a richer dataset of human gameplay videos with text annotations, improving its understanding of complex behaviors. This integration of visual and textual data during training gives it the ability to understand and act upon natural language instructions. However, it can perform only short tasks that take several seconds.

Attention mechanism. In transformer models, *attention weights* are computed by taking the softmax of the dot product of queries (Q) and keys (K), indicating how much focus each part of the input should receive. *Attention outputs* are then obtained by multiplying these weights by the values (V). This allows the model to weigh the input elements appropriately when generating representations or predictions. VPT has 4 attention layers with 16 heads in each. We will use "head 2.3" to refer to layer 2 attention head 3.

5.2 Related work

Mechanistic interpretability. Most mechanistic interpretability work has focused on reverse engineering circuits in LLMs [11, 40, 60]. There has been some limited work on understanding transformer-based vision models. For example, [18] found that attention heads can have specialized roles, such as focusing on shapes, colors, and object counts.

Figure 5.2: (Middle) Visualization of a trajectory up to crafting a stone pickaxe. The leftmost pixel of each frame corresponds to the time step in the attention plots. (Top) Attention weights of attention head 2.2—note the different pattern above the 3rd frame. This coincides with the camera moving up. The vertical axis is the 128 attention weights, the horizontal axis is time. (Bottom) Max attention weights over all attention heads. This shows that most attention is paid to 3–4 past frames and some key-frames. See Video02 and Video03.

Interpretable reinforcement learning. Reinforcement learning techniques can sometimes be substantially easier to interpret depending on what kind of model is used for the policy and value function. Techniques such as linear function approximation, decision trees, and tabular methods often provide clear and understandable representations of the learned policies and value functions; however, almost all state-of-the-art RL uses deep RL [12, 21, 39, 46].

Recent advancements in interpretable RL include the development of frameworks and methodologies that aim to balance the performance and transparency of RL models. For example, the SINDy-RL framework combines sparse dictionary learning with deep reinforcement learning to create interpretable and capable agents. This approach leverages the Sparse Identification of Nonlinear Dynamics (SINDy) method to construct data-driven models that are not only effective but also offer insight into the underlying decision-making processes [66].

Hierarchical reinforcement learning is another avenue that enhances interpretability. This method splits tasks, where a higher-level controller decides on a sequence of subtasks while a lower-level controller handles specific actions within each subtask. This hierarchical structure allows for a more intuitive understanding of the policy's behavior by breaking down complex decisions into simpler, more manageable parts [5, 62].

Additionally, explainable reinforcement learning techniques focus on post-hoc methods to explain the decisions of black-box models. These include generating visualizations, using model-agnostic techniques like LIME [55], Shapley values [6], and developing algorithms that provide explanations for specific actions taken by the RL agent [53].

5.3 Attention visualization

We start our analysis by visualizing parts of the transformer attention mechanism. VPT uses a transformer architecture with a context length of 128 frames (6.4 in-game seconds). The agent takes between 3 to 10 minutes to make a diamond pickaxe from scratch when it succeeds at the task. This involves a long sequence of subtasks, such as finding and chopping a tree, crafting different pickaxes, mining stone, finding and smelting iron, and others. As a comparison, a proficient human takes on average 20 minutes to solve this task [4]. How does VPT "keep the thread" for so long? That is, how does it know which part of the long sequential task it is currently solving? We cannot help but notice the similarity to the famous musician Clive Wearing, who was struck by an extremely strong form of amnesia, which left him with about 7 seconds of memory. Yet he could play long piano pieces [58].

It could be that a single frame is enough to recognize what the agent has to do next. For example, if it sees a stone pickaxe in the hotbar, but no iron pickaxe, it might be looking for iron ore. However, the agent is quite robust to the items being placed in different slots of the hotbar, or even in the parts of inventory that are visible only when the inventory menu is open. This would imply a more sophisticated mechanism, likely using its memory of

past frames. Given these constraints, it is likely that VPT has learned to play Minecraft in a different way than humans; something that we expect to be common among many large-scale AI agents.

5.3.1 Attention weights

Visualizing attention weights between tokens in LLMs has enabled the discovery of many interesting circuits, such as induction heads [50]. We use a very similar technique in the sequential decision making domain in our work. The main difference comes from the short context window of VPT—it does not have access to frames beyond the previous 128, while its task takes many thousands.

The top plot in Figure 5.2 shows how much attention is being paid to each of the past 128 frames (vertical axis) over the first 1,000 frames of an episode (horizontal axis) by the attention head 2 on layer 2 (4 layers, 16 heads each). The darker the color, the higher the attention weight, with 0% represented by white and 100% by black. 1,000 frames is 10–20% of an average episode in which the agent obtains a diamond pickaxe. The frames in the middle show the progression over that time—from chopping a tree to crafting a stone pickaxe. The leftmost pixel of each frame corresponds to that exact moment in time on the above plot. A diagonal line indicates that a specific frame is being attended to as the agent continues to interact with the environment.

The bottom part of the top plot shows a short horizontal line (above the third visualized frame). This indicates that the attention head is paying attention to the last frame it saw during the rollout. This coincides perfectly with the agent looking up to chop the logs above it. This does not happen when the agent looks down shortly after. We can also observe how attention head 2.2 operates in two modes: one of paying attention to some specific frames in the past and the second of mainly looking at the last frame. Many other attention heads show the same two-mode pattern (see Figure 5.6–Figure 5.9 in the appendix). The patterns are consistent between episodes. Also, see Figure 5.3 for a visualization of attention weights at a single point in time.

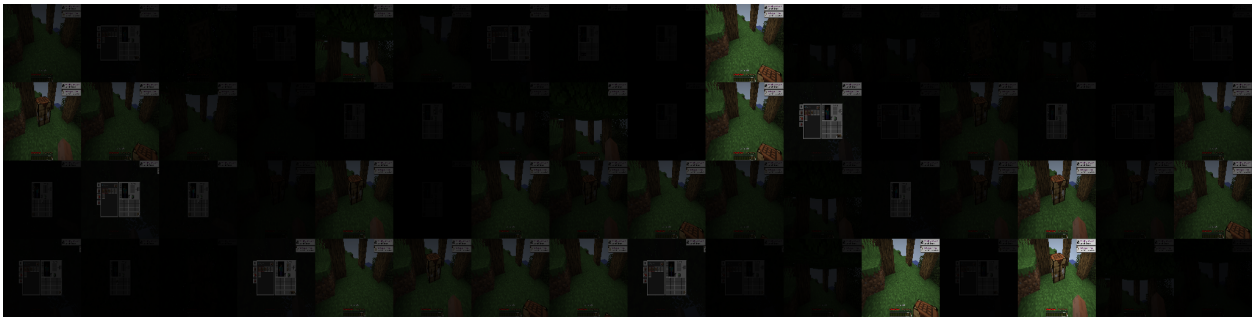

Figure 5.3: The frame that each attention head is paying the most attention to at a single point in time, right after placing a crafting table. Brightness indicates the magnitude of the attention. For example, heads 0.9 (1st row, 10th frame) and 1.9 are looking at the previous frame; heads 2.13, and 3.13 are looking at the current frame (crafting table placed). Other heads are looking at the inventory menu at different earlier times, some with the recipe book open, and some closed. See Video03.

To visualize what VPT pays the most attention to more clearly, we overlaid the attention weight patterns for all 64 attention heads by taking a maximum for each frame and each memory position (see the bottom plot of Figure 5.2). We can see that the past 3–4 frames are always attended to, indicated by the thick dark line at the bottom; as well as some key-frames, seen as diagonal lines. We surmise that the recent frames help describe its immediate past, while the key-frames allow it to recognize which part of the full task it is currently doing. Some example key-frames include: a dirt block that was just destroyed (starting to dig down?), a cobblestone block that was just destroyed (dug further down?), and an achievement popup after making a stone pickaxe (can now mine iron?). These are of course speculative and require more evidence.

5.3.2 Attention outputs

In addition to visualizing attention weights (Figure 5.2), we can also gain insights by visualizing attention outputs. For instance, attention head 2.2 again shows a distinct pattern when the camera is moved up (see Figure 5.4). We discovered some of the patterns in the attention weight plots after initially noticing them in the attention output visualizations.

Below we list some of the patterns we found in both the attention weight and the attention output visualizations:

- Attention head 0.8—looks at the previous frame every time the inventory menu is open (Figure 5.6, Figure 5.10).
- Attention head 0.9—looks at the previous frame all the time, except for each time when the inventory menu state is changing, when it looks at the current frame (Figure 5.6).
- Attention head 1.2—mostly looks at every 4th frame (Figure 5.7, Figure 5.10).
- Attention head 2.2—looks at the current frame when the agent is looking up (Figure 5.6).
- Attention head 3.3—shows a distinct pattern when the agent decides that it needs to open the crafting table menu (Figure 5.11).

While visualizations can suggest what certain attention heads are doing, one of the main goals is to be able to control the agent's behavior in a predictable manner, which we explore in the next section.

5.4 CNN Feature visualization

5.4.1 Filter Visualization by Optimization

Our study leverages filter visualization by optimization to elucidate the specific patterns and features that various layers of the VPT model detect. The goal of this optimization is to generate an image that maximizes the activation of a specific kernel map. The loss function is the output of that kernel map. In Figure 5.12, we show the visualizations of the fifth CNN layer's filters. These visualizations reveal that filters at this depth in

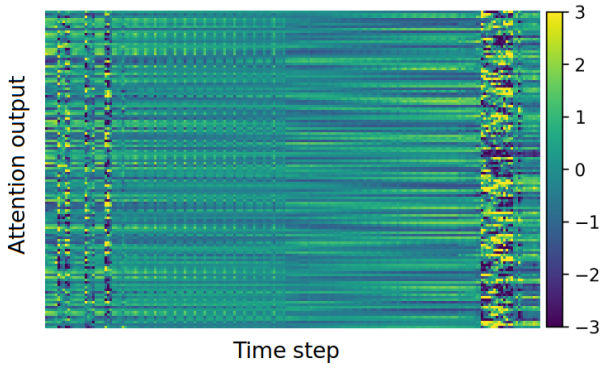

Figure 5.4: All 128 output z-scores of attention head 2.2 over the first 200 frames of a regular episode. The different pattern on the right coincides with the agent looking up. Regular vertical lines in the first half match the attacking arm looping every 4 frames. These disappear in the second half. The agent is still attacking, but the arm is replaced by a smaller object—an oak log it just chopped. See Video04.

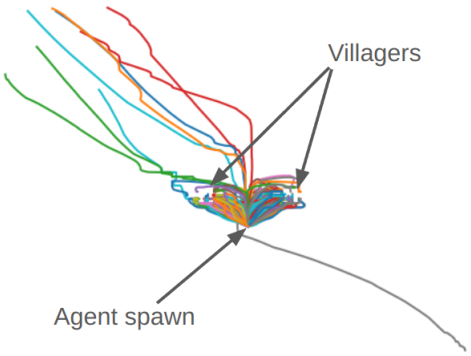

Figure 5.5: Top-down view of trajectories when VPT is presented with a choice between two villagers standing under tree leaves, one on the left, one on the right. The identical scenarios produce different trajectories due to stochastic actions, including one trajectory where the agent turns around and goes in the other direction.

the network capture more complex textures and patterns compared to initial layers (see Figure 5.13).

Figure 5.13 provides filter visualizations for the first CNN layer. These filters detect basic features such as edges, simple textures, and colors. The simplicity of these features aligns with the layer's role in capturing foundational visual elements, which are progressively combined and abstracted in the deeper layers to form more complex representations.

5.4.2 Receptive Field Attention

Figure 5.14 shows the receptive field attention of layer 6 mapped onto an input image. This mapping shows how the activations from this layer correspond to specific regions in the input image, highlighting areas of interest that the model focuses on during gameplay.

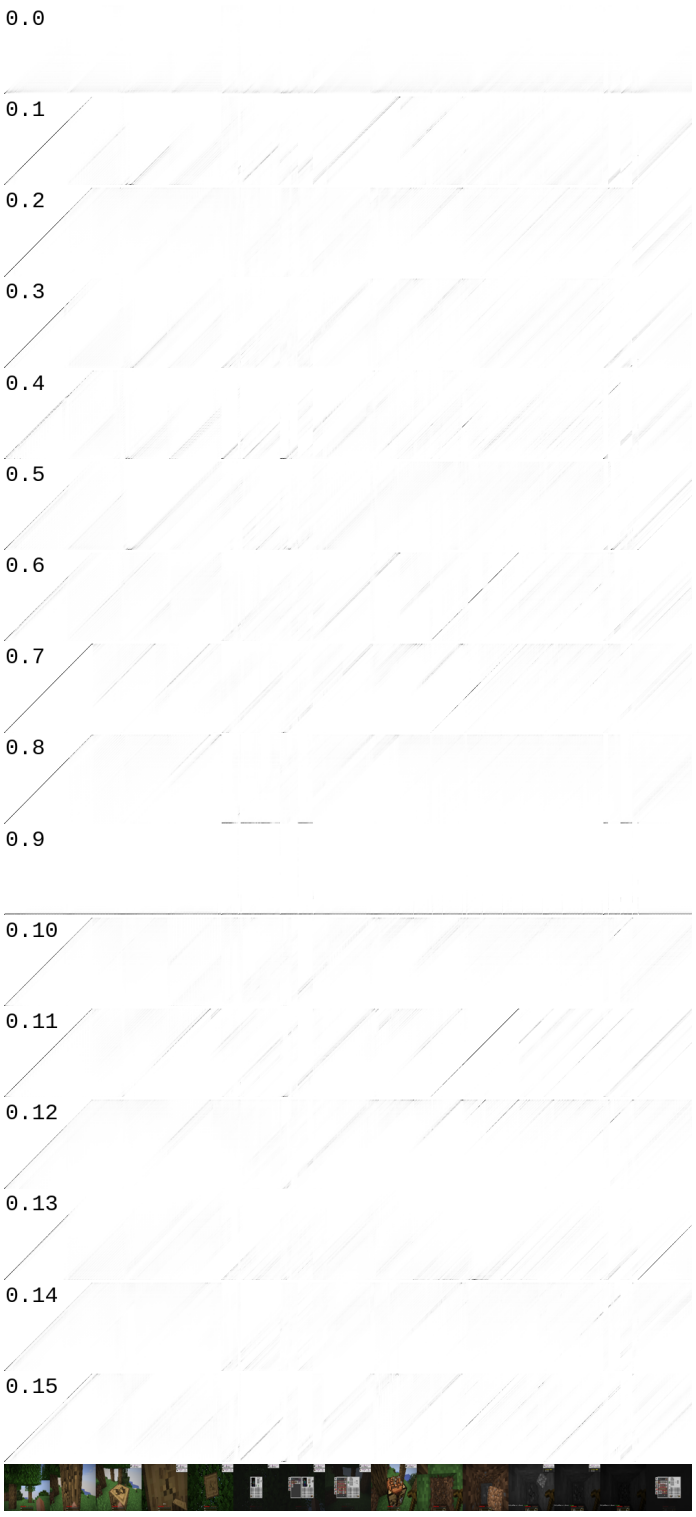

Figure 5.6: Attention weights of attention heads in layer 0.

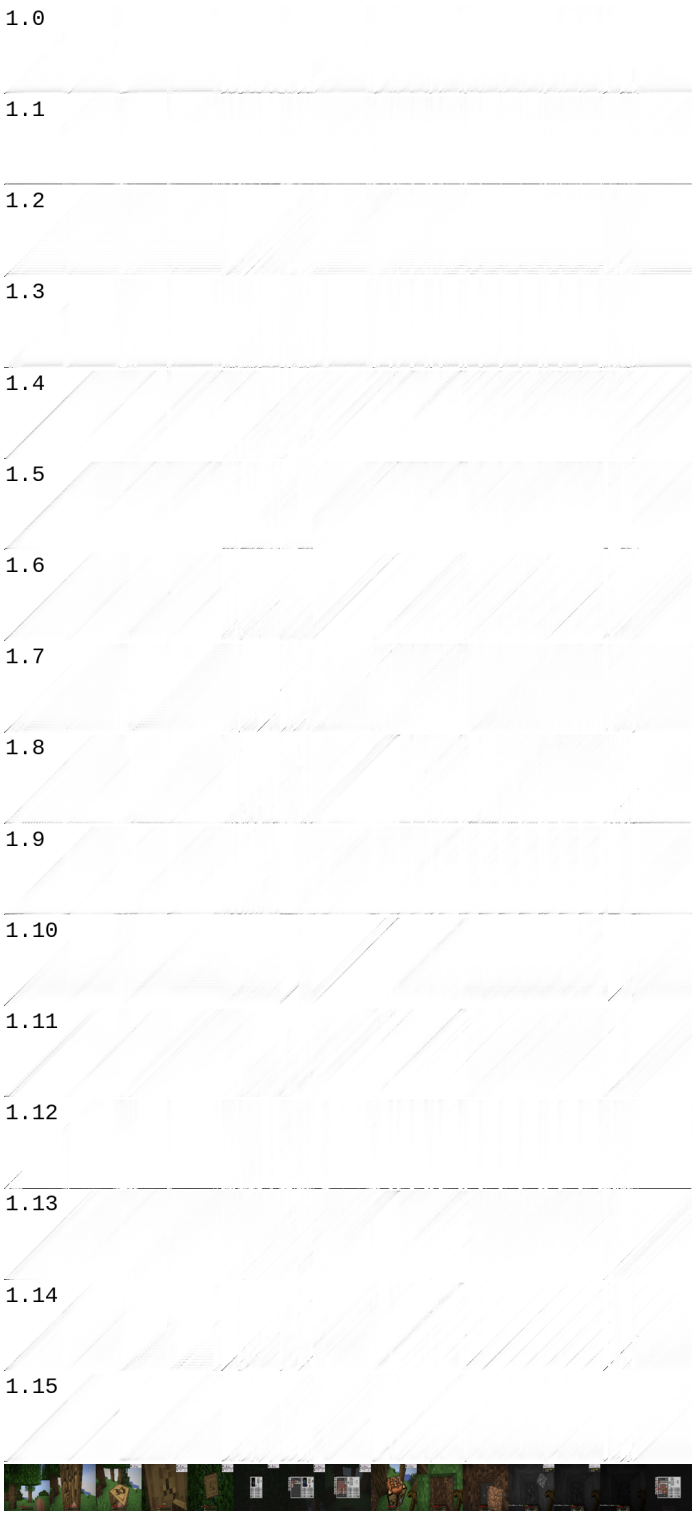

Figure 5.7: Attention weights of attention heads in layer 1.

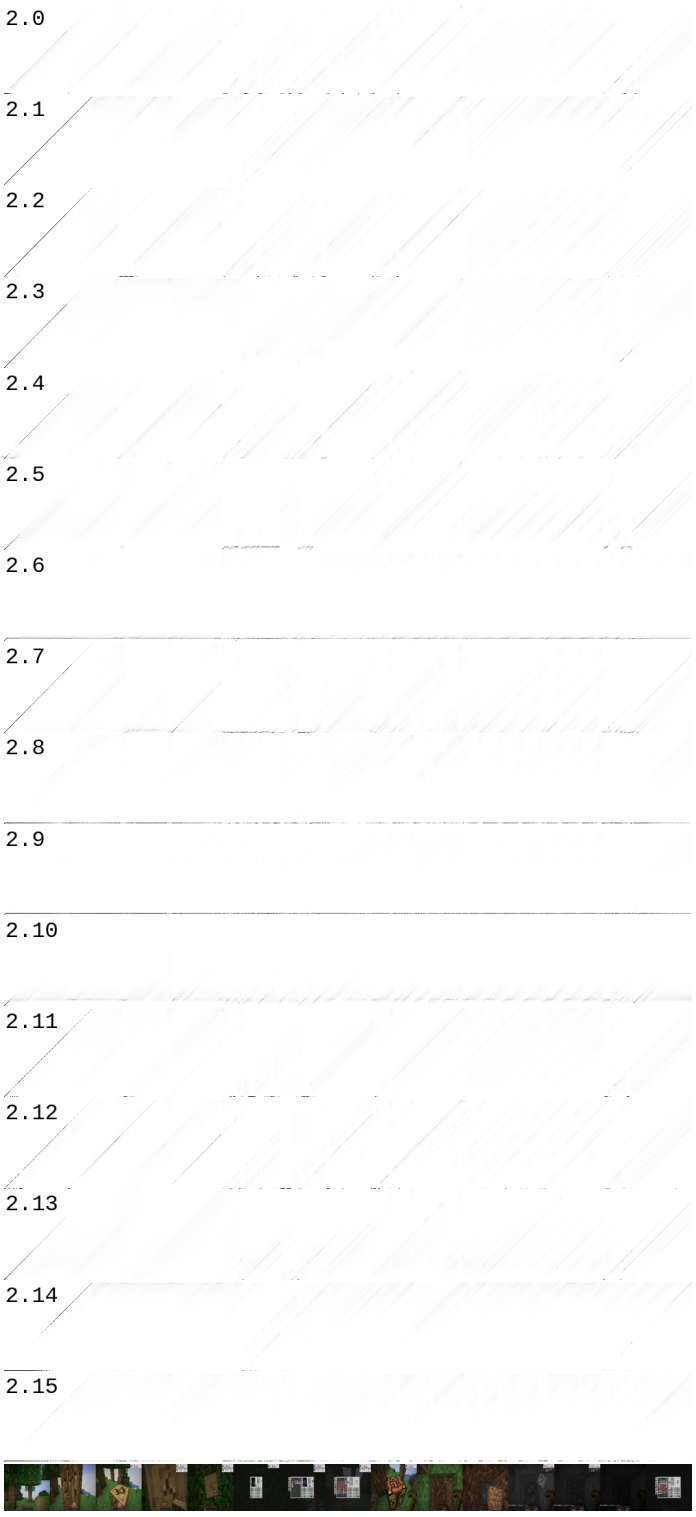

Figure 5.8: Attention weights of attention heads in layer 2.

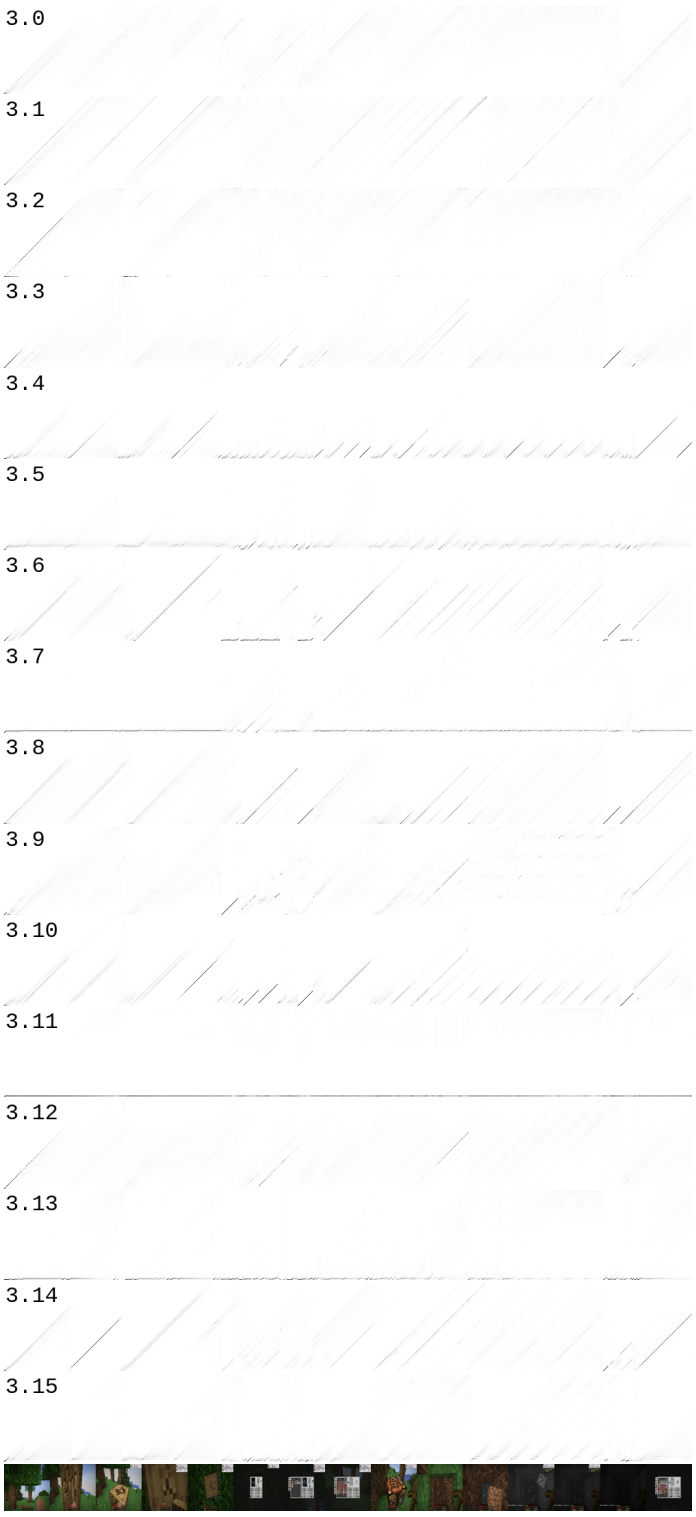

Figure 5.9: Attention weights of attention heads in layer 3.

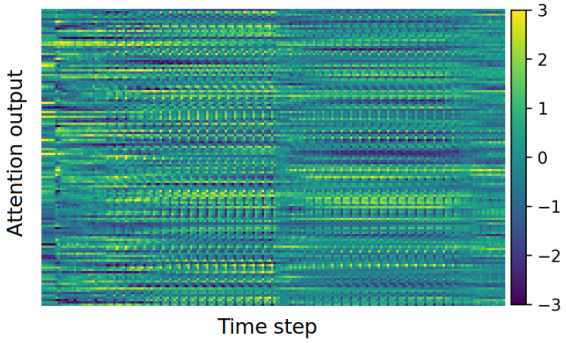

Figure 5.10: Attention head 1.2 shows this unique pattern likely due to paying attention to every 4th frame, which can be seen in the attention weight plots (Figure 5.7).

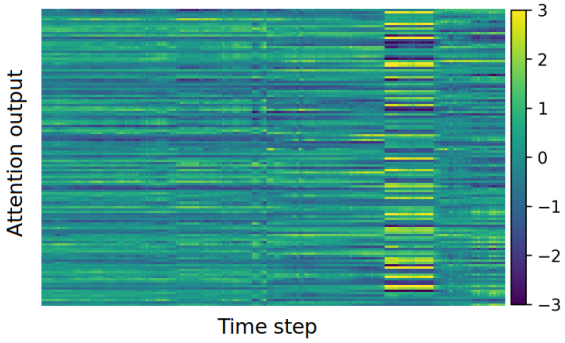

Figure 5.11: Attention head 3.3 shows a distinct pattern (the one on the right) while the agent tries to open the crafting table menu. It starts right after the previous subtask is finished. The agent switches to the crafting table on the hotbar, jumps up, places it under itself, and opens it, after which the pattern stops. This pattern happens in multiple parts of the episode, whenever the crafting table menu is needed. See Video13.

The receptive field R of a convolutional layer refers to the region in the input image that affects a particular feature in the output. It can be calculated using the kernel size K, the stride S, and the padding P. The basic formula for a single layer is straightforward:

$$R = K$$

For deeper layers, the calculation becomes recursive. Let's denote:

- R_L as the receptive field of layer L
- K_L as the kernel size of layer L
- S_L as the stride of layer L
- P_L as the padding of layer L

The receptive field of a layer L depends on the receptive field of the previous layer R_{L-1} and can be calculated as:

$$R_L = R_{L-1} + (K_L - 1) \times \prod_{i=1}^{L} S_i$$

Where $\prod_{i=1}^{L} S_i$ is the product of all the strides from layer 1 to layer L.

This recursive method allows us to determine the receptive field of any layer in a deep convolutional network by considering the kernel size, stride, and padding of all preceding layers. The mapping of activations to specific regions in the input image (like gameplay frames) helps in understanding which parts of the input the model focuses on. For instance, the activations in a higher layer of the network will correspond to larger regions of the input image, indicating broader, more abstract features such as objects or actions within the game. In Figure 5.15, we present heatmaps of the receptive fields associated with the top 500 activations of the sixth layer.

5.4.3 Kernel Visualization

The kernel visualization for the fifth layer, shown in Figure 5.16, demonstrates how the model interprets an image containing a villager and a tree. For this figure, we combined the outputs of multiple kernels and applied PCA to reduce the dimensionality to three principal components. This allowed us to visualize complex, high-dimensional features in an RGB format, highlighting how different components contribute to the learned representations. This technique revealed the intricate patterns learned by the network and how different kernels collaborate to encode features. Interestingly, the visualization indicates that the model does not distinctly separate the villager from the tree, suggesting a limitation in the model's ability to discern boundaries between overlapping objects.

5.4.4 Filters Overlay

Figure 5.17 showcases the overlay of filter activations, providing a comprehensive view of how different filters respond to the same input image. Here's how we generate these filter overlays

- Pass an image through the network: the input image is fed through the CNN, and the activations of a specific layer are recorded.
- Extract filter activations: For the chosen layer, the activations of each filter are extracted. These activations are essentially the feature maps generated by each filter when applied to the input image.
- Overlay the activations on the input image: Each filter's activation map is overlaid on top of the input image. This can be done by upscaling the activation maps to the original image size and blending them with the image using different visualization techniques like heatmaps or transparency overlays (we used transparency overlays).

This overlay helps understand the cumulative effect of multiple filters working together, contributing to the overall feature extraction and decision-making process in the VPT model.

5.4.5 Layer-Specific Activations

The kernel visualization of all 256 kernels for an inventory image in the sixth layer, depicted in Figure 5.18, emphasizes how the model processes and prioritizes different parts of the inventory interface. The activations suggest that the model pays particular attention to areas with high informational content, such as item slots and inventory shapes (filters 15, 35, 39, 142, 143).

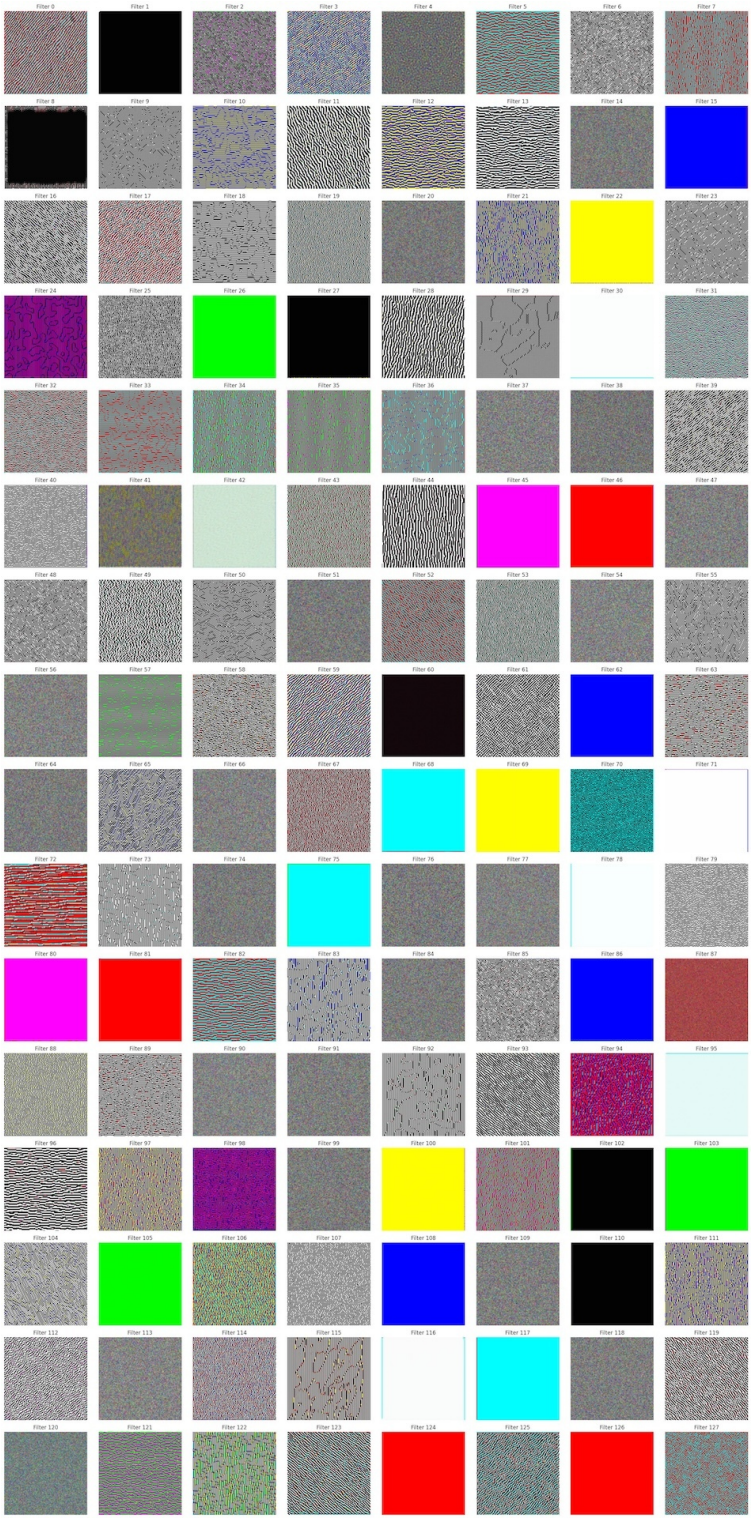

Figure 5.12: Filter visualization by optimization for the 5th CNN layer of VPT.

Figure 5.13: Filter visualization by optimization for the 1st CNN layer of VPT.

Figure 5.14: Receptive field attention of layer 6 mapped into an input image.

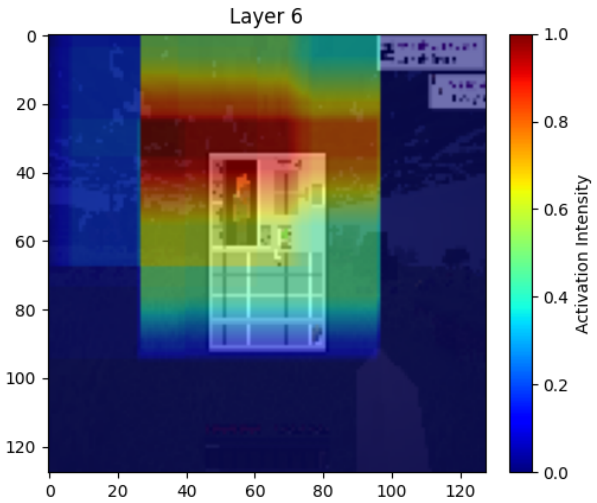

Figure 5.15: Receptive fields associated to top 500 activations of 6th layer. In other words, the receptive field attention of layer 6 mapped into an input image.

Figure 5.16: How kernels visualize the image for VPT in the 5th layer, this is the villager-tree input image and lack of boundary in the image suggests that VPT didn't highlight the difference between the villager and tree.

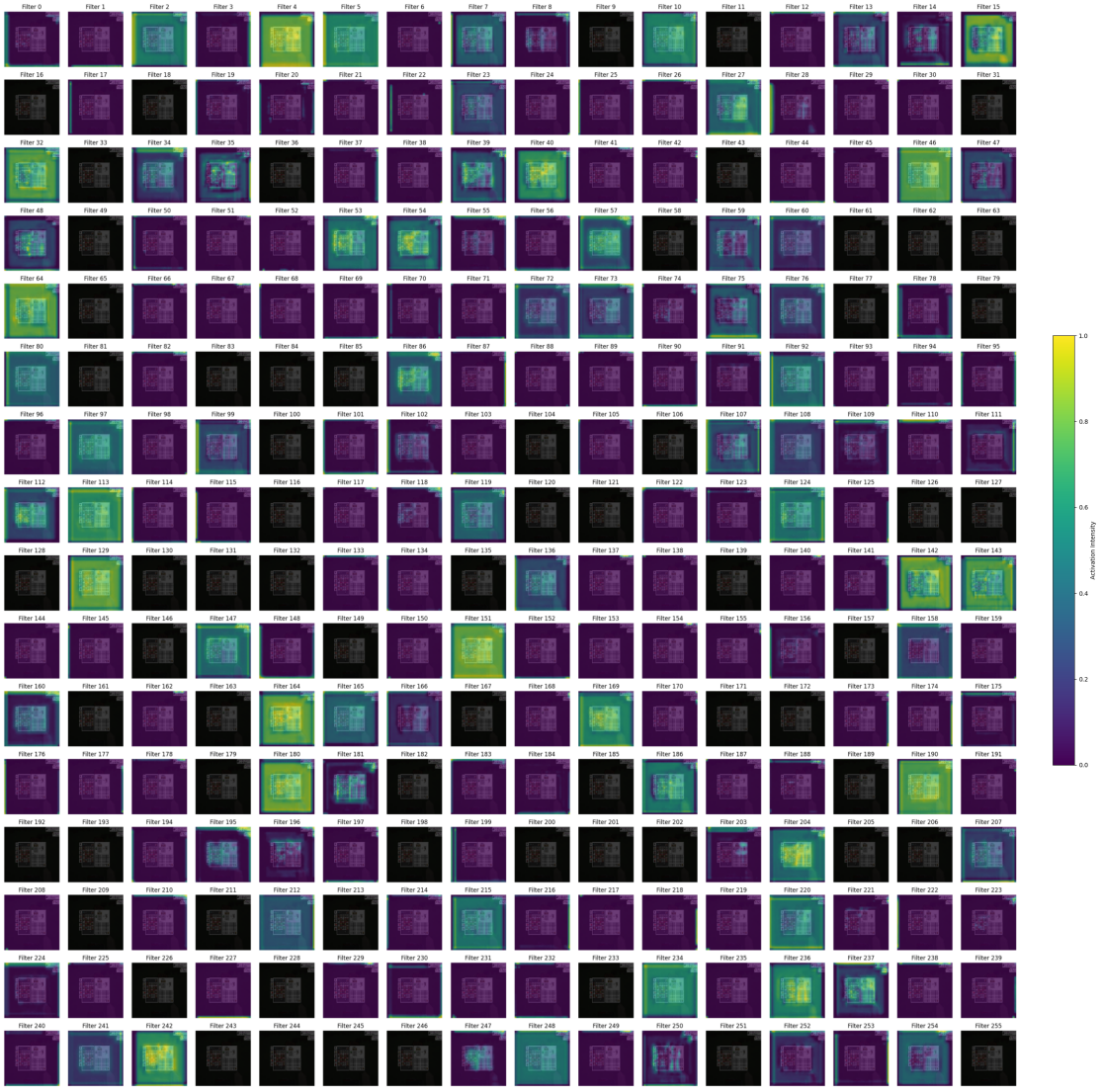

Figure 5.17: Filters overlay.

5.5 Discussion

In this chapter, we undertook a practical exploration of interpretability techniques applied to the VPT model, a large transformer-based agent operating within the Minecraft environment. By employing methods such as attention visualization, and CNN feature visualization, we aimed to uncover insights into the decision-making processes of VPT. While these techniques provided intriguing observations, our analysis also highlighted

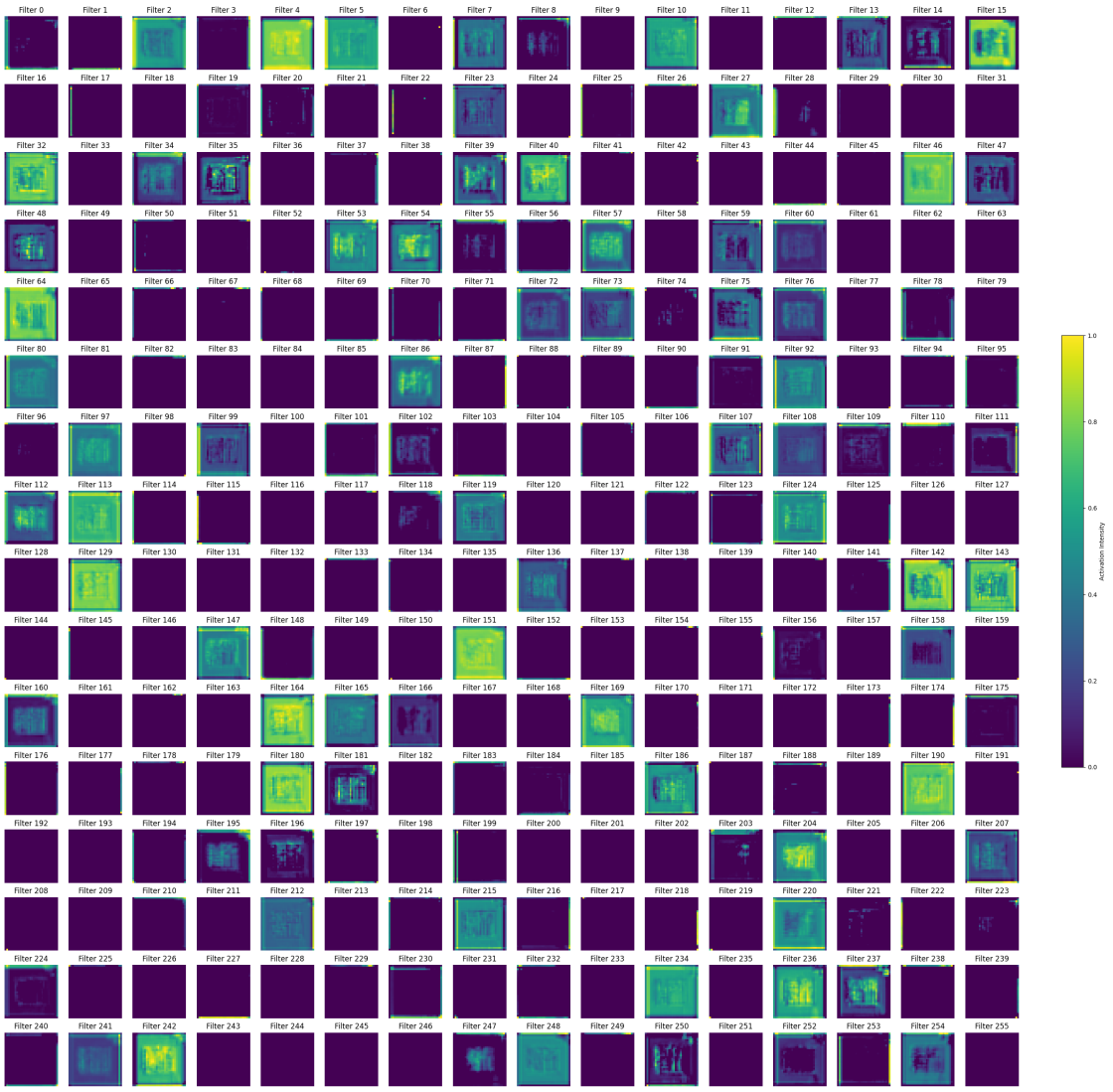

Figure 5.18: Kernel visualization for an inventory image of the 6th layer.

significant challenges in interpreting empirical results without a solid theoretical foundation.

One of the key observations was the apparent specialization of certain attention heads within the transformer architecture. For instance, specific heads seemed to focus on recent frames or particular environmental cues, suggesting a form of temporal or task-specific processing. However, without a robust theoretical framework, it is difficult to ascertain the exact role these attention patterns play in the agent's overall behavior. The visual pat-

terns observed in attention weights and outputs offer only a superficial understanding, leaving us to speculate about the underlying mechanisms.

Similarly, CNN feature visualization techniques revealed that the model detects various visual features at different layers. Early layers captured basic textures and edges, while deeper layers responded to more complex patterns. Notably, the model did not distinctly differentiate between overlapping objects such as a villager and a tree, as evidenced by the lack of boundary in the kernel visualizations. This raises questions about the model's ability to form abstract representations of objects and whether it truly understands the environment in a human-like manner.

Our empirical findings, while suggestive, are challenging to interpret definitively. The patterns observed could be artifacts of the training data or the architecture rather than indicative of meaningful cognitive processes within the model. Without theoretical models that can predict or explain these patterns, we are left with descriptive accounts that lack explanatory power. This limitation hampers our ability to generalize insights or to apply them in guiding the design of more interpretable or robust agents.

Moreover, the stochastic nature of agentic models and the complexity of dynamic environments like Minecraft exacerbate the difficulty of interpretation. Unlike language models where carefully crafted prompts can elicit specific behaviors for study, agentic models interact with an environment that introduces variability and unpredictability. Reproducing specific scenarios or isolating variables is non-trivial, making it hard to establish causal relationships between model internals and observed behaviors.

The lack of solid theoretical underpinnings also limits the practical utility of these interpretability techniques. While visualizations can highlight where the model focuses its attention or which features activate certain neurons, they do not necessarily inform us how to modify the model to achieve desired behaviors or to correct undesired ones. This gap reduces the actionable value of interpretability in improving model performance or safety.

Our work underscores the pressing need for developing theoretical frameworks that can bridge the gap between empirical observations and underlying mechanisms. Such frameworks could provide formal definitions and models that predict how architectural choices or training regimes influence model behavior. They could also offer explanations for why certain patterns emerge in visualizations and how they relate to the model's functional capabilities.

In conclusion, while practical interpretability techniques offer a window into the complex workings of large neural agents like VPT, their insights remain limited without a solid theoretical foundation. The empirical results are challenging to interpret definitively, and their implications for understanding or improving agent behavior are uncertain. Advancing the field will require a concerted effort to develop theoretical models that can explain and predict the phenomena observed empirically, thereby enhancing our ability to interpret, trust, and effectively deploy complex AI agents.

Chapter 6

Conclusion

In this thesis, we have embarked on a comprehensive exploration of mechanistic interpretability and theorical explainability in neural networks, combining theoretical developments with practical applications. Our investigation was motivated by the pressing need to understand the inner workings of complex neural models, particularly as they become increasingly prevalent in critical decision-making processes. We aimed to bridge the gap between theoretical explainability and empirical interpretability, recognizing that a deep theoretical understanding must be complemented by practical insights to effectively unravel the complexities of modern neural networks.

6.1 Summary of Contributions

Our work can be broadly divided into two main parts: theoretical advancements in the analysis of ReLU networks and practical interpretability applied to large-scale transformer-based agents.

6.1.1 Theoretical Advancements

In the initial chapters, we developed a formal framework for understanding ReLU neural networks through the lens of polyhedral geometry. We introduced notions of dimension

specific to neural networks, such as the convex hull dimension, unbounded convex hull dimension, and max cell dimension. These definitions provided a mathematical foundation for quantifying the complexity and structural properties of networks at different layers.

We proved key propositions, such as Proposition 4.5.4, which establishes an upper bound on the dimension of the image of a cell under the network mapping, based on the number of active neurons. This result highlighted how ReLU activations can reduce the effective dimensionality of data as it propagates through the network, offering insights into the role of network architecture in shaping data transformations.

6.1.2 Practical Interpretability

In the latter part of the thesis, we applied interpretability techniques to the VPT agent, a large transformer-based model trained to play Minecraft. By visualizing attention weights and outputs, performing CNN feature visualization, and analyzing saliency maps, we sought to uncover how VPT makes decisions in a complex, dynamic environment.

Our practical exploration revealed patterns in attention mechanisms, with certain heads exhibiting specialized behaviors that could correspond to task-specific processing. Feature visualization techniques provided glimpses into the types of visual features the model attends to at different layers. However, we also encountered challenges in interpreting these empirical results without a solid theoretical foundation.

6.2 Discussion of Findings

6.2.1 Bridging Theory and Practice

One of the central themes emerging from our work is the noticeable gap between theoretical explainability and empirical interpretability. While our theoretical contributions

offer a rigorous mathematical framework for analyzing neural networks, applying these insights directly to practical models like VPT proved challenging.

The theoretical constructs, such as convex hull dimensions and cell dimensions, provide valuable abstractions for understanding the potential behaviors of networks. However, the complexity and stochastic nature of large-scale agentic models introduce nuances that are not fully captured by our theoretical models. This disconnect underscores the limitations of current theoretical frameworks when faced with the intricacies of real-world neural agents.

6.2.2 Challenges in Empirical Interpretability

Our empirical analysis of VPT highlighted several challenges inherent in interpreting complex models:

- **Stochasticity and Environment Interaction**: Agentic models like VPT operate in dynamic environments with inherent randomness. Reproducing specific scenarios or isolating variables is difficult, complicating the task of establishing causal relationships between internal mechanisms and observed behaviors.

- **Limited Context Window**: VPT's attention mechanism has a limited context window, restricting its ability to retain long-term dependencies. Understanding how the model maintains coherence over extended tasks requires further investigation.

- **Interpretation of Visualizations**: While attention and feature visualizations provide some insights, they often lead to speculative conclusions without rigorous backing. The lack of clear theoretical explanations for observed patterns limits the depth of our understanding.

- **Complexity of Agentic Behavior**: Agentic models are trained to maximize long-term rewards, adding layers of complexity to their decision-making processes com-

pared to models optimized for next-token prediction. This makes it harder to decompose their behaviors into understandable components.

6.3 Significance of the Work

Despite the challenges, our work contributes to the field of mechanistic interpretability in several ways:

- **Theoretical Foundations**: We provided formal definitions and propositions that enhance our understanding of how ReLU networks transform data, particularly in terms of dimensionality reduction and the role of activation patterns.
- **Empirical Insights**: Our practical analysis of VPT offers a starting point for interpreting large-scale agentic models. The patterns observed in attention mechanisms and feature activations can inform future research and guide the development of more interpretable models.
- **Highlighting the Theory-Practice Gap**: By attempting to apply theoretical insights to practical models, we have illuminated the existing gap between theoretical explainability and empirical interpretability. Recognizing this gap is a crucial step toward addressing it in future research.
- **Methodological Contributions**: We adapted interpretability techniques commonly used in language models to the domain of agentic models operating in visual environments, demonstrating the potential and limitations of these methods in new contexts.

6.4 Limitations

Our work has several limitations that must be acknowledged:

- **Scope of Theoretical Models**: The theoretical frameworks developed are specific to feedforward ReLU networks and may not generalize to other architectures or activation functions commonly used in practice.
- **Limited Practical Application**: We were unable to directly apply our theoretical results to the practical analysis of VPT. This highlights a limitation in the applicability of current theoretical models to complex, real-world neural agents.
- **Interpretability of Empirical Results**: The empirical observations made in the analysis of VPT are challenging to interpret definitively without a stronger theoretical basis. This limits the actionable insights that can be drawn from our practical work.
- **Complexity of the Environment**: The dynamic and stochastic nature of the Minecraft environment adds layers of complexity that are difficult to control and analyze systematically.

6.5 Future Work

Our findings suggest several avenues for future research:

- **Developing Unified Theoretical Frameworks**: There is a need for theoretical models that can better capture the complexities of agentic models and are applicable across different architectures and modalities.
- **Enhancing Practical Interpretability Tools**: Improving interpretability techniques to provide more definitive insights into model behavior, perhaps by integrating theoretical predictions with empirical observations.
- **Exploring Other Modalities and Models**: Applying interpretability methods to a wider range of models, including those with different architectures, activation functions, and operating in other environments, to test the generality of our observations.

- **Bridging Theory and Practice**: Focusing on creating methods that can directly apply theoretical insights to practical models, reducing the gap identified in our work.
- **Dynamic Environment Analysis**: Developing techniques to better handle the stochasticity and complexity of dynamic environments, enabling more controlled experiments and clearer interpretations.

6.6 Final Remarks

This thesis represents an initial step toward a deeper understanding of mechanistic interpretability in neural networks, combining theoretical developments with practical explorations. While we have made progress in formalizing aspects of neural network behavior and applying interpretability techniques to complex models, the journey is far from complete.

The gap between theoretical explainability and empirical interpretability remains a significant challenge. Bridging this gap will require concerted efforts to develop theories that are informed by practical realities and empirical methods that are grounded in solid theoretical foundations. By highlighting this disconnect, we hope to encourage future research that integrates these perspectives, ultimately leading to more transparent, trustworthy, and effective neural networks.

In an era where neural networks are increasingly integral to critical applications, understanding their inner workings is not just an academic pursuit but a necessity for ensuring reliability and accountability. We believe that the insights and reflections presented in this thesis contribute to this important endeavor and will inspire further advances in the field of interpretability.

Bibliography

[1] Abi Raad, M., Ahuja, A., Barros, C., Besse, F., Bolt, A., Bolton, A., Brownfield, B., Buttimore, G., Cant, M., Chakera, S., Chan, S. C. Y., Clune, J., Collister, A., Copeman, V., Cullum, A., Dasgupta, I., de Cesare, D., Di Trapani, J., Donchev, Y., Dunleavy, E., Engelcke, M., Faulkner, R., Garcia, F., Gbadamosi, C., Gong, Z., Gonzales, L., Gregor, K., Gupta, K., Hallingstad, A. O., Harley, T., Haves, S., Hill, F., Hirst, E., Hudson, D. A., Hudson, J., Hughes-Fitt, S., Rezende, D. J., Jasarevic, M., Kampis, L., Ke, R., Keck, T., Kim, J., Knagg, O., Kopparapu, K., Lampinen, A., Legg, S., Lerchner, A., Limont, M., Liu, Y., Loks-Thompson, M., Marino, J., Martin Cussons, K., Matthey, L., Mcloughlin, S., Mendolicchio, P., Merzic, H., Mitenkova, A., Moufarek, A., Oliveira, V., Oliveira, Y., Openshaw, H., Pan, R., Pappu, A., Platonov, A., Purkiss, O., Reichert, D., Reid, J., Richemond, P. H., Roberts, T., Ruscoe, G., Sanchez Elias, J., Sandars, T., Sawyer, D. P., Scholtes, T., Simmons, G., Slater, D., Soyer, H., Strathmann, H., Stys, P., Tam, A. C., Teplyashin, D., Terzi, T., Vercelli, D., Vujatovic, B., Wainwright, M., Wang, J. X., Wang, Z., Wierstra, D., Williams, D., Wong, N., York, S., and Young, N. Scaling instructable agents across many simulated worlds. *arXiv preprint arXiv:2404.10179* (2024).

[2] Arora, R., Basu, A., Mianjy, P., and Mukherjee, A. Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491* (2016).

[3] BADRINARAYANAN, V., MISHRA, B., AND CIPOLLA, R. Symmetry-invariant optimization in deep networks. *arXiv preprint arXiv:1511.01754* (2015).

[4] BAKER, B., AKKAYA, I., ZHOKHOV, P., HUIZINGA, J., TANG, J., ECOFFET, A., HOUGHTON, B., SAMPEDRO, R., AND CLUNE, J. Video pretraining (vpt): Learning to act by watching unlabeled online videos. In *NeurIPS 2022* (2022).

[5] BARTO, A. G., AND MAHADEVAN, S. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems 13*, 4 (2003), 341–379.

[6] BEECHEY, D., SMITH, T. M., AND ŞIMŞEK, Ö. Explaining reinforcement learning with shapley values. In *International Conference on Machine Learning* (2023), PMLR, pp. 2003–2014.

[7] BUI THI MAI, P., AND LAMPERT, C. Functional vs. parametric equivalence of relu networks. In *8th International Conference on Learning Representations* (2020).

[8] CANTILLON, P., WOOD, D. F., AND YARDLEY, S. *ABC of Learning and Teaching in Medicine*. John Wiley & Sons, 2017.

[9] CARLINI, N., JAGIELSKI, M., AND MIRONOV, I. Cryptanalytic extraction of neural network models. In *Advances in Cryptology–CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part III* (2020), Springer, pp. 189–218.

[10] CHAUKAIR, M., SCHÜTTE, C., AND SUNKARA, V. On the activation space of relu equipped deep neural networks. *Procedia Computer Science 222* (2023), 624–635.

[11] CONMY, A., MAVOR-PARKER, A., LYNCH, A., HEIMERSHEIM, S., AND GARRIGA-ALONSO, A. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems 36* (2023), 16318–16352.

[12] DAO, G., MISHRA, I., AND LEE, M. Deep reinforcement learning monitor for snapshot recording. In *Proceedings of the 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA '18)* (Los Alamitos, CA, 2018), IEEE, pp. 591–598.

[13] DE LOERA, J., RAMBAU, J., AND SANTOS, F. *Triangulations: structures for algorithms and applications*, vol. 25. Springer Science & Business Media, 2010.

[14] ELHAGE, N., NANDA, N., OLSSON, C., HENIGHAN, T., JOSEPH, N., MANN, B., ASKELL, A., BAI, Y., CHEN, A., CONERLY, T., ET AL. A mathematical framework for transformer circuits. *Transformer Circuits Thread 1* (2021), 1.

[15] ENTEZARI, R., SEDGHI, H., SAUKH, O., AND NEYSHABUR, B. The role of permutation invariance in linear mode connectivity of neural networks. *arXiv preprint arXiv:2110.06296* (2021).

[16] FEFFERMAN, C. Reconstructing a neural net from its output. *Revista Matemática Iberoamericana 10*, 3 (1994), 507–555.

[17] FIGURE. Building ai powered robots, 2024.

[18] GANDELSMAN, Y., EFROS, A. A., AND STEINHARDT, J. Interpreting clip's image representation via text-based decomposition. In *International Conference on Learning Representations* (2024).

[19] GIRYES, R., SAPIRO, G., AND BRONSTEIN, A. M. Deep neural networks with random gaussian weights: A universal classification strategy? *IEEE Trans. Signal Process. 64*, 13 (2016), 3444–3457.

[20] GIRYES, R., SAPIRO, G., AND BRONSTEIN, A. M. Deep neural networks with random gaussian weights: A universal classification strategy? *IEEE Transactions on Signal Processing 64*, 13 (2016), 3444–3457.

[21] GLANOIS, C., WENG, P., ZIMMER, M., LI, D., YANG, T., HAO, J., AND LIU, W. A survey on interpretable reinforcement learning. *arXiv preprint arXiv:2112.13112* (2021).

[22] GRIGSBY, E., LINDSEY, K., AND ROLNICK, D. Hidden symmetries of relu networks. In *International Conference on Machine Learning* (2023), PMLR, pp. 11734–11760.

[23] GRIGSBY, J. E., AND LINDSEY, K. On transversality of bent hyperplane arrangements and the topological expressiveness of relu neural networks. *SIAM Journal on Applied Algebra and Geometry 6*, 2 (2022), 216–242.

[24] GRIGSBY, J. E., LINDSEY, K., AND MASDEN, M. Local and global topological complexity measures of relu neural network functions. *arXiv preprint arXiv:2204.06062* (2022).

[25] GRIGSBY, J. E., LINDSEY, K., MEYERHOFF, R., AND WU, C. Functional dimension of feedforward relu neural networks. *arXiv preprint arXiv:2209.04036* (2022).

[26] GRÜNBAUM, B., KLEE, V., PERLES, M. A., AND SHEPHARD, G. C. *Convex polytopes*, vol. 16. Springer, 1967.

[27] GRÜNBAUM, B., AND SHEPHARD, G. C. Convex polytopes. *Bulletin of the London Mathematical Society 1*, 3 (1969), 257–300.

[28] GRUNERT, R. *Piecewise linear Morse theory*. PhD thesis, 2017.

[29] GUSS, W. H., HOUGHTON, B., TOPIN, N., WANG, P., CODEL, C., VELOSO, M., AND SALAKHUTDINOV, R. Minerl: A large-scale dataset of minecraft demonstrations. *CoRR abs/1907.13440* (2019).

[30] HANIN, B. Universal function approximation by deep neural nets with bounded width and relu activations. *Mathematics 7*, 10 (2019), 992.

[31] HANIN, B., AND ROLNICK, D. How to start training: The effect of initialization and architecture. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada* (2018), pp. 569–579.

[32] HANIN, B., AND ROLNICK, D. Complexity of linear regions in deep networks. In *International Conference on Machine Learning* (2019), PMLR, pp. 2596–2604.

[33] HANIN, B., AND ROLNICK, D. Deep relu networks have surprisingly few activation patterns. *Advances in neural information processing systems 32* (2019).

[34] HANIN, B., AND ROLNICK, D. Deep relu networks have surprisingly few activation patterns. *Advances in neural information processing systems 32* (2019).

[35] HANNA, M., LIU, O., AND VARIENGIEN, A. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In *NeurIPS 2023* (2023).

[36] HILTON, J., CAMMARATA, N., CARTER, S., GOH, G., AND OLAH, C. Understanding rl vision. *Distill* (2020). https://distill.pub/2020/understanding-rl-vision.

[37] JAKUB, C., AND NICA, M. Depth degeneracy in neural networks: Vanishing angles in fully connected ReLU networks on initialization. arXiv:stat.ML/2302.09712.

[38] JAKUB, C., AND NICA, M. Depth degeneracy in neural networks: Vanishing angles in fully connected relu networks on initialization. *arXiv preprint arXiv:2302.09712* (2023).

[39] KENNY, E. M., TUCKER, M., AND SHAH, J. Towards interpretable deep reinforcement learning with human-friendly prototypes. In *The Eleventh International Conference on Learning Representations* (2023).

[40] LIEBERUM, T., RAHTZ, M., KRAMÁR, J., IRVING, G., SHAH, R., AND MIKULIK, V. Does circuit analysis interpretability scale? evidence from multiple choice capabilities in chinchilla. *arXiv preprint arXiv:2307.09458* (2023).

[41] LIFSHITZ, S., PASTER, K., CHAN, H., BA, J., AND MCILRAITH, S. Steve-1: A generative model for text-to-behavior in minecraft. *arXiv preprint arXiv:2306.00937* (2023).

[42] LIN, H. W., TEGMARK, M., AND ROLNICK, D. Why does deep and cheap learning work so well? *Journal of Statistical Physics 168* (2017), 1223–1247.

[43] MASDEN, M. Algorithmic determination of the combinatorial structure of the linear regions of relu neural networks. *arXiv preprint arXiv:2207.07696* (2022).

[44] MCDOUGALL, C., CONMY, A., RUSHING, C., MCGRATH, T., AND NANDA, N. Copy suppression: Comprehensively understanding an attention head. *arXiv preprint arXiv:2310.04625* (2023).

[45] MILANI, S., KANERVISTO, A., RAMANAUSKAS, K., SCHULHOFF, S., HOUGHTON, B., MOHANTY, S., GALBRAITH, B., CHEN, K., SONG, Y., ZHOU, T., ET AL. Towards solving fuzzy tasks with human feedback: A retrospective of the minerl basalt 2022 competition. *arXiv preprint arXiv:2303.13512* (2023).

[46] MILANI, S., TOPIN, N., VELOSO, M., AND FANG, F. Explainable reinforcement learning: A survey and comparative review. *ACM Comput. Surv. 56*, 7 (apr 2024).

[47] MILLI, S., SCHMIDT, L., DRAGAN, A. D., AND HARDT, M. Model reconstruction from model explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency* (2019), pp. 1–9.

[48] MIXON, D. G., PARSHALL, H., AND PI, J. Neural collapse with unconstrained features. *Sampling Theory, Signal Processing, and Data Analysis 20*, 2 (2022), 11.

[49] MOJANG. Minecraft. `https://www.minecraft.net/`, 2011.

[50] OLSSON, C., ELHAGE, N., NANDA, N., JOSEPH, N., DASSARMA, N., HENIGHAN, T., MANN, B., ASKELL, A., BAI, Y., CHEN, A., CONERLY, T., DRAIN, D., GANGULI, D., HATFIELD-DODDS, Z., HERNANDEZ, D., JOHNSTON, S., JONES, A., KERNION, J., LOVITT, L., NDOUSSE, K., AMODEI, D., BROWN, T., CLARK, J., KAPLAN, J., MCCANDLISH, S., AND OLAH, C. In-context learning and induction heads, 2022.

[51] OLSSON, C., ELHAGE, N., NANDA, N., JOSEPH, N., DASSARMA, N., HENIGHAN, T., MANN, B., ASKELL, A., BAI, Y., CHEN, A., ET AL. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895* (2022).

[52] PAPYAN, V., HAN, X., AND DONOHO, D. L. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences 117*, 40 (2020), 24652–24663.

[53] PUIUTTA, E., AND VEITH, E. M. Explainable reinforcement learning: A survey. *arXiv preprint arXiv:2005.06247* (2020).

[54] RAGHU, M., POOLE, B., KLEINBERG, J., GANGULI, S., AND SOHL-DICKSTEIN, J. On the expressive power of deep neural networks. In *international conference on machine learning* (2017), PMLR, pp. 2847–2854.

[55] RIBEIRO, M. T., SINGH, S., AND GUESTRIN, C. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2016), KDD '16, Association for Computing Machinery, p. 1135–1144.

[56] ROLNICK, D., AND KORDING, K. Reverse-engineering deep relu networks. In *International Conference on Machine Learning* (2020), PMLR, pp. 8178–8187.

[57] ROLNICK, D., AND TEGMARK, M. The power of deeper networks for expressing natural functions. *arXiv preprint arXiv:1705.05502* (2017).

[58] SACKS, O. *Musicophilia: Tales of Music and the Brain*. Knopf, New York, 2007.

[59] SIMON, J. B., ANAND, S., AND DEWEESE, M. Reverse engineering the neural tangent kernel. In *International Conference on Machine Learning* (2022), PMLR, pp. 20215–20231.

[60] WANG, K., VARIENGIEN, A., CONMY, A., SHLEGERIS, B., AND STEINHARDT, J. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593* (2022).

[61] WATSON, D. S. Conceptual challenges for interpretable machine learning. *Synthese 200*, 2 (2022), 65.

[62] XU, D., AND FEKRI, F. Interpretable model-based hierarchical reinforcement learning using inductive logic programming. *CoRR abs/2106.11417* (2021).

[63] YARAS, C., WANG, P., ZHU, Z., BALZANO, L., AND QU, Q. Neural collapse with normalized features: A geometric analysis over the riemannian manifold. *Advances in neural information processing systems 35* (2022), 11547–11560.

[64] ZHANG, C., BENGIO, S., HARDT, M., RECHT, B., AND VINYALS, O. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM 64*, 3 (2021), 107–115.

[65] ZHU, Z., DING, T., ZHOU, J., LI, X., YOU, C., SULAM, J., AND QU, Q. A geometric analysis of neural collapse with unconstrained features. *Advances in Neural Information Processing Systems 34* (2021), 29820–29834.

[66] ZOLMAN, N., FASEL, U., KUTZ, J. N., AND BRUNTON, S. L. Sindy-rl: Interpretable and efficient model-based reinforcement learning. *arXiv preprint arXiv:2403.09110* (2024).