Comparing Linear Regions in ReLU-Type Networks: Theory and Monte Carlo Methods

Yuan Wang
Independent Researcher
Singapore
yuanwang2011@outlook.com

Abstract

In this work, we establish the uniqueness of linear patterns for generic fully connected neural networks activated by ReLU or leaky ReLU functions. With this result, together with several tools from geometry and probability, we compare the numbers of linear regions for a wide variety of networks from both theoretical and practical perspectives, with minimal prerequisites. More specifically, on the theoretical side, we show that networks with neurons arranged as $pq \times 2$ (2 hidden layers, each having pq neurons) are expected to have more linear regions than those arranged as $p \times 2q$ for $p \geqslant 1$ and $q \geqslant 2$; on the empirical side, we show the validity of a Monte-Carlo approach for comparing linear regions via linear patterns, and carry out experiments for networks in various scenarios. Our results indicate that neurons closer to the input layer tend to have more impact on the number of linear regions. Moreover, along the way, we find the precise value of the expected number of linear regions for networks with 2 hidden layers, and an explicit upper bound for networks whose layer widths are non-increasing. The implementations for this paper are available through the following link:
https://github.com/yuanwang-research/Regions-ReLU-Network

1 Introduction

As neural networks become increasingly widely used, there is growing demand to understand their internal structure. Among all network types, those activated by ReLU and leaky ReLU functions have drawn particular attention. One of the key properties of such networks is that their activation functions, and hence the function represented by the whole network, are piecewise linear. Because of this, the complexity of such networks is reflected by the number of linear regions. In recent years, the linear regions of ReLU and leaky ReLU networks have been studied from various perspectives, including qualitative characterization, quantitative estimation, and empirical calculation (e.g. [MPCB14], [RPK+17], [HR19], [XHY+20], [Wan22], [Ber23], [VAS+24], [SHR25]).

For convenience, we call a network activated by either the ReLU or the leaky ReLU function a network of *ReLU-type*. Despite numerous studies on ReLU-type networks, the understanding of their linear regions is still far from ideal. In particular, direct comparisons between the expected numbers of linear regions for networks with different neuron arrangements are still largely missing. Related works worth mentioning include [HR19], which gives an upper bound for the expected density of linear regions, and [Wan22], which compares the expected number of linear regions for a very limited class of networks. This problem is challenging because calculating the precise number of linear regions for a ReLU-activated network is NP-hard ([Wan22, Theorem 3.2], [SHR25, Section 5]).

In this paper, we provide several results and tools from both theoretical and empirical perspectives that are capable of comparing the numbers of linear regions for a wide range of ReLU-type networks. Along the way, we also reveal some fundamental properties of those networks. More specifically, in

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

the theoretical part, we show that a gap exists between the expected numbers of linear regions for network structures with hidden neurons arranged as $pq \times 2$ and $p \times 2q$. The rigorous statement of the result can be cleanly put in a self-contained way as follows:

Theorem 1.1. *Denote by $\mathcal{N}_{n\times l}$ a fully-connected ReLU-type neural network that has input dimension M, output dimension 1, and l hidden layers, each containing n neurons, and whose weights and biases are all independent continuous random variables with even density functions. Denote by $E(\mathcal{N}_{n\times l})$ the expectation of the number of linear regions of $\mathcal{N}_{n\times l}$ with respect to its weights and biases. Then $E(\mathcal{N}_{pq\times 2}) > E(\mathcal{N}_{p\times 2q})$ for all integers $p \geqslant 1$, $q \geqslant 2$, and $M \geqslant pq$.*

Theorem 1.1 corresponds to Theorem 3.7, Part 3. Observe that there is no extra condition besides the arrangements of neurons in the theorem. In particular, the assumption for the distribution of weights and bias is a generalization of many typical networks upon initialization. For example, Theorem 1.1 (in fact, the entire Theorem 3.7) directly applies to the case when one initializes a module with PyTorch[1] using the function `torch.nn.Linear`, as the default distribution for this function is a variation of the Kaiming uniform distribution [HZRS15], which is a special case of that in Theorem 1.1.

In the process of proving Theorem 1.1, we give an explicit upper bound for the number of linear regions for general networks with non-increasing layer widths (Theorem 3.7, Part 1). Moreover, we find the precise value of the expected number of linear regions for networks with two hidden layers. Surprisingly, for the leaky ReLU case, the number of linear regions is generically constant (Theorem 3.7, Part 2), but this no longer holds for networks with depth 3 or more (Example 3.9).

One of the key observations that contributes to Theorem 1.1 is Theorem 3.3, which establishes the uniqueness of linear patterns for networks with generic parameters. More specifically, Theorem 3.3 says that every non-constant linear region for a network has a generically unique linear pattern, and this also holds for constant linear regions for networks with depth $\leqslant 2$ and the width of the input layer is no less than that of the first layer, but not for those beyond that (by Example 3.5 and 3.6). This fact justifies, from a theoretical perspective, a Monte-Carlo approach to compare linear regions for different networks. Roughly speaking, we randomly sample a set of input data and send them through different networks, and compare the numbers of distinct linear patterns corresponding to the inputs. Multiple experiments are then carried out for networks with random parameters as well as those during training processes.

In recent years, some research has suggested that the expressibility gained from increased depth is smaller than expected, especially compared with the gain from increased width (e.g. [HR19], [Wan22]). Our theorems, examples, and experimental results not only provide stronger support for this view, but also indicate a more fundamental tendency: *hidden layers closer to the input layer tend to exert greater influence on the number of linear regions* (especially in settings with non-constant widths, such as Example 3.8 and the part of Section 4 for networks with varying sizes of layers). In particular, adding neurons to hidden layers closer to the input layer usually results in a larger increase in the number of linear regions than adding them to more distant layers.

We summarize the contribution of this paper as follows:

- A theoretical result comparing the number of linear regions for a large group of ReLU-type networks ($pq \times 2$ vs. $p \times 2q$), with byproduct results including an upper bound for the expected number of linear regions, as well as precise values for the expectations (resp. numbers) for ReLU (resp. leaky ReLU) networks with 2 hidden layers.
- A practical approach to use Monte-Carlo method to compare the numbers of linear regions for arbitrary ReLU-type networks, together with a result on the uniqueness of linear patterns that supports this approach theoretically.
- Although this is not among the main novel contributions of the paper, we provide an open-source tool to explicitly construct linear regions for ReLU-type networks (accessible through the link in the abstract). In particular, for a network with input dimension 2, the tool can explicitly draw its linear regions as a 3D model. In fact, all 3D models of linear regions presented in later sections of this paper were produced with this tool (and are also open-sourced).

[1] version 2.11.0, the latest stable version by the time this paper was finished

1.1 Comparison with related works

For the theoretical part, the result closest to ours is [Wan22, Theorem 1.1], which is, as far as we know, the only direct comparison result for concrete network structures with random parameters. However, the scope of Theorem 1.1 is far broader than that of loc. cit., since we are able to handle multiple widths for the networks on the right-hand side of the inequality, rather than the fixed width of 2 in loc. cit. Moreover, the upper bound for arbitrary non-increasing widths (Theorem 3.7, Part 1), on which this comparison relies, has no counterpart in loc. cit., which only treats deep networks of width 1 or 2 and networks with 2 hidden layers. It is also worth noting that Theorem 3.7, Part (2a) gives the exact expectation for networks with 2 layers, whereas [Wan22, Theorem 1.2, Part 2] only gives a range. Furthermore, we handle the leaky ReLU case in addition to the ReLU case throughout, and observe the surprising phenomenon that the number of linear regions is generically constant for the 2-layer case (which is no longer true for depth $\geqslant 3$).

In a stochastic framework, [GEU24] bounds the expected density of linear regions along a one-dimensional path for networks with general piecewise linear activations. In contrast, we study the expected total number of linear regions over the entire input space, which depends on neither a path nor an input distribution. The two works are complementary: loc. cit. covers a broader class of activations, whereas our specialization to ReLU-type networks allows direct comparisons between architectures.

Approximation-theoretic results show that deep networks can approximate certain functions with far fewer units than shallow ones (e.g. [LS17], [Yar17]). They do not conflict with ours: they measure the error of approximating a prescribed target on a bounded domain with target-dependent parameters, whereas we count all linear regions over the entire input space for generic random parameters. Since approximation also depends on where the regions are placed, our results do not suggest that shallow networks are universally preferable; rather, the number of linear regions, a common expressivity proxy whose worst-case maximum is often used to argue for depth [MPCB14], can favor width in expectation for typical random parameters.

For the empirical part, it is well known that the linear pattern of a generic point reflects the linear region it belongs to. The innovative part here is that we provide theoretical support (Theorem 3.3, which has no counterpart in [Wan22]), which makes linear patterns a theoretically sound tool for comparing linear regions, and run experiments to compare networks (both randomly generated and during training processes), from which we obtain insights beyond the reach of previous analyses as far as we know. Unlike the exact algorithm of [Wan22], this method scales to large networks.

1.2 Limitations

For the theoretical part, there is still a gap between Theorem 1.1 and the ideal goal of comparing the expected numbers of linear regions for arbitrary network structures. Bridging this gap may be challenging, given the NP-hardness of computing the number of linear regions in general. For the empirical Monte-Carlo method, although it is theoretically supported and performs well in our experiments, it only detects regions reached by the samples and hence gives a lower bound on the number of non-constant linear regions; its resolution depends on the sample size and the input distribution (Subsection 3.3), and comparisons are inconclusive when the count saturates.

2 Preliminaries and notations

In this section, we rigorously set up notations and concepts used in the theorems, examples, and experiments in the following sections.

2.1 The basics

Throughout the paper, by *network*, we always mean multi-layer perceptrons (MLP). A network consists of the input layer, several hidden layers and an output layer, and each of the hidden layers contains some neurons. For the numbering of the layers, we call the input layer the 0-th layer, and call the i-th hidden layer the i-th layer (here i starts from 1), and finally call the output layer the $(l+1)$-th layer given there are l hidden layers in total. We use n_i to denote the number of neurons in the i-th layer and use N_i^j to denote the j-th neuron in the i-th hidden layer.

For a network $\mathcal{N}$, we define $w_i^{j,k}$ as the weight from the j-th neuron of the $(i-1)$-th layer to the k-th neuron of the i-th layer of $\mathcal{N}$. In particular, when $i = 1$, $w_1^{j,k}$ corresponds to the weights between the input layer and the first hidden layer. Denote by W_i the matrix of shape (n_{i-1}, n_i) whose (j,k)-th entry is $w_i^{j,k}$. We define b_i^j as the bias for the j-th neuron in the i-th layer and B_i as the vector formed by $\{b_i^j\}$.

For a network $\mathcal{N}$ with input dimension n and output dimension m, we use F to denote the piecewise linear function (from $\mathbb{R}^n$ to $\mathbb{R}^m$) determined by $\mathcal{N}$. Precisely, if $\mathcal{N}$ has l hidden layers in total, then $F = g \circ f_l \circ f_{l-1} \circ ... \circ f_2 \circ f_1$. Here $f_i(x) = a(W_i^\top x + B_i)$ (from $\mathbb{R}^{n_{i-1}}$ to $\mathbb{R}^{n_i}$) is the operation corresponding to the i-th layer, where $a(x)$ is the activation function (applied to each element of the vector) which can be either ReLU or Leaky ReLU, and $g(x) = W_{l+1}^\top x + B_{l+1}$ (from $\mathbb{R}^{n_l}$ to $\mathbb{R}^m$) is the operation in the output layer (where no activation is applied). Moreover, we define F_i as $F_i = f_i \circ f_{i-1} \circ ... \circ f_2 \circ f_1$ which is the part of F from the input to the i-th hidden layer. We next define $\tilde{F}_i = \tilde{f}_i \circ \tilde{f}_{i-1} \circ ... \circ \tilde{f}_2 \circ \tilde{f}_1$ for $i = 1, 2, ..., l+1$ as the variation of F_i where all the activations are removed, i.e. $\tilde{f}_i(x) = W_i^\top x + B_i$. In particular, $\tilde{F}_i$ does not rely on the choice of activation function. We denote F_i^j ($\tilde{F}_i^j$) as the j-th entry of the output of F_i (corresponding to N_i^j).

We define a *generic point* in an ambient space as a point in the space possibly with a subset of measure zero removed. We say that a network is *generic* if the weights and biases of the network are chosen as a generic point in the parameter space. In fact, throughout this paper we focus on networks with generic parameters which is usually the case in reality.

2.2 Activation regions, linear regions and linear patterns

We define an *activation region* as the union of points sharing a common activation pattern (i.e. the activation states of all neurons, whether activated or not; see [SHR25, Section 2]) that has full dimension (i.e. the same dimension as the input space). Recently, [SHR25] collected and compared commonly used definitions of linear region and activation region, and our definition of activation region corresponds to [SHR25, Definition 2]. Our definition is also the same as that in [LCPK23, Section 2.2], with the full-dimension requirement. It is well known that an activation region is convex; see [LCPK23, Section 2.3] for a proof.

We define a *linear region* as the closure of an open, inclusion-wise maximal connected subset of the input space on which F is affine, i.e. the closure of a region in [SHR25, Definition 4]. We define the *linear pattern* for a generic point P as the linear function obtained by restricting F to a sufficiently small neighborhood of P.

2.3 The hyperplanes, orthants and strata formed by layers

Suppose that the numbers of neurons $n_0, n_1, n_2, ...$ in layers (including the input layer) are non-increasing and the weight matrices $\{W_i\}$ are all of full rank. In this case, for any i and j, $\tilde{F}_i^j(x) = 0$ (resp. $\tilde{F}_i^j(x) \geqslant 0$) defines a hyperplane (resp. half-space) in the input space $\mathbb{R}^n$, which we define as H_i^j (resp. ${}^+H_i^j$). Moreover, for a fixed layer index i, the hyperplanes H_i^j's are in general position, thus forming an Euclidean coordinate which we call the *coordinate* for the i-th layer. In particular, there are 2^{n_i} orthants in the coordinate which we call the *orthants* for the i-th layer. We also define the *positive orthant* for the i-th layer, denoted by ${}^+\mathcal{Q}_i$, as

$$ {}^+\mathcal{Q}_i := \begin{cases} \mathbb{R}^n, & i = 0 \\ \bigcap_{j=1}^{n_i} {}^+H_i^j, & i > 0 \end{cases} $$

Next, we define a d-dimensional *stratum* associated to distinct indices $j_1, j_2, ..., j_{n_i - d}$ for layer i as $(\bigcap_{k=1,2,...,n_i-d} H_i^{j_k}) \cap {}^+\mathcal{Q}_i$. Especially, the n_i-dimensional stratum is just ${}^+\mathcal{Q}_i$ itself. The d-dimensional *strata* is defined simply as the set containing every stratum of dimension d. When $n_i < n$, the hyperplanes H_i^j do not form a coordinate system of $\mathbb{R}^n$; the orthants and strata are then the preimages, under the affine surjection $\tilde{F}_i : \mathbb{R}^n \to \mathbb{R}^{n_i}$, of those in $\mathbb{R}^{n_i}$ (with dimensions measured there). As the layers before the i-th one are linear in our proofs, F factors through $\tilde{F}_i$, and taking preimages gives a bijection between regions preserving convexity, connectedness and distinct linear patterns, so we may count in $\mathbb{R}^{n_i}$.

We say that two orthants $\mathcal{Q}'$ and $\mathcal{Q}''$ for the i-th layer are *adjacent* if the intersection of $\mathcal{Q}'$ and $\mathcal{Q}''$ is of dimension $n-1$. We define the *distance* from an arbitrary orthant $\mathcal{Q}$ to the positive orthant ${}^+\mathcal{Q}_i$ as the minimum number of steps to move from $\mathcal{Q}$ to ${}^+\mathcal{Q}_i$ by moving to an adjacent orthant each time.

For any $x \in \mathbb{R}^n \backslash {}^+\mathcal{Q}_1$, define L_x as the line segment projecting x to $\partial^+\mathcal{Q}_1$, the boundary of ${}^+\mathcal{Q}_1$, along the coordinates for layer 1. Rigorously, for any real number c, define affine spaces ${}_cH_1^j = \{x \in \mathbb{R}^n | F_1^j(x) = c\}$. By the fact that F_1^j is a linear function activated by either a ReLU or leaky-ReLU function only once, we know that ${}_cH_1^j$ is connected. We then define T_x as the point closest to x with $\tilde{F}_1^j(T_x) = F_1^j(x)$ for all j, so that $T_x \in {}_{F_1^j(x)}H_1^j$. Since $F_1^j(x)$ is always non-negative, T_x belongs to ${}^+\mathcal{Q}_1$. Then L_x is the line segment connecting x and T_x.

Lemma 2.1. *For a ReLU-activated network, for a generic $x \in \mathbb{R}^n \backslash {}^+\mathcal{Q}_1$, F is constant along L_x. In particular, all the points in L_x always belong to the same linear region.*

Proof. Let $K = \{k | F_1^k(x) > 0\}$. For all $k \in K$, by definition, we have that x and T_x both belong to ${}_{F_1^k(x)}H_1^k$. So F_1^k remains constant along L_x. On the other hand, for all $j \notin K$, x and T_x both belong to ${}_0H_1^j$, so F_1^j equals to 0 along L_x. Therefore F_1^i (for all $i = 0, 1, 2, ..., n_1$), hence F, remains constant along L_x. □

By Lemma 2.1, for a ReLU-activated network, every activation region can be reduced to an activation region in a stratum of ${}^+\mathcal{Q}_1$ by projecting each point P in that region along the line L_P. This reduction is essential for handling linear regions in ReLU-activated networks in the proofs of the two main theoretical results (Theorem 3.3 and Theorem 3.7) in this paper.

In the appendix (Section A.1), we provide an illustration of the concepts introduced in this section.

3 Theoretical results

For convenience, we extract the notations and conditions commonly used in the statements of the theorems as follows.

Notation-Condition 3.1. *Let $\mathcal{N}$ be a fully-connected ReLU-type neural network that has input dimension M, output dimension 1, and l hidden layers. The notation $\mathcal{N}_{(n_1,n_2,...,n_l)}$ means that the i-th hidden layer of $\mathcal{N}$ has n_i neurons; the notation $\mathcal{N}_{n\times l}$ means that each of the l hidden layers has n neurons.*

Notation-Condition 3.2. *Assume Notation-Condition 3.1. Assume $l \geqslant 2$ and the weights and biases of $\mathcal{N}$ are all independent continuous random variables with even density functions. Define $E(\mathcal{N})$ as the expectation of the linear regions of $\mathcal{N}$ with respect to the weights and biases.*

For Notation-Condition 3.1 and 3.2, we can say that assume one of them with a specific activation function (e.g. ReLU or leaky ReLU), which applies to all the activation functions in the network.

3.1 Generic uniqueness of linear regions

We now present the theorem on the generic uniqueness of linear patterns as follows.

Theorem 3.3. *Assume Notation-Condition 3.1. For generic weights and biases, we have*

1. *The non-constant linear regions of $\mathcal{N}$ are exactly the same as the non-constant activation regions. In particular, they are convex, and there are no two linear regions having the same linear pattern.*

2. *Suppose that $M \geqslant n_1$ where n_1 is the width of the first layer of $\mathcal{N}$. For constant linear regions, there are no two linear regions of $\mathcal{N}$ that have the same linear pattern when $l \leqslant 2$.*

Remark 3.4. For networks with leaky ReLU activation functions, there is no constant linear region as long as the weights and biases are generic. This is easy to see, as all computations an input vector encounters are full-rank affine transformations, as well as the leaky ReLU function (for each entry of an input vector), which is either identity or scalar multiplication by a nonzero scalar. So generically speaking, different activation patterns result in different linear patterns. Therefore, the conclusion of Part 1 holds for all linear regions in the network.

The proof of Theorem 3.3 can be found in the appendix. For Part 1, we show that two distinct activation patterns, at least one of which is non-constant, induce linear patterns given by distinct polynomials in the parameters. If we remove either the condition $M \geqslant n_1$ or the one that $l \leqslant 2$, Theorem 3.3 Part 2 no longer holds. For the former condition, we can construct a very simple counterexample as follows.

Example 3.5. Let $\mathcal{N}$ be a ReLU-activated network with input dimension 1, output dimension 1, and 2 hidden layers each of which has 2 neurons. The weights and biases for each layer of $\mathcal{N}$ are set as follows. In particular, the weights towards the output layer are all 1 and the bias for the output neuron is 0.

$$W_1 = \begin{bmatrix} 1 & 1 \end{bmatrix},\ B_1 = (b_1^1 = 0, b_1^2 = 2),\ W_2 = \begin{bmatrix} w_2^{1,1} = -2 & w_2^{1,2} = 1 \\ w_2^{2,1} = 1 & w_2^{2,2} = 2 \end{bmatrix},\ B_2 = (-1, -8)$$

Denote the input for this network as x. There are 5 linear regions in total which are $(-\infty, -1]$, $[-1, 0]$, $[0, 1]$, $[1, \frac{4}{3}]$ and $[\frac{4}{3}, \infty)$. Among them, only the first and the fourth are constant (both have output value 0) and they are disconnected. It is easy to see that the values of the parameters are generic.

For the latter one, counterexamples already exist among networks with depth 3 and input dimension 2. In particular, we have

Example 3.6. Let $\mathcal{N}$ be a ReLU-activated network with input dimension 2, output dimension 1, and 3 hidden layers each of which has 2 neurons. The weights and biases are set as follows, and in particular, those for the output layer are the same as Example 3.5.

$$W_1 = \begin{bmatrix} 1 & 0 \\ -2 & 1 \end{bmatrix},\ B_1 = (0, 0),\ W_2 = W_3 = \begin{bmatrix} 2 & -1 \\ 5 & -2 \end{bmatrix},\ B_2 = B_3 = (-6, 2)$$

The top view of the linear regions of $\mathcal{N}$ (from above the xy-plane), as well as a link to the 3D model, are shown in Figure 1. In particular, there are 2 disconnected linear regions with zero value. The parameters of this network are generic: a small perturbation may cause small changes in the shapes and values of the nonzero linear regions, but the constancy/non-constancy and their relative positions remain the same, and the two zero-valued linear regions remain disconnected.

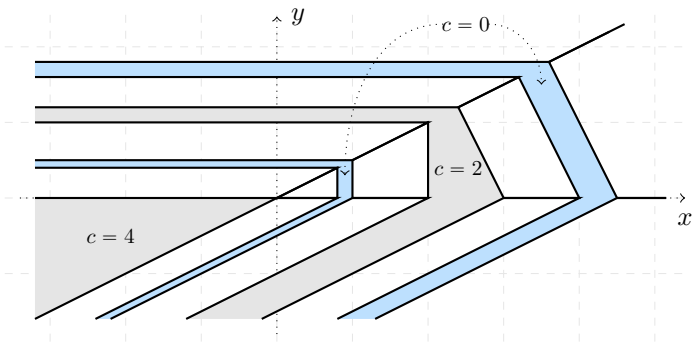


Figure 1: Linear regions of $\mathcal{N}$. The linear regions are divided precisely by the solid lines. The linear regions filled with colors are the ones with constant values (denoted as c in the image), and the others are non-constant. The 3D model can be accessed through this link:
https://html-preview.github.io/?url=https://github.com/yuanwang-research/Regions-ReLU-Network/blob/main/media/disconnected_zero_regions.html

3.2 Estimation and comparison with respect to network structures

With the help of Theorem 3.3, we show the following result that estimates and compares the numbers of linear regions for various network structures.

Theorem 3.7. *Assume Notation-Condition 3.2. Define functions G and T as*

$$G(m, n) \coloneqq \frac{1}{2^m} \sum_{i=0}^{m} \binom{m+n}{i};\ \ T(m, n) \coloneqq \frac{1}{2^m} \sum_{i=0}^{m} \binom{m}{i} G(i, n)$$

Then we have

1. *Let $n_1, n_2, ..., n_l$ be positive integers such that $M \geqslant n_1 \geqslant n_2 \geqslant ... \geqslant n_l$.*

   (a) *Assume that $\mathcal{N}$ is activated by ReLU function. Then $E(\mathcal{N}_{(n_1,n_2,...,n_l)}) \leqslant 2^{n_1} \prod_{i=1}^{l-1} T(n_i, n_{i+1})$.*

   (b) *Assume that $\mathcal{N}$ is activated by leaky ReLU function. Then $E(\mathcal{N}_{(n_1,n_2,...,n_l)}) \leqslant 2^{n_1} \prod_{i=1}^{l-1} G(n_i, n_{i+1})$.*

2. *Let n_1, n_2 be positive integers such that $M \geqslant n_1$.*

   (a) *Assume that $\mathcal{N}$ is activated by ReLU function. Then $E(\mathcal{N}_{(n_1,n_2)}) = 2^{n_1}(1 - \frac{1}{2^{n_2}})T(n_1, n_2) + \frac{1}{2^{n_2}} G(n_1, n_2)$.*

   (b) *Assume that $\mathcal{N}$ is activated by leaky ReLU function. Then with probability 1, the number of linear regions of $\mathcal{N}_{(n_1,n_2)}$ is $2^{n_1} G(n_1, n_2)$.*

3. *$E(\mathcal{N}_{pq\times 2}) > E(\mathcal{N}_{p\times 2q})$ for $p \geqslant 1$, $q \geqslant 2$ and $M \geqslant pq$. In particular, if $\mathcal{N}$ is activated by ReLU function, then $E(\mathcal{N}_{pq\times 2}) > \frac{2^{pq}-1}{2^{pq-q+1}} E(\mathcal{N}_{p\times 2q})$*

The proof of Theorem 3.7 can be found in the appendix (Section A.2). A sketch is as follows. The logical relationship among the three parts is similar for both the ReLU and leaky ReLU cases, but the technical treatments differ. We first prove Part 1, which gives an upper bound for $E(\mathcal{N}_{p\times 2q})$ (the right-hand side of the inequality in Part 3). We then prove Part 2, which gives the precise value of $E(\mathcal{N}_{pq\times 2})$ (the left-hand side in Part 3). Finally, we compare $E(\mathcal{N}_{pq\times 2})$ with the upper bound of $E(\mathcal{N}_{p\times 2q})$, which yields Part 3. More specifically, for Part 1, we use an inductive approach where we consider a series of states that starts with the network whose activation functions for all neurons are set to identity, and then gradually restores the activations from the l-th to the first hidden layer. When the neurons of a layer are activated, all linear regions correspond to intersections between the linear regions formed in (each orthant of) the next layer (previously activated) and the strata of ${}^+\mathcal{Q}_i$ for the ReLU case (resp. the orthants themselves for the leaky ReLU case). Then, by symmetry of the orthants and convexity of activation regions, we can inductively derive the upper bound for the number of linear regions in Part 1. This approach also contributes to the proof of Part 2, and since only two layers are involved there, no induction is needed and we obtain the exact count. Still, for the ReLU case we need to remove duplicated counts of zero linear regions; for the leaky ReLU case we use a deformation that turns the regions into those divided by the hyperplanes $\{H_i^j\}$. Part 3 mainly consists of arithmetic calculations comparing the values obtained from Parts 1 and 2, and the ReLU and leaky ReLU cases are treated separately.

The condition that the numbers of neurons in layers are non-increasing in Part 1 as well as the one that $M \geqslant n_1$ in Part 2 are used in the proof to ensure that coordinates and orthants are properly built in the layers considered. More specifically, for a certain layer (say the i-th layer) in the network, as long as the number of neurons in the $(i+1)$-th layer is no more than that of the current layer, the hyperplanes $\{H_{i+1}^j\}$ generically form a coordinate inside the coordinate of the i-th layer, and when the parameters are continuous independent random variables with even density functions, the distribution of the orthants are symmetric with respect to the strata of ${}^+\mathcal{Q}_i$. Our proof relies on such symmetry which does not hold without the condition.

For 2 hidden layers, the bounds in Part 1 are attained for leaky ReLU and nearly attained for ReLU (with relative gap less than $\frac{1}{2^{n_2}}$ by Part (2a)), but we expect them to be looser for depth at least 3.

One might think that the total number of parameters reflects the magnitude of the number of linear regions in a network. While a quite rough correlation like this seems to exist, this is not always the case, and a simple counterexample can be constructed using Part 2, as follows.

Example 3.8. Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be two networks both with input dimension 10, output dimension 1, and 2 hidden layers. $\mathcal{N}_1$ has hidden dimensions $n_1 = 10, n_2 = 10$, so the total number of parameters is $10 \cdot 10 + 10 + 10 \cdot 10 + 10 + 10 \cdot 1 + 1 = 231$, and the expected number of linear regions by Part 2 is 176694.49 for ReLU and 616666 for leaky ReLU cases. On the other hand, $\mathcal{N}_2$ has hidden

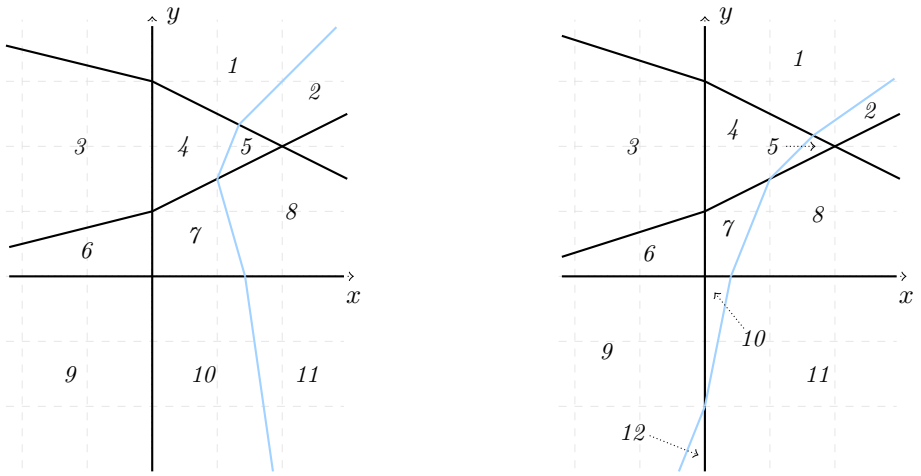
dimensions $n_1 = 5, n_2 = 30$, so it carries 266 parameters and is expected to have 37480.00 (resp. 384168) linear regions for ReLU (resp. leaky ReLU) cases.

This phenomenon, where one network has a less than or equal number of parameters but more linear regions than another, is also observed later in the experiments in Section 4.

While Part (2b) depicts the unusual fact that the number of linear regions is (generically) constant for leaky-ReLU-activated networks of depth 2, this no longer holds for depth greater than 2. We give a simple counterexample with depth 3 and input dimension 2 as follows.

Example 3.9. Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be two leaky-ReLU-activated networks (with negative slope 0.5) with input dimension 2, output dimension 1 and 3 hidden layers of sizes $(2, 2, 1)$. Their weights and biases are as follows. In particular, they have the same parameters except for $w_3^{2,1}$ which is 0.5 for $\mathcal{N}_1$ and 1 for $\mathcal{N}_2$. The linear regions for $\mathcal{N}_1$ and $\mathcal{N}_2$ are shown in Figure 2.

$$W_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad B_1 = (0, 0), \quad W_2 = \begin{bmatrix} -1 & -3 \\ -2 & 6 \end{bmatrix}, \quad B_2 = (6, -6),$$
$$W_3 = \begin{bmatrix} 1 & (0.5 \text{ or } 1) \end{bmatrix}^\top, \quad B_3 = (-2), \quad W_4 = [1], \quad B_4 = (0)$$



(a) $w_3^{2,1} = 0.5$ (b) $w_3^{2,1} = 1$

Figure 2: Different linear regions resulting from the change of $w_3^{2,1}$. The blue line on each side represents the partition induced by the 3rd hidden layer. The labels do not cover all linear regions; instead, they cover those affected by the parameter change. Thus, all unlabeled regions are the same for both $\mathcal{N}_1$ and $\mathcal{N}_2$ (the two networks contain a total of 15 and 16 regions, respectively). The 3D models are as follows.
Left: https://html-preview.github.io/?url=https://github.com/yuanwang-research/Regions-ReLU-Network/blob/main/media/leaky_15_regions.html
Right: https://html-preview.github.io/?url=https://github.com/yuanwang-research/Regions-ReLU-Network/blob/main/media/leaky_16_regions.html

3.3 Monte-Carlo approach for comparing linear regions

By Theorem 3.3 Part 1, generically speaking, two inputs induce the same non-constant linear pattern through a network if and only if they belong to the same linear region. Therefore, comparing the numbers of distinct non-constant linear patterns induced from a random sample reflects the comparison of the numbers of linear regions from a probabilistic perspective. We call these distinct non-constant linear patterns the *detected linear patterns*. Generically, their number equals the number of non-constant linear regions containing at least one sample, which is a lower bound for the number of non-constant linear regions. More precisely, for M samples drawn from an input distribution μ, the expected number of detected linear patterns is $\sum_R \big(1 - (1 - \mu(R))^M\big)$, where R ranges over the non-constant linear regions. Hence the method weights regions by their probability mass under μ rather than counting them equally: regions of small mass contribute little, and the count saturates at

M once almost every sample falls into a distinct region. When μ represents the inputs of interest, the comparison thus reflects the effective region complexity on those inputs, with the sample size and the scale of μ (e.g. σ) controlling its resolution (see Subsection A.3). Moreover, the computational cost for each input is linear with respect to the parameters of the network, which is theoretically minimal. These advantages make this a feasible approach for bypassing the NP-hardness barrier of this problem.

For the ReLU case, constant linear regions are also involved (they are absent in the leaky ReLU case by Remark 3.4). Still, intuitively, the nontriviality of the network is mainly caused by non-constant linear regions, and this is also supported empirically by the experiments in Section 4, where we count the numbers of constant regions for each batch of input data (Table 2). Therefore, comparing the numbers of non-constant regions seems to reflect the relationship among the total numbers of linear regions.

4 Experiments

In this section, we use the methodology described in Subsection 3.3 to compare linear regions for networks of different structures. More specifically, we create networks with weights and biases randomly generated according to the default distribution in PyTorch. For each network, we feed input data drawn from the normal distribution $N(0, \sigma^2)$ for various choices of σ, and obtain the numbers of detected linear patterns induced by these data.

We first compare these numbers of linear patterns between a pair of network structures: width $= n$, depth $= n - 1$ (wider), and width $= n - 1$, depth $= n$ (deeper), for $n = 20, 40, 60, 80$, and 100. We choose the standard deviations $\sigma = 0.01, 0.1, 0.5$, and 1, and send 10000 data points for each choice of σ. We then take the average number of detected linear patterns across 1000 networks. The average numbers are shown in Table 1. From the experimental results, we can see that although the difference between width and depth is only 1, as long as the input samples are balanced with the actual number of linear regions (there are multiple, yet not too many, samples belonging to the same linear region), there are clear gaps within each pair of networks, and the wider network always yields more detected linear patterns. Entries equal to 10000 mean that every sampled input yields a distinct detected linear pattern; such saturated entries are inconclusive at the current sampling budget, whereas unsaturated entries remain informative. Besides the detected linear patterns, we also present the average number of input data points having constant linear patterns in Table 2. Note that constant linear regions are detected only for the case $n = 20$ (for other choices of n, the corresponding numbers are all 0), and the proportion is clearly very small.

Theorem 3.3 Part 1 imposes no restriction on network structure, and so is the theoretical support it provides. Here we also run experiments for networks with 3 hidden layers, with input dimensions 90/100/110 and hidden sizes (120, 100, 80), (110, 100, 90), (100, 100, 100), (90, 100, 110), and (80, 100, 120), using 10000 input data points drawn from $N(0, 0.01^2)$ across 1000 networks as in the previous example. The average values are shown in Table 3. Note that for the last 2 cases (the rightmost 2 columns), the total numbers of parameters are nearly equal (30291 vs. 30311), and yet there is a clear difference between the average numbers of detected linear patterns. The overall trend across all cases here is that networks with more neurons in layers closer to (rather than farther from) the input layer yield more detected linear patterns.

To calibrate the method, we also apply it to small networks with known numbers of linear regions, $\mathcal{N}_{4\times1}$ and $\mathcal{N}_{2\times2}$ with input dimension 4 (Subsection A.3). The detected numbers never exceed the exact ones, and with a properly chosen σ, increasing the sample size drives them toward the exact values. For example, for the leaky ReLU $\mathcal{N}_{2\times2}$ with $\sigma = 512$, the average rises from 9.25 (with 10^4 samples) to 10.76 (with 7×10^7 samples) out of 11.

Although this paper mainly discusses the case where network parameters are randomly generated, we also ran experiments for networks during training. Since training breaks the independence and symmetry assumptions of our theory, these experiments are exploratory; in the appendix (Subsection A.4), we present an experiment in which we estimate the numbers of linear regions during training on MNIST and compare different model structures at selected epochs. All experiments in this paper were carried out using an A100 GPU (80GB) and were finished within one day.

	w: width, l: depth	n=20	n=40	n=60	n=80	n=100
ReLU $\sigma = 0.01$	$w = n, l = n - 1$	9.088	86.736	507.301	1783.418	4378.618
	$w = n - 1, l = n$	8.469	78.719	462.739	1705.87	4175.408
ReLU $\sigma = 0.1$	$w = n, l = n - 1$	2251.212	9873.949	9999.987	10000	10000
	$w = n - 1, l = n$	1881.251	9845.077	9999.99	10000	10000
ReLU $\sigma = 0.5$	$w = n, l = n - 1$	9961.832	10000.0	10000.0	10000	10000
	$w = n - 1, l = n$	9928.469	10000.0	10000.0	10000	10000
ReLU $\sigma = 1$	$w = n, l = n - 1$	9997.062	10000	10000	10000	10000
	$w = n - 1, l = n$	9993.463	10000	10000	10000	10000
Leaky ReLU $\sigma = 0.01$	$w = n, l = n - 1$	9.148	85.969	508.847	1793.196	4386.606
	$w = n - 1, l = n$	8.562	79.584	460.548	1704.272	4193.33
Leaky ReLU $\sigma = 0.1$	$w = n, l = n - 1$	2250.96	9874.343	9999.988	10000	10000
	$w = n - 1, l = n$	1881.343	9845.735	9999.992	10000	10000
Leaky ReLU $\sigma = 0.5$	$w = n, l = n - 1$	9961.623	10000	10000	10000	10000
	$w = n - 1, l = n$	9928.613	10000	10000	10000	10000
Leaky ReLU $\sigma = 1$	$w = n, l = n - 1$	9997.016	10000	10000	10000	10000
	$w = n - 1, l = n$	9993.527	10000	10000	10000	10000

Table 1: Average numbers of detected linear patterns from 10000 input data across 1000 networks

	$\sigma = 1$	$\sigma = 0.5$	$\sigma = 0.1$	$\sigma = 0.05$	$\sigma = 0.01$
$w = 20,\ l = 19$	36	24	3	0	0
$w = 19,\ l = 20$	40	16	1	0	0

Table 2: Average number of input data points with constant linear patterns for the ReLU case (out of 10000)

Input dim	100	100	100	100	100	90	110
1st layer width	120	110	100	90	80	110	90
2nd layer width	100	100	100	100	100	100	100
3rd layer width	80	90	100	110	120	90	110
ReLU	4849.78	4238.84	3814.85	3096.16	2515.47	3836.11	3325.80
Leaky ReLU	4857.19	4230.54	3809.89	3086.98	2507.46	3837.56	3330.12

Table 3: Average numbers of detected linear patterns (various input dimensions and common differences)

Acknowledgments and Disclosure of Funding

The author thanks the anonymous reviewers and the area chair for their constructive comments, which helped improve this paper.

References

[Ber23] Arturs Berzins. Polyhedral complex extraction from ReLU networks using edge subdivision. *International Conference on Machine Learning (ICML)*, 202:2234–2244, 23–29 Jul 2023.

[GEU24] Alexis Goujon, Arian Etemadi, and Michael Unser. On the number of regions of piecewise linear neural networks. *Journal of Computational and Applied Mathematics*, 441:115667, 2024.

[HR19] Boris Hanin and David Rolnick. Deep relu networks has surprisingly few activation patterns. *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[LCPK23] Y. Liu, C. Cole, C. Peterson, and M. Kirby. Relu neural networks, polyhedral decompositions, and persistent homology. *Proceedings of the 2 nd Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML) at the 40 th International Conference on Machine Learning*, 2023.

[LS17] Shiyu Liang and R. Srikant. Why deep neural networks for function approximation? *International Conference on Learning Representations (ICLR)*, 2017.

[MPCB14] Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural network. *Conference on Neural Information Processing Systems (NeurIPS)*, 2014.

[RPK+17] Maithra Raghu, Ben Poole, Jon Klenberg, Surya Ganguli, and Jascha Sohl Dickstein. On the expressive power of deep neural networks. *International Conference on Machine Learning (ICML)*, 2017.

[SHR25] M. Stargalla, C. Hertrich, and D. Reichman. The computational complexity of counting linear regions in relu neural networks. *Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

[VAS+24] Mattia Jacopo Villani, Emanuele Albini, Shubham Sharma, Saumitra Mishra, Salim Ibrahim Amoukou, Daniele Magazzeni, and Manuela Veloso. Pice: Polyhedral complex informed counterfactual explanations. *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 7(1):1502–1513, Oct. 2024.

[Wan22] Y. Wang. Estimation and comparison of linear regions for relu networks. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.

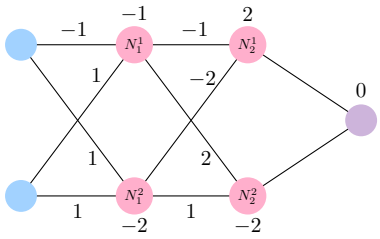
[XHY+20] Huan Xiong, Lei Huang, Mengyang Yu, Li Liu, Fan Zhu, and Ling Shao. On the number of linear regions of convolutional neural networks. *International Conference on Machine Learning (ICML)*, 2020.

[Yar17] Dmitry Yarotsky. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94:103–114, 2017.

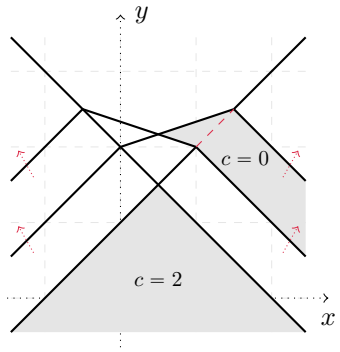
A Appendix

A.1 An illustration of related concepts

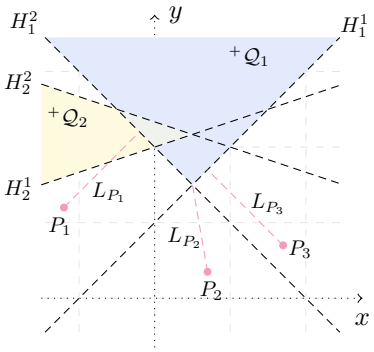
Here we provide an illustration for the notations and concepts set up in Section 2 using a simple network. The 2D figures of various aspects of the network as well as a link to the 3D model of its linear regions are shown as Figure 3.



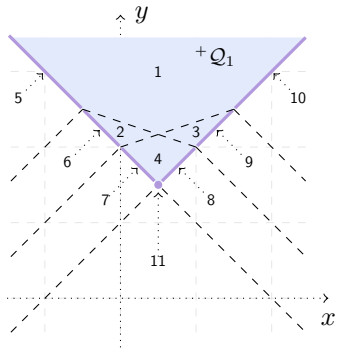
(a) The model structure with the values for the corresponding weights (beside the edges) and biases (beside the nodes)



(b) The linear regions for the ReLU case are the regions bounded by the solid black lines, and there are 2 constant ones filled with grey color. For the leaky-ReLU case (with non-zero negative slope), the four region boundaries on the sides would move up (see the red arrows), and the constant linear region on the upper right side would be divided into two by the red dashed line and become non-constant.



(c) Hyperplanes, orthants and strata for the two hidden layers. For both layers, the 2-dim stratum is $^+\mathcal{Q}_i$, the 1-dim strata are $H_1^i \cap {}^+\mathcal{Q}_i$ and $H_2^i \cap {}^+\mathcal{Q}_i$ and the 0-dim stratum is $H_1^i \cap H_2^i$, $i = 1, 2$. P_j $(j = 1, 2, 3)$ are random points in the orthants for the first layer, and L_{P_j}'s are the corresponding projection lines towards $^+\mathcal{Q}_1$ along the coordinates.



(d) The representatives of activation regions in the strata of $^+\mathcal{Q}_1$ (labeled with number 1-11) for the ReLU case. Activation regions 5-10 are represented by the 1-dim line segments they project to, and region 11 is represented by the point (0-dim) on the top of it (pointed to by the dashed arrows). They are formed by the intersection between the strata of $^+\mathcal{Q}_1$ and the orthants of $^+\mathcal{Q}_2$.

Figure 3: A 2D demonstration of the network in different aspects. Here is the link to the 3D model of the linear regions:
ReLU case: https://html-preview.github.io/?url=https://github.com/yuanwang-research/Regions-ReLU-Network/blob/main/media/basic_illustration_relu.html
Leaky ReLU case (with negative slope 0.2): https://html-preview.github.io/?url=https://github.com/yuanwang-research/Regions-ReLU-Network/blob/main/media/basic_illustration_leaky.html

A.2 Proofs of the theoretical results

Here we provide the proofs of the theorems in Section 3, namely Theorem 3.3 (the connectivity theorem) and Theorem 3.7 (the main comparison theorem), together with some supporting lemmas.

Proof of Theorem 3.3. First, in this proof we say that a neuron is *activated* (resp. *unactivated*) at an input point if the output through that neuron at that point is positive (resp. less than or equal to 0). We also define the negative half-space ${}^{-}H_i^j = \{x \in \mathbb{R}^M | \tilde{F}_i^j(x) \leqslant 0\}$, and the negative quadrant for the first layer ${}^{-}\mathcal{Q}_1 = \bigcap_{j=1}^{n_1} {}^{-}H_1^j$, which is intuitively the quadrant opposite to ${}^{+}\mathcal{Q}_1$. We prove the two parts as follows.

Part 1: Let α be the negative slope of the activation function, with $\alpha = 0$ in the ReLU case and $\alpha \in (0,1)$ in the leaky ReLU case. Fix an activation pattern A, and for each neuron N_t^j, let $s_A(N_t^j) = 1$ if N_t^j is activated under A and $s_A(N_t^j) = \alpha$ otherwise. On the activation region with pattern A, the output of N_t^j is $s_A(N_t^j)(\sum_k w_t^{k,j} x_{t-1}^k + b_t^j)$, where x_{t-1}^k is the output of N_{t-1}^k (or the k-th input coordinate if $t = 1$). So a straightforward induction shows that the restriction of F to this region is the affine map $F_A(x) = \sum_{i=1}^M c_{A,i} x_i + d_A$, where

$$c_{A,i} = \sum_{i_1,\ldots,i_l} \Big(\prod_{r=1}^{l} s_A(N_r^{i_r}) \Big) w_1^{i,i_1} w_2^{i_1,i_2} \cdots w_l^{i_{l-1},i_l} w_{l+1}^{i_l,1},$$

$$d_A = b_{l+1}^1 + \sum_{t=1}^{l} \sum_{i_t,\ldots,i_l} \Big(\prod_{r=t}^{l} s_A(N_r^{i_r}) \Big) b_t^{i_t} w_{t+1}^{i_t,i_{t+1}} \cdots w_{l+1}^{i_l,1},$$

and the sums run over all sequences of neuron indices, i.e., over all paths from an input coordinate (resp. a neuron) to the output. In particular, $c_{A,i}$ and d_A are polynomials in the weights and biases. Each monomial above determines the path it comes from, as it contains every weight (and possibly the bias) along that path. So distinct paths give distinct monomials, which cannot merge or cancel, and $c_{A,i}$ (resp. $d_A - b_{l+1}^1$) is the zero polynomial if and only if all the coefficients $\prod_r s_A(N_r^{i_r})$ in it vanish. In the ReLU case, such a coefficient is 1 if all neurons along the path are activated under A (we then call the path *active* under A) and 0 otherwise. We call A *non-constant* if $(c_{A,1}, \ldots, c_{A,M})$ is not the zero polynomial. In the ReLU case, this means that there is an active path from the input to the output, while in the leaky ReLU case, every A is non-constant.

Now let $A_1 \neq A_2$ be two activation patterns, at least one of which is non-constant. We show that $c_{A_1,i} - c_{A_2,i}$ for some i, or $d_{A_1} - d_{A_2}$, is not the zero polynomial.

- *The ReLU case:* If only one of A_1 and A_2 is non-constant, this is clear by definition. Otherwise, without loss of generality, there is a neuron N_p^q that is activated under A_1 but not under A_2. Let $(i_1, \ldots, i_l)$ be an active path under A_1. Since the network is fully connected, replacing i_p by q yields another path, which is still active under A_1; in other words, every activated neuron of a non-constant activation pattern lies on an active path. Then for every i, the monomial $w_1^{i,i_1} \cdots w_p^{i_{p-1},q} w_{p+1}^{q,i_{p+1}} \cdots w_{l+1}^{i_l,1}$ (with $i_0 = i$) appears in $c_{A_1,i}$ but not in $c_{A_2,i}$, since N_p^q is not activated under A_2.
- *The leaky ReLU case:* Let t be the largest index of the hidden layers in which A_1 and A_2 differ, let $N_t^{i_t}$ be a neuron with different activation states under A_1 and A_2, and choose arbitrary indices $i_{t+1}, \ldots, i_l$. By the choice of t, each $N_r^{i_r}$ with $r > t$ has the same activation state under A_1 and A_2. Hence the coefficients of the monomial $b_t^{i_t} w_{t+1}^{i_t,i_{t+1}} \cdots w_{l+1}^{i_l,1}$ in d_{A_1} and d_{A_2} are α^k and α^{k+1} (in some order) for some integer $k \geqslant 0$, which are different since $\alpha \in (0,1)$.

In both cases, we obtain a nonzero polynomial whose zero set contains all parameters for which $F_{A_1} = F_{A_2}$, and the zero set of a nonzero polynomial has measure zero. Let S be the union of these zero sets over all pairs (A_1, A_2) as above, together with the zero sets of the nonzero polynomials among $\{c_{A,i}\}$. As a finite union of sets of measure zero, S has measure zero. For parameters outside S, an activation pattern induces a non-constant linear pattern if and only if it is non-constant, and two distinct activation patterns, at least one of which is non-constant, induce different linear patterns.

Consequently, two generic points having the same non-constant linear pattern must have the same activation pattern.

Therefore, the union of the activation regions sharing the same non-constant linear pattern, which contains the linear region(s) with that linear pattern, is the same as the single activation region with the corresponding activation pattern (as derived above). So we are done.

Part 2: By *Part 1*, we only need to consider constant regions. When $l = 1$, the only constant region is ${}^{-}\mathcal{Q}_1$, which is convex (hence connected). So for the rest of the proof, we assume $l = 2$.

By Remark 3.4, we only need to consider the ReLU case. In this case, since $M \geqslant n_1$ and the weights are full-rank, we have a valid ${}^{+}\mathcal{Q}_1$. Then by Lemma 2.1, every linear region that is not inside ${}^{+}\mathcal{Q}_1$ is connected to ${}^{+}\mathcal{Q}_1$ (via the line L_P for any point P). So we only need to consider the intersection of ${}^{+}\mathcal{Q}_1$ with that linear region. For any x in ${}^{+}\mathcal{Q}_1$ belonging to a constant linear region, either all the neurons in layer 1 are unactivated, or all in layer 2 are unactivated. In the former case, given that x is in ${}^{+}\mathcal{Q}_1$, x can only lie in the intersection of all the H_1^i, which we denote by O^1 (the origin of the coordinate for layer 1, which is an affine subspace of dimension $M - n_1$). In the latter case, by definition, x belongs to the region $(\bigcap_i {}^{-}H_2^i) \cap {}^{+}\mathcal{Q}_1$, which we denote by ${}^{-}\mathcal{Q}$ for simplicity [2]. Then it suffices to show that either ${}^{-}\mathcal{Q}$ and O^1 have non-empty intersection (in this case, O^1 would actually be inside ${}^{-}\mathcal{Q}$, since F_1 vanishes on O^1, so the activation states of layer 2 are constant there) or F has different (constant) values on them. If ${}^{-}\mathcal{Q} = \varnothing$ or $O^1 \subset {}^{-}\mathcal{Q}$, then we are done. Otherwise, at least one neuron of the second layer is activated on O^1 (say N_2^j), so the value of F on O^1 changes with the biases $\{b_2^j\}$ for the neurons in the second layer through the (generic) weights $\{w_3^{j,1}\}$, and hence is different from that on ${}^{-}\mathcal{Q}$ (which is b_3^1, as all neurons in layer 2 are unactivated there) for generic values of those biases. So we are done. □

Lemma A.1. *Consider an n-dimensional Euclidean space with coordinates $(x_1, x_2, ..., x_n)$. Fix an orthant $\mathcal{Q} = \{(x_1, ..., x_n) | \, \mu_i x_i \geqslant 0 \ \forall i = 1, 2, ..., n\}$, where $\mu_i \in \{-1, 1\}$. Let m be a positive integer, and let w_{ij} and b_k $(i, k = 1, 2, ..., m, \ j = 1, 2, ..., n)$ be independent continuous random variables with even density functions. Then the expected number of regions into which the hyperplanes $H_i = \{\sum_{j=1}^{n} w_{ij} x_j + b_i = 0\}$ divide $\mathcal{Q}$ is*

$$\frac{1}{2^n} \sum_{i=0}^{n} \binom{n+m}{i}.$$

Proof. By symmetry, the expected number of linear regions cut by the hyperplanes is the same for every orthant. Moreover, the sum of the expectations over all orthants (there are 2^n orthants in total) is the expected total number of regions formed by $\{H_i\}$ together with the coordinate hyperplanes $H'_j = \{(x_1, x_2, ..., x_n) | \, x_j = 0\}$. On the other hand, it is easy to see that the H_i's and H'_j's are in general position with probability 1, and in this case they divide the whole space into $\sum_{i=0}^{n} \binom{n+m}{i}$ regions[3]. Therefore, the expectation for each orthant is $\frac{1}{2^n} \sum_{i=0}^{n} \binom{n+m}{i}$. □

Proof of Theorem 3.7. Throughout the proof, we denote the negative slope of the leaky ReLU function by α ($\alpha \in (0, 1)$). We prove all parts as follows.

Part 1: We actually show that the expected number of activation regions (hence also that of linear regions) is bounded by the right-hand side.

Consider the following process for $\mathcal{N}$. First, we fix all parameters $\{w_i^{j,k}\}$ and $\{b_i^j\}$ except those with $i = l + 1$, where $\{w_{l+1}^{j,k}\}$ and $\{b_{l+1}^j\}$ remain random variables with even density functions. For these fixed parameters, we ensure that the weight matrices $\{W_i\}$ for $i = 1, 2, ...l$ are full rank (this is reasonable because we will eventually restore the parameters as random variables, so for now it is fine to choose generic values). Moreover, we disable the activation of all neurons $\{N_i^j\}$ by setting each activation function to the identity $f(x) = x$ (instead of ReLU or leaky ReLU). We call this state of the network ST_l. Then for t in $l, l-1, l-2, ..., 1$, each time we restore the original setting of the

[2] Note that we do not assume the number of neurons is non-increasing, so $\{{}^{1}H_i\}$ and $\{{}^{2}H_j\}$ do not necessarily form coordinates and $\bigcap_i {}^{2}_{-} H_i$ may not really be an orthant.

[3] This is also why we require a continuous distribution. For example, if the biases are all 0 (still an even distribution but not continuous), then the hyperplanes are in a special position and the number of regions is smaller than the generic one.

network for the t-th hidden layer by restoring $\{w_t^{j,k}\}$ and $\{b_t^j\}$ as continuous independent random variables with even density functions, and by re-enabling activation (to either ReLU or leaky ReLU accordingly). We call this single step OP_t, and call the state of the network after OP_t by ST_{t-1}. Note that in state ST_0, the network $\mathcal{N}$ is the same as in the original setting for the theorem. In other words, in state ST_t, the weights and biases of layers $t+1, \ldots, l+1$ are independent random variables with even density functions and the neurons in layers $t+1, \ldots, l$ use the original activation function, whereas the parameters of layers $1, \ldots, t$ are fixed generic values and the neurons in these layers use the identity function. In particular, each time Lemma A.1 is applied below, the random variables in loc. cit. are the weights and biases of a single layer, which are independent of each other and of the parameters of the other layers. The whole process is illustrated as follows:

$$\text{(linear function) } ST_l \xrightarrow{OP_l} ST_{l-1} \xrightarrow{OP_{l-1}} \ldots \xrightarrow{OP_2} ST_1 \xrightarrow{OP_1} ST_0 \text{ (original network)}$$

Now we start with the transition from ST_l to ST_{l-1} through OP_l. In ST_l, the whole network is just a linear function, as there is no activation at all. In particular, there is only one linear region, namely the whole input space $\mathbb{R}^M$. Then in ST_{l-1}, only layer l has activation, so we have 2^{n_l} activation regions, which are exactly the orthants for the l-th layer.

Next, we move on to state ST_{l-2}. There, the neurons in the $(l-1)$-th layer are activated. By the assumption that the numbers of neurons in layers are non-increasing, the coordinates for the $(l-1)$-th layer, as well as the positive orthant ${}^+\mathcal{Q}_{l-1}$, exist. Moreover, in ${}^+\mathcal{Q}_{l-1}$, the activation regions are just the orthants for the l-th layer (intersected with ${}^+\mathcal{Q}_{l-1}$), as all points there are activated in layer $l-1$. Here we handle the ReLU and leaky ReLU cases separately:

- *Part* (1a) (the ReLU case): Since layer $l-1$ is the first hidden layer that is activated, we can apply Lemma 2.1, with the ${}^+\mathcal{Q}_1$ in loc. cit. corresponding to ${}^+\mathcal{Q}_{l-1}$ here (because the neurons before the $(l-1)$-th layer are still unactivated). By Lemma 2.1, every activation region that is not in the interior of ${}^+\mathcal{Q}_{l-1}$ can be reduced to the intersection between the region and a stratum of ${}^+\mathcal{Q}_{l-1}$. Therefore, we only need to consider the intersection between the orthants for the l-th layer and the strata of ${}^+\mathcal{Q}_{l-1}$. For each d-dimensional stratum of the form $(\bigcap_{k=1,2,\ldots,n_{l-1}-d} H_{l-1}^{j_k}) \cap {}^+\mathcal{Q}_{l-1}$ for the $(l-1)$-th layer, we can extend it to the corresponding d-dimensional affine space $\bigcap_{k=1,2,\ldots,n_{l-1}-d} H_{l-1}^{j_k}$ (e.g., by removing the part intersected with ${}^+\mathcal{Q}_{l-1}$), and view the stratum as an orthant of that affine space. From this perspective, we can apply Lemma A.1, with the x_i in loc. cit. being the outputs of the i-th neuron of the $(l-1)$-th layer (i does not belong to $\{j_k\}$ that form the stratum), and w_{ij} and b_i in loc. cit. being $w_l^{i,j}$ and b_l^i, respectively (in particular, they are independent random variables). By Lemma A.1, the expectation of the number of orthants in the d-dimensional stratum is $\frac{1}{2^d}\sum_{i=0}^{d}\binom{n_l+d}{i}$, and by the reverse correspondence from Lemma 2.1, it is also the expectation of the number of activation regions in any orthant for the $(l-1)$-th layer that has distance $n_{l-1}-d$ from ${}^+\mathcal{Q}_{l-1}$. Note that there are $\binom{n_{l-1}}{d}$ such orthants for layer $l-1$ in total.
- *Part* (1b) (the leaky ReLU case): In this case, the expected number of activation regions is the sum of the expectations over all orthants for the $l-1$-th layer (this is also true in Part (1a), but there we handle them by projection to lower-dimensional strata). Within an orthant $\mathcal{Q}$ for the $(l-1)$-th layer, the positive and negative coordinates correspond to positive and negative outputs from the neurons, respectively, with the negative outputs multiplied by the negative slope α of the leaky ReLU function. This is equivalent to multiplying $w_l^{j,i}$ by α for indices j with negative outputs, and it does not change the fact that $w_l^{j,i}$ is an independent random variable with an even density function. In particular, Lemma A.1 still applies to calculate the activation regions within $\mathcal{Q}$, and in the leaky ReLU case all orthants are in the same situation, with expectation for each of them equal to $\frac{1}{2^{n_{l-1}}}\sum_{i=0}^{n_{l-1}}\binom{n_{l-1}+n_l}{i}$.

More generally, in ST_t, only layers from $t+1$ to l contribute to forming activation regions, as the part of the network before that is still linear. Again, since the numbers of neurons are non-increasing,

we generically have valid coordinates and quadrants for both the $(t+1)$-th and the $(t+2)$-th layers. We handle the ReLU and leaky ReLU cases separately:

- *Part* (1a) (the ReLU case): We define E_d^t as the expected number of activation regions in an orthant for the $(t+1)$-th layer at distance $n_{t+1}-d$ from ${}^+\mathcal{Q}_{t+1}$ (i.e. whose stratum is d-dimensional) in state ST_t. From the previous argument, we know that

$$E_d^{l-1}=1,\ E_d^{l-2}=\frac{1}{2^d}\sum_{i=0}^{d}\binom{n_l+d}{i} \tag{1}$$

In general, we can obtain an upper bound for E_d^t by induction. We denote this upper bound by $\hat{E}_d^t$ and set $\hat{E}_d^{l-1}=E_d^{l-1}$ and $\hat{E}_d^{l-2}=E_d^{l-2}$. We can still use Lemma A.1 to get that $\frac{1}{2^d}\sum_{i=0}^{d}\binom{n_{t+2}+d}{i}$ is the expected number of orthants of layer $t+2$ that a d-dimensional stratum of layer $t+1$, denoted by S, passes through within ${}^+\mathcal{Q}_{t+1}$. Then, inside an orthant of layer $t+2$, S passes through each activation region at most once (by convexity of those regions). By induction, we have upper bounds $\{\hat{E}_d^{t+1}\}$ for the number of activation regions in state ST_{t+1} when considering strata of layer $t+2$, and there are $\binom{n_{t+2}}{d}$ orthants at distance d from ${}^+\mathcal{Q}_{t+2}$. So the types of orthants, with respect to the distance of intersection with S, follow a binomial distribution $B(n_{t+2},\frac{1}{2})$ (using that random variables $w_{t+2}^{i,j}$ and b_{t+2}^i are independent). Therefore, we have

$$\hat{E}_d^t\leqslant\frac{1}{2^d}\left(\sum_{i=0}^{d}\binom{n_{t+2}+d}{i}\right)\left(\frac{1}{2^{n_{t+2}}}\sum_{d'=0}^{n_{t+2}}\binom{n_{t+2}}{d'}\hat{E}_{d'}^{t+1}\right) \tag{2}$$

for $d=0,1,...,n_{t+1}$.

Finally, upon reaching ST_0, we obtain an upper bound for the total number of activation regions from (2), namely

$$\begin{aligned}
&\sum_{i=0}^{n_1}\binom{n_1}{i}\hat{E}_i^0=2^{n_1}\left(\frac{1}{2^{n_1}}\sum_{i=0}^{n_1}\binom{n_1}{i}\hat{E}_i^0\right)\\
&\leqslant 2^{n_1}\left(\frac{1}{2^{n_1}}\sum_{i=0}^{n_1}\binom{n_1}{i}\frac{1}{2^i}(\sum_{j=0}^{i}\binom{n_2+i}{j})(\frac{1}{2^{n_2}}\sum_{k=0}^{n_2}\binom{n_2}{k}\hat{E}_k^1)\right)\\
&=2^{n_1}\left(\frac{1}{2^{n_1}}\sum_{i=0}^{n_1}\binom{n_1}{i}\frac{1}{2^i}(\sum_{j=0}^{i}\binom{n_2+i}{j})\right)(\frac{1}{2^{n_2}}\sum_{k=0}^{n_2}\binom{n_2}{k}\hat{E}_k^1)\\
&=2^{n_1}T(n_1,n_2)(\frac{1}{2^{n_2}}\sum_{k=0}^{n_2}\binom{n_2}{k}\hat{E}_k^1)\\
&\leqslant ...=2^{n_1}T(n_1,n_2)T(n_2,n_3)(\frac{1}{2^{n_3}}\sum_{k=0}^{n_3}\binom{n_3}{k}\hat{E}_k^2)\\
&\leqslant ...\leqslant 2^{n_1}\prod_{i=1}^{l-1}T(n_i,n_{i+1})\ \text{(the right side)}
\end{aligned}$$

- *Part* (1b) (the leaky ReLU case): We define $\tilde{E}^t$ as the expected number of activation regions in an (arbitrary) orthant for the $(t+1)$-th layer in state ST_t. From the previous argument, we know that

$$\tilde{E}^{l-1}=1,\ \tilde{E}^{l-2}=\frac{1}{2^{n_{l-1}}}\sum_{i=0}^{n_{l-1}}\binom{n_{l-1}+n_l}{i} \tag{3}$$

Then we can also get an upper bound for $\tilde{E}^t$ by induction. Compared to Part (1a), instead of considering strata for the $(t+1)$-th layer, we consider orthants for that layer themselves,

intersected with orthants for layer $t+2$. Still, we apply Lemma A.1 to obtain that the expected number of such intersections is $\frac{1}{2^{n_{t+1}}}\sum_{i=0}^{n_{t+1}}\binom{n_{t+1}+n_{t+2}}{i}$. Again, inside an orthant of layer $t+2$, an orthant of layer $t+1$ passes through each activation region at most once, so the expectation there is at most $\tilde{E}^{t+1}$. Moreover, since the numbers of neurons are non-increasing and, by induction, coordinates and orthants exist for every layer, we can use induction to deduce that the total expected number is

$$E(\mathcal{N}_{(n_1,n_2,\ldots,n_l)}) = 2^{n_1}\tilde{E}^0 \leqslant 2^{n_1}\cdot\frac{1}{2^{n_1}}\sum_{i=0}^{n_1}\binom{n_1+n_2}{i}\tilde{E}^1$$

$$= 2^{n_1}G(n_1,n_2)\tilde{E}^1 \leqslant 2^{n_1}G(n_1,n_2)G(n_2,n_3)\tilde{E}^2\ldots \leqslant 2^{n_1}\prod_{i=1}^{l-1}G(n_i,n_{i+1}).$$

This is exactly what we want to prove.

Part 2: We consider the ReLU and leaky ReLU cases separately as follows:

- *Part* (2a) (the ReLU case): First, by the argument up to (1) in the proof of Part (1a), we know that the total number of activation regions is

$$\sum_{k=0}^{n_1}\binom{n_1}{k}E_k^0 = \sum_{k=0}^{n_1}\binom{n_1}{k}(\frac{1}{2^k}\sum_{i=0}^{k}\binom{k+n_2}{i}) = 2^{n_1}T(n_1,n_2) \tag{4}$$

  By Theorem 3.3 Part 1, the non-constant linear regions are the same as the activation regions. Then by Theorem 3.3 Part 2 and its proof, the constant activation regions on which all neurons of layer 2 are unactivated (the zero regions) form a single linear region when nonempty. The only other possible constant region lies over $^-\mathcal{Q}_1$; it forms a linear region by itself and is kept in the count below. We discard the zero regions in all strata of layer 1 except the n_1-dimensional one. By Lemma 2.1, the n_1-dimensional zero locus must exist for the other zero loci to exist, so we retain the former. For the computation, we first subtract the proportion of zero linear regions from (4) and then add back the contribution from the n_1-dimensional stratum. By symmetry, the 2^{n_2} orthants of layer 2 behave identically with respect to their intersections with the strata of layer 1. Therefore, in the subtraction step, we subtract a proportion $\frac{1}{2^{n_2}}$ from the $\frac{1}{2^k}\sum_{i=0}^{k}\binom{n_2+k}{i}$ term (equivalently, from the $T(n_1,n_2)$ term). To add back the n_1-dimensional part, we use the term corresponding to $k=n_1$ in (4), multiplied by $\frac{1}{2^{n_2}}$:

$$\binom{n_1}{n_1}(\frac{1}{2^{n_2}}\cdot\frac{1}{2^{n_1}}\sum_{i=0}^{n_1}\binom{n_1+n_2}{i}) = \frac{1}{2^{n_2}}G(n_1,n_2).$$

  This gives the desired result.

- *Part* (2b) (the leaky ReLU case): Consider an orthant for the first layer, which we denote by $\mathcal{Q}$. $\mathcal{Q}$ is characterized by a subset $I^-\subset\{1,2,\ldots,n_1\}$ as $\mathcal{Q}=\{x\in\mathbb{R}^M|\tilde{F}_1^j(x)\leqslant 0 \text{ if } j\in I^-, \tilde{F}_1^j(x)\geqslant 0 \text{ otherwise}\}$. Then $F_1^j(x)=\alpha\tilde{F}_1^j(x)$ for $j\in I^-$ and $F_1^j(x)=\tilde{F}_1^j(x)$ otherwise. On the other hand, a linear region, which is generically the same as an activation region by Theorem 3.3 Part 1 and Remark 3.4, is characterized by the signs of a collection of linear combinations of $F_1^j(x)$ (determined by W_2 and b_2). When we change the negative slope α while keeping it positive, the linear regions are stretched along the coordinates corresponding to I^-, but their existence does not change. In particular, this is also the case when we change α to 1. Moreover, in that case, F_i^j is equal to $\tilde{F}_i^j$, and the linear regions are formed exactly by the n_1+n_2 hyperplanes $\{H_i^j\}$. They are the preimages under $\tilde{F}_1$ of n_1+n_2 hyperplanes in general position in $\mathbb{R}^{n_1}$, so they generically divide $\mathbb{R}^M$ into $2^{n_1}G(n_1,n_2)$ parts.

Part 3: We consider the ReLU and leaky ReLU cases separately as follows.

- *The ReLU case:* First, observe that

$$\begin{aligned} E(\mathcal{N}_{pq\times 2}) &\overset{\text{Part (2a)}}{=} (1-\frac{1}{2^{pq}})2^{pq}T(pq,pq)+\frac{1}{2^{n_1+n_2}}\sum_{i=0}^{n_1}\binom{n_1+n_2}{i} \\ &> (2^{pq}-1)T(pq,pq) \end{aligned} \tag{5}$$

Next we show that $T(p,p)\leqslant 2^{p-1}$. Observe that for any $i\leqslant p$, by basic properties of binomial coefficients, we have $\sum_{j=0}^{i}\binom{p+i}{j}\leqslant\frac{1}{2}\sum_{j=0}^{p+i}\binom{p+i}{j}$. Hence

$$\begin{aligned} T(p,p) &= \frac{1}{2^p}\sum_{i=0}^{p}\binom{p}{i}(\frac{1}{2^i}\sum_{j=0}^{i}\binom{i+p}{j})\leqslant\frac{1}{2^p}\sum_{i=0}^{p}\binom{p}{i}(\frac{1}{2^{i+1}}\sum_{j=0}^{i+p}\binom{i+p}{j}) \\ &= \frac{1}{2^p}\sum_{i=0}^{p}\binom{p}{i}(\frac{1}{2^{i+1}}2^{i+p})=2^{p-1} \end{aligned} \tag{6}$$

Next, we prove that

$$T(p,p)^q\leqslant T(pq,pq) \tag{7}$$

as follows:

$$\begin{aligned} T(p,p)^q &= \prod_{k=1}^{q}(\frac{1}{2^p}\sum_{i=0}^{p}\binom{p}{i}(\frac{1}{2^i}\sum_{j=0}^{i}\binom{p+i}{j}))=\frac{1}{2^{pq}}\prod_{k=1}^{q}(\sum_{i=0}^{p}\binom{p}{i}(\frac{1}{2^i}\sum_{j=0}^{i}\binom{p+i}{j})) \\ &= \frac{1}{2^{pq}}\sum_{i=0}^{pq}(\sum_{\substack{i_1,i_2,\ldots,i_q\in[0,p],\\ \sum_{k=1}^{q}i_k=i}}(\prod_{k=1}^{q}\binom{p}{i_k})(\prod_{k=1}^{q}(\frac{1}{2^{i_k}}\sum_{j=0}^{i_k}\binom{p+i_k}{j}))) \\ &= \frac{1}{2^{pq}}\sum_{i=0}^{pq}(\sum_{\substack{i_1,i_2,\ldots,i_q\in[0,p],\\ \sum_{k=1}^{q}i_k=i}}(\prod_{k=1}^{q}\binom{p}{i_k})(\frac{1}{2^i}\prod_{k=1}^{q}(\sum_{j=0}^{i_k}\binom{p+i_k}{j}))) \\ &= \frac{1}{2^{pq}}\sum_{i=0}^{pq}(\sum_{\substack{i_1,i_2,\ldots,i_q\in[0,p],\\ \sum_{k=1}^{q}i_k=i}}(\prod_{k=1}^{q}\binom{p}{i_k})(\frac{1}{2^i}\sum_{j=0}^{i}(\sum_{\substack{j_1,j_2,\ldots,j_q\\ 0\leqslant j_k\leqslant i_k\\ \sum_{k=1}^{q}j_k=j}}\prod_{k=1}^{q}\binom{p+i_k}{j_k}))) \\ &\leqslant \frac{1}{2^{pq}}\sum_{i=0}^{pq}(\sum_{\substack{i_1,i_2,\ldots,i_q\in[0,p],\\ \sum_{k=1}^{q}i_k=i}}\prod_{k=1}^{q}\binom{p}{i_k}(\frac{1}{2^i}\sum_{j=0}^{i}(\sum_{\substack{j_1,j_2,\ldots,j_q,\\ \sum_{k=1}^{q}j_k=j}}\prod_{k=1}^{q}\binom{p+i_k}{j_k}))) \\ &= \frac{1}{2^{pq}}\sum_{i=0}^{pq}(\sum_{\substack{i_1,i_2,\ldots,i_q\in[0,p],\\ \sum_{k=1}^{q}i_k=i}}\prod_{k=1}^{q}\binom{p}{i_k})(\frac{1}{2^i}\sum_{j=0}^{i}\binom{i+pq}{j}) \\ &= \frac{1}{2^{pq}}\sum_{i=0}^{pq}\binom{pq}{i}(\frac{1}{2^i}\sum_{j=0}^{i}\binom{i+pq}{j}) \\ &= T(pq,pq) \end{aligned}$$

where the only inequality in the middle is due to the removal of the restriction $0\leqslant j_k\leqslant i_k$. Finally,

$$\begin{aligned} E(\mathcal{N}_{p\times 2q}) &\overset{\text{Part (1a)}}{\leqslant} 2^pT(p,p)^{2q-1}\overset{(6)}{\leqslant}2^p\cdot(2^{p-1})^{q-1}\cdot T(p,p)^q\overset{(7)}{\leqslant}2^p\cdot(2^{p-1})^{q-1}T(pq,pq) \\ &= 2^{pq-q+1}T(pq,pq)=\frac{2^{pq-q+1}}{2^{pq}-1}(2^{pq}-1)T(pq,pq)\overset{(5)}{<}\frac{2^{pq-q+1}}{2^{pq}-1}E(\mathcal{N}_{pq\times 2}) \end{aligned}$$

- *The leaky ReLU case:* Obviously, we have $G(p,p)<2^p$. So

$$E(\mathcal{N}_{p\times 2q})\overset{\text{Part (1b)}}{\leqslant}2^pG(p,p)^{2q-1}<2^p\cdot(2^p)^{q-1}G(p,p)^q=2^{pq}(\frac{1}{2^p}\sum_{j=0}^{p}\binom{2p}{j})^q$$

$$= \prod_{i=1}^{q} (\sum_{j=0}^{p} \binom{2p}{j}) = \sum_{i=0}^{pq} (\sum_{\substack{i_1,\dots,i_q \in [0,p], \\ \sum_{k=1}^{q} i_k = i}} \prod_{k=1}^{q} \binom{2p}{i_k}) < \sum_{i=0}^{pq} \binom{2pq}{i} = 2^{pq} G(pq, pq) = E(\mathcal{N}_{pq \times 2})$$

□

A.3 Calibration of the Monte-Carlo method on small networks

Here we calibrate the Monte-Carlo method on two small networks with input dimension 4 whose numbers of linear regions are known. The first one, $\mathcal{N}_{4\times1}$, generically has $2^4 = 16$ linear regions, one of which is constant in the ReLU case. The second one, $\mathcal{N}_{2\times2}$, has 6.5 linear regions in expectation in the ReLU case, one or two of which are constant, and generically has exactly 11 linear regions in the leaky ReLU case, none of which is constant (Theorem 3.7, Part 2 and Remark 3.4). Recall that only non-constant linear patterns are counted as detected linear patterns. As in Section 4, the parameters are generated according to the default distribution in PyTorch, and the input data are drawn from $N(0, \sigma^2)$.

Table 4 shows the means and standard deviations of the numbers of detected linear patterns from 10000 input data across 1000 networks for various choices of σ. For $\mathcal{N}_{4\times1}$, the means approach the numbers of non-constant linear regions (15 for ReLU and 16 for leaky ReLU) once σ is large enough. For $\mathcal{N}_{2\times2}$, whose regions are more heterogeneous in probability mass, 10000 input data are not enough to detect all regions, but the results are reasonable when σ is properly chosen.

We further vary the sample size for the leaky ReLU networks with $\sigma = 512$. For $\mathcal{N}_{4\times1}$, all 16 regions are detected for 79.9%, 89.4%, and 97.8% of the networks with 10^4, 5×10^4, and 5×10^5 input data, respectively. For $\mathcal{N}_{2\times2}$, 7×10^7 input data yield an average of 10.76 detected linear patterns (out of 11), and all 11 regions are detected for 78.8% of the networks. In both cases, the number of detected linear patterns never exceeds the actual number of linear regions, as predicted by the theory. Regions with large probability mass under the input distribution are detected quickly, whereas regions with very small mass (e.g. tail regions) require many more samples. Hence the sample size and σ control the resolution of the comparison, which is consistent with the intended use of the method: comparing networks rather than counting all of their linear regions.

	ReLU		Leaky ReLU	
σ	$\mathcal{N}_{4\times1}$	$\mathcal{N}_{2\times2}$	$\mathcal{N}_{4\times1}$	$\mathcal{N}_{2\times2}$
1	13.650 (1.414)	4.224 (1.822)	14.550 (1.491)	6.587 (1.808)
2	14.321 (0.942)	4.630 (1.794)	15.265 (0.998)	7.154 (1.813)
4	14.586 (0.691)	4.876 (1.746)	15.557 (0.715)	7.512 (1.777)
8	14.669 (0.637)	4.948 (1.721)	15.650 (0.656)	7.704 (1.747)
16	14.681 (0.669)	4.965 (1.688)	15.658 (0.703)	7.894 (1.670)
32	14.683 (0.689)	4.918 (1.650)	15.657 (0.731)	8.215 (1.529)
64	14.686 (0.678)	4.788 (1.589)	15.660 (0.719)	8.645 (1.288)
128	14.689 (0.674)	4.653 (1.506)	15.662 (0.716)	9.093 (1.130)
256	14.687 (0.682)	4.561 (1.452)	15.661 (0.722)	9.321 (0.989)
512	14.685 (0.687)	4.430 (1.410)	15.659 (0.726)	9.248 (0.953)

Table 4: Means (standard deviations) of the numbers of detected linear patterns from 10000 input data across 1000 networks. The numbers of non-constant linear regions are 15 (ReLU) and 16 (leaky ReLU) for $\mathcal{N}_{4\times1}$, and 11 for the leaky ReLU $\mathcal{N}_{2\times2}$.

A.4 Comparison of networks during training processes

In this subsection, experiments are carried out for networks during training. We consider networks with structures such as 512×1, 256×2, 128×4, and 64×8, trained on the MNIST dataset. We then feed 10000 input data points and record the number of distinct linear patterns after each epoch.

The average numbers of linear patterns across all epochs are shown in Table 5. Table 6 shows the numbers of detected linear patterns together with the training and test accuracies at epochs 0, 20, 40, 60, 80, and 100. Figure 4 shows moving averages (window size 5) of the numbers of linear patterns detected during training. The tendency that wider networks have more linear patterns than deeper ones also holds in this setting.

$\sigma = 0.01$ w: width, l: depth	$w = 512$ $l = 1$	$w = 256$ $l = 2$	$w = 128$ $l = 4$	$w = 64$ $l = 8$
ReLU	7071.44	4524.75	2651.69	752.67
Leaky ReLU	7202.11	5701.66	2119.13	766.77

Table 5: Average number of linear patterns detected from the sample during training

scale 0.001	0	20	40	60	80	100
ReLU $w = 512, l = 1$	- 9.26% 872	98.67% 98.15% 9254	99.97% 98.43% 6645	99.98% 98.55% 7970	100.00% 98.58% 8197	100.00% 98.56% 4188
ReLU $w = 256, l = 2$	- 9.95% 128	99.50% 98.08% 2782	99.97% 98.57% 6207	99.86% 98.21% 1269	100.00% 98.57% 5880	100.0% 98.58% 7198
ReLU $w = 128, l = 4$	- 11.81% 10	99.24% 97.38% 3085	99.89% 97.99% 6111	99.40% 97.72% 6823	99.99% 97.52% 7589	100.00% 97.99% 8251
ReLU $w = 64, l = 8$	- 10.09% 2	98.98% 96.78% 935	99.85% 97.49% 524	99.83% 97.41% 666	100.00% 97.78% 702	100.00% 97.83% 386
Leaky ReLU $w = 512, l = 1$	- 9.19% 872	99.69% 98.11% 8808	99.93% 98.11% 7524	100.00% 98.53% 5470	100% 98.49% 7432	100% 98.53% 7115
Leaky ReLU $w = 256, l = 2$	- 9.93% 126	99.68% 98.03% 3057	99.95% 98.53% 8631	99.91% 98.26% 5836	99.98% 97.8% 1667	100% 98.55% 3754
Leaky ReLU $w = 128, l = 4$	- 11.79% 10	99.38% 97.88% 3425	99.85% 98.03% 4442	99.98% 98.41% 1898	100% 98.46% 1848	100% 98.53% 1310
Leaky ReLU $w = 64, l = 8$	- 10.09% 2	99.03% 97.51% 1315	99.85% 97.63% 503	99.88% 97.77% 682	99.97% 97.66% 2017	100.00% 97.93% 657

Table 6: Accuracies and numbers of linear patterns at certain steps

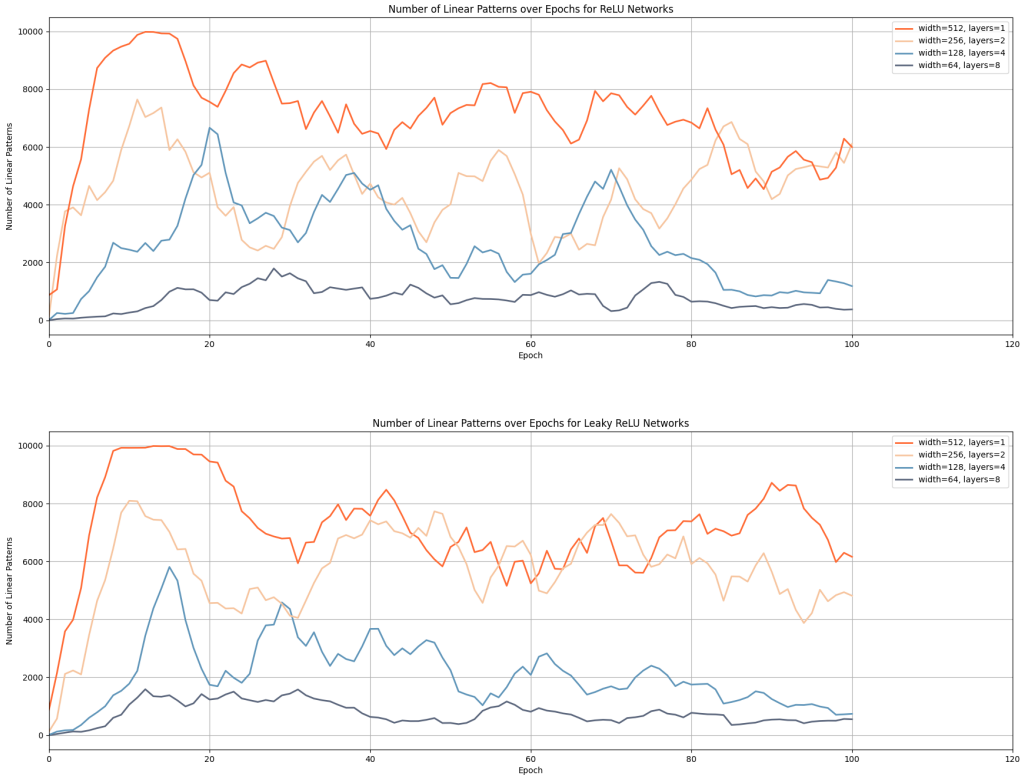


Figure 4: Moving averages of numbers of linear patterns for MNIST

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately present the primary claims of the paper

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of the work is discussed in Section 1.2

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: All of the detailed proof, if not present in the main body of the paper, are provided in the appendix, including theorems and the supporting lemmas.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have provided the complete code as well as the instructions to reproduce the experiments in the GitHub repository

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The data and code are provided with sufficient instructions on the paper as well as on the GitHub repo.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We have provided the code that is able to reproduce all the experiments, and all the details are already there.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Each case of the experiments are carried out in the same setting for sufficiently many randomized networks

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: This is mentioned at the end of Section 4

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We are aware of the Code of Ethics and conform with it in this paper.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All the code, data and models used in this paper are properly credited and respected.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The code we implemented for this paper is well documented.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.