Exact Count of Boundary Pieces of ReLU Classifiers: Towards the Proper Complexity Measure for Classification

Paweł Piwek[1] **Adam Klukowski**[2] **Tianyang Hu**[2]

[1]University of Oxford, pawel.piwek@maths.ox.ac.uk
[2]Huawei Noah's Ark Lab, hutianyang1@huawei.com

Abstract

Classic learning theory suggests that proper regularization is the key to good generalization and robustness. In classification, current training schemes only target the complexity of the classifier itself, which can be misleading and ineffective. Instead, we advocate directly measuring the complexity of the decision boundary. Existing literature is limited in this area with few well-established definitions of boundary complexity. As a proof of concept, we start by analyzing ReLU neural networks, whose boundary complexity can be conveniently characterized by the number of affine pieces. With the help of tropical geometry, we develop a novel method that can explicitly count the exact number of boundary pieces, and as a by-product, the exact number of total affine pieces. Numerical experiments are conducted and distinctive properties of our boundary complexity are uncovered. First, the boundary piece count appears largely independent of other measures, e.g., total piece count, and l_2 norm of weights, during the training process. Second, the boundary piece count is negatively correlated with robustness, where popular robust training techniques, e.g., adversarial training or random noise injection, are found to reduce the number of boundary pieces.

1 BACKGROUND

Despite deep learning's huge success in image classification, naturally trained deep classifiers are found to be adversarially vulnerable [Goodfellow et al., 2014, 2016]. By adding a small perturbation (adversarial attack) to an image, which is almost imperceptible to humans, the neural network's predicted class can be arbitrarily manipulated. The prevalence of adversarial examples for state-of-the-art deep classifiers, even on small datasets such as CIFAR [Krizhevsky, 2009], suggests overfitting, where decision boundaries of trained deep neural networks (DNNs) are *overly complicated* and within a small distance to almost all the training instances. Ideally, we want our model to generalize well on unseen data and be robust against small input perturbations, i.e., the prediction doesn't change much in case of small random noises. For regression, the requirement loosely translates to the smoothness of the predictor function. However, it becomes drastically different for classification, due to the discrete nature of class labels.

The goal of classification is to recover the Bayes optimal decision boundary with the lowest misclassification rate (0-1 loss). Decision boundary corresponds to certain level sets of the classifiers, which is more difficult to control than the classifier itself. As is often the case, especially in image classification, the classes can be thought of as separable with positive margins, i.e., the class labels have no randomness and images in different classes reside in non-overlapping regions with positive pairwise distances. In this case, there are infinitely many possible decision boundaries with zero misclassification error, but only some of them are robust with good generalization properties. Current training methods offer little control over the selection process and the resulting decision boundaries often turn out to be unsatisfactory. For natural data, it is commonly believed that an ideal decision boundary (e.g., human's), which offers both good accuracy and robustness, should not be too complicated. In practice, how to effectively find such decision boundaries can be a real challenge.

Let $\mathcal{F}$ denote some function space. In learning theory, the model complexity (how large is $\mathcal{F}$) is of critical importance, especially for model generalization and robustness [Vapnik, 1999, Bousquet and Elisseeff, 2002, James et al., 2013]. Certain types of regularization are necessary to prevent overcomplication and overfitting of the training data. The same is also true in deep learning, where modern networks are usually overparametrized. Various regularization techniques have been developed for training DNNs, e.g., weight decay, dropout [Srivastava et al., 2014], batch normalization [Ioffe and Szegedy, 2015], early stopping [Prechelt, 1998], etc. Though their regularization effects are largely implicit, a variety of implicit biases have been recently identified [Woodworth et al., 2019, Chizat and Bach, 2020, Razin and Cohen, 2020, Hu et al., 2021b, Ding et al., 2023]. Nevertheless, without exception, all aforementioned types of regularization are on the *functional* level, i.e., regularizing $\mathcal{F}$ with respect to some complexity measurement. However, as we will point out in the next section, the complexity of $\mathcal{F}$ itself is not of the most interest in classification. Instead, what matters the most are the *level sets* of $\mathcal{F}$.

2 PROPER REGULARIZATION FOR CLASSIFICATION

For a function $f : \mathbb{R}^d \to \mathbb{R}$, let $\|f\|_\infty = \sup_{\boldsymbol{x}\in\mathbb{R}^d} |f(\boldsymbol{x})|$. Let $\mathbb{P}$ be a probability measure on $\mathbb{R}^d$ and denote $d_\triangle(G_1, G_2) = \mathbb{P}(G_1 \triangle G_2) = \mathbb{P}\left((G_1\backslash G_2) \cup (G_2\backslash G_1)\right)$ as the measure of the symmetric difference of sets in $\mathbb{R}^d$.

Consider the binary classification setting where $\boldsymbol{x} \in \mathbb{R}^d$, $y \in \{-1, 1\}$. Let the conditional probability $\eta(\boldsymbol{x}) = \mathbb{P}(y = 1|\boldsymbol{x})$. Given $\eta(\boldsymbol{x})$, the Bayes optimal decision rule is to assign label 1 if $\eta(\boldsymbol{x}) \geq 1/2$ and label -1 if $\eta(\boldsymbol{x}) < 1/2$. If the two classes are separated (the supports of two class distributions are disjoint), η is a piecewise constant function taking values only from $\{0, 1\}$. The 0-1 loss is not friendly for optimization [Bartlett et al., 2006]. Thus, various surrogate losses are employed in practice, e.g., cross-entropy, hinge loss, etc. In statistics literature, there are two types of assumptions for classification [Audibert and Tsybakov, 2007], one on the conditional probability and the other on the decision boundary. Classification by estimating the conditional probability is usually referred to as "plug-in" classifiers and it's worth noting that it essentially reduces classification to regression. In comparison, estimating the decision boundary is more fundamental [Hastie et al., 2009]. Hence, characterizing the decision boundary is of critical importance.

2.1 FROM FUNCTION SPACE TO LEVEL SET

The goal of classification is to recover the Bayes optimal decision boundary, which divides the input space into non-overlapping regions with respect to labels. Therefore, classification is better to be thought of as *estimation of sets* in $\mathbb{R}^d$, rather than estimation of functions on $\mathbb{R}^d$. This is because the set difference reflects the 0-1 loss much more directly than functional norms on $\mathcal{F}$. To be more specific, if $f \in \mathcal{F}$ approximates η so well that $\|f(\boldsymbol{x}) - \eta(\boldsymbol{x})\|_\infty \leq 2\epsilon$, there is still no guarantee of matching the sign of $\eta(\boldsymbol{x}) - 1/2$ close to the decision boundary. Consider a noisy scenario, where the label we observe is flipped relative to the true label with probability $\left(\frac{1}{2} - \epsilon\right)$. Then the misclassification rate of f could be arbitrarily bad. In contrast, if we have a good estimation of the set $G^* = \{\boldsymbol{x} \in \mathbb{R}^d : \eta(\boldsymbol{x}) \geq 1/2\}$ such that $d_\triangle(\widehat{G}, G^*) \leq \epsilon$, the misclassification probability can be directly bounded by ϵ.

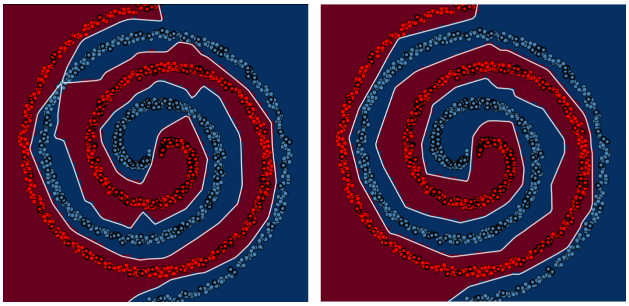

Figure 1: Illustration of a difficult classification task in $[-1, 1]^2$ using ReLU classifiers. Two classes (blue and red) are separated. Among all the points, only 300 in each class are training samples, marked with **thickened** outline. The left figure is from regular training, achieving 99.65% test accuracy; the right figure is from adversarial training, achieving 100% test accuracy. The decision boundary on the right is more robust and noticeably less complicated.

In practice, the deep classifier is parametrized by a neural network $f \in \mathcal{F}$ and the decision boundary is its *level set*, $G_f := \{\boldsymbol{x} \in \mathbb{R}^d : f(\boldsymbol{x}) = 0\}$, which is modeled *implicitly*. Let $\mathcal{G} = \{G_f : f \in \mathcal{F}\}$. Notice that regularizing f may have no effect on G_f since the level set is invariant to scaling of f. To be more specific, $f(\boldsymbol{x})$ and $\lambda \cdot f(\boldsymbol{x})$ have the same level set, and as $\lambda \to 0$, the majority of commonly used function norms $\|f(\boldsymbol{x})\|$ will tend to zero. Hence, the complexity of $\mathcal{F}$ and the complexity of $\mathcal{G}$ may not be closely connected.

When explicit regularization is absent in training deep classifiers, one may hope the decision boundary complexity is implicitly regularized, either from the model architecture or the training techniques. Unfortunately, this is not supported by empirical evidence in robust transfer learning [Shafahi et al., 2019]. Given an adversarially robust teacher model, e.g., from adversarial training, only by vanilla knowledge distillation [Hinton et al., 2015] and fitting the input-output relationship, the resulting student model, no matter the size, does not retain robustness. To achieve comparable robustness, data augmentation on the input space such as mixup samples [Muhammad et al., 2021], or matching intermediate features [Goldblum et al., 2020] seems indispensable. While matching the classifiers cannot transfer robustness, matching the decision boundary from teacher to student obviously can. From this perspective, various data augmentations can be viewed as regularization of the input space, on the decision boundary.

Adversarial training, noise injection, and margin maximization can all be viewed as means of boundary regularization, pushing decision boundaries away from training samples.

We show empirically that these methods lead to a significant reduction in boundary complexity, even though their design motivation was different. Adversarial training can be also viewed as a special form of gradient regularization [Lyu et al., 2015], or data-dependent operator norm regularization [Roth et al., 2019]. Among others, Chan et al. [2019] proposed to directly regularize the saliency of the classifier's Jacobian to improve robustness. Adversarial robustness is also shown to improve by replacing the ReLU activation with smooth functions [Xie et al., 2020], and modifying the loss function [Pang et al., 2019, Bao et al., 2020, Hu et al., 2021a]. Although the classifier gradient is more related to boundary complexity, these types of regularization methods inspired by adversarial training are not directly targeting the decision boundary.

In this work, we advocate that for classification, the proper complexity to regularize is the boundary complexity of $\mathcal{G}$, rather than the functional complexity of $\mathcal{F}$. A complexity measurement directly targeting the decision boundary will better reflect classification properties and may be largely independent of known metrics on the function space.

2.2 MEASURING BOUNDARY COMPLEXITY

Now that we have established boundary complexity as the proper, yet missing regularization in classification, the next question is how to measure it. Compared to functions, boundary complexity measurement is far less explored. In statistics literature, classification has been analyzed as a nonparametric estimation of sets problem where the convergence rate critically depends on the complexity of the hypothesis class and the estimator class [Mammen and Tsybakov, 1999]. However, the typical complexity measurements, e.g., bracketing entropy, covering number, Rademacher complexity, etc. are on the group level and cannot evaluate a single set (decision boundary). For general classifiers, how to properly quantify the boundary complexity remains an open problem. Chen et al. [2019] utilized persistent homology to measure the topological complexity of decision boundaries. Lei et al. [2022] characterized boundary complexity by their variability with respect to data and algorithm randomness. Yang et al. [2020] proposed the concept of boundary thickness and demonstrated its relationship to classification robustness. However, the aforementioned characterizations of boundary complexity are highly abstract and not explicitly calculable.

To this end, we consider specifically classifiers with Rectified Linear unit (ReLU) activation, whose decision boundary is piecewise linear, and the boundary complexity can be conveniently characterized by the number of affine pieces, which is intuitive and visually accessible. In Figure 1, the left decision boundary has 491 affine pieces while the right one has only 254. As can be seen in the figure, the less complicated boundary generalizes better and is more robust.

Remark 1 (Boundary pieces)**.** The count of boundary pieces of ReLU networks might be overly simplified for classification problems, since it does not take the length of each piece and their overall structure into consideration. However, it does offer unique benefits. Besides being intuitive and visually accessible, it also bridges the complexity of the ReLU network itself. It would be interesting to see the relationship between the count of boundary pieces and the total number of linear pieces during training. Other boundary complexities, e.g., boundary thickness, have no counterpart in the function space.

For ReLU neural networks, the structure of the affine pieces and, in particular, the number of distinct pieces have been objects of interest. Sharp bounds (exponential with depth) on the maximum number of affine regions have been investigated [Montufar et al., 2014], demonstrating the benefit of deeper networks. Hanin and Rolnick [2019] provided a framework to count the number of linear regions of a piecewise linear network. A method for upper-bounding the number of affine regions *locally* in a ball around a data point was developed in Zhu et al. [2020]. Interestingly, both experiments of Zhu et al. [2020] on local number of affine regions and ours on global count of boundary pieces indicate a two-stage behaviour during training.

In classification, we are interested in the boundary pieces (level set) more than in affine regions, and existing literature there is scarce. For counting, previous works only compute a *superset* of the decision boundary and therefore give only upper bounds on the exact number (see Proposition 6.1. in Zhang et al. [2018] and Alfarra et al. [2020]). For linking the count to classification, to the best of the authors' knowledge, the only relevant work is Hu et al. [2020], where a teacher-student classification setting is considered and upper bounds on boundary pieces (bracketing entropy) in ReLU classifiers are utilized to bound the generalization error. Interestingly, Hu et al. [2020] showed that when the student network is larger than the teacher, if the boundary complexity is not regularized, the 0-1 loss excess risk convergence rate will not be rate-optimal.

As we illustrated before, a ReLU network and its level set may share little connection. Calculating the number of boundary pieces is a new and technically challenging problem. Although there might be other ways to characterize the boundary complexity, the boundary piece count does provide a valid starting point for this problem.

2.3 CONTRIBUTIONS

In this work, we study the boundary complexity of ReLU classifiers and investigate the number of affine pieces in the decision boundary. The contributions are

- With the help of tropical geometry, we provide a novel explicit algorithm for counting the exact number of boundary pieces and affine regions of ReLU networks. In contrast to Zhang et al. [2018] and Alfarra et al. [2020], we do not require the weights to be integer-valued. Unlike the algorithm of Zhu et al. [2020], which discards some information at each layer, our approach preserves a complete representation of a neural network's functional form.

- We empirically investigate our proposed boundary complexity during training and interesting properties are revealed. First, the boundary piece count is largely independent of other measures during training. They (e.g., boundary count, total piece count, and l_2 norm of weights) share little similarity during the training process. Second, the boundary piece count is negatively correlated with robustness. Adversarial training and noise injection are found to have significant regularizing effects on boundary complexity.

3 BOUNDARY COMPLEXITY OF RELU NETWORKS

A few works Alfarra et al. [2020], Charisopoulos and Maragos [2018], Hertrich et al. [2021], Maragos et al. [2021], Montúfar et al. [2021], Trimmel et al. [2020], Zhang et al. [2018] on this topic used the ideas of *tropical geometry* - an area of algebraic geometry studying surfaces over the max-plus semi-ring Maclagan and Sturmfels [2009]. The connection to ReLU networks comes from them being compositions of affine transformations and the rectified linear unit $\sigma(x) = \max\{0, x\}$. This enables us to write the network as a difference between two convex piece-wise affine functions. These, in turn, can be interpreted in a useful way in a *dual space*, where affine functions are points and maximum functions correspond to upper convex hulls. This interpretation allowed Zhang et al. to reprove the best bounds for the largest possible number of affine regions a ReLU network with a given architecture may have.

This section expands on the tropical geometry perspective of ReLU networks. Our main theoretical result is a way to explicitly compute the zero set of a difference of two convex piecewise-affine functions—and therefore compute the exact count of boundary pieces of a ReLU network. To improve the readability, we include necessary preliminary results and rephrase them into consistent technical language. The proofs are mostly omitted and can be found in the appendix.

Let's start with a proposition taken from Magnani and Boyd [2009].

Proposition 2. A function of the form

$$f(\mathbf{x}) = \max_{i=1,\dots,n} \{A_i\mathbf{x} + b_i\}$$

is convex and piecewise-affine. Also, every convex piecewise-affine function with a finite number of linear pieces is of this form.

We will proceed to abbreviate "convex piecewise-affine" to CPA and "difference of convex piecewise-affine" to DCPA. To be precise, by a ReLU network we mean a neural network where every activation function is the rectified linear unit.

Proposition 3. Given any ReLU network, the function defined by it can be written as a DCPA function.

Conversely, Ovchinnikov [2002] proved that any piecewise-affine function with a finite number of linear regions is a min-max polynomial in its component affine functions. This implies that it can be written as a DCPA function and so – represented by a ReLU network.

3.1 TROPICAL GEOMETRY

In this section, we introduce the aforementioned interpretation of CPAs in the *dual space* $\mathbf{D}$. It may resemble a projective involution, which makes it even more surprising that notions such as convex hull turn out useful. We make no distinction between affine functions $f : \mathbf{x} \mapsto \mathbf{a}^\intercal\mathbf{x} + b$ and their graphs $\{(\mathbf{x}, y) \in \mathbb{R}^{d+1} \mid y = f(\mathbf{x})\}$. Thus, we identify affine functions $\mathbb{R}^d \to \mathbb{R}$ with hyperplanes in $\mathbb{R}^{d+1}$ containing no vertical lines ($\{\mathbf{x}_0\} \times \mathbb{R} \subseteq \mathbb{R}^{d+1}$ for some $\mathbf{x}_0 \in \mathbb{R}^d$); this ambient $\mathbb{R}^{d+1}$ will be called the *real space* and denoted $\mathbf{R}$.

We make effort to distinguish between $\mathbf{R}$ and $\mathbf{D}$ as both are copies of $\mathbb{R}^{d+1}$ which may cause confusion.

Definition 4. We say that $(\mathbf{x}, y)$ lies above (the graph of) f when $y > f(\mathbf{x})$. We denote it by $(\mathbf{x}, y) \succ f$.

Definition 5. For an affine function $f : \mathbb{R}^d \to \mathbb{R}$ given by $f(\mathbf{x}) = \mathbf{a}^\intercal\mathbf{x} + b$, we define its *dual* $\mathcal{R}^{-1}(f)$ as the point $(\mathbf{a}, b) \in \mathbb{R}^{d+1} =: \mathbf{D}$. Accordingly, this $\mathbb{R}^{d+1}$ will be called the *dual space* and denoted $\mathbf{D}$. Conversely, for a dual point $\mathbf{c} = (\mathbf{a}, b) \in \mathbf{D}$, we define $\mathcal{R}(\mathbf{c})$ to be the affine function $\mathbf{x} \mapsto \mathbf{a}^\intercal\mathbf{x} + b$ (i.e. a hyperplane in $\mathbf{R}$).

As we will see from Proposition 7, $\mathcal{R}$ turns out to interchange the relations of collinearity and concurrence, extend to planes of any dimensionalities, preserve orthogonality and sides of hyperplanes. For consistency, we set:

Definition 6. To a real point $\mathbf{z} = (\mathbf{x}, y) \in \mathbf{R}$, we associate as its dual the following hyperplane in $\mathbf{D}$

$$\mathcal{R}^{-1}(\mathbf{z}) = (\mathbf{a} \mapsto (-\mathbf{x})^\intercal\mathbf{a} + y).$$

Conversely, to a dual hyperplane $H = (\mathbf{a} \mapsto \mathbf{x}^\intercal\mathbf{a} + y) \subset \mathbf{D}$, we associate the real point

$$\mathcal{R}(H) = (-\mathbf{x}, y) \in \mathbf{R}.$$

Note that the correspondence between dual hyperplanes and real points has an extra sign not present in the pairing of dual points with real planes.

Proposition 7. The duality $\mathcal{R}$ has the following properties:

1. A dual point $\mathbf{c} \in \mathbf{D}$ lies on a dual hyperplane $H \subset \mathbf{D}$ if and only if the corresponding real hyperplane $\mathcal{R}(\mathbf{c}) \subset \mathbf{R}$ contains the point $\mathcal{R}(H) \in \mathbf{R}$. I.e.
$$\mathbf{c} \in H \Leftrightarrow \mathcal{R}(\mathbf{c}) \ni \mathcal{R}(H).$$
2. Points of a dual k-dimensional plane F are precisely the duals of real hyperplanes containing some $(d-k)$-dimensional real plane. We denote this common real $(d-k)$-dimensional hyperplane as $\mathcal{R}(F)$.
3. Duality is containment-reversing, i.e.,
$$F \subseteq G \Leftrightarrow \mathcal{R}(F) \supseteq \mathcal{R}(G)$$
for dual planes F, G, and analogously for $\mathcal{R}^{-1}$.
4. For any real hyperplane f, the projection $p(\mathcal{R}^{-1}(f))$ of its dual $\mathcal{R}^{-1}(f)$ onto the first d coordinates is normal to its isolines $\{\mathbf{x} \mid f(\mathbf{x}) = \text{const.}\}$.
5. Dual point $\mathbf{c} \in \mathbf{D}$ lies above the graph of $H \subset \mathbf{D}$ if and only if the real point $\mathcal{R}(H) \in \mathbf{R}$ lies below the graph of $\mathcal{R}(\mathbf{c}) \subset \mathbf{R}$. In symbols
$$\mathbf{c} \succ H \Leftrightarrow \mathcal{R}(\mathbf{c}) \succ \mathcal{R}(H).$$
6. Points $\mathbf{c}, \mathbf{c}'$ that differ only in the $(d+1)$-th coordinate (lie exactly above/below each other) correspond precisely to parallel planes (both under $\mathcal{R}$ and $\mathcal{R}^{-1}$).

The next proposition shows another property of the duality, crucial to our framework.

Definition 8. Let $S \subset \mathbb{R}^{d+1}$ be a finite set of points. The convex hull of S will be denoted $\mathcal{C}(S)$. Furthermore, we will call the set of points

$$\{(\mathbf{x}, y) \in \mathcal{C}(S) \mid (\mathbf{x}, y+\epsilon) \notin \mathcal{C}(S) \text{ for any } \epsilon > 0\}$$

the *upper hull* of S and denote it $\mathcal{U}(S)$. Finally, the set of vertices of $\mathcal{U}(S)$ will be denoted $\mathcal{U}^*(S)$.

Proposition 9. Let $S \subset \mathbf{D}$ be a finite set of points. Then, for every point $\mathbf{x} \in \mathbf{D}$ lying below $\mathcal{U}(S)$, we have (in $\mathbf{R}$)

$$\mathcal{R}(\mathbf{x}) \le \max\{\mathcal{R}(\mathbf{s}) \mid \mathbf{s} \in \mathcal{U}(S)\},$$

i.e. the affine function in $\mathbf{R}$ dual to $\mathbf{x}$ lies fully below the maximum of the affine functions whose duals lie on $\mathcal{U}(S)$.

Example 10 gives us a useful correspondence—each CPA function can be represented uniquely as an upper-convex hull in the dual space. This allows us to implicitly simplify the notation as well, as illustrated in Example 10.

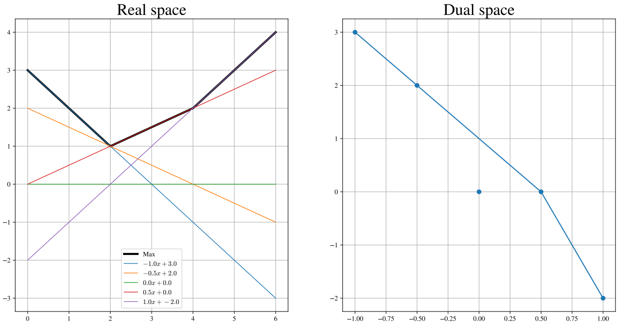

Figure 2: Real and dual diagrams in Example 10.

Example 10. Let us consider the function

$$f(x) = \max\Big\{-x+3,\ -\tfrac{1}{2}x+2,\ \tfrac{1}{2}x,\ x-2,\ 0\Big\}.$$

Figure 2 draws it in both the real and dual space.

We can see that the points $(-\frac{1}{2}, 2), (0,0) \in \mathbf{D}$ corresponding to the functions $y = -\frac{1}{2}x+2$ and $y=0$ lie respectively on and under the upper hull of the other points. This means that the functions $y = -\frac{1}{2}x+2, y=0$ never exceed the maximum of $-x+3, \frac{1}{2}x,\ x-2$, but $y = -\frac{1}{2}x+2$ matches it at some point.

In particular, we can write the maximum using just three of the functions.

$$\begin{aligned}&\max\Big\{-x+3,\ -\tfrac{1}{2}x+2,\ \tfrac{1}{2}x,\ x-2,\ 0\Big\}\\&=\max\Big\{-x+3,\ \tfrac{1}{2}x,\ x-2\Big\}\end{aligned}$$

3.2 RELU NETWORKS IN THE CONTEXT OF TROPICAL GEOMETRY

This section shows precisely how to generate the dual diagram of a function defined by a neural network.

Let us denote by $F_l : \mathbb{R}^d \to \mathbb{R}^{w_l}$ the function defined by the network taking the input to the post-activation values on the l-th layer (here w_l is the width of the l-th layer). This means that

$$F_l(\mathbf{x}) = \sigma(A_l F_{l-1}(\mathbf{x})).$$

Let us assume that $F_{l-1} = \mathcal{R}(P_{l-1}) - \mathcal{R}(N_{l-1})$ for P_{l-1} and N_{l-1} being *vectors* (ordered tuples) of sets of points. We want to write $F_l = \mathcal{R}(P_l) - \mathcal{R}(N_l)$ for P_l and N_l computed in terms of P_{l-1} and N_{l-1}. For this, we need to introduce some notation.

Definition 11. Given sets of points $X, Y \subset \mathbf{D} \cong \mathbb{R}^{d+1}$, we define

- $X \oplus Y = \{\mathbf{x}+\mathbf{y} \mid \mathbf{x} \in X, \mathbf{y} \in Y\}$ to be the *Minkowski sum of X and Y*;

- $X \cup Y$ to be the standard union of X and Y as sets.

We also define these operations on vectors of sets of points to be the coordinate-wise operations. These have important interpretations in our correspondence.

In the following, for a finite set $X \subset \mathbf{D}$ we identify $\mathcal{R}(X)$ with the function $\max\{\mathcal{R}(\mathbf{x}) \mid \mathbf{x} \in X\}$ being a maximum of hyperplanes in $\mathbf{R}$.

Proposition 12. For any sets of points $X, Y \subset \mathbf{D}$, we have

- $\mathcal{R}(X \cup Y) = \max\{\mathcal{R}(X), \mathcal{R}(Y)\}$;
- $\mathcal{R}(X \oplus Y) = \mathcal{R}(X) + \mathcal{R}(Y)$.

Proof. The first one is clear from the definition. For the second one, we have

$$\begin{aligned}&\max\{x_1, \dots, x_n\} + \max\{y_1, \dots, y_m\} \\ &= \max\{x_1 + y_1, x_1 + y_2, \dots, x_n + y_m\}.\end{aligned}$$

□

Now, we need to define matrix multiplication for vectors of sets of points.

Definition 13. Given $S \subset \mathbf{D}$, we define the scalar multiplication $\lambda \cdot S$ in the usual way. For a vector $X = (X_i)_{1 \leq i \leq n}$ of sets of points in the dual space and for an $n \times m$ matrix A we define the *Minkowski matrix product of* X *by* A through

$$(A \otimes X)_i = \bigoplus_{j=1}^{n} A_{ij} \cdot X_j.$$

Notice that we could run into problems with just using the Minkowski operations, since as long as S has at least 2 points, we will have $2 \cdot S \neq S \oplus S$. However, if we restrict ourselves to the vertices of upper convex hulls and non-negative matrices the operations are 'well-behaved'.

Proposition 14. For matrices A, B with non-negative values and vectors of points X, Y_1, Y_2, the following hold.

- $\mathcal{U}^*\big((A + B) \otimes X\big) = \mathcal{U}^*\big((A \otimes X) \oplus (B \otimes X)\big)$;
- $A \otimes (Y_1 \oplus Y_2) = (A \otimes Y_1) \oplus (A \otimes Y_2)$;
- $AB \otimes X = A \otimes (B \otimes X)$;
- $X \oplus (Y_1 \cup Y_2) = X \oplus Y_1 \cup X \oplus Y_2$.

This seems useful, but quite restrictive, since we need to operate with non-negative matrices. However, every matrix A can be written as a difference between its positive part and its negative part $A = A^+ - A^-$, where both A^+ and A^- are non-negative.

We also have an interpretation for the matrix multiplication, similar to Proposition 12. Here, when passing a vector of sets of points to the operator $\mathcal{R}$, we apply it coordinate-wise getting a vector of maximums of affine functions.

Proposition 15. Given a vector X of sets of points in $\mathbf{D}$ and a non-negative matrix A, we have

$$A\, \mathcal{R}(X) = \mathcal{R}(A \otimes X).$$

Proof.

$$[A\, \mathcal{R}(X)]_i = \bigoplus_j A_{ij} [\mathcal{R}(X)]_j = \bigoplus_j [\mathcal{R}(A_{ij} X_j)]$$

$$= \mathcal{R}(\oplus_j A_{ij} X_j) = \mathcal{R}([A \otimes X]_i) = [\mathcal{R}(A \otimes X)]_i$$

□

We can now characterise the function $F_l = \mathcal{R}(P_l) - \mathcal{R}(N_l)$ in terms of vectors of points P_{l-1} and N_{l-1}.

Proposition 16. Let's assume that $F_l = \sigma(A_l\, F_{l-1})$ and $F_{l-1} = \mathcal{R}(P_{l-1}) - \mathcal{R}(N_{l-1})$. Then, after writing $A_l = A_l^+ - A_l^-$, we get $F_l = \mathcal{R}(P_l) - \mathcal{R}(N_l)$ for

$$N_l = (A_l^- \otimes P_{l-1}) \oplus (A_l^+ \otimes N_{l-1})$$

$$\text{and} \quad P_l = (A_l^+ \otimes P_{l-1}) \oplus (A_l^- \otimes N_{l-1}) \,\cup\, N_l.$$

Proposition 16 is the key to our counting algorithm. Given a neural network, we apply it to all the layers successively, and in the end we obtain a representation of the NN as a DCPA function. Having a DCPA form, we can use proposition 20 and 21 to count the number of boundary and affine pieces.

3.3 TROPICAL HYPERSURFACES

In this section, we explore the regions into which a CPA function partitions the plane, which is called the *tessellation* of a CPA. We define it formally below.

Definition 17. Given a CPA

$$F(\mathbf{x}) = \max\{f_1(\mathbf{x}), \dots, f_n(\mathbf{x})\}$$

where f_i are affine functions, an *affine region of* F is

$$\Big\{\mathbf{x} \in \mathbb{R}^d \Big| f_i(\mathbf{x}) = f_{i'}(\mathbf{x}) > f_j(\mathbf{x}) \text{ for all } i, i' \in I, j \in J\Big\},$$

where I, J are disjoint sets whose union is $\{1, \dots, n\}$. Its *dimension* is the smallest dimension of an affine subspace of $\mathbb{R}^d$ containing it. The set of all regions of dimension k (k-cells) will be denoted as $\mathcal{T}_k(F)$, and $\mathcal{T}(F) = \bigcup_k \mathcal{T}_k(F)$.

For a set of points S in the dual space we will denote by $\mathcal{T}(S)$ the tessellation of $\mathcal{R}(S)$. For example, $\mathcal{T}_0$ is the set of all vertices of $\mathcal{T}(S)$, $\mathcal{T}_1$ is the set of all its lines, rays and segments.

Proposition 18. k-cells of $\mathcal{T}(S)$ are in one-to-one correspondence with $(d-k)$-cells of $\mathcal{U}(S)$. Each k-cell σ of $\mathcal{T}(S)$ is of the form

$$p(\mathcal{R}(\text{dual planes tangent to } \mathcal{U}(S) \text{ containing } \sigma')),$$

where σ' is a $(d-k)$-cell of $\mathcal{U}(S)$, and $p : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ is the projection onto first d coordinates.

By H being *tangent* we mean that the whole of $\mathcal{U}(S)$ lies under or on H and that $H \cap \mathcal{U}(S) \neq \emptyset$.

3.4 DECISION BOUNDARY

Let $F = \mathcal{R}(P)$ and $G = \mathcal{R}(N)$ be CPA functions $\mathbb{R}^d \to \mathbb{R}$. We are interested in being able to describe the zero set D of a DCPA function $F - G$. The proposition below expands on the idea of Proposition 6.1 in [Zhang et al., 2018].

Proposition 19. Let us assume that no points of P lie on $\mathcal{U}(N)$ and vice versa. The set D is a union of precisely these $(d-1)$-dimensional cells of $\mathcal{T}(P \cup N)$ which correspond to the edges of $\mathcal{U}(P \cup N)$ with one end in P and the other end in N.

This means that to draw the decision boundary, all we have to do is draw the hypersurface $\mathcal{T}(P \cup N)$ and identify which cells come from the intersection of the graphs of $\mathcal{R}(P)$ and $\mathcal{R}(N)$.

Proposition 19 deals with the case most likely to happen in general situations, but it is possible that some points of P lie on $\mathcal{U}(N)$ or vice versa. Proposition 20 describes this more difficult case too. We compute the boundary count of a neural network by applying 20 to the DCPA representation of a NN (from proposition 16).

Proposition 20. Let $F = \mathcal{R}(P), G = \mathcal{R}(N)$ be CPA functions. Then the zero set $D = \{\mathbf{x} \in \mathbb{R}^d \mid F(\mathbf{x}) = G(\mathbf{x})\}$ consists precisely of this cells of $\mathcal{T}(P \cup N)$, which correspond to the cells of $\mathcal{U}(P \cup N)$ containing points from both P and N.

3.5 AFFINE PIECES

Our formalism also allows us to count the exact total number of affine pieces. To do this for a neural network, we apply the corollary 21 to the DCPA form obtained from proposition 16.

Corollary 21. The number of affine pieces (d-cells) of a DCPA function $\mathcal{R}(P) - \mathcal{R}(N)$ is equal to the number of vertices of $\mathcal{U}(P \oplus N)$.

Corollary 21 is a special case of a more general result stated below.

Proposition 22. Each k-cell σ of $\mathcal{R}(P) - \mathcal{R}(N)$ is of the form

$$\sigma = p(\mathcal{R}(\text{hyperplanes tangent to } \mathcal{U}(P \oplus N) \text{ containing } \sigma'))$$

where σ' is a $(d-k)$-cell of $\mathcal{U}(P \oplus N)$. The correspondence $\sigma \leftrightarrow \sigma'$ is bijective.

To the best of the authors' knowledge, this explicit formula for counting the total number of affine pieces has not been spelled out in existing literature, where the scaling of the count with respect to neural network structures is usually the focus.

Remark 23. In ReLU neural networks it is possible to have a degenerate situation, where on two regions the network computes the same affine function, but these regions differ in activation patterns. Our approach will see such regions as separate. We do not know of any literature where this would be treated differently.

4 NUMERICAL EXPERIMENTS

In this section, as a proof of concept, we conduct numerical experiments on 2D synthetic data. The aim of this section is two-fold. Firstly, we compare the proposed boundary complexity (#Boundary) to various other complexity measurements, e.g., the total number of affine pieces (#Total), the sum of weights squared (F-norm), and evaluate their trends during training. The results show that our boundary complexity is quite unique, with distinctive features. Secondly, we demonstrate a negative correlation between the number of boundary pieces and classification robustness, where popular robust training methods, specifically noise injection and adversarial training, can both diminish the number of boundary pieces.

We choose ReLU neural networks with 2 hidden layers of different widths across all our simulations. Three training schemes are considered: regular training with cross-entropy (CE), CE with Gaussian noise injection (Noisy), and CE with l_∞-adversarial training by fast gradient sign attacks [Goodfellow et al., 2014] (Adv). Two synthetic datasets are constructed in 2-dimensional space, one is 3-by-3 Gaussian mixture (Figure 3) and the other is spiral-shaped (Figure 1). The Gaussian case provides a baseline while the spiral case is much more challenging and may better reflect complicated data structures in practice. To measure robustness, we choose Gaussian distributed random noise injection with standard deviation σ. 2000 test points are used to approximate the expectation and this empirical robustness measure is denoted (in percentile) by $R(\sigma)$.

The quantities at initialization are shown in Table 1 and Table 2. We can see that the initial #Boundary is usually much smaller, with larger variations. This is to be expected as the boundary is only a level set of the initialized classifier,

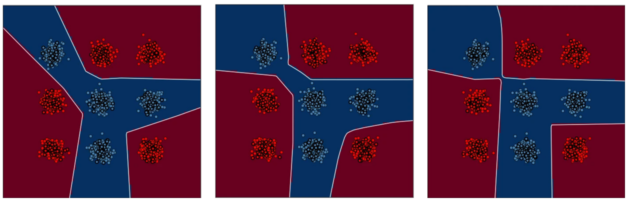

Figure 3: Decision boundaries in the 3×3 Gaussian mixture case in $[-2,2]^2$. From left to right are instances of CE (#Boundary=46), Noisy (#Boundary=41), Adv (#Boundary=40), respectively.

which can be very sensitive to constant shifts. The initial #Total is usually larger. This is interesting and indicates that the initial classifier is more random in terms of linear region arrangement. Like #Boundary, the F-norm at initialization is much smaller, but with much smaller variations. This is to be expected as the F-norm is directly linked to initialized weights.

4.1 TRENDS DURING TRAINING

For different tasks, we can observe the overall trend for #Boundary to be: first increase, then decrease and finally stabilize. Similar behaviors can also be observed for #Total and F-norm during training, but their movements are not synchronized. Among the training methods, the overall trends share more similarities than differences, except for with or without weight decay. Typical instances are shown in Figure 4 and 5.

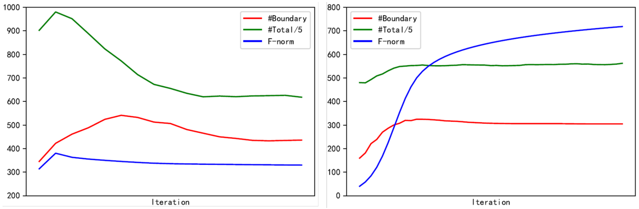

Figure 4: Training trends of #Boundary (red), #Total (green), F-norm (red) vs. iteration in the 2D spiral case. Left: CE with weight decay; Right: CE without weight decay.

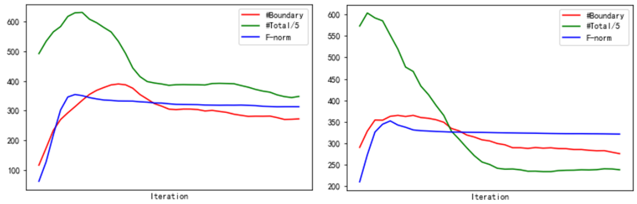

Figure 5: Training trends of #Boundary (red), #Total (green), F-norm (red) vs. iteration in the 2D spiral case. Left: Noisy with weight decay; Right: Adv with weight decay.

Table 1: Comparison of boundary piece counts in the Gaussian mixture case for ReLU network with layer widths 2-10-10-1. The reported number is an average (standard deviation) of 10 repetitions.

	#Boundary	#Total	F-norm	Acc%	$R(0.2)$
Initial	29 (17)	290 (29)	6.8 (0.61)	50.1 (1.1)	-
CE	43 (5.3)	190 (24)	57 (3.6)	100	96.4
Noisy	41 (3.1)	216 (26)	67 (2.6)	100	97.0
Adv	36 (4.6)	172	73 (2.1)	100	97.2

Table 2: Comparison of boundary piece counts in the 2D spiral case for ReLU network with layer widths 2-30-30-1. The reported number is an average (standard deviation) of 10 repetitions.

	#Boundary	#Total	F-norm	Acc%	$R(0.02)$
Initial	90 (61)	2432 (179)	20 (0.71)	50.2 (1.2)	-
CE	377 (31)	1915 (207)	283 (11)	93.60 (1.8)	94.3 (2.2)
Noisy	272 (33)	1493 (114)	322 (17)	99.15 (0.56)	98.1 (0.51)
Adv	259 (21)	1241 (135)	356 (19)	99.35 (0.38)	98.9 (0.36)

#Boundary vs others. The left figure in Figure 4 shows the typical trends in the Noisy case with weight decay, where we can clearly see that #Boundary lags behind the others. When the training starts, #F-norm and #Total peak much earlier than #Boundary. In most cases, we observe that F-norm peaks first, then #Total, and lastly #Boundary. When robust training is applied (Noisy, Adv), the gaps among them widen. In the later stage, F-norm stabilizes much faster than the others, while we can consistently observe that #Boundary flattens slower than #Total. Overall, #Boundary appears to change much slower than the others, taking more time to peak, and more time to plateau.

Role of weight decay. The right figure in Figure 4 shows a typical trend in the CE case without weight decay, which demonstrates drastically different behaviors. #Boundary and #Total plateau much earlier and do not change much once the classifier has overfit the training data. In comparison, F-norms keep getting larger, which is to be expected due to the use of cross-entropy loss. Weight decay is found to play an important role in the forming of ReLU networks' geometric structures. This is surprising as naively shrinking a ReLU network does not change its affine piece arrangement.

4.2 CLASSIFICATION ROBUSTNESS

In this section, we aim to investigate the relationship between robustness and #Boundary. However, in the absence of practical algorithms to regularize the boundary complexity, we turn to popular robust training methods and evaluate whether they can significantly reduce #Boundary. Results for the Gaussian mixture and spiral case are reported in Table 1 and Table 2, respectively.

In the simpler Gaussian mixture case, the strength for Noisy and Adv are both set at 0.1, the same as the variance of each mixing component. Figure 3 shows the decision boundaries for CE, Noisy and Adv. Despite the apparent visual difference, the #Boundary does not differ that much. In Table 1, we can observe #Boundary to be smaller on average for Noisy and especially Adv.

The effects of Noisy and Adv become more significant in the harder, more challenging spiral case. CE does not perform as consistently as Noisy or Adv and sometimes will miss the spiral shape. The strength for Noisy and Adv are both set at 0.01, which is roughly the size of the margin. As can be seen from Table 2, both #Boundary and #Total significantly dropped while F-norm stays relatively on the same level.

On both datasets, compared with CE, Noisy and Adv have strong effects on reducing the boundary complexity. The same is not true for function complexity such as F-norm.

5 DISCUSSION

We advocate that proper regularization on the decision boundary is of critical importance to classification. As a proof of concept, we choose the number of linear pieces of ReLU networks to measure the boundary complexity, due to its well-definedness. The main technical contribution is the explicit formula to count the exact number of boundary pieces as well as total affine pieces. Empirical evaluation and justification are made on synthetic data and interesting properties of the boundary piece count are revealed.

Limitations and extensions. (1) While the main focus of this work is on rectified linear units, our method can easily be extended to leaky ReLU activation, and basically all other piecewise linear functions. (2) In the experiments, we only evaluated binary classification. However, it is also quite straightforward to count the boundaries between any two given classes in the multi-class classification scenario. (3) In the present form, the computation scaling with respect to the network size is impractical for large models, especially with input dimension and depth. The most time-consuming part is the Minkowski sum. However, most of them do not directly contribute to the level set. We believe that further optimizations could shed more light on the mechanics of training procedures. Moreover, incorporating differentiability would give a penalty term that regularizes a previously unaddressed aspect of the network. (4) Though intuitive, the number of boundary pieces may not be the best choice for the complexity measurement in classification, since it doesn't take finer details such as piece arrangement into consideration. How to better quantify boundary complexity remains an open question.

Regularizing the boundary complexity. Given a measurable boundary complexity, regularizing it during the training process can be challenging. Adversarial training or noise injection can act as a regularization for boundary complexity, as verified in our experiment. Defining suitable boundary complexity measurement and proposing direct and more efficient ways to control it is an open question. The aim of this work is to identify such an important problem and convince the readers that boundary complexity is indeed proper to regularize for classification robustness. Such regularization is not at odds with other established methods, but a healthy complement to existing literature. The level set sampling method proposed in Atzmon et al. [2019] may be a good starting point. Uncovering the link of our work to persistent homology Chen et al. [2019] is also interesting. We hope that further work will lead to achieving our ultimate goal – designing practical and scalable algorithms for effective regularization and thus improving state-of-the-art performance in classification.

References

Motasem Alfarra, Adel Bibi, Hasan Hammoud, Mohamed Gaafar, and Bernard Ghanem. On the decision boundaries of neural networks: A tropical geometry perspective. *arXiv preprint arXiv:2002.08838*, 2020.

Matan Atzmon, Niv Haim, Lior Yariv, Ofer Israelov, Haggai Maron, and Yaron Lipman. Controlling neural level sets. *arXiv preprint arXiv:1905.11911*, 2019.

Jean-Yves Audibert and Alexandre B Tsybakov. Fast learning rates for plug-in classifiers. *The Annals of statistics*, 35(2):608–633, 2007.

Han Bao, Clay Scott, and Masashi Sugiyama. Calibrated surrogate losses for adversarially robust classification. In *Conference on Learning Theory*, pages 408–451. PMLR, 2020.

Peter L Bartlett, Michael I Jordan, and Jon D McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

Olivier Bousquet and André Elisseeff. Stability and generalization. *The Journal of Machine Learning Research*, 2: 499–526, 2002.

Alvin Chan, Yi Tay, Yew Soon Ong, and Jie Fu. Jacobian adversarially regularized networks for robustness. *arXiv preprint arXiv:1912.10185*, 2019.

Vasileios Charisopoulos and Petros Maragos. A tropical approach to neural networks with piecewise linear activations. *arXiv preprint arXiv:1805.08749*, 2018.

Chao Chen, Xiuyan Ni, Qinxun Bai, and Yusu Wang. A topological regularizer for classifiers via persistent homology. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2573–2582. PMLR, 2019.

Lenaic Chizat and Francis Bach. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. In *Conference on Learning Theory*, pages 1305–1338. PMLR, 2020.

Liang Ding, Tianyang Hu, Jiahang Jiang, Donghao Li, Wenjia Wang, and Yuan Yao. Random smoothing regularization in kernel gradient descent learning. *arXiv preprint arXiv:2305.03531*, 2023.

Micah Goldblum, Liam Fowl, Soheil Feizi, and Tom Goldstein. Adversarially robust distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3996–4003, 2020.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Boris Hanin and David Rolnick. Complexity of linear regions in deep networks. In *International Conference on Machine Learning*, pages 2596–2604. PMLR, 2019.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

Christoph Hertrich, Amitabh Basu, Marco Di Summa, and Martin Skutella. Towards lower bounds on the depth of relu neural networks. *arXiv preprint arXiv:2105.14835*, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Tianyang Hu, Zuofeng Shang, and Guang Cheng. Sharp rate of convergence for deep neural network classifiers under the teacher-student setting. *arXiv preprint arXiv:2001.06892*, 2020.

Tianyang Hu, Jun Wang, Wenjia Wang, and Zhenguo Li. Understanding square loss in training overparametrized neural network classifiers. *arXiv preprint arXiv:2112.03657*, 2021a.

Tianyang Hu, Wenjia Wang, Cong Lin, and Guang Cheng. Regularization matters: A nonparametric perspective on overparametrized neural network. In *International Conference on Artificial Intelligence and Statistics*, pages 829–837. PMLR, 2021b.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 2009.

Shiye Lei, Fengxiang He, Yancheng Yuan, and Dacheng Tao. Understanding deep learning via decision boundary. *arXiv preprint arXiv:2206.01515*, 2022.

Chunchuan Lyu, Kaizhu Huang, and Hai-Ning Liang. A unified gradient regularization family for adversarial examples. In *2015 IEEE international conference on data mining*, pages 301–309. IEEE, 2015.

Diane Maclagan and Bernd Sturmfels. Introduction to tropical geometry. *Graduate Studies in Mathematics*, 161, 2009.

Alessandro Magnani and Stephen P Boyd. Convex piecewise-linear fitting. *Optimization and Engineering*, 10:1–17, 2009.

Enno Mammen and Alexandre B Tsybakov. Smooth discrimination analysis. *The Annals of Statistics*, 27(6): 1808–1829, 1999.

Petros Maragos, Vasileios Charisopoulos, and Emmanouil Theodosis. Tropical geometry and machine learning. *Proceedings of the IEEE*, 109(5):728–755, 2021.

Guido Montúfar, Yue Ren, and Leon Zhang. Sharp bounds for the number of regions of maxout networks and vertices of minkowski sums. *arXiv preprint arXiv:2104.08135*, 2021.

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932, 2014.

Awais Muhammad, Fengwei Zhou, Chuanlong Xie, Jiawei Li, Sung-Ho Bae, and Zhenguo Li. Mixacm: Mixup-based robustness transfer via distillation of activated channel maps. *Advances in Neural Information Processing Systems*, 34, 2021.

Sergei Ovchinnikov. Max-min representation of piecewise linear functions. *Contributions to Algebra and Geometry*, 43(1):297–302, 2002.

Tianyu Pang, Kun Xu, Yinpeng Dong, Chao Du, Ning Chen, and Jun Zhu. Rethinking softmax cross-entropy loss for adversarial robustness. *arXiv preprint arXiv:1905.10626*, 2019.

Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 1998.

Noam Razin and Nadav Cohen. Implicit regularization in deep learning may not be explainable by norms. *arXiv preprint arXiv:2005.06398*, 2020.

Kevin Roth, Yannic Kilcher, and Thomas Hofmann. Adversarial training is a form of data-dependent operator norm regularization. *arXiv preprint arXiv:1906.01527*, 2019.

Ali Shafahi, Parsa Saadatpanah, Chen Zhu, Amin Ghiasi, Christoph Studer, David Jacobs, and Tom Goldstein. Adversarially robust transfer learning. *arXiv preprint arXiv:1905.08232*, 2019.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Martin Trimmel, Henning Petzka, and Cristian Sminchisescu. Tropex: An algorithm for extracting linear terms in deep neural networks. In *International Conference on Learning Representations*, 2020.

Vladimir N Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5): 988–999, 1999.

Blake Woodworth, Suriya Genesekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Kernel and deep regimes in overparametrized models. In *Conference on Learning Theory (COLT)*, 2019.

Cihang Xie, Mingxing Tan, Boqing Gong, Alan Yuille, and Quoc V Le. Smooth adversarial training. *arXiv preprint arXiv:2006.14536*, 2020.

Yaoqing Yang, Rajiv Khanna, Yaodong Yu, Amir Gholami, Kurt Keutzer, Joseph E Gonzalez, Kannan Ramchandran, and Michael W Mahoney. Boundary thickness and robustness in learning models. *Advances in Neural Information Processing Systems*, 33:6223–6234, 2020.

Liwen Zhang, Gregory Naitzat, and Lek-Heng Lim. Tropical geometry of deep neural networks. In *International Conference on Machine Learning*, pages 5824–5832. PMLR, 2018.

Rui Zhu, Bo Lin, and Haixu Tang. Bounding the number of linear regions in local area for neural networks with relu activations. *arXiv preprint arXiv:2007.06803*, 2020.