Exact Visualization of Deep Neural Network Geometry and Decision Boundary

Ahmed Imtiaz Humayun IMTIAZ@RICE.EDU
Rice University
Randall Balestriero RBAL@FB.COM
FAIR
Richard Baraniuk RICHB@RICE.EDU
Rice University

Abstract

Visualizing Deep Network (DN) geometry and decision boundaries remains a key challenge even today. In fact, despite the dire need for such methods e.g. to assess the quality of a trained model, to compare models, to interpret decisions, the community at large still relies on crude approximations and proxies. For example, computing the decision boundary of a model, say on a 2d slice of their input space, is done through gradient descent and sampling with dichotomy search. In this paper, we lean on the rich theory of Continuous Piece-Wise Linear (CPWL) DNs to provide, for the first time, a method that provably produces the exact geometry (CPWL partition) and decision boundary of any DN employing nonlinearities such as ReLU, Leaky-ReLU, and max-pooling. Using our method we are able to not only visualize the decision boundary but also obtain its spanning space, i.e., we can sample arbitrarily many inputs that provably lie on the model's decision boundary, up to numerical precision. We explore how such methods can be used to interpret architectural choices e.g. using convolutional architectures instead of fully-connected neural networks.

1. Introduction

Deep learning and in particular Deep Networks (DNs) have redefined the landscape of machine learning and pattern recognition. Although current DNs employ a variety of techniques that improve their performances, their core definition has changed much and simply consist in sequentially mapping an input vector $\boldsymbol{x}$ to a sequence of L *feature maps* $\boldsymbol{z}^{\ell}$, $\ell = 1, \dots, L$ by successively applying the simple nonlinear transformation (termed a DN *layer*)

$$\boldsymbol{z}^{\ell+1} = \boldsymbol{a}\left(\boldsymbol{W}^{\ell}\boldsymbol{z}^{\ell} + \boldsymbol{b}^{\ell}\right), \quad \ell = 0, \dots, L-1 \tag{1}$$

with $\boldsymbol{z}^0 = \boldsymbol{x}$, $\boldsymbol{W}^{\ell}$ the weight matrix, $\boldsymbol{b}^{\ell}$ the bias vector, and $\boldsymbol{a}$ an activation operator that applies a scalar nonlinear activation function a to each element of its vector input. The parametrization of $\boldsymbol{W}^{\ell}, \boldsymbol{b}^{\ell}$ controls the type of layer employed e.g. circulant matrix for convolutional layer.

Interpreting the geometry of a DN is not a trivial task since many different parameters can lead to the same input-output mapping. A common example is obtained by permuting the rows of $\boldsymbol{W}^{\ell}, \boldsymbol{b}^{\ell}$ and the columns of $\boldsymbol{W}^{\ell+1}$ for some layer(s). It is clear that the overall mapping remains unchanged, yet the parameters are different. As a result, practitioners have relied on different solutions to interpret what has been learned by a model without directly looking at their weights. One important example is to find the closest point to a training sample $\boldsymbol{x}$ that lies on the model's decision boundary, which finds practical use for active learning and adversarial robustness (He et al., 2018). In this setting, one commonly perform gradient updates on $\boldsymbol{x}$ from an objective that is minimized when an input lies on the model's decision boundary. Although alternative and more efficient solutions have been developed, most of these methods rely on search and sampling. Therefore, there still exists a requirement for exact methods of computing the decision boundary.

In this paper, we propose a special family of DNs obtained by restricting $\boldsymbol{a}$, the activation functions, to be Continuous Piece-Wise Linear (CPWL) as is the case with the eponymous (leaky-)ReLU, absolute value, max-pooling. In this setting, the entire DN will itself becomes a CPWL operator, and we will demonstrate how from this observation alone, it is possible to probe the DN's geometry in an exact manner. Echoing our previous example, we will be able to obtain the exact space of the DN's input space where its decision boundary lies and therefore we are able to sample from the decision boundary. Furthermore, leaning on the CPWL form of the DN, we demonstrate that its geometry can be entirely described by its *input space partition* and *per-region affine mappings* which again, we will be able to obtain in closed-form. All in all, we provide a tractable and efficient method to visualize and quantitatively interpret the geometry of Deep Neural Networks, opening new doors to interpretability and visualization.

2. The Geometry and Decision Boundary of Continuous Piece-Wise Linear Deep Networks

2.1. Deep Networks as Continuous Piece-Wise Linear Operators

One rich class of functions emerges from piecewise polynomials: spline operators. In short, given a partition Ω of a domain $\mathbb{R}^S$, a spline of order k is a mapping defined by a polynomial of order k on each region $\omega \in \Omega$ with continuity constraints on the entire domain for the derivatives of order $0,\dots,k-1$. As we will focus entirely on affine splines ($k = 1$) we only formally define this case for concreteness. An affine spline S produces its output via

$$\boldsymbol{S}(\boldsymbol{x}) = \sum_{\omega \in \Omega} (\boldsymbol{A}_\omega \boldsymbol{x} + \boldsymbol{b}_\omega) \mathbb{1}_{\{\boldsymbol{x} \in \omega\}}, \tag{2}$$

with $\boldsymbol{A}_\omega, \boldsymbol{b}_\omega$ the per-region *slope* and *offset* parameters respectively and with constraints that the entire mapping is continuous on the domain i.e. $S \in \mathcal{C}^0(\mathbb{R}^S)$. Spline operators and especially affine spline operators have been extensively used in function approximation theory (Cheney and Light, 2009), optimal control (Egerstedt and Martin, 2009), statistics (Fantuzzi et al., 2002) and related fields.

2.2. Exact Computation of Their Partition and Decision Boundary

To visualize the decision boundary of a Deep Neural Network, we must first find the spline partition induced by the Neural Network and project the decision boundary to the input space. Given any Neural Network, we first represent the per layer operation as affine operations by the network. While this is trivial for multi-layer perceptrons, convolution and residual blocks can also be expressed as affine operations (Balestriero and Baraniuk,

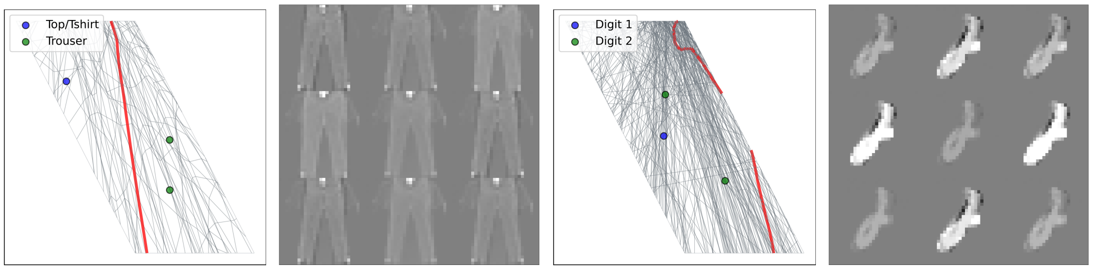

Figure 1: **(Left)** Decision boundary visualization for an MLP with width 50 and depth 3, trained on fashion-MNIST. Dark red line represents the learned decision boundary while black lines represent the spline partition of the network. **(Middle Left)** Samples from the decision boundary between classes Top and Trouser. The samples have distinct attributes present from both classes. **(Middle Right)** Decision boundary and partition visualization of a convolutional neural network trained on MNIST, with two convolutional layers and one hidden fully connected layer of width 50. Both of the digit 2 samples are misclassified by the network as digit 1. **(Right)** Samples from the decision boundary between digits 1 and 2 of MNIST.

2020). Let, our network can be represented as a set of L affine layers, with layerwise affine parameters $\{\boldsymbol{W}^\ell, \boldsymbol{b}^\ell\}_{\ell=0}^{L-1}$. For ease of discussion, suppose the network has a ReLU non-linearity right after the first $L-1$ layers. Note that our implementation is generalized for any CPWL activation, e.g. ReLU, Leaky-ReLU, sawtooth. Each layerwise affine operation can be considered a projection of incoming vectors z^ℓ onto a set of hyperplanes in the input space of the layer, defined by $\langle \boldsymbol{w}_i^\ell, z^\ell \rangle + b_i^\ell = 0$, where $\boldsymbol{w}_i^\ell$ is the i-th element of $\boldsymbol{W}^\ell$ and b_i^ℓ is the i-th element of $\boldsymbol{b}^\ell$. Each hyperplane projection gives the pre-activation of that layer for the corresponding output dimension. For a subsequent ReLU activation, vectors from the negative half-space created by the hyperplane are mapped to zero, while vectors from the positive half-space are mapped linearly. Therefore, for each layer, we have a set of hyperplanes in the input space of the layer which define the position of the non-linearities. Building on this intuition, for each layer our algorithm finds the set of convex polytopes formed by the intersection of hyperplanes, where each polytope represents a linear region. The pseudocode for finding the complete spline partition and the decision boundary can therefore be summarized as below:

- Given a bounded input domain, iterate through the first layer ($\ell = 1$) hyperplanes $\langle \boldsymbol{w}_i^\ell, \boldsymbol{z}^\ell \rangle + b_i^\ell = 0$ and cut the input domain to form convex V-polytopes. For each polytope ω the operation performed by the layer is an affine operation via parameters $\boldsymbol{A}_\omega^\ell = \boldsymbol{q}_\omega^\ell \odot \boldsymbol{W}^\ell$ and $\boldsymbol{b}_\omega^\ell = \boldsymbol{q}_\omega^\ell \odot \boldsymbol{b}^\ell$, where $\boldsymbol{q}_\omega^\ell$ is the activation pattern corresponding to the ω polytope of the ℓ-th layer. Project the V-polytopes to the next layer via region-wise affine operation $\boldsymbol{z}^{\ell+1} = \boldsymbol{A}_\omega^\ell \boldsymbol{z}^\ell + \boldsymbol{b}^\ell$

- For layers $\{1, 2 ... L-2\}$ use the layer hyperplanes to partition the incoming polytopes and update the region-wise affine parameters. For each new region ω in layer ℓ input formed by cutting region ω' from layer $\{\ell' = \ell - 1\}$ output, the affine parameters will be $\{\boldsymbol{A}_\omega^\ell, \boldsymbol{b}_\omega^\ell\}$=$\{\boldsymbol{q}_\omega^\ell \odot \boldsymbol{W}^\ell \boldsymbol{A}_{\omega'}^{\ell'}, \boldsymbol{q}_\omega^\ell \odot \boldsymbol{b}^\ell + \boldsymbol{q}_\omega^\ell \odot \boldsymbol{W}^\ell \boldsymbol{b}_{\omega'}^{\ell'}\}$.

Architecture	Dataset	Parameters	Avg Vol	Avg Number of Vertices	Ecc	Number of Regions
MLP	MNIST	44,860	3.144e-4	4	102e7	318
	Fashion-MNIST	44,860	4.991e-4	4	36e7	1364
CONV	MNIST	39,780	1.134e-5	4	17e7	8814
	Fashion-MNIST	39,780	3.54e-5	4	14e7	28222

Table 1: Statistics of the spline partitions formed by fully-connected (MLP) and Convolutional neural networks. For each dataset, the same 2D slice and input domain is used to find the partition regions. Convolutional neural networks form a significantly higher number of regions compared to MLPs even with less parameters. The mean eccentricity and volume across regions is also significantly lower for convolutional neural networks.

- For networks trained with softmax output activations, during inference we can consider the output activation as a max since the decision boundary between classes occurs when the $\arg\max$ differs. Therefore, the decision boundary between classes $\{i, j\}$ at the final layer input can be expressed by the hyperplane $\langle \boldsymbol{w}_i^\ell - \boldsymbol{w}_j^\ell, \boldsymbol{z}^\ell \rangle + b_i^\ell - b_j^\ell$. We therefore project the null space of the hyperplanes in layer $L-1$ to the input space to get the decision boundaries in the input space.

While the above method is generalized for arbitrary input dimensionality, it can be computationally expensive. In our implementation, given three points $\boldsymbol{x}_1, \boldsymbol{x}_2$ and $\boldsymbol{x}_3$ we can define 2d slice of the input space through those points i.e.

$$S = \{\boldsymbol{x}_1 + \alpha_1(\boldsymbol{x}_2 - \boldsymbol{x}_1) + \alpha_2(\boldsymbol{x}_3 - \boldsymbol{x}_1) : \alpha_1 \in \mathbb{R}, \alpha_2 \in \mathbb{R}\},$$

so that at $(0, 0)$ we are at $\boldsymbol{x}_1$, at $(1, 0)$ we are at $\boldsymbol{x}_2$, and at $(0, 1)$ we are at $\boldsymbol{x}_3$. We consider a bounded domain on this slice to compute the partition boundary.

2.3. Impact of Architecture on Partitions Properties

We see that the choice of architecture can have significant effect on the partitioning induced by a deep neural network (Tab. 1). For a given dataset, we fix the input domain and switch between convolutional and fully connected architectures to draw emphasis on the effect of the symmetries induced by a convolutional layer. We see that in convolutional architectures, there is a significantly higher number of partition regions formed, which is an indication of higher complexity of the learned model. We also see that the eccentricity and volume of the polytopes are significantly smaller for convolutional architectures compared to fully connected architectures, indicating more uniform partition shapes and higher partition density. These can also be visualized in Fig. 1.

3. Conclusions

We present the first provable method to visualize and sample the decision boundary of deep neural networks with CPWL non-linearities. Our presented methods may allow many future avenues of exploration and understanding of neural network geometries.

References

Randall Balestriero and Richard G Baraniuk. Mad max: Affine spline insights into deep learning. *Proceedings of the IEEE*, 2020.

Elliott Ward Cheney and William Allan Light. *A course in approximation theory*, volume 101. American Mathematical Soc., 2009.

Magnus Egerstedt and Clyde Martin. *Control theoretic splines: optimal control, statistics, and path planning*. Princeton University Press, 2009.

Cesare Fantuzzi, Silvio Simani, Sergio Beghelli, and Riccardo Rovatti. Identification of piecewise affine models in noisy environment. *International Journal of Control*, 75(18): 1472–1485, 2002.

Warren He, Bo Li, and Dawn Song. Decision boundary analysis of adversarial examples. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BkpiPMbA-.