Provable Instance Specific Robustness via Linear Constraints

Ahmed Imtiaz Humayun [*1] **Josue Casco-Rodriguez** [*1] **Randall Balestriero** [2] **Richard G. Baraniuk** [1]

Abstract

Deep Neural Networks (DNNs) trained for classification tasks are vulnerable to adversarial attacks. But not all the classes are equally vulnerable. Adversarial training does not make all classes or groups equally robust as well. For example, in classification tasks with long-tailed distributions, classes are asymmetrically affected during adversarial training, with lower robust accuracy for less frequent classes. In this regard, we propose a provable robustness method by leveraging the continuous piecewise-affine (CPA) nature of DNNs. Our method can impose linearity constraints on the decision boundary, as well as the DNN CPA partition, without requiring any adversarial training. Using such constraints, we show that the margin between the decision boundary and minority classes can be increased in a provable manner. We also present qualitative and quantitative validation of our method for class-specific robustness. Our code is available at https://github.com/Josuelmet/CROP

1. Introduction

Deep Neural Networks (DNNs) while ubiquitous, have a concerning vulnerability to adversarial attacks. Recent studies (Zhao et al., 2022; Kim et al., 2019) have shown that subgroups within the data can be affected by adversarial attacks in a non-uniform manner. For example, minority classes have been shown to be more susceptible to adversarial attacks as well as less robust even when adversarial training is performed (Wu et al., 2021). Similar phenomenon can be seen for subgroups within the data, e.g., demographic groups (Xu et al., 2021). Based on the application of choice, one might require a DNN to be robust for a minority class, a subgroup or specific samples.

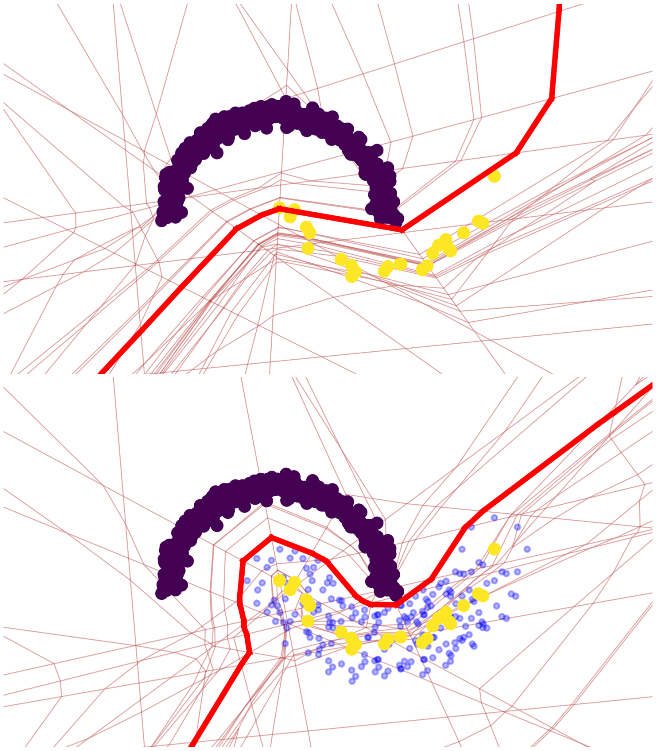

Figure 1: SplineCam (Humayun et al., 2023) visualization of DNN trained on 2D two moons classification task with **(bottom)** and without **(top)** our proposed constraint. Our method termed *CROP*, is applied to the minority class samples (yellow) which are undersampled by $20 : 1$. The dark red line represents the decision boundary *exactly*, whereas light red lines represent boundaries of each ReLU neuron from the network. Blue dots represent constraint points as described in Sec. 3. Without the constraint, we see that the decision boundary is biased towards the majority class, i.e., the decision boundary has a larger margin from the majority class samples.

We present **CROP**, or linear **C**onstraints for **P**rovable **RO**bustness. CROP can ensure robustness in an instance specific or point-wise manner, i.e., the user can constrain the network to be provably robust in the locality of any given sample or set of samples. For a region prescribed in the input space as the convex hull of a set of vertices, our method can provably ensure that the decision boundary does not in-

[*] Equal contribution [1] Rice University [2] Independent Researcher. Correspondence to: Ahmed Imtiaz Humayun <imtiaz@rice.edu>.

2^{nd} *AdvML Frontiers workshop at* 40^{th} *International Conference on Machine Learning*, Honolulu, Hawaii, USA. PMLR 202, 2023.

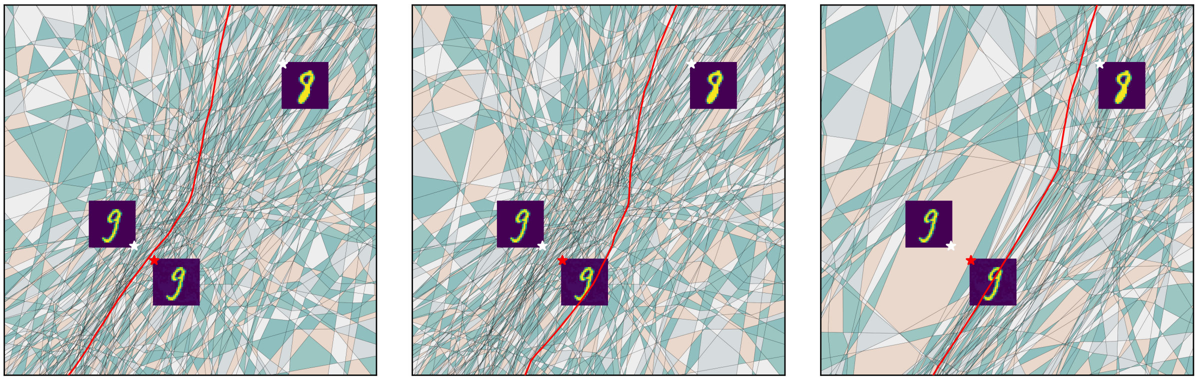

Figure 2: SplineCam (Humayun et al., 2023) visualizations of the input space for an MLP trained on a MNIST classification task without constraints **(left)**, with constraints only on the decision boundary **(middle)** and with constraints on all the neurons of the network **(right)**. Two training samples are shown with white markers, belonging to classes 9 and 8. An adversarial example generated using PGD attack with epsilon $8/255$ and 10 steps is shown with a red marker. Constraining either the decision boundary or the network ensures that the adversarial example is on the digit 9 side of the boundary, ensuring robustness. For this example, we only use 20 vertices around the digit 9 to constrain the network.

tersect the region. By prescribing such regions centered on training samples, we show how the network can be forced to increase its margin for minority classes while improving the robustness for out-of-training samples within the group as well. CROP is exact for DNNs with CPA non-linearities, and can provably defend against data poisoning attacks as well.

Our contributions can be listed as follows:

- We present a provable method to guarantee robustness in an instance specific manner against adversarial attacks as well as data poisoning attacks,
- We provide qualitative visualizations representing the exact spline partition of a DNN with and without robustness constraints,
- We present quantitative results showing superior performance against a number of state-of-the-art attacks in a class-specific robustness task.

2. Background

2.1. Adversarial Robustness

Adversarial attacks fool a classifier $f_{\boldsymbol{\theta}}$ by starting with a real data sample $\boldsymbol{x}$ and adding optimized perturbations to produce an adversarial example $\boldsymbol{x}_i$ such that the real and adversarial examples have different predicted labels but look similar, $f_{\boldsymbol{\theta}}(\boldsymbol{x}) \neq f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$. (Szegedy et al., 2014; Papernot et al., 2017; Yuan et al., 2019; Schott et al., 2019). Adversarial attacks are particularly concerning in long-tailed (i.e., imbalanced) datasets (Wu et al., 2021), since it has been empirically shown that neural network training dynamics incentivize the learned decision boundary to be closer to minority samples. In line with other literature, we define ϵ-robustness as successful classification of all potential adversarial examples within an ϵ-ball around a data sample $\boldsymbol{x}$.

Several adversarial attack methods exist with which to evaluate the robustness of a network $f_{\boldsymbol{\theta}}$. In this paper, we quantitatively validate the performance of our proposed method, against the following attacks. First is the Fast Gradient Sign Method (FGSM), which calculates the gradient in the direction of the decision boundary once, then applies noise in the direction of the gradient to generate an adversarial sample (Goodfellow et al., 2014). Next is the Projected Gradient Descent (PGD) attack and its Kullback-Leibler variant Trades' PGD (TPGD) (Madry et al., 2017; Zhang et al., 2019). While FGSM calculates the adversarial gradient once, PGD and TPGD iteratively descend the adversarial gradient while ensuring that the resulting adversarial example remains within the specified ℓ_∞ ball of the original sample, thus performing a constrained adversarial optimization. Finally, AutoAttack uses an ensemble of four parameter-free attacks, including two step size-free versions of PGD (Croce & Hein, 2020).

2.2. Deep Networks are Continuous Piecewise-Affine Operators

The core operation of DNNs primarily consists of sequentially mapping an input vector $\boldsymbol{x}$ to a sequence of L *feature maps* $\boldsymbol{z}^\ell$, $\ell = 1, \dots, L$ by successively applying simple nonlinear transformations, as in

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

Algorithm 1 CROP Forward Pass

```
Input: Mini-batch $\boldsymbol{X} \in \mathbb{R}^{N \times D}$
Constraints $\boldsymbol{C} \in \mathbb{R}^{R \times V \times D}$, where $R$ is the number of constraint regions, and $V$ is the number of vertices per region
$L$-layer neural network $f_{\boldsymbol{\theta}}$ with weights $\boldsymbol{W}^\ell$ and biases $\boldsymbol{b}^\ell$ at layer $\ell$
Numerical error tolerance constant $\tau = 0.001$
Output: $\boldsymbol{X}^\ell$
Initialize $\boldsymbol{X}^1 \leftarrow \boldsymbol{X}, \boldsymbol{C}^1 \leftarrow \boldsymbol{C}$    ▷ $\boldsymbol{X}^1$ and $\boldsymbol{C}^1$ are the inputs to layer $\ell = 1$
for layer $\ell = 1$ to $L$ do
    Initialize $\boldsymbol{c} \leftarrow \boldsymbol{0}$    ▷ $\boldsymbol{c}$ is the extra bias
    $\boldsymbol{H} \leftarrow \boldsymbol{C}^\ell(\boldsymbol{W}^\ell)^T + (\boldsymbol{b}^\ell)^T$    ▷ Calculate the constraints' pre-activations, with shape $(R, V, -1)$
    if layer $\ell$ should be constrained then
        $\boldsymbol{d} \leftarrow \text{where}\left(|\text{sgn}(\boldsymbol{H}).\text{sum}(\text{axis} = 1)| \neq V\right).\text{any}(0)$    ▷ Find which dimensions need extra bias
        $\boldsymbol{s} \leftarrow \text{sgn}(\text{sgn}(\boldsymbol{H}).\text{sum}(\text{axis} = 0, 1))[\boldsymbol{d}]$    ▷ Get the majority sign of each dimension in $\boldsymbol{d}$
        $\boldsymbol{c}[\boldsymbol{d}] \leftarrow (1 + \tau)\text{ReLU}(\boldsymbol{H}[:, :, \boldsymbol{d}] \odot \boldsymbol{s}).\text{max}(\text{axis} = 0, 1) \odot \boldsymbol{s}$    ▷ Calculate bias for non-agreement.
                                                                                              ▷ $\odot$ is elementwise multiplication.
    end if
    $\boldsymbol{X}^{\ell+1} \leftarrow \sigma\left(\boldsymbol{X}^{\ell+1}(\boldsymbol{W}^\ell)^T + \boldsymbol{1}_N(\boldsymbol{b}^\ell - \boldsymbol{c})^T\right)$    ▷ Pass $\boldsymbol{X}$ through layer $\ell$ with extra bias $\boldsymbol{c}$
    $\boldsymbol{C}^{\ell+1} \leftarrow \sigma\left(\boldsymbol{H} - \boldsymbol{c}^T\right)$    ▷ Pass $\boldsymbol{C}$ through layer $\ell$ with extra bias $\boldsymbol{c}$
end for
Return $\boldsymbol{X}^L$ to evaluate loss and perform back-propagation
```

starting with $\boldsymbol{z}^0 = \boldsymbol{x}$. Here $\boldsymbol{W}^\ell$ and $\boldsymbol{b}^\ell$ denotes the weight matrix and the bias vector for layer ℓ, and σ is an activation operator that applies an element-wise nonlinear activation function. One popular choice for σ is the Rectified Linear Unit (ReLU) (Glorot et al., 2011) that takes the elementwise maximum between its entry and 0. The parametrization of $\boldsymbol{W}^\ell, \boldsymbol{b}^\ell$ controls the type of layer, e.g., circulant matrix for convolutional layer.

Let $\boldsymbol{S}$ be a DNN with L layers and parameters $\{\boldsymbol{W}^\ell, \boldsymbol{b}^\ell\}_{\ell=1}^L$. $\boldsymbol{S}$ employs continuous piecewise affine (CPA) activation σ at each layer, i.e., layer ℓ outputs are given by Eq. 1, with $\boldsymbol{z}^0$ equal to input $\boldsymbol{x} \in \mathbb{R}^S$.

The layer 1 to ℓ composition of a DNN $\boldsymbol{S}$, denoted as $\boldsymbol{S}^\ell$ with output space $\mathbb{R}^\ell$, can be expressed as:

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

with indicator function $\mathbb{1}_{\{.\}}$ and per-region affine parameters given by,

$$\boldsymbol{A}^\ell_\omega = \prod_{i=1}^{\ell} \text{diag}\left(\boldsymbol{q}^i_\omega\right) \boldsymbol{W}^i, \tag{3}$$

$$\boldsymbol{b}^\ell_\omega = \text{diag}\left(\boldsymbol{q}^\ell_\omega\right) \boldsymbol{b}^\ell + \sum_{i=1}^{\ell-1} \left(\prod_{j=i+1}^{\ell} \text{diag}\left(\boldsymbol{q}^j_\omega\right) \boldsymbol{W}^j \right) \text{diag}\left(\boldsymbol{q}^i_\omega\right) \boldsymbol{b}^i. \tag{4}$$

Here, $\boldsymbol{q}^\ell_\omega$ is the point-wise derivative of σ at pre-activation $\boldsymbol{W}^\ell \boldsymbol{z}^{\ell-1} + \boldsymbol{b}^\ell$, and diag$(.)$ operator given a vector argument creates a matrix with the vector values along the diagonal. As a consequence of Thm. 1 from Balestriero & Baraniuk, $\boldsymbol{q}^\ell_\omega$ is unique for any region $\omega \in \Omega$.

Such formulations of DNNs have previously been employed to make theoretical studies amenable to actual DNNs, e.g. in generative modeling (Humayun et al., 2022a;b), DNN complexity analysis (Hanin & Rolnick, 2019) and network pruning (You et al., 2021). The spline formulation of DNNs allow leveraging the rich literature on spline theory, e.g., in approximation theory (Cheney & Light, 2009), optimal control (Egerstedt & Martin, 2009), statistics (Fantuzzi et al., 2002) and related fields.

3. Provable Constraints on the Decision Boundary

Suppose, $\boldsymbol{w}^\ell_i$,b^ℓ_i are the i-th rows of $\boldsymbol{W}^\ell, \boldsymbol{b}^\ell$. Therefore, there exists a hyperplane $h^\ell_i \in \mathbb{R}^{\ell-1}$ from layer ℓ with parameters $\boldsymbol{w}^\ell_i, b^\ell_i$, expressed as,

$$h^\ell_i \triangleq \{\boldsymbol{z} \in \mathbb{R}^{\ell-1} : \langle \boldsymbol{w}^\ell_i, \boldsymbol{z} \rangle + b^\ell_i = 0\}. \tag{5}$$

Let, $\boldsymbol{S}$ is a binary classifier DNN therefore with a single output neuron with sigmoid activation. In the input space of the last layer, $\mathbb{R}^{L-1}$, the decision boundary is also a hyperplane h^L_1, that can be expressed as:

$$h^L_1 \triangleq \{\boldsymbol{z} \in \mathbb{R}^{L-1} : \langle \boldsymbol{w}^L_1, \boldsymbol{z} \rangle + b^L_1 = 0\}. \tag{6}$$

Suppose we have an arbitrarily ordered set of vertices $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots \boldsymbol{v}_p]^T$ in the input space of the network. The vertices

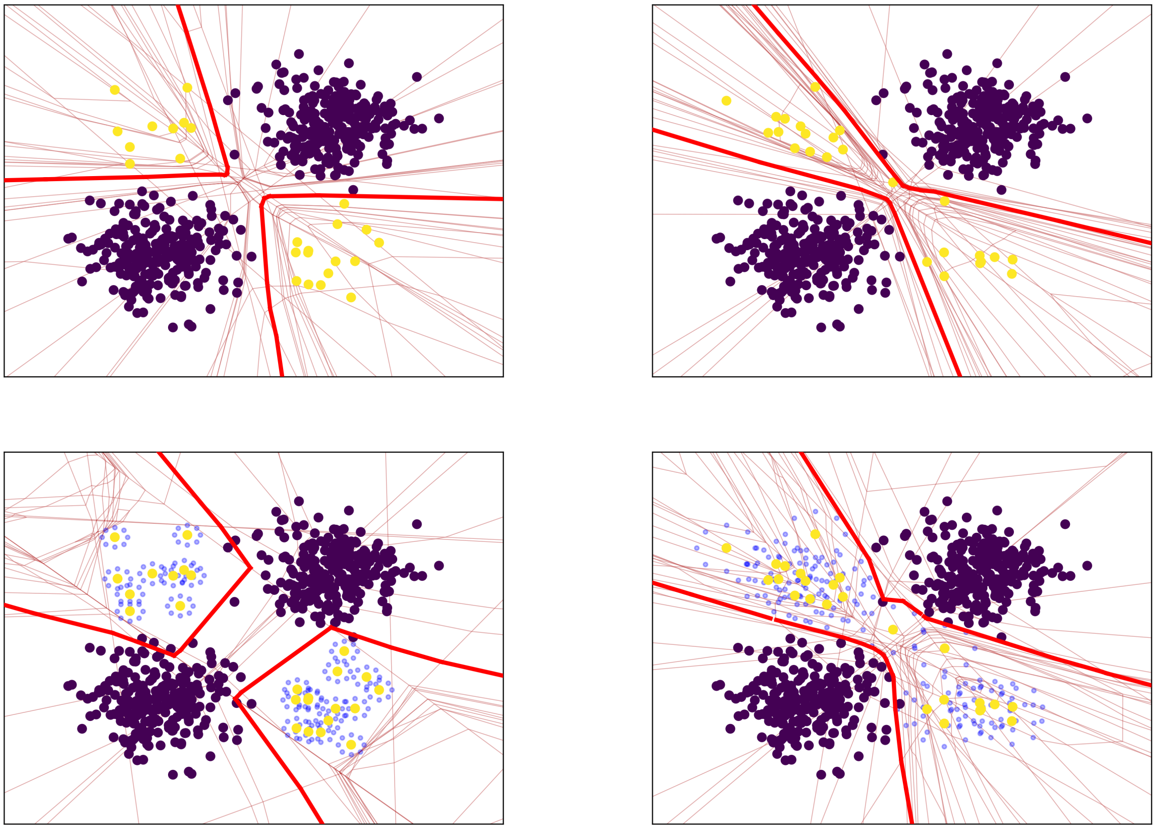

Figure 3: SplineCam visualizations of the DNN decision boundary (red) and neurons for a 2D toy binary classification task with four Gaussian distributions. One of the distributions (yellow) is under sampled with the ratio $20 : 1$. **Top row** shows models trained without constraints. **Bottom row** presents visualizations for DNNs trained with constraints. **Bottom left** presents results for smaller ϵ constraints and constraints applied to every layer. **Bottom right** presents constraints applied to only the decision boundary with larger ϵs.

lie on the surface of a sphere with radius ϵ centered on the input vector x.

Consider the set of pre-activation signs of the output neuron:

$$\{sign(\langle \boldsymbol{w}_1^L, \boldsymbol{S}^{L-1}(\boldsymbol{v})\rangle + b_1^L) : \forall \boldsymbol{v} \in \boldsymbol{V}\}. \tag{7}$$

Therefore, a sufficient condition for the decision boundary to not intersect the convex hull of $\boldsymbol{V}$ in the input space, is for all the elements of the set in Eq. 7 to be the same. This is a direct result of Thm 1. by Balestriero & LeCun.

The above discussions give us the necessary framework to define robustness for an input sample x and provide the following Theorem for robustness.

Definition 1. A binary classifier DNN $\boldsymbol{S}$ is ϵ-robust for input x if the minimum ℓ_2 distance from x to the decision boundary, i.e., the margin, is lower bounded by ϵ.

Theorem 1. Let, there exists a S-polytope containing an arbitrary input vector $x \in \mathbb{R}^S$, with vertices $\boldsymbol{V} = \{\boldsymbol{v}_i\}_{i=1}^p$ where $p > S$. A binary classifier $\boldsymbol{S}$ is at least ϵ-robust if the following is satisfied,

$$||v_i - x||_2 > \epsilon \qquad \forall \boldsymbol{v}_i \in \boldsymbol{V}, \tag{8}$$

$$|\sum_{i=1}^{p} sign(\langle \boldsymbol{w}_1^L, \boldsymbol{S}^{L-1}(\boldsymbol{v_i})\rangle + b_1^L)| = p, \tag{9}$$

where, $|.|$ is the absolute value operation.

The proof of Thm 1. is direct, since Eq. 9 ensures that the decision boundary in the input space of the final layer $\mathbb{R}^{L-1}$ does not intersect the S-polytope embedded in $\mathbb{R}^{L-1}$ and Eq. 8 ensures that all the vertices of the polytope are at least ϵ distance away.

Our proposed algorithm, *CROP* can be summarized as follows. To make a network epsilon robust for a given sample, we first define a polytope in terms of its vertices, such that the vertices are at least ϵ distance away from x. We call these the constraint vertices. Following that we compute the

Table 1: Performance comparison between our proposed method and standard adversarial training methods. We pre-train an MLP on MNIST without constraints and then fine-tune the network for 5 epochs using our proposed constraints or with adversarial training. We use the least robust class for the pre-trained model (digit 9) as our target class.

	Evaluation	Pre-training	Fine-tuning (5 epochs)					
			Ours Constr.	Gaussian w\ RandSmooth	FGSM	PGD-10	PGD-100	TPGD-100
All Class	Clean Train	**100**	91.28	99.44	99.65	99.28	99.12	99.19
	Clean Test	**97.88**	90.18	97.02	97.05	96.83	96.66	96.73
Target Class	Clean Test	97.52	**99.81**	98.21	98.91	99.11	99.01	98.91
	FGSM	68.88	**93.65**	68.28	89.39	92.66	92.47	92.66
	PGD-10	40.93	**88.61**	65.23	77.66	86.22	85.47	84.55
	PGD-100	40.83	**88.40**	64.73	77.26	85.88	84.87	83.98
	TPGD-100	44.79	**91.77**	65.74	81.80	91.06	88.22	88.82
	AutoAttack	28.34	**86.81**	69.18	72.25	82.95	81.56	81.57

pre-activations for each of these vertices for the neuron we want to constrain (e.g., the output neuron). We compute the majority vote of the signs of the pre-activation and add an extra bias to the learned bias of the neuron to ensure that the signs of the pre-activation for all the vertices are the same. We perform this during training for every iteration while the network is being trained. This way, for every forward pass, it is ensured that the constrained vertices have the same sign. Our method can be applied to constrain any neuron, therefore ensuring that features are also robust to ϵ perturbation of the input. In Algorithm 1, we present our proposed algorithm for sample specific robust training. Note that we can easily apply multiple constraints on multiple samples as well, as long as we want the signs for all the constraints to be identical. Our proposed method can be viewed as a method of constraining the continuous piecewise affine spline learned by the DNN (as discussed in Sec. 2.2) to be linear in a prescribed region for either the whole network or only the layer we wish to constrain (e.g., output layer).

4. Evaluation

Qualitative Validation. We start evaluation of our method by qualitative validation. To qualitatively validate the effect of CROP, we us SplineCam (Humayun et al., 2023) to visualize the exact decision boundaries of the network on a 2D subspace of the input space. SplineCam computes the decision boundary analytically therefore it does not perform any approximations, allowing us to qualitatively assess the effect of the constraints on the decision boundary. In Fig. 1 and Fig. 3 we present SplineCam visualizations on toy 2D tasks with imbalanced data. We present results for constraints applied on all the layers as well as only the decision boundary. We see that applying the constraints visibly shifts the decision boundary to ensure that the constraints are met. In Fig. 2, we present qualitative visualizations of the constraint applied on a single sample from MNIST. We use 20 vertices to define a constraint region around a training sample, as well as generate an adversarial sample using the same training sample. We also take the nearest neighbor to the training sample in the training dataset, that belongs to class the adv. sample is misclassified to by the network. Using these three samples, we define a square 2D input domain in the input space of the network and compute the spline partition induced by the network via SplineCam. This allows us to visualize individual neurons that intersect the 2D plane as well as the decision boundary. We see that applying constraint visibly shifts the decision boundary as well as intermediate layer neurons when all the layers are constrained. This is visual proof that even with a smaller number of constrain points, CROP can increase the margin in a locality of the input space.

Quantitative Validation. We quantitatively validate the performance of CROP on a single class robustness task on MNIST. We first start by training a 5 layer MLP with width 64 on MNIST. We train the network until interpolation without any constraints. We refer to this as the base network. After training, we perform PGD attack and find that the most susceptible class is the digit 9. Choosing the digit 9 as our target class, we fine-tune the base network for 5 epochs with a learning rate of 10^{-5} using CROP constraints or with standard adversarial training methods. We apply CROP on the pre-trained model and fine-tune for 3 training epochs. We use 50 vertices, an ℓ_2 distance of 1 away from each training sample from the digit 9 class to define the constrained regions. Following this we remove the constraint and fine-tune the network on the original training dataset for 2 epochs as a cooldown step.

In Table 1 we present performance of CROP for single class robustness and compare with standard adversarial attacks and defenses. For all the adversarial attacks, we use ℓ_∞ as the distance metric with a step size of $2/255$, attack epsilon of $8/255$. We use FGSM (Goodfellow et al., 2014) PGD with 10 and 100 steps (Madry et al., 2017), TPGD with 100 steps (Zhang et al., 2019) and AutoAttack (Croce & Hein, 2020) to measure the single class robustness of CROP. For defense, we fine-tune the base network on the training set with only the target class samples augmented via adversarial attacks. We use FGSM, PGD and TPGD during adversarial training in this setting. We also compare with Randomized Smoothing (Cohen et al., 2019), for which we first fine-tune the base model with Gaussian noise (variance 1) augmentation for the target class. During inference, we evaluate the fine-tuned network on 50 noisy versions of each test sample and do a majority vote.

We see that even with 5 fine-tuning epochs, CROP surpasses all the other adversarial defense methods on the single class robustness task. This is possible because CROP constraints are analytic and can be instantaneously applied, whereas the adversarial training methods require longer training runs to acquire robustness. We see that PGD based adversarial training methods acquire robustness performance closest to CROP. The overall test accuracy for CROP is lower compared to others.

5. Conclusion and future works

We provide a provable way to constraint samples to be robust towards adversarial attacks. Our proposed method *CROP*, can be applied to all samples from a specific class to ensure robustness towards targeted attacks, as well as, ensure robustness for minority classes in an imbalance dataset. CROP also ensures robustness against data poisoning attacks since the constraints are always ensured regardless of training. Future work includes applying constraints to all the classes in an iterative manner where the constraints are applied by alternating target classes.

Acknowledgements

Humayun, Casco-Rodriguez and Baraniuk were supported by NSF grants CCF1911094, IIS-1838177, and IIS-1730574; ONR grants N00014- 18-12571, N00014-20-1-2534, and MURI N00014-20-1-2787; AFOSR grant FA9550-22-1-0060; and a Vannevar Bush Faculty Fellowship, ONR grant N00014-18-1-2047.

References

Balestriero, R. and Baraniuk, R. A spline theory of deep networks. In *Proc. ICML*, pp. 374–383, 2018.

Balestriero, R. and LeCun, Y. Police: Provably optimal linear constraint enforcement for deep neural networks. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023.

Cheney, E. W. and Light, W. A. *A Course In Approximation Theory*, volume 101. American Mathematical Soc., 2009.

Cohen, J., Rosenfeld, E., and Kolter, Z. Certified adversarial robustness via randomized smoothing. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pp. 2206–2216. PMLR, 2020.

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

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

Glorot, X., Bordes, A., and Bengio, Y. Deep sparse rectifier neural networks. In *AISTATS*, pp. 315–323, 2011.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Hanin, B. and Rolnick, D. Complexity of linear regions in deep networks. *arXiv preprint arXiv:1901.09021*, 2019.

Humayun, A. I., Balestriero, R., and Baraniuk, R. MaGNET: Uniform sampling from deep generative network manifolds without retraining. In *ICLR*, 2022a. URL https://openreview.net/forum?id=r5qumLiYwf9.

Humayun, A. I., Balestriero, R., and Baraniuk, R. Polarity sampling: Quality and diversity control of pre-trained generative networks via singular values. In *CVPR*, pp. 10641–10650, 2022b.

Humayun, A. I., Balestriero, R., Balakrishnan, G., and Baraniuk, R. G. Splinecam: Exact visualization and characterization of deep network geometry and decision boundaries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3789–3798, June 2023.

Kim, J., Jeong, J., and Shin, J. Imbalanced classification via adversarial minority over-sampling. 2019.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 506–519, 2017. ISBN 9781450349444. doi: 10.1145/3052973.3053009. URL https://doi.org/10.1145/3052973.3053009.

Schott, L., Rauber, J., Bethge, M., and Brendel, W. Towards the first adversarially robust neural network model on MNIST. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1EHOsC9tX.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2014.

Wu, T., Liu, Z., Huang, Q., Wang, Y., and Lin, D. Adversarial robustness under long-tailed distribution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8659–8668, 2021.

Xu, H., Liu, X., Li, Y., Jain, A., and Tang, J. To be robust or to be fair: Towards fairness in adversarial training. In *International Conference on Machine Learning*, pp. 11492–11501. PMLR, 2021.

You, H., Balestriero, R., Lu, Z., Kou, Y., Shi, H., Zhang, S., Wu, S., Lin, Y., and Baraniuk, R. Max-affine spline insights into deep network pruning. *arXiv preprint arXiv:2101.02338*, 2021.

Yuan, X., He, P., Zhu, Q., and Li, X. Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 30(9): 2805–2824, 2019. doi: 10.1109/TNNLS.2018.2886017.

Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., and Jordan, M. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pp. 7472–7482. PMLR, 2019.

Zhao, W., Li, H., Wu, L., Zhu, L., Zhang, X., and Zhao, Y. Robustness of classifier to adversarial examples under imbalanced data. In *2022 7th International Conference on Computer and Communication Systems (ICCCS)*, pp. 156–161, 2022. doi: 10.1109/ICCCS55155.2022.9846074.