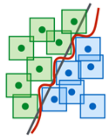
Large Norms of CNN Layers Do Not Hurt Adversarial Robustness

Youwei Liang, Dong Huang
SCAU

This work discusses the connections of

- **adversarial robustness** of neural nets, and
- their **Lipschitz constants**, and
- the **norms** of convolutional layers.

Adversarially robust classifiers are *provably* realizable using neural nets.



Assumption 2 (2-epsilon separable). *The data points of any two different classes are 2-epsilon separable:* $\inf\{d(x^{(i)}, x^{(j)}) : x^{(i)} \in \mathcal{X}^{(i)}, x^{(j)} \in \mathcal{X}^{(j)}, i \neq j\} > 2\epsilon$.

Theorem 2 (Realizability of robust classifiers). *Let* $\rho : \mathbb{R} \to \mathbb{R}$ *be any non-affine continuous function which is continuously differentiable at at least one point, with nonzero derivative at that point. If Assumption 2 holds, then there exists a feedforward neural network with* ρ *being the activation function that has robust accuracy* 1.

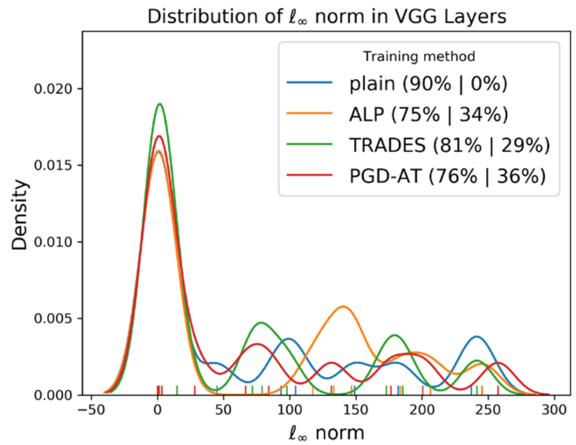
Robust classifiers need not have small Lipschitz constants.

Proposition 1. *There exists a feedforward network with ReLU activation where the norms of all layers can be arbitrarily large while the Lipschitz constant of the network is 0.*

Proposition 2. *Let* $\rho : \mathbb{R} \to \mathbb{R}$ *be any non-affine continuous function which is continuously differentiable at at least one point, with nonzero derivative at that point. If Assumption 2 holds, then for all* $\xi > 0$*, there exists a feedforward neural network with* ρ *being the activation function that achieves robust accuracy* 1 *and its Lipschitz constant is at least* ξ.

Our theories and experiments refute the argument that large norms of neural net layers are bad for adversarial robustness.

Robust classifiers need not have small layer norms.



We provide a theorem to compute and regularize (with a norm-decay algorithm) the layer norms of CNNs.

Theorem 1. *Suppose Assumption 1 holds. Then the* ℓ_1 *norm and* ℓ_∞ *norm and an upper bound of the* ℓ_2 *norm of conv are given by*

$$\|\operatorname{conv}\|_1 = \max_{1\le j\le d_{in}} \max_{\mathcal{A}\in\mathcal{S}} \sum_{(k,t)\in\mathcal{A}} \sum_{i=1}^{d_{out}} |K_{i,j,k,t}|, \tag{1}$$

$$\|\operatorname{conv}\|_\infty = \max_{1\le i\le d_{out}} \sum_{j=1}^{d_{in}} \sum_{k=1}^{k_1} \sum_{t=1}^{k_2} |K_{i,j,k,t}|, \tag{2}$$

Algorithm 1 Norm Decay

Input: loss function $\mathcal{L}$ (assuming it is to be minimized), parameters θ, momentum γ, regularization parameter β
Output: parameters θ
1: $h \leftarrow \mathbf{0}$ (initialize the gradient of norms of layers)
2: **repeat**
3: $\quad g \leftarrow \nabla_\theta \mathcal{L}$
4: $\quad$ Compute p, the gradient of ℓ_1 or ℓ_∞ norm of each fully-connected and convolutional layer
5: $\quad h \leftarrow \gamma \cdot h + (1-\gamma)\cdot p$
6: $\quad g \leftarrow g + \beta/N \cdot h$
7: $\quad \theta \leftarrow \mathrm{SGD}(\theta, g)$
8: **until** convergence

Norm-regularization tends to hurt robust accuracy.

		plain	weight decay				singular value clipping				ℓ_1 norm decay				ℓ_∞ norm decay			
model	ACC	—	10^{-2}	10^{-3}	10^{-4}	10^{-5}	0.5	1.0	1.5	2.0	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
vgg	Clean	90.4	91.6	91.7	90.1	90.2	87.6	89.1	90.0	89.9	88.1	91.1	90.6	90.8	91.8	91.1	90.8	90.6
	Robust	60.2	56.3	60.5	60.6	60.3	48.8	52.2	54.1	56.7	56.5	62.5	61.1	60.1	56.9	60.0	60.8	60.1
resnet	Clean	93.2	94.3	94.1	93.1	92.7	93.6	94.0	94.2	93.8	92.5	93.4	93.5	93.4	93.0	93.8	93.1	93.0
	Robust	37.0	28.2	33.7	33.9	40.9	35.2	41.7	43.2	39.8	24.5	37.7	38.3	37.5	20.0	34.7	38.9	37.6
senet	Clean	93.1	94.2	93.9	93.0	92.4	93.8	94.2	93.8	94.2	92.3	93.8	93.3	93.3	93.0	93.6	92.8	93.2
	Robust	35.7	23.5	32.8	37.0	34.8	30.5	35.6	35.2	37.4	33.6	36.0	38.2	36.7	28.6	31.0	37.6	37.4
regnet	Clean	91.8	93.6	94.4	92.3	91.3	93.9	93.4	93.0	92.4	93.7	92.3	91.6	91.9	93.4	92.0	91.8	91.9
	Robust	34.8	23.7	30.3	30.0	31.0	27.7	28.8	29.0	28.8	29.2	31.1	28.1	34.3	23.2	27.7	27.9	30.6

https://arxiv.org/abs/2009.08435