Adversarial Robustness of Convolutional Networks

Fergus O'Hanlon
Mathematical Institute
University of Oxford
Oxford, OX2 6GG
fergus.ohanlon@st-annes.ox.ac.uk

Abstract

Convolutional Neural Networks (CNNs) have been immensely successful in the past few decades. In particular, the introduction of Residual Networks (ResNets) was significantly influential. However, one of the flaws with CNNs is that they are susceptible to adversarial attacks. It is known that there is a relationship between natural accuracy and robust accuracy. Furthermore, increasing the width of ResNet is known to increase performance. After presenting "Convolutional Normalization", we explore the effect of normalization techniques on natural and robust accuracy and investigate their relationship with width for ResNet.

1 Introduction

Convolutional Neural Networks (CNNs) have been a major focus in the field of deep learning since the introduction of LeNet-5 [1]. Major breakthroughs in image, video, speech, and audio processing have been made in the past couple of decades with CNNs [2]. Since then, it has been found that certain small perturbations to a neural network's input can cause it to misclassify the output [3]. Such perturbations are found by optimizing the input to maximise the prediction error and the perturbed examples are so called "adversarial examples". Normalization plays an important role in modern network architecture. For example, batch normalization (BN) [4] is an essential component for training deep CNNs. Convolutional Normalization (ConvNorm) is a new technique specific for CNNs, which this report aims to present and explore.

2 Framework

2.1 Deep Networks

A deep network $f(\cdot)$ of L layers is a composition of a series of mappings: $f(\boldsymbol{x}) = f^{L-1} \circ \cdots \circ f^0(\boldsymbol{x})$. Each layer $f^\ell(\cdot)$ consists of a linear transforma $\mathcal{A}^\ell(\cdot)$ followed by a nonlinear activation function $\varphi(\cdot)$. The output of each layer is given by the following recursion:

$$\boldsymbol{z}^{\ell+1} := f^\ell(\boldsymbol{z}^\ell) = \varphi \circ \mathcal{A}^\ell(\boldsymbol{z}^\ell)$$

for $\ell = 0, 1, \ldots, L-1$, with $\boldsymbol{z}_0 = \boldsymbol{x}$. The goal of deep learning is to train a network which can be captured by the following minimisation problem:

$$\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathcal{L}\left(\boldsymbol{\theta}; \left\{\boldsymbol{x}^i, \boldsymbol{y}^i\right\}_{i=1}^m\right) := \frac{1}{m} \sum_{i=1}^m \ell\left(f(\boldsymbol{x}^i, \boldsymbol{\theta}), \boldsymbol{y}^i\right)$$

where $\boldsymbol{\theta}$ denotes the networks parameters in $\left\{\mathcal{A}^\ell(\cdot)\right\}_{\ell=0}^{L-1}$ and ℓ is a chosen loss function.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

2.2 Convolutional Neural Networks and Convolution Operators

CNNs consist of convolutional layers which implement the linear transform by convolving the input. Suppose that the input data $\boldsymbol{x}$ has C output channels, $\boldsymbol{x} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_C)$. The linear operator for the ℓ'th layer, $\mathcal{A}^\ell(\cdot) : \mathbb{R}^{C_\ell \times m} \mapsto \mathbb{R}^{C_{\ell+1} \times m}$ is a convolution operation with $C_{\ell+1}$ output channels,

$$\boldsymbol{z}^{\ell+1} = \left(\boldsymbol{z}_1^{\ell+1}, \cdots, \boldsymbol{z}_{C_\ell+1}^{\ell+1}\right), \quad \boldsymbol{z}_k^{\ell+1} = \varphi\left(\sum_{j=1}^{C_\ell} \boldsymbol{a}_{kj}^\ell * \boldsymbol{z}_j^\ell\right) (k = 1, \ldots, C_{\ell+1})$$

Now, since any linear convolution can be reduced to a circular convolution via zero-padding, we consider $*$ as a circular convolution such that

$$\boldsymbol{y} = \boldsymbol{a} * \boldsymbol{x} = \boldsymbol{C_a} \cdot \boldsymbol{x}$$

where $\boldsymbol{C_a} \in \mathbb{R}^{C \times C}$ is a circulant matrix whose first column is $\boldsymbol{a}$. $C_{\boldsymbol{a}}$ can be decomposed via the discrete Fourier Transform (DFT) matrix $\boldsymbol{F}$:

$$\boldsymbol{C_a} = \boldsymbol{F}^* \operatorname{diag}(\widehat{\boldsymbol{a}})\boldsymbol{F}, \quad \widehat{\boldsymbol{a}} = \boldsymbol{F}\boldsymbol{a}$$

This means that the convolution $\boldsymbol{a} * \boldsymbol{x}$ can be computed efficiently in the frequency domain using the fast fourier transform (FFT).

2.3 Adversarial Attacks

2.3.1 Fast Gradient Sign Method

The following method to generate adversarial examples is referred to as the "fast gradient sign method" (FGSM) [5]:

$$\tilde{\boldsymbol{x}} := \epsilon \operatorname{sign}(\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, y))$$

Where $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, y)$ is the cost, $\boldsymbol{\theta}$ is the parameter vector, and y is the target associated with $\boldsymbol{x}$.

2.3.2 Projected Gradient Descent

Where FGSM is a one-step scheme, [6] established a more powerful, multi-step scheme by considering projected gradient descent (PGD) on the negative loss function:

$$\boldsymbol{x}^{t+1} = \Pi_{x+\mathcal{S}}\left(\boldsymbol{x}^t + \alpha \operatorname{sign} \nabla_{\boldsymbol{x}} \mathcal{L}(\theta, \boldsymbol{x}, \boldsymbol{y})\right)$$

where $\Pi_{x+\mathcal{S}}$ denotes projecting perturbations into the set $\mathcal{S}$ and α is the step size.

3 Convolutional Normalization

ConvNorm was proposed by [7] which exploits the convolution structures of CNNs and exploits translation-invariance properties of convolutional operators. The technique improves training and robustness through reducing the network's Lipschitzness [8] by normalizing each output channel using preconditioning.

To introduce this method, consider a vanilla CNN using 1-stride and a layer ℓ that has C_I and C_O input and output channels, respectively. The output of the k-th channel before the nonlinear activation can be written as (omitting the layer number ℓ):

$$\boldsymbol{z}_{k,out} = \sum_{j=1}^{C_I} \boldsymbol{a}_{kj} * \boldsymbol{z}_{j,in} = \begin{bmatrix}\boldsymbol{C}_{\boldsymbol{a}_{k1}} & \boldsymbol{C}_{\boldsymbol{a}_{k2}} & \ldots & \boldsymbol{C}_{\boldsymbol{a}_{kC_I}}\end{bmatrix} \cdot \begin{bmatrix}\boldsymbol{z}_{1,in} \\ \boldsymbol{z}_{2,in} \\ \vdots \\ \boldsymbol{z}_{C_I,in}\end{bmatrix} = \boldsymbol{A}_k \cdot \boldsymbol{z}_{in}$$

where $\boldsymbol{A}_k := \begin{bmatrix}\boldsymbol{C}_{\boldsymbol{a}_{k1}} & \boldsymbol{C}_{\boldsymbol{a}_{k2}} & \ldots & \boldsymbol{C}_{\boldsymbol{a}_{kC_I}}\end{bmatrix}$ and $\boldsymbol{z}_{in} := \begin{bmatrix}\boldsymbol{z}_{1,in} & \boldsymbol{z}_{2,in} & \ldots & \boldsymbol{z}_{C_I,in}\end{bmatrix}^\top$

The idea of convnorm is to multiply each output channel $\boldsymbol{z}_{k,out}$ $(k = 1, \ldots, C_O)$ by a preconditioning matrix $\boldsymbol{P}_k$, as defined in equation (1) below. This is inspired by [9] which showed that normalizing

the output $\boldsymbol{z}$ eliminates bad local minimzers and dramatically improves the optimization landscape for learning the kernel $\boldsymbol{a}$.

$$\boldsymbol{P}_k = \left(\sum_{j=1}^{C_I} \boldsymbol{C}_{\boldsymbol{a}_{kj}} \boldsymbol{C}_{\boldsymbol{a}_{kj}}^\top \right)^{-1/2} = \left(\boldsymbol{A}_k \boldsymbol{A}_k^\top \right)^{-1/2} \tag{1}$$

The resulting normalized output is thus given by

$$\widetilde{\boldsymbol{z}}_{k,out} := \boldsymbol{P}_k \boldsymbol{z}_{k,out} = (\boldsymbol{A}_k {\boldsymbol{A}_k}^\top)^{-1/2} \boldsymbol{A}_k \cdot \boldsymbol{z}_{in} = \boldsymbol{Q}_k(\boldsymbol{A}_k) \cdot \boldsymbol{z}_{k,in}$$

where $\boldsymbol{Q_k}(\boldsymbol{A}_k) := (\boldsymbol{A}_k {\boldsymbol{A}_k}^\top)^{-1/2} \boldsymbol{A}_k$. ConvNorm is essentially a reparameterization of the kernels $\{\boldsymbol{a}_{kj}\}_{j=1}^{C_I}$ for the k-th channel. Expanding the definition of $\boldsymbol{C}_{\boldsymbol{a}_{kj}}$ in equation (1) provides:

$$\begin{aligned} \boldsymbol{P}_k &= \left(\sum_{j=1}^{C_I} \boldsymbol{F}^* \operatorname{diag}(\widehat{\boldsymbol{a}}_{kj}) \boldsymbol{F} \boldsymbol{F}^* \operatorname{diag}(\overline{\widehat{\boldsymbol{a}}}_{kj}) \boldsymbol{F} \right)^{-1/2} \\ &= \boldsymbol{F}^* \left(\operatorname{diag} \left(\sum_{j=1}^{C_I} |(\widehat{\boldsymbol{a}}_{kj})|^{\odot 2} \right)^{\odot -1/2} \right) \boldsymbol{F} = \boldsymbol{F}^* \left(\operatorname{diag}(\boldsymbol{F} \boldsymbol{v}_k) \right) \boldsymbol{F} = \boldsymbol{C}_{\boldsymbol{v}_k} \end{aligned}$$

with $\boldsymbol{v}_k := \boldsymbol{F}^{-1} \left(\sum_{i=1}^{C_I} |\widehat{\boldsymbol{a}}_{ki}|^{\odot 2} \right)^{\odot -1/2}$ and $\odot$ denoting entrywise operation. Rewriting $\boldsymbol{Q}_k(\boldsymbol{A}_k)$ as:

$$\boldsymbol{Q}_k(\boldsymbol{A}_k) = \boldsymbol{P}_k \boldsymbol{A}_k = \begin{bmatrix} \boldsymbol{C}_{\boldsymbol{v}_k * \boldsymbol{a}_{k1}} & \cdots & \boldsymbol{C}_{\boldsymbol{v}_k * \boldsymbol{a}_{kC_I}} \end{bmatrix}$$

means that the reparameterization can be written in terms of convolutions which can be efficiently computed using FFT. Specifically, for each $j = 1, \ldots, C_I$:

$$\boldsymbol{C}_{v_k * \boldsymbol{a}_{kj}} = \boldsymbol{F}^* \operatorname{diag}(\widehat{\boldsymbol{g}}(\boldsymbol{a}_{kj})) \boldsymbol{F}, \quad \widehat{\boldsymbol{g}}(\boldsymbol{a}_{kj}) = \widehat{\boldsymbol{a}}_{kj} \odot \left(\sum_{i=1}^{C_I} |\widehat{\boldsymbol{a}}_{ki}|^{\odot 2} \right)^{\odot -1/2}$$

The matrix $\boldsymbol{A}_k$ can't be be normalized to exact orthogonal since it is overcomplete. However it can be normalized to tight frame with $\boldsymbol{Q}_k \boldsymbol{Q}_k^\top = \boldsymbol{I}$ and so $\|\boldsymbol{Q}_k\|$ can be normalized in each channel to unity.

Applying the normalization over every channel provides the overall ConvNorm normalization as follows:

$$\widetilde{\boldsymbol{z}}_{out} = \begin{bmatrix} \boldsymbol{P}_1 \boldsymbol{z}_{1,out} \\ \vdots \\ \boldsymbol{P}_{C_O} \boldsymbol{z}_{C_O,out} \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{Q}_1 \\ \vdots \\ \boldsymbol{Q}_{C_O} \end{bmatrix}}_{:= \boldsymbol{Q}} \boldsymbol{z}_{in} \tag{2}$$

Proposition 3.1. The spectral norm of $\boldsymbol{Q}$ induced in (2) can be bounded by

$$\|\boldsymbol{Q}\| \leq \sqrt{\sum_{k=1}^{C_O} \|\boldsymbol{Q}_k\|^2}$$

and so the ConvNorm actually reduces $\|\boldsymbol{Q}\|$, improving the Lipschitzness in each layer.

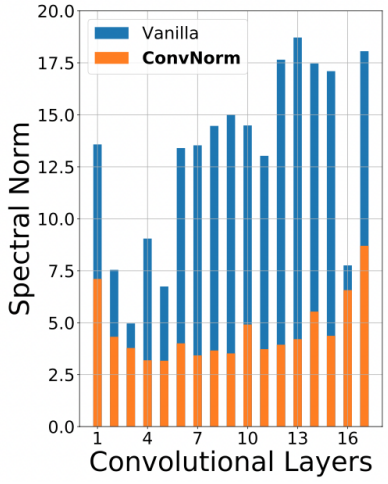

Figure 1: **Spectral norm for each layer on ResNet18** [7]. ConvNorm reduces the spectral norm

Proof.

$$\sigma_1^2(\boldsymbol{Q}) = \lambda_1 \left(\boldsymbol{Q}^\top \boldsymbol{Q} \right) = \lambda_1 \left(\sum_{i=1}^{C_O} \boldsymbol{Q}_i^\top \boldsymbol{Q}_i \right) \leq \sum_{i=1}^{C_O} \lambda_1 \left(\boldsymbol{Q}_i^\top \boldsymbol{Q}_i \right) = \sum_{i=1}^{C_O} \sigma_1^2(\boldsymbol{Q}_i)$$

Taking the square root of both sides provides the desired result. □

4 Experiments and Results

The ConvNorm repository [7] was imported and modified to implement the WideResNet architecture [10]. A new script was created based on the existing model for ResNet and adapted to implement the 'basic-wide' block [10]. Two normalization techniques were focused on, namely BN [4] and the previously discussed ConvNorm. Similar to [7], the training procedure described in [11] was used for robust training. Experiments were performed to investigate the effectiveness of the normalization techniques on WideResNet16 and explore how they performs against the width factor k.

For every experiment, the models were trained on the CIFAR-10 dataset [12] for 40 epochs using cross-entropy loss and AdamW [12] for optimization with an initial learning rate of 0.001. 10% of the training set is used for validation and the validation set is treated as a held-out test set. PGD and FGSM attacks are performed with total $\epsilon = 8/255$ where PGD is implemented with 10 steps of size $\alpha = 2/255$. Computations were executed with a NVIDIA Tesla P100 PCIe 16 GB GPU.

4.1 Adjusting to Computational Limitations

WideResNet is much more efficient than ResNet due to the parallel nature of GPU computations [10]. We utilize this computational discount and chose this architecture to increase our capacity to explore more networks. With limited computational resources, AdamW was chosen as an optimizer with the aim of reducing training time [13].

4.2 Results

4.2.1 Natural Accuracy

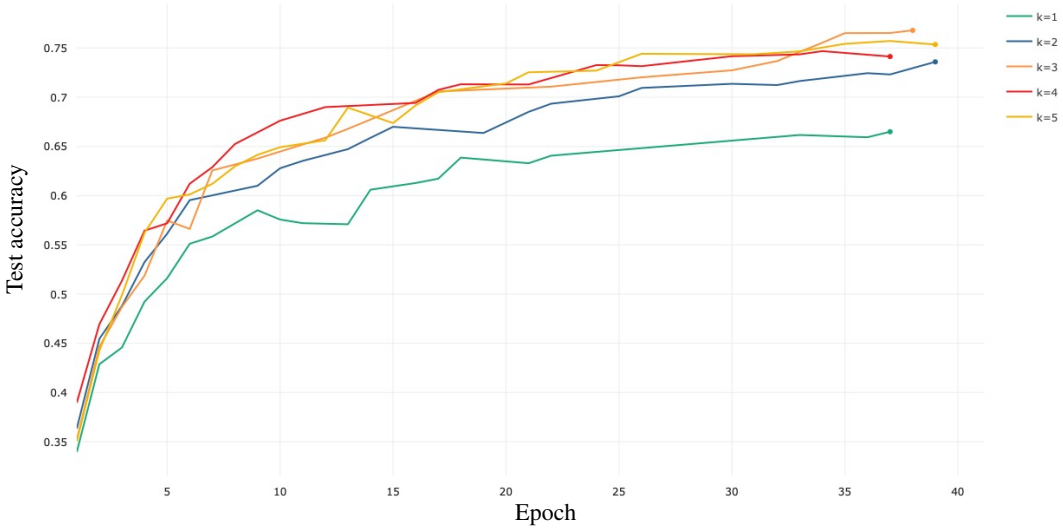

Figure 2: **Test accuracy of WideResNet-16 using various widening factors k.** Increasing the width of the model increases the natural accuracy. (Note: some of the data for the later epochs is missing due to upload rate limits of comet.ml)

4.2.2 Robust Accuracy

5 Discussion and Conclusion

Through the implentation of WideResNet, the relationship between ConvNorm and network width was explored. The empirical data suggests that ConvNorm improves natural accuracy but there is a significant decrease in robustness. While there may a trade-off between natural accuracy and robustness [14] [15], BN also increases natural accuracy but doesn't exhibit the a similar proportional drop in robustness. While these initial results may indicate that ConvNorm does not improve

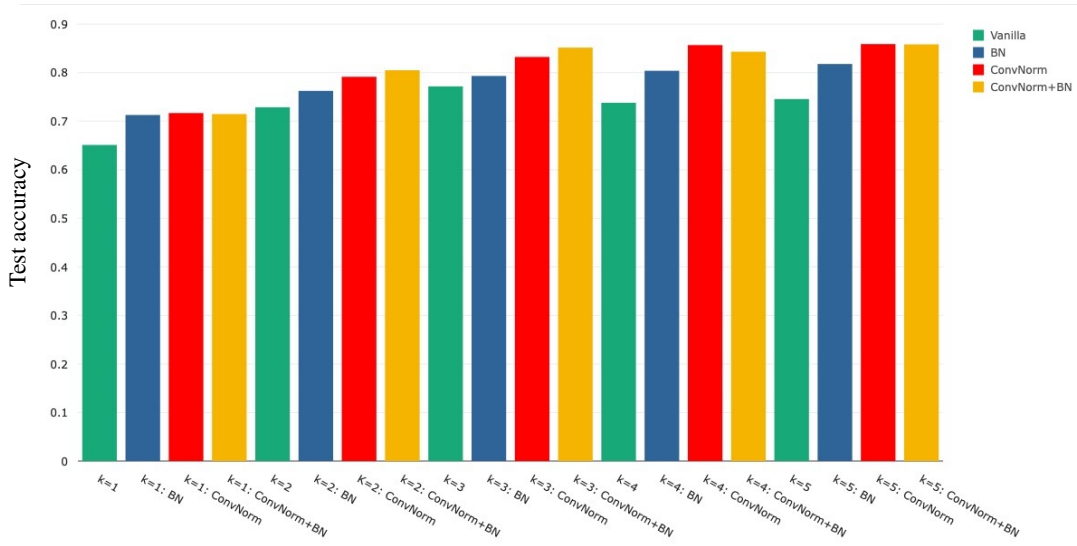

Figure 3: **Top-1 test accuracy of WideResNet-16-k with various normalization techniques.** ConvNorm and BN both improve the natural accuracy for increased widening factors.

Table 1: **Comparison of normalization methods under different adversarial attacks.**

Top-N	Attack	k	Vanilla	BN	ConvNorm	BN+ConvNorm
Precision@1	PGD	1	29.64	37.05	38.00	37.03
		2	35.23	38.09	41.72	1.17
		3	32.95	41.18	0.26	3.19
		4	34.33	40.67	8.03	1.31
		5	33.82	40.36	2.81	2.44
	FGSM	1	36.27	41.92	42.50	41.66
		2	47.30	44.20	47.41	75.73
		3	55.17	47.42	71.27	81.98
		4	43.21	48.80	66.40	83.58
		5	43.31	47.36	87.79	74.45
Precision@5	PGD	1	85.49	90.12	90.65	90.80
		2	88.93	91.47	91.93	63.53
		3	89.43	92.19	46.56	68.35
		4	87.55	92.29	78.07	56.49
		5	88.78	92.80	68.16	62.81
	FGSM	1	88.32	91.34	91.75	91.79
		2	92.57	92.45	93.28	98.57
		3	94.98	93.42	98.05	99.04
		4	90.81	93.78	97.80	99.29
		5	91.75	93.89	99.44	98.35

The model trained with ConvNorm exhibits an increase in robustness against FGSM attacks as the widen factor k increases. However, it performs badly against the stronger PGD attacks for larger widths. The robustness of the vanilla model fluctuates slightly for both attacks as the width increases. BN almost always results in an increase of robustness in comparison to the vanilla model.

robustness of WideResNet, further experiments are worthwhile before conclusion. Possible avenues of further investigation include training using: CIFAR-100, more epochs, SGD with a learning rate schedule.

References

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[3] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[4] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

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

[6] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[7] Sheng Liu, Xiao Li, Yuexiang Zhai, Chong You, Zhihui Zhu, Carlos Fernandez-Granda, and Qing Qu. Convolutional normalization: Improving deep convolutional network robustness and training. *arXiv preprint arXiv:2103.00673*, 2021.

[8] Robert M Gray. Toeplitz and circulant matrices: A review. 2006.

[9] Qing Qu, Xiao Li, and Zhihui Zhu. A nonconvex approach for exact and efficient multichannel sparse blind deconvolution. *arXiv preprint arXiv:1908.10776*, 2019.

[10] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[11] A Shafahi, M Najibi, MA Ghiasi, Z Xu, J Dickerson, C Studer, LS Davis, G Taylor, and T Goldstein. Adversarial training for free!" in advances in neural information processing systems. 2019.

[12] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[13] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[14] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

[15] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pages 7472–7482. PMLR, 2019.