Batch Normalization Increases Adversarial Vulnerability and Decreases Adversarial Transferability: A Non-Robust Feature Perspective Supplementary Material

Philipp Benz* pbenz@kaist.ac.kr
Chaoning Zhang* chaoningzhang1990@gmail.com
In So Kweon iskweon77@kaist.ac.kr

Korea Advanced Institute of Science and Technology (KAIST)

A. Relation to frequency perspective

Our work focuses on the feature perspective [7] to analyze the model robustness. A Fourier perspective on robustness is introduced in [15]. With the analysis of corruption analysis in Sec. 3.2, our explanation from the feature perspective is "noise corruptions mainly corrupt the NRFs while contrast and fog mainly corrupt the RFs". Their explanation from the frequency perspective can be summarized as: noise corruptions mainly corrupt the high-frequency (HF) features while contrast and fog mainly corrupt the low-frequency (LF). These two explanations align well with each other in the sense that NRFs are widely recognized to have HF property, which motivated the exploration of several defense methods [2, 9]. Moreover, our work is the first to demonstrate that the model learns the order from RFs to NRFs. Meanwhile, it has been shown in [14] that the model learns first LF component then HF component, which aligns well with our finding by perceiving NRFs having HF properties.

B. Experimental setup

B.1. Setup for training models in Sec. 3.2

The models for CIFAR10 and SVHN used in Sec. 3.2 were trained with SGD with the training parameters listed in Table A1. The ResNet50 models in Table 1 and Table 2 of the main manuscript were trained for 350 epochs with an initial learning rate of 0.1, which was decreased by a factor of 10 at epochs 150 and 250, while the other parameters are the same as before. For ImageNet, the VGG models were obtained from the `torchvision` library, while the ResNet models are trained with the same parameters as in [6].

*Equal contribution

Table A1. Parameters to train a standard model on CIFAR10/SVHN.

Parameter	Value
Learning rate	0.01
Batch size	128
Weight Decay	0.0005
Epochs	300
Learning rate decay epochs	200
Learning rate decay factor	0.1

Table A2. Training parameters for adversarial training for CIFAR10/SVHN.

Parameter	Value
Learning rate	0.01
Batch size	128
Weight Decay	0.0005
Epochs	150
Learning rate decay epochs	100
Learning rate decay factor	0.1
PGD-variant	l_2
PGD step size (α)	0.1
PGD perturbation magnitude (ϵ)	0.5
PGD iterations	7

B.2. PGD attack for evaluating the adversarial robustness

In Table 1, we evaluate the robustness of models with the l_2 and l_∞ variants of the PGD-attack [10] and Carlini & Wagner (CW) attack [1]. For the l_2 and l_∞ attack we use $\epsilon = 0.25$ and $\epsilon = 1/255$ for images within a pixel range of $[0, 1]$, respectively. The attacks are run for 20 iteration steps and we calculate the step size with 2.5ϵ/steps. For the CW-attack [1], we follow [16] to adopt the PGD approach with the CW loss and the same hyper parameters as above.

The robust accuracy values in all figures in Sec. 5 are obtained with l_2-PGD as above, but for 10 iteration steps on 1000 evaluation samples (100 samples per class) to reduce

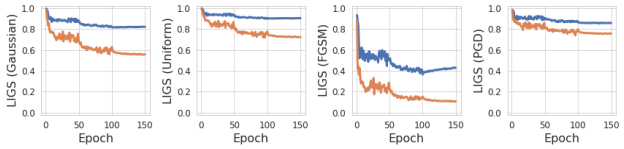

Figure A1. Trend of LIGS with different perturbations: Gaussian, Uniform, FGSM, PGD (left to right).

Table A3. Cross-evaluation of the features extracted from source models on target models with the baseline VGG16 for standard models.

Target \ Source	None	BN	IN	LN	GN
None	–	45.6	29.6	52.1	45.9
BN	**75.3**	–	35.8	58.9	54.3
IN	**66.1**	50.4	–	53.0	61.9
LN	**79.4**	59.4	37.9	–	63.4
GN	**73.3**	54.4	43.1	61.3	–

Table A4. Cross-evaluation of the features extracted from source models on target models with the baseline VGG16 for adversarially trained models.

Target \ Source	None	BN	IN	LN	GN
None	–	85.0	59.8	75.8	65.9
BN	81.3	–	58.1	73.7	62.5
IN	75.8	78.8	–	69.4	62.9
LN	82.9	84.5	60.0	–	64.8
GN	80.7	83.7	63.8	73.7	–

computation cost.

B.3. LIGS metric

By default, the LIGS values are calculated with ν being set to Gaussian noise with mean $\mu = 0$ and standard deviation $\sigma = 0.01$. In Figure A1 ν is set to either Gaussian noise, uniform noise in the range of $[-0.01, 0, 01]$, FGSM with $\epsilon = 0.01$ or l_∞-PGD with $\epsilon = 0.01$, 7 step iterations and a step size of 2.5ϵ/steps.

C. Feature Transferability

In Sec. 3.1 we formulated the conjecture, that BN shifts the model to rely more on NRFs instead of RFs and provided empirical evidence for this conjecture. Additional to the empirical evidence given in Sec. 3.2, we provide one additional piece of evidence via a feature transferability analysis. We extract the features out of the trained models as a new dataset and perform cross-evaluation on the remaining models (details in the supplementary). The results are shown in Table A3. As a control study, we perform the same analysis on adversarially trained robust models, see Table A4. It has been shown in [11] that robust models are superior to normally trained models for transfer learning, which demonstrates that RFs can transfer better. Here, we find that features extracted from robust models (right) can transfer better than the features extracted from standard models (left). For the normally trained models, we observe that the features extracted from the model w/o normalization transfer better (indicated in bold) than those models with BN/IN/LN/GN, especially IN. Recognizing the extracted features have both RF and NRFs, our observation suggests that the models with normalization rely more on NRFs than that w/o BN.

C.1. Extracting features as a dataset from a model

To demonstrate feature transferability in Table A3 and Table A4, we extract features from standard and adversarially trained models as a dataset. For robust models in Table A4 we follow the adversarial training strategy from [10] with l_2-PGD and we list the parameters in Table A2. We follow the procedure and hyperparameter choices in [7] and generate dataset $\hat{\mathcal{D}}$, given a model C:

$$\mathbb{E}_{(x,y)\sim\hat{\mathcal{D}}}[y \cdot f(x)] = \begin{cases} \mathbb{E}_{(x,y)\sim\mathcal{D}}[y \cdot f(x)] & \text{if } f \in F_C \\ 0 & \text{otherwise,} \end{cases} \tag{A1}$$

where F_C is the set of features utilized by C. The set of activations in the penultimate layer $g(x)$ corresponds to F_C in the case of DNNs. Thus, to extract the robust features from $\mathcal{C}$ we perform the following optimization:

$$\min_{\delta} ||g(x) - g(x' + \delta)||_2. \tag{A2}$$

The optimization is performed for each sample x from $\mathcal{D}$. Likewise, x' is drawn from $\mathcal{D}$ but with a label other than that of x. The optimization process is realized using the l_2 variant of PGD. We set the step size to 0.1 and the number of iterations to 1000 and we do not apply a restriction on the perturbation magnitude ϵ.

D. Influence of other normalization techniques on LIGS

In Fig. 1, we show the influence of BN on the robust accuracy and LIGS over the model training process. Additionally, Fig. A2 shows the results of repeating this experiment with IN, LN, and GN. Similar trends to those of BN are observed.

E. Results on ImageNet with LIGS trend

Fig. A3 shows the influence of normalization for models trained on ImageNet. It can be observed that the model with IN always exhibits the lowest accuracy, while the model with BN has the highest accuracy. Similar to the results on CIFAR10, BN/IN/LN/GN consistently leads to lower LIGS.

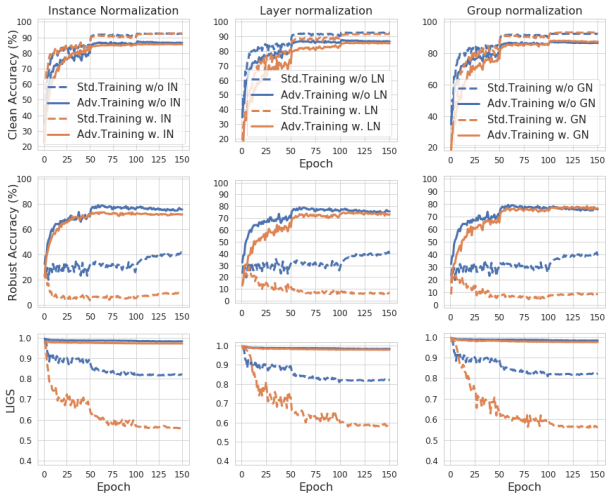

Figure A2. Trend of clean accuracy (top), robust accuracy (middle), LIGS (bottom) for ResNet18 on CIFAR10 with different normalization techniques (IN, LN, GN) applied.

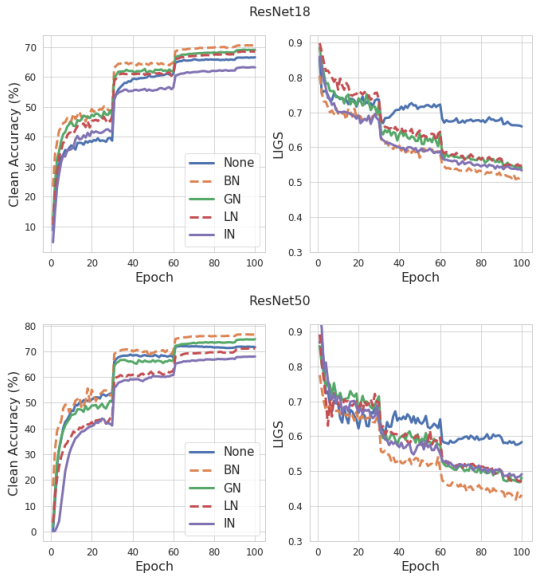

Figure A3. Comparison of different normalization techniques for ResNet18 (top) and ResNet50 (bottom) trained on ImageNet.

F. Description of $\widehat{\mathcal{D}}_R$ / $\widehat{\mathcal{D}}_{NR}$ / $\widehat{\mathcal{D}}_{rand}$ / $\widehat{\mathcal{D}}_{det}$

[7] introduced a methodology to extract feature datasets from models. In particular the datasets $\widehat{\mathcal{D}}_R$, $\widehat{\mathcal{D}}_{NR}$, $\widehat{\mathcal{D}}_{rand}$ and $\widehat{\mathcal{D}}_{det}$ were introduced, which we will describe here briefly. $\widehat{\mathcal{D}}_R$ indicates a dataset containing mainly RFs relevant to a robust model, and $\widehat{\mathcal{D}}_{NR}$ indicates that with standard model. During the extraction of $\widehat{\mathcal{D}}_{NR}$, the magnitude ϵ was not constraint, thus $\widehat{\mathcal{D}}_{NR}$ has both RFs and NRFs. $\widehat{\mathcal{D}}_{\text{rand}}$ and $\widehat{\mathcal{D}}_{\text{det}}$ are datasets consisting of "useful" NRFs represented through adversarial examples for a standard model. The target classes of $\widehat{\mathcal{D}}_{\text{rand}}$ were chosen randomly, while the ones

Table A5. Hyperparameters for training the extracted datasets.

Dataset	LR	Batch size	LR Drop	Data Aug.	Momentum	Weight Decay
$\widehat{\mathcal{D}}_R$	0.01	128	Yes	Yes	0.9	$5 \cdot 10^{-4}$
$\widehat{\mathcal{D}}_{NR}$	0.01	128	Yes	Yes	0.9	$5 \cdot 10^{-4}$
$\widehat{\mathcal{D}}_{rand}$	0.01	128	Yes	Yes	0.9	$5 \cdot 10^{-4}$
$\widehat{\mathcal{D}}_{det}$	0.1	128	Yes	No	0.9	$5 \cdot 10^{-4}$
$\widehat{\mathcal{D}}_{conflict}$	0.1	128	Yes	No	0.9	$5 \cdot 10^{-4}$

for $\widehat{\mathcal{D}}_{\text{det}}$ were selected with an offset of $t+1$ to the original sample ground-truth class. Note that these datasets are labeled with the target class for which the adversarial example was generated. We follow the procedure described in [7] and extract the datasets from a ResNet50 model. The hyperparameters used to train a model on one of the above datasets are listed in Table A5. We use SGD as an optimizer and train the models for 150 epochs with a learning rate decrease by a factor of 10 at epochs 50 and 100.

Fig. A4 shows the trends for training ResNet18 on $\widehat{\mathcal{D}}_R$, $\widehat{\mathcal{D}}_{NR}$ and $\widehat{\mathcal{D}}_{rand}$. As seen before, the model trained on $\widehat{\mathcal{D}}_R$ achieves a relatively high LIGS, while the model trained on $\widehat{\mathcal{D}}_{rand}$ exhibits relatively low LIGS values. The LIGS values for the model trained on $\widehat{\mathcal{D}}_{NR}$ are in the middle of $\widehat{\mathcal{D}}_R$ and $\widehat{\mathcal{D}}_{rand}$, which is expected because $\widehat{\mathcal{D}}_{NR}$ has RFs and NRFs.

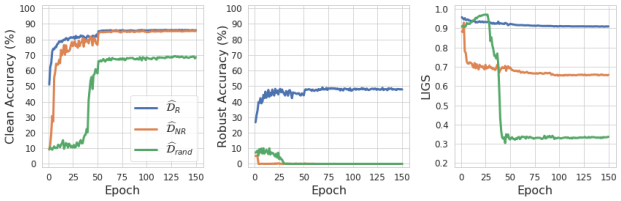

Figure A4. Comparison of $\widehat{\mathcal{D}}_R$, $\widehat{\mathcal{D}}_{NR}$ and $\widehat{\mathcal{D}}_{rand}$.

Fig. 6 shows the trends for training a ResNet18 on $\widehat{\mathcal{D}}_{Conflict}$, consisting of conflicting features. $\widehat{\mathcal{D}}_{Conflict}$ differs from $\widehat{\mathcal{D}}_{\text{det}}$ in that it draws x' from a robust dataset $\widehat{\mathcal{D}}_R$ instead of $\mathcal{D}$. The same experiment with $\widehat{\mathcal{D}}_{\text{det}}$ is shown in Fig. A5. The results resemble those of $\widehat{\mathcal{D}}_{Conflict}$. However, we used $\widehat{\mathcal{D}}_{Conflict}$ to avoid the influence of the NRFs in $\mathcal{D}$.

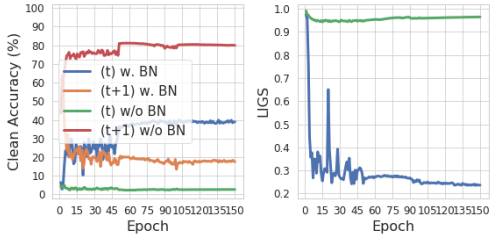

Figure A5. Result on $\widehat{\mathcal{D}}_{\text{det}}$.

G. Evaluating adversarial robustness with FGSM attack

Motivated by the (local) linearity assumption, [5] proposed the one-step FGSM attack. FGSM efficiently attacks the model but is not as effective as PGD attack [10] because the DNN is not fully linear. With iterative nature to overcome this linearity assumption, PGD is a very strong attack and de facto benchmark standard for evaluating the adversarial robustness, due to which PGD is adopted in our work. Here, we discuss the effect of BN with FGSM. BN reduces the LIGS value, which indicates the model with BN has low local linearity. Since the success of FGSM is highly dependent on the linearity assumption, the FGSM attack is conjectured to be less effective on the model with BN than w/o BN. This conjecture is supported by the results in the supplementary. As shown in Table A6, we find that with FGSM attack, the model with BN has higher adversarial robustness than that w/o BN.

Table A6. Robust accuracy comparison of models with and w/o BN under FGSM attack.

Network	Acc	FGSM 4/255
VGG11 (None)	90.06	32.51
VGG11 (BN)	92.48	40.86
VGG16 (None)	91.89	23.26
VGG16 (BN)	93.7	51.28
ResNet50 (None)	92.15	28.23
ResNet50 (BN)	95.6	38.07

H. Additional Transferability Results

In Section 7 we demonstrated that more strong transferable adversarial examples can be generated for models without BN. In Table A7 we demonstrate that adversarial examples generated on adversarially trained models transfer better than normal models. Compared to the ResNet50 model with BN, both adversarially trained models transfer better for all I-FGSM variants. Except for DI-FGSM, the adversarially trained models do also outperform the RN50 models without BN. The results for DI-FGSM and TI-FGSM for CIFAR10 are shown in Table A8. The results resemble the ones originally presented in Table 4 of the main manuscript.

In Figure 9 of the main manuscript, we demonstrated that the transferability of adversarial examples generated for a ResNet18 on CIFAR10 decreases with ongoing model training. In Figure A6 we provide the complementary result for a ResNet18 trained on ImageNet. For ResNet18 trained on ImageNet, we observe a similar trend as on CIFAR10. The transferability initially increases and then decreases gradually during training.

Table A7. Influence of BN on the transferability of robustly trained ResNet50 models. Results on ImageNet with various baselines: I-FGSM [8], MI-FGSM [3], DI-FGSM [13] and TI-FGSM [4].

	Variant	BN	RN50	DN121	VGG19	RN152	MN-V2	I-V3	Avg
I	Standard	Y	100	80.1	71.6	86.2	73.4	34.2	74.2
	Standard	N	98.6	94.3	87.0	95.5	94.4	72.1	90.3
	$L_2 = 3.0$	Y	98.9	98.6	94.6	98.3	98.1	96.5	97.5
	$L_\infty = 4$	Y	97.3	95.9	92.5	96.1	96.6	92.7	95.2
MI	Standard	Y	100	88.8	81.9	92.8	83.0	50.7	82.9
	Standard	N	98.9	95.4	88.7	95.5	96.2	78.5	92.2
	$L_2 = 3.0$	Y	97.6	97.1	92.9	96.7	97.6	94.8	96.1
	$L_\infty = 4$	Y	95.5	94.6	89.9	93.7	95.1	89.3	93.0
DI	Standard	Y	100	98.1	96.9	97.9	94.4	59.8	91.2
	Standard	N	99.4	99.1	95.8	98.1	98.8	90.3	96.9
	$L_2 = 3.0$	Y	98.0	97.5	91.9	96.3	97.2	94.3	95.9
	$L_\infty = 4$	Y	93.9	93.9	85.8	91.3	94.1	88.6	91.3
TI	Standard	Y	100	82.4	75.4	88.6	77.1	40.3	77.3
	Standard	N	98.7	95.0	87.0	95.7	95.2	77.6	91.5
	$L_2 = 3.0$	Y	98.5	98.3	96.2	97.9	98.6	95.8	97.5
	$L_\infty = 4$	Y	96.4	96.4	92.8	95.5	97.1	93.9	95.4

Table A8. Influence of BN on the transferability. Results on CIFAR10 with two baselines: DI-FGSM [13], TI-FGSM [4].

	Source	BN	AlexN	VGG16	RN50	DN	RNext	WRN	Avg
DI	VGG16	Y	28.7	100*	91.8	89.7	90.9	90.8	82.0
	VGG 16	N	42.0	99.9	99.8	99.1	99.3	99.6	90.0
	ResNet18	Y	25.3	80.7	88.1	87.2	91.6	92.1	77.5
	ResNet18	N	41.4	99.6	99.7	98.5	99.3	99.1	89.6
TI	VGG16	Y	33.9	100.0	80.8	74.9	80.7	78.8	74.9
	VGG 16	N	51.2	99.6	99.3	97.2	98.3	98.4	90.7
	ResNet18	Y	26.7	67.7	76.5	73.9	80.5	81.7	67.8
	ResNet18	N	53.1	99.1	99.4	97.0	98.7	98.0	90.9

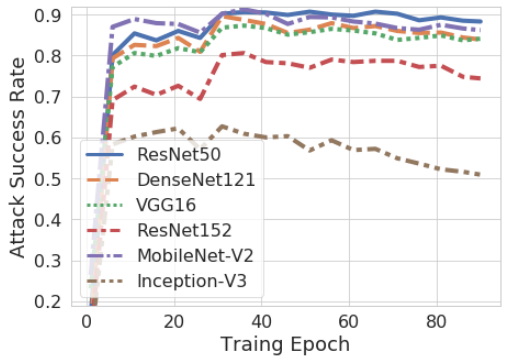

Figure A6. Performance of a substitute ResNet18 (ImageNet) model measured across different training epochs on 5 black-box models.

I. Visualization of the optimization landscape

Following [12], we visualize the optimization landscape. The results on ResNet18 and VGG16 are shown in Fig.A7 and Fig.A8, respectively. On ResNet18, only BN leads to a more predictive and stable gradient; on ResNet50, BN/IN/LN/GN lead to a more stable gradient, however, the effect of IN/LN/GN is significantly smaller than that

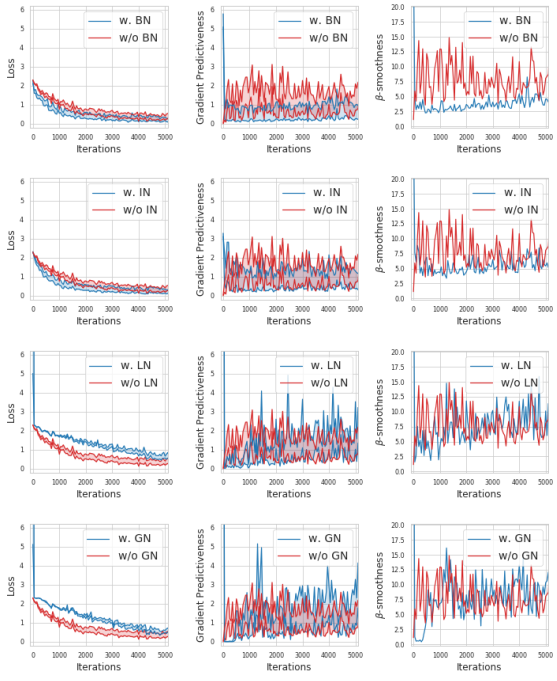

Figure A7. Optimization landscape of ResNet18 with and without normalization. Variation in loss (left); l_2 gradient change (center); β-smoothness (right).

of BN. The results demonstrating the influence of shortcut are shown in Fig.A9 where the shortcut is found to have a trivial influence on the gradient stability. The results comparing FixupIni and BN are shown in Fig.A10, where FixupIni leads to a less stable gradient than BN.

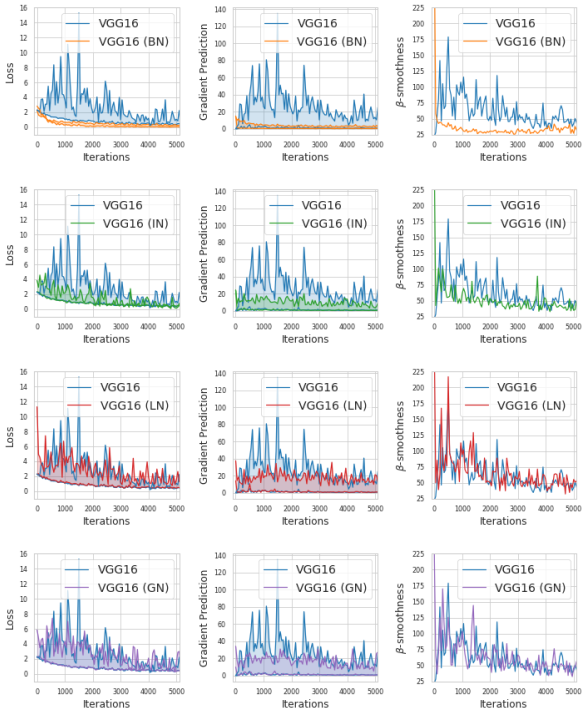

Figure A8. Optimization landscape of VGG16 with and without BN. Variation in loss (left); l_2 gradient change (center); β-smoothness (right).

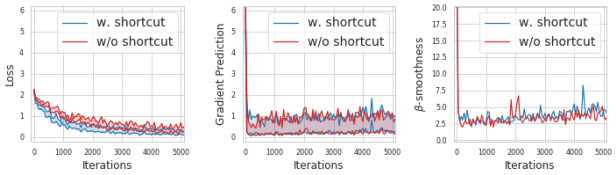

Figure A9. ResNet18 shortcut comparison.

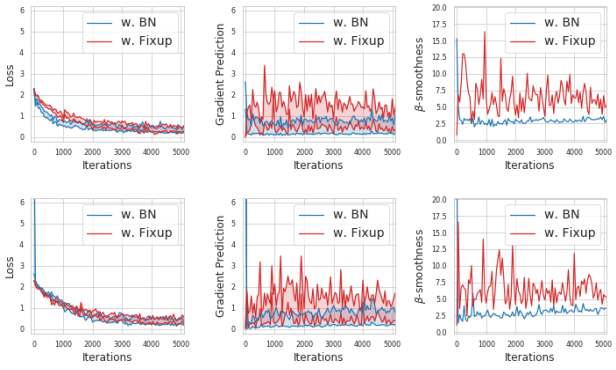

Figure A10. Comparison of BN and FixupIni on Resnet20 (top) and ResNet56 (bottom).

References

[1] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *SP*, 2017. 1

[2] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Siwei Li, Li Chen, Michael E. Kounavis, and Duen Horng Chau. Shield: Fast, practical defense and vaccination for deep learning using jpeg compression. In *KDD*, 2018. 1

[3] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *CVPR*, 2018. 4

[4] Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *CVPR*, 2019. 4

[5] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015. 4

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[7] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *NeurIPS*, 2019. 1, 2, 3

[8] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *ICLR*, 2017. 4

[9] Chihuang Liu and Joseph JaJa. Feature prioritization and regularization improve standard accuracy and adversarial robustness. In *IJCAI*, 2019. 1

[10] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018. 1, 2, 4

[11] Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? In *NeurIPS*, 2020. 2

[12] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *NeurIPS*, 2018. 4

[13] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L Yuille. Improving transferability of adversarial examples with input diversity. In *CVPR*, 2019. 4

[14] Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *arXiv preprint arXiv:1901.06523*, 2019. 1

[15] Dong Yin, Raphael Gontijo Lopes, Jon Shlens, Ekin Dogus Cubuk, and Justin Gilmer. A fourier perspective on model robustness in computer vision. In *NeurIPS*, 2019. 1

[16] Haichao Zhang and Jianyu Wang. Defense against adversarial attacks using feature scattering-based adversarial training. In *NeurIPS*, 2019. 1