Batch Normalization Increases Adversarial Vulnerability and Decreases Adversarial Transferability: A Feature Perspective

Philipp Benz[*], **Chaoning Zhang**[*], **Adil Karjauv & In So Kweon**
Korea Advanced Institute of Science and Technology (KAIST)
{pbenz, iskweon77}@kaist.ac.kr, {chaoningzhang1990, mikolez}@gmail.com

Abstract

Batch normalization (BN) has been widely used in modern deep neural networks (DNNs) due to fast convergence. BN is observed to increase the model accuracy while at the cost of adversarial robustness. We conjecture that the increased adversarial vulnerability is caused by BN shifting the model to rely more on non-robust features (NRFs). Our exploration finds that other normalization techniques also increase adversarial vulnerability and our conjecture is also supported by analyzing the model corruption robustness and feature transferability. With a classifier DNN defined as a feature set F we propose a framework for disentangling F robust usefulness into F usefulness and F robustness. We adopt a local linearity-based metric, termed LIGS, to define and quantify F robustness. Measuring the F robustness with the LIGS provides direct insight on the feature robustness shift independent of usefulness. Moreover, the LIGS trend during the whole training stage sheds light on the order of learned features, *i.e.* from RFs (robust features) to NRFs, or vice versa. Our work analyzes how BN and other factors influence the DNN from the feature perspective. We also show that a substitute model without BN significantly outperforms its counterpart for a transfer-based black-box attack.

1 Introduction

Batch normalization (BN) (Ioffe & Szegedy, 2015) has been considered as a milestone technique in the development of deep neural networks (DNNs) pushing the frontier in computer vision due to fast convergence. We evaluate the behavior of the models with and without BN in Table 1, as expected, BN improves the clean accuracy, *i.e.* accuracy on clean images. However, this improvement comes at the cost of lower robust accuracy, *i.e.* accuracy on adversarial images (Szegedy et al., 2013).

Table 1: Comparison of models with and w/o BN on accuracy and robustness

	Network	Acc	PGD l_2	PGD l_∞	CW l_2	CW l_∞
			0.25	$1/255$	0.25	$1/255$
ImageNet	VGG16 (None)	71.59	15.55	1.79	16.66	0.23
	VGG16 (BN)	73.37	6.04	0.55	6.82	0.02
	VGG19 (None)	72.38	16.52	2.18	17.46	0.30
	VGG19 (BN)	74.24	6.94	0.69	7.66	0.03
	ResNet18 (None)	66.51	30.44	1.24	30.43	0.93
	ResNet18 (BN)	70.50	16.79	0.14	17.40	0.07
	ResNet50 (None)	71.60	28.00	2.17	28.26	0.88
	ResNet50 (BN)	76.54	19.50	0.53	20.19	0.19
CIFAR10	VGG11 (None)	90.06	51.30	70.47	51.75	70.40
	VGG11 (BN)	92.48	39.31	63.87	39.04	63.85
	VGG16 (None)	91.89	34.01	63.18	34.37	63.46
	VGG16 (BN)	93.7	28.61	56.05	24.01	54.58
	ResNet50 (None)	92.15	29.24	49.33	17.09	49.24
	ResNet50 (BN)	95.6	9.15	36.37	8.72	36.64

One recent work (Ilyas et al., 2019) has shown that adversarial vulnerability can be attributed to highly predictive yet brittle non-robust features (NRFs) that are inherent to the dataset. Under their feature framework (Ilyas et al., 2019), the dataset contains robust features (RFs) and non-robust features (NRFs) that are both useful for classification. Given the observation that BN increases adversarial vulnerability in Table 1, their finding (Ilyas et al., 2019) motivates us to explore BN from the feature robustness perspective, towards explaining the increased adversarial vulnerability (Galloway et al., 2019).

Given NRFs are predictive (higher accuracy) yet brittle (lower robustness), we conjecture that BN shifts the model to rely more on NRFs than RFs for classification. We explore BN and similar normalization techniques and find that they increase adversarial vulnerability in standard training. Moreover, our conjecture is also corroborated by the analysis of corruption robustness and feature transferability.

[*]Equal Contribution

Given both RFs and NRFs are useful, it would be desirable to have a metric to measure "pure" robustness independent of usefulness for providing direct evidence on the shift towards NRFs. To this end, with a classifier DNN defined as a feature set F, we propose a framework for disentangling F robust usefulness into F usefulness and F robustness. F usefulness and F robust usefulness can be measured by clean accuracy and robust accuracy, respectively. To measure F robustness, we adopt a metric termed *Local Input Gradient Similarity* (LIGS) (see Sec. 3.1), measuring the local linearity of a DNN as an indication for feature robustness. Note that these metrics describe the entire feature set of a DNN instead of any single feature. We find that BN (or IN/LN/GN) reduces F robustness, which naturally explains the lower robust accuracy caused by them. With this disentangled representation, we investigate and compare the behavior of models trained on a dataset that mainly has either RFs or NRFs, which shows that NRFs are difficult to learn w/o BN suggesting BN is essential for learning NRFs. Further investigation on the dataset with RFs and NRFs conflicting for different classification reveals that the model learns first RFs and then NRFs, and the previous learned RFs can be partially forgotten while the model learns NRFs in the later stage. Beyond normalization, we also analyze other network structure and optimization factors, most of which have no significant influence on F robustness. Finally, we show that a substitute model without BN outperforms its counterpart by a large margin for a transfer-based black-box attack.

2 Does BN shift the model to more rely on RFs or NRFs?

2.1 Feature perspective on the increased adversarial vulnerability

Santurkar et al. explored how BN helps optimization, revealing that BN smooths the optimization landscape instead of reducing internal covariate shift (Ioffe & Szegedy, 2015). Given that DNN learns a set of features (Ilyas et al., 2019), this improved optimization is expected to lead to a model with more useful features, consequently improving the accuracy in standard training. The byproduct of increasing adversarial vulnerability needs explanation. The cause of adversarial vulnerability has been attributed to the existence of NRFs in the (training) dataset (Ilyas et al., 2019), which motivates us to explore the increased adversarial vulnerability from the feature perspective.

Features in DNN: RFs vs. NRFs. *Feature* is one of the most widely used terms in computer vision. Hand-crafted feature design (Lowe, 2004; Dalal & Triggs, 2005) was essential for many vision tasks until deep learning gained popularity, shifting the trend to exploit DNNs to extract features automatically. Despite different interpretations, there is folklore belief that a classification DNN can be perceived as a function utilizing useful features (Ilyas et al., 2019). Specifically, Ilyas et al. define a *feature* to be a function mapping from the input space $\mathcal{X}$ to real numbers, *i.e.* $f : \mathcal{X} \rightarrow \mathbb{R}$, and the set of all features is thus $\mathcal{F} = \{f\}$. A feature f is ρ-useful ($\rho > 0$) if it is correlated with the true label in expectation, *i.e.* $\mathbb{E}_{(\mathrm{x},\mathrm{y})\sim\mathcal{D}}[y \cdot f(x)] \geq \rho$. Given a ρ-useful feature f, robust features (RFs) and non-robust features (NRFs) are formally defined as follows:

- *Robust feature:* a feature f is robust if there exists a $\gamma > 0$ for it to be γ-robustly useful under some specified set of valid perturbations Δ, *i.e.* $\mathbb{E}_{(\mathrm{x},\mathrm{y})\sim\mathcal{D}}[\inf_{\delta\in\Delta(x)} y \cdot f(x+\delta)] \geq \gamma$.
- *Non-robust feature:* a feature f is non-robust if $\gamma > 0$ does not exist.

Conjecture. With the above feature framework, we conjecture that BN shifts the model to rely more on NRFs instead of RFs.

2.2 Normalization leads to the utilization more NRFs

The observation in Table 1 constitutes important yet isolate evidence for our conjecture, which can be corroborated as follows. First, BN is no more complex than normalizing the DNN intermediate feature layers; thus if our conjecture is correct, other normalization techniques (such as LN, IN, and GN) are also likely to mirror the same behavior. Second, with the link (Gilmer et al., 2019) between adversarial robustness and corruption robustness, our conjecture can be more convincing if the corruption robustness phenomenon also supports it. Third, given RFs transfer better, extra evidence can be provided with feature transferability analysis. For the above discussed three points, we provide the empirical result in the Appendix. Overall, the results provide solid evidence that normalization, such as BN, shifts the model to rely more on NRFs instead of RFs.

3 Disentangling usefulness and robustness of model features

With the above evidence to corroborate our conjecture that BN shifts the model to rely more on NRFs, it would be desirable to have a metric to measure "pure" robustness independent of usefulness. Given both RFs/NRFs are useful and their core difference lies in robustness, such a metric is crucial for providing direct evidence on the shift towards NRFs by showing a lower "pure" robustness. Moreover, the LIGS trend during the whole training stage also sheds light on the learned order of features, *i.e.* from RFs to NRFs or vice versa. Evaluating adversarial robustness by robust accuracy demonstrates how robustly useful the model features are. Thus, disentangling robust usefulness into usefulness and robustness provides a better understanding of adversarial robustness.

3.1 Framework for disentangling usefulness and robustness

Following (Ilyas et al., 2019), we define a DNN classifier as a set of features, *i.e.* $F = \{f\}$. The definitions of f usefulness and robust usefulness in Sec. 2.1 can be readily extended to F.

- F usefulness: F is ρ-useful ($\rho > 0$) if it is correlated with the true label in expectation, *i.e.* $\mathbb{E}_{(\mathrm{x},\mathrm{y})\sim\mathcal{D}}[y \cdot F(x)] \geq \rho$;
- F robust usefulness: F is γ-robustly useful if there exists a $\gamma > 0$ under some specified set of valid perturbations Δ, *i.e.* $\mathbb{E}_{(\mathrm{x},\mathrm{y})\sim\mathcal{D}}[\inf_{\delta\in\Delta(x)} y \cdot F(x+\delta)] \geq \gamma$.

For being orthogonal to usefulness, we can not trivially define F robustness by measuring its correlation with the true label in expectation. With a locally quadratic approximation, prior work (Moosavi-Dezfooli et al., 2019) provided theoretical evidence of a strong relationship between robustness and local linearity. Thus, we define F robustness as follows.

- F robustness: A feature set F is β-robust if the local linearity is large than β ($\beta > 0$), *i.e.* $\mathbb{E}_{(\mathrm{x},\mathrm{y})\sim\mathrm{D},\nu\sim\Delta}\left[dist(\nabla l(x,y), \nabla l(x+\nu,y))\right] \geq \beta$.

The local linearity indicated by the distance ($dist$) between $\nabla(l(x,y)$ and $\nabla l(x+\nu,y)$ can be represented in different forms, such as calculating the norm of their difference (Moosavi-Dezfooli et al., 2019). We adopt the cosine similarity (Zhang et al., 2020b) to quantify this distance as:

$$\mathbf{E}_{(x,y)\sim D,\nu\sim\Delta}\left[\frac{\nabla l(x,y)\cdot\nabla l(x+\nu,y)}{\|\nabla l(x,y)\|\cdot\|\nabla l(x+\nu,y)\|}\right]. \quad (1)$$

The adopted distance metric indicates similarity (or linearity) between the original and locally perturbed input gradient and is thus termed *Local Input Gradient Similarity* (LIGS).

The overall trend in Fig. 1 shows that robust accuracy is influenced by both clean accuracy and LIGS. For example, for adversarial (adv.) training, the LIGS stays close to 1 during the whole training stage, and the robust accuracy is highly influenced by the clean accuracy. For standard (std.) training, however, the LIGS is much lower, leading to a much smaller robust accuracy despite slightly higher clean accuracy. The influence of BN is mainly observed on LIGS. During the whole training stage, BN leads to a significantly lower LIGS, consequently lower robust accuracy. As (standard) training evolves, the LIGS value decreases, *i.e.* the feature robustness decreases, suggesting the model relies more NRFs as training evolves. The influence of BN in adv training, however, is very limited. Here, only BN on CIFAR10 is reported. We provide more results with IN/LN/GN and results on ImageNet in Appendix I and Appendix J. The results mirror the trend in Fig. 1.

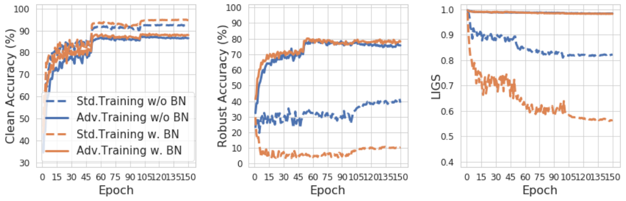

Figure 1: Trend of clean accuracy, robust accuracy, LIGS with ResNet18 on CIFAR10.

Disentangling RFs and NRFs. Following the procedure of Ilyas et al. we extract $\widehat{\mathcal{D}}_R$, $\widehat{\mathcal{D}}_{NR}$ and $\widehat{\mathcal{D}}_{rand}$ (Description in Appendix K). Note that $\widehat{\mathcal{D}}_R$ mainly (if not exclusively) has RFs, while $\widehat{\mathcal{D}}_{rand}$ only has NRFs. $\widehat{\mathcal{D}}_{NR}$ has both RFs and NRFs (see Appendix K for results on $\widehat{\mathcal{D}}_{NR}$). Here, to demonstrate the effect of BN on either NRFs or NRFs, we report the results trained on $\widehat{\mathcal{D}}_R$ and $\widehat{\mathcal{D}}_{rand}$ in Fig. 2, where the clean accuracy and robust accuracy results echo the findings in (Ilyas et al., 2019). There are two major observations regarding the LIGS result. First, the LIGS on $\widehat{\mathcal{D}}_R$ is very high (more than 0.9), which explains why a model (normally) trained on $\widehat{\mathcal{D}}_R$ has relatively high robust accuracy, while the LIGS on $\widehat{\mathcal{D}}_{rand}$ eventually becomes very low because $\widehat{\mathcal{D}}_{rand}$ only has NRFs. Second, w/o BN, the model is found to not converge on $\widehat{\mathcal{D}}_{rand}$, leading to 10% accuracy, *i.e.* equivalent to random guess. This finding also indirectly supports our conjecture. The model with BN starts to converge (achieving an accuracy of higher than 10%) after around 25 epochs and the LIGS is observed to increase before the model starts converging. This suggests that the model is learning features that are robust yet hardly useful. This "warmup" phenomenon is not accidental and repeatedly happens with different random training seeds. After the model starts to converge, the LIGS plummets to a very low value quickly. Refer to the Appendix for more results and analysis, such as training on a dataset of conflicting RFs and NRFs, the influence of other actors on the behavior of DNN.

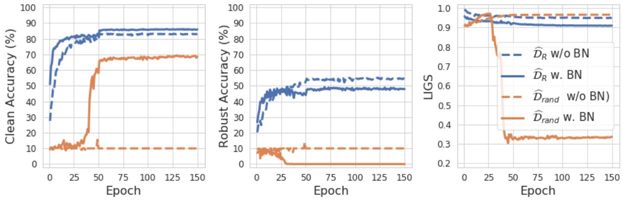

Figure 2: Analysis of BN with ResNet18 on datasets of disentangled features.

4 Influence of BN on Adversarial Transferabilty

Recent works show that robust models are more suitable for downstream tasks because robust features are more transferable (Salman et al., 2020). One natural conjecture is that models with more robust features might also be more suitable for being used as the substitute model for generating transferable adversarial examples across models. Since one of the main takeaways from this work is that BN shifts the model to utilize more NRFs than RFs, we remove BN from the substitute model to craft adversarial examples with increased transferability. We adopt MI-FGSM (Dong et al., 2018) as a strong baseline for transfer-based attacks. The results on CIFAR10 and ImageNet are shown in Table 2 and Table 3, respectively. We observe that a substitute model without BN outperforms its counterpart by a large margin. Moreover, the results here also provide additional evidence that BN indeed shifts the model to rely more on NRFs.

Table 2: Influence of BN on the transferability. Results on CIFAR-10

Source Model	BN	resnet18	resnet18 id	vgg16	vgg16 bn	densenet
VGG16	yes	95.6	87.0	87.3	100	90.7
VGG 16	no	99.6	99.9	100	99.7	98.8
ResNet18	yes	100	72.6	68.5	85.7	90.1
ResNet18	no	99.7	99.8	99.6	99.6	98.0

Table 3: Influence of BN on the transferability. Results on ImageNet.

Source Model	BN	WB	resnet50 bn	densenet121	vgg19 bn	resnet152	mobilenet v2	inception v3
VGG16	yes	100	69.4	64.4	100	46.4	72.9	34.6
VGG 16	no	99.9	75.3	77.5	98.9	61.7	86.8	49.6
ResNet50	yes	100	100	85.8	80	82.6	85.4	54.9
ResNet50	no	100	98.3	96.3	89.5	96.7	96.6	82.2

5 Conclusion

We found that BN and other normalizations increase adversarial vulnerability in standard training. We attribute the reason to the shift of the model to rely more on NRFs, for which evidence from adversarial robustness, corruption robustness, and feature transferability is provided. We are the first to investigate and successfully disentangle usefulness and robustness of model features. With the disentangled interpretation, we found (a) BN and IN/LN/GN decreases the F robustness, explaining why they lead to lower robust accuracy despite higher clean accuracy; (b) the model learns first RFs and then NRFs because RFs features are essential for training the model in the early stage; (c) BN is crucial for learning NRFs, especially, when the dataset only has NRFs. (d) other network structures and optimization factors have no significant influence on F robustness, leaving normalization as the only factor found to have a significant and consistent influence on the F robustness; (g) ResNet shortcut and Fixup initialization does not lead to lower LIGS, suggesting stabilizing training does not necessarily lead to learning more NRFs.*i.e.* lower LIGS. Finally, we reveal that a substitute model without BN outperforms its counterpart by a large margin, which is a practical relevance of our feature perspective understanding on BN.

References

Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 2018.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Philipp Benz, Chaoning Zhang, Tooba Imtiaz, and In-So Kweon. Universal adversarial perturbations are not bugs, they are features. *CVPR workshop on Adversarial Machine Learning in Computer Vision*, 2020a.

Philipp Benz, Chaoning Zhang, Tooba Imtiaz, and In-So Kweon. Data from model: Extracting data from non-robust and robust models. *CVPR workshop on Adversarial Machine Learning in Computer Vision*, 2020b.

Philipp Benz, Chaoning Zhang, Tooba Imtiaz, and In So Kweon. Double targeted universal adversarial perturbations. In *ACCV*, 2020c.

Philipp Benz, Chaoning Zhang, Adil Karjauv, and In So Kweon. Revisiting batch normalization for improving corruption robustness. *WACV*, 2021a.

Philipp Benz, Chaoning Zhang, Adil Karjauv, and In So Kweon. Universal adversarial training with class-wise perturbations. *ICME*, 2021b.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Symposium on Security and Privacy (SP)*, 2017.

Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. In *Proceedings of Machine Learning Research*, 2019.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

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.

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

Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: from adversarial to random noise. In *NeurIPS*, 2016.

Angus Galloway, Anna Golubeva, Thomas Tanay, Medhat Moussa, and Graham W Taylor. Batch normalization is a cause of adversarial vulnerability. *arXiv preprint arXiv:1905.02161*, 2019.

Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow. Adversarial spheres. *arXiv preprint arXiv:1801.02774*, 2018.

Justin Gilmer, Nicolas Ford, Nicholas Carlini, and Ekin Cubuk. Adversarial examples are a natural consequence of test error in noise. In *ICML*, 2019.

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

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

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

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016.

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

David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004.

Ping Luo, Xinjiang Wang, Wenqi Shao, and Zhanglin Peng. Towards understanding regularization in batch normalization. In *ICLR*, 2019.

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

Saeed Mahloujifar, Dimitrios I Diochnos, and Mohammad Mahmoody. The curse of concentration in robust learning: Evasion and poisoning attacks from concentration of measure. In *AAAI*, 2019.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *CVPR*, 2017.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard. Robustness via curvature regularization, and vice versa. In *CVPR*, 2019.

Hyeonseob Nam and Hyo-Eun Kim. Batch-instance normalization for adaptively style-invariant neural networks. In *NeurIPS*, 2018.

Guillermo Ortiz-Jimenez, Apostolos Modas, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Hold me tight! influence of discriminative features on deep network boundaries. In *NeurIPS*, 2020.

Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. Generative adversarial perturbations. In *CVPR*, 2018.

Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli. Adversarial robustness through local linearization. In *NeurIPS*, 2019.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *NeurIPS*, 2017.

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

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

Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In *NeurIPS*, 2018.

Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. *NeurIPS*, 2020.

Ali Shafahi, W. Ronny Huang, Christoph Studer, Soheil Feizi, and Tom Goldstein. Are adversarial examples inevitable? In *ICLR*, 2019.

Yash Sharma, Gavin Weiguang Ding, and Marcus A. Brubaker. On the effectiveness of low frequency perturbations. In *IJCAI*, 2019.

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.

Thomas Tanay and Lewis Griffin. A boundary tilting persepective on the phenomenon of adversarial examples. *arXiv preprint arXiv:1608.07690*, 2016.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *ICLR*, 2019.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018.

Cihang Xie and Alan Yuille. Intriguing properties of adversarial training at scale. *ICLR*, 2020.

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.

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.

Chaoning Zhang, Francois Rameau, Seokju Lee, Junsik Kim, Philipp Benz, Dawit Mureja Argaw, Jean-Charles Bazin, and In So Kweon. Revisiting residual networks with nonlinear shortcuts. In *BMVC*, 2019a.

Chaoning Zhang, Philipp Benz, Tooba Imtiaz, and In-So Kweon. Cd-uap: Class discriminative universal adversarial perturbation. In *AAAI*, 2020a.

Chaoning Zhang, Philipp Benz, Tooba Imtiaz, and In-So Kweon. Understanding adversarial examples from the mutual influence of images and perturbations. In *CVPR*, 2020b.

Chaoning Zhang, Philipp Benz, Dawit Mureja Argaw, Seokju Lee, Junsik Kim, Francois Rameau, Jean-Charles Bazin, and In So Kweon. Resnet or densenet? introducing dense shortcuts to resnet. In *WACV*, 2021a.

Chaoning Zhang, Philipp Benz, Adil Karjauv, and In So Kweon. Universal adversarial perturbations through the lens of deep steganography: Towards a fourier perspective. *AAAI*, 2021b.

Chaoning Zhang, Philipp Benz, Chenguo Lin, Adil Karjauv, Jing Wu, and In So Kweon. A survey on universal adversarial attack. *IJCAI*, 2021c.

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

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. In *Proceedings of Machine Learning Research*, 2019b.

Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. In *ICLR*, 2019c.

A Additional empirical evidence

Adversarial Robustness. Table 4 shows that the phenomenon of increased adversarial vulnerability is not limited to BN but also to IN, LN, GN. Overall, except for the result of ResNet50 on CIFAR10, IN consistently achieves the lowest robust accuracy. We suspect that this can be attributed to the prior finding (Ulyanov et al., 2016; Nam & Kim, 2018) that IN excludes style information by performing instance-wise normalization. The style information is likely RFs (note that changing style, color for instance, normally requires large pixel intensity change), thus IN discarding style can result in the least robust model.

Table 4: Influence of various normalization techniques on accuracy (left/) and robustness (/right).

Data	Network	None	BN	IN	LN	GN
SVHN	VGG11	95.42/63.91	96.27/51.22	95.89/45.82	96.29/56.77	96.30/56.37
	VGG16	95.76/62.24	96.43/52.90	96.64/47.43	96.18/59.55	96.21/59.50
CIFAR10	VGG11	90.06/51.30	92.48/39.31	88.42/31.38	90.54/42.41	90.68/39.43
	VGG16	91.89/34.02	93.70/28.61	90.73/13.44	92.51/28.92	92.83/26.73
	ResNet50	92.15/29.24	95.60/9.15	93.40/10.80	90.37/7.24	92.61/6.43
ImageNet	ResNet18	66.51/30.44	70.50/16.79	63.14/14.29	68.36/19.72	69.02/19.76
	ResNet50	71.60/28.00	76.54/19.50	67.97/13.65	71.08/17.38	74.69/20.34

Gilmer et al. revealed that adversarial training (and Gaussian data augmentation) significantly improve the robustness against noise corruptions, *i.e*. Gaussian/Speckle/Shot while decreasing the robustness against contrast and fog, which is confirmed in Figure 3. Following (Ilyas et al., 2019), a standard model is perceived to learn a sufficient amount of NRFs, while a robust model (robustified through adversarial training) mainly has RFs. Perceiving from the feature perspective, the following explanation arises: noise corruptions mainly corrupt the NRFs while contrast and fog mainly corrupt the RFs. Our explanation from the feature perspective echos with a prior explanation from the frequency perspective (Yin et al., 2019). We discuss their link in Appendix Sec. C.

Corruption Robustness. We find that the model without normalization is more robust to noise corruptions than their counterparts with normalization while a reverse trend is observed for fog and contrast. Given our explanation, this contrasting behavior suggests that the models with normalization learn more NRFs instead of RFs. Another observation from Figure 3 is that IN leads to extra high-robustness against contrast corruption, suggesting less IN is the least dependent on RFs in this context. This aligns well with the adversarial robustness result that the model with IN is generally the least robust.

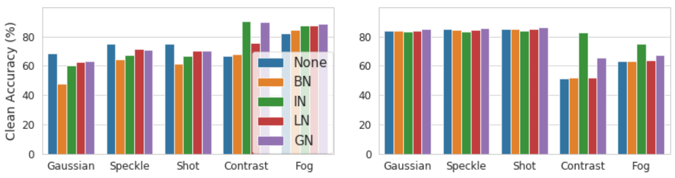

Figure 3: Corruption robustness of VGG16 with *standard training* (left) and *adversarial training* (right)

Feature Transferability. We extract the features out of the trained models as a new dataset and perform cross-evaluation on the remaining models (details in Appendix H.3). The results are shown in Table 5 (left). As a control study, we perform the same analysis on adversarially trained robust models, see Table 5 (right). It has been shown in (Salman et al., 2020) 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 among 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.

Table 5: Cross-evaluation of the features extracted from source models on target models with the baseline VGG16. The left reports the results for standard models, and the right for adversarially trained 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	–

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	–

B Disentangling Usefulness and Robustness of Model Features

Interpretation and Relationship. Informally but intuitively, the usefulness of F can be perceived as the number of features if we assume that each feature is equally useful for classification; and the robustness of F can be seen as the ratio of RFs to NRFs in F. This is illustrated schematically in Fig. 4, where a DNN located in the top right region has high robust usefulness, *i.e.* high robust accuracy, indicating the model learns sufficient features and among them, a high percentage belongs to RFs. A low robust accuracy can be caused by either low F usefulness or low F robustness. Fig. 4 also shows the difference between standard training (green) and adversarial training (blue). Both start from the state of high F robustness and low F usefulness; compared with standard training, adversarial training eventually leads to a model of higher F robustness and lower F usefulness. For standard training, removing BN also increases F robustness. By definition, F robustness, F usefulness, and F robust usefulness can be measured by LIGS, clean accuracy, and robust accuracy, respectively. The schematic illustration in Fig. 4 aligns well with the results in Fig. 1.

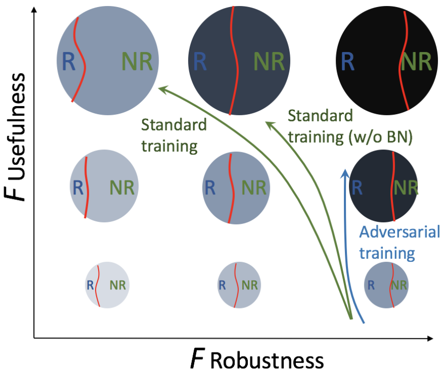

Figure 4: Schematic of disentangling F usefulness and robustness with ball color representing robust usefulness, *i.e.* the darker, the more robustly useful. Ball size indicates usefulness while red line divides RFs and NRFs.

Perturbation Choice of LIGS. We investigate the influence of perturbation type by setting ν to Gaussian noise, uniform noise, FGSM perturbation, and PGD perturbation (details in Appendix H.4) and the results are shown in Figure 5. Among all the chosen types of perturbation, we observe a general trend that the LIGS decreases with training, and consistently the LIGS w/o BN is higher than that with BN. Unless specified, we sample the ν from a Gaussian distribution to measure the LIGS in this work.

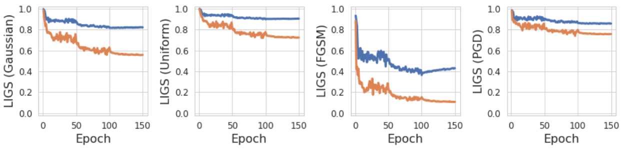

Figure 5: Trend of LIGS with different perturbations: Gaussian, Uniform, FGSM, PGD (left to right).

Relation to Prior Works. The primary motivation of adopting LIGS in this work is to define and quantify the F robustness. Directly maximizing the local linearity as a new regularizer has been shown to improve adversarial robustness on par with adversarial training (Moosavi-Dezfooli et al., 2019). A similar finding has also been shown in (Qin et al., 2019). Note that "adversarial robustness" mostly refers to "robust usefulness" instead of pure "robustness". To avoid confusion, we highlight that F *robustness* is orthogonal to usefulness. Contrary to the prior works (Moosavi-Dezfooli et al., 2019; Qin et al., 2019), which improve "adversarial robustness" by investigating (and establishing) the link between robust usefulness with local linearity, we adopt the local linearity as a measure of pure "robustness". By definition, local linearity does not imply usefulness because it is not related to the correlation with the true label. Nonetheless, their observation that maximizing local linearity can help improve robust usefulness (measured by robust accuracy), can be seen as a natural consequence of increasing F robustness.

C Relation to Frequency Perspective

Our work focuses on the feature perspective (Ilyas et al., 2019) to analyze the model robustness. A Fourier perspective on robustness is introduced in (Yin et al., 2019). With the analysis of corruption analysis in Sec. 2.2, 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 recognized to have HF property, which motivated the exploration of multiple defense methods (Das et al., 2018; Liu & JaJa, 2019). Note, that NRFs do not necessarily have to lie in the HF domain (Sharma et al., 2019; Ortiz-Jimenez et al., 2020). Moreover, our work is the first to demonstrate that the model learns the order from RFs to NRFs. Meanwhile, it has been shown in (Xu et al., 2019) that the model learns first LF component then HF component, which aligns well with our finding by perceiving NRFs having HF property. This phenomenon can also be interpreted as catastrophic forgetting and has also been observed and further discussed in (Ortiz-Jimenez et al., 2020).

D Training on a Dataset of Disentangled RFs and NRFs

It is worth mentioning that by default the experiment setup is the same by only changing the variable of interest (*e.g.* testing with and without BN). Training on a dataset of disentangled RFs and NRFs with BN as the control variable highlights the effect of BN on them while excluding mutual influence.

Training on a Dataset of Conflicting RFs and NRFs. In the original dataset, $\mathcal{D}$, abundant RFs and NRFs co-exist and the model learns both for classification. It is interesting to understand the order of the learned feature, *i.e.* from RFs to NRFs or vice versa, as well their influence on each other. The decreasing trend of LIGS in Fig. 1 suggests that the model learns mainly RFs first. Here, we provide another evidence with the metric of clean accuracy. In the $\mathcal{D}$, RFs and NRFs are cued for the same classification, thus no insight can be deduced from the clean accuracy. To this end, we design a dataset $\widehat{\mathcal{D}}_{Conflict}$ of conflicting RFs and NRFs. Specifically, we exploit the generated $\widehat{\mathcal{D}}_R$ of target class t as the starting images and generate the NRFs of the target class of $t+1$. In other words, in the $\widehat{\mathcal{D}}_{Conflict}$ RFs are cued for class t while NRFs are cued for class $t+1$. Figure 6 shows that with BN the clean accuracy aligned with RFs increases significantly in the first few epochs and peaks around 80% followed by a sharp decrease, while the accuracy aligned with NRFs slowly increases until saturation. It supports that the model learns from RFs to NRFs. Eventually, the accuracy aligned with NRFs surpasses that aligned with RFs, indicating the model forgets most of the first learned RFs during the later stage. W/o BN, we find that the model in the whole stage learns RFs while ignores NRFs. It clearly shows that BN is crucial for learning NRFs, which naturally explains why BN shifts the models towards learning more NRFs. We also discuss the results of $\widehat{\mathcal{D}}_{det}$ (Ilyas et al., 2019) in Appendix K.

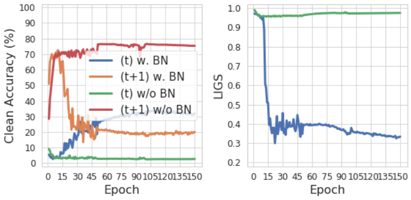

Figure 6: Analysis on dataset with Conflicting RFs and NRFs.

E Exploration Beyond (Batch) Normalization

Network Structure Factors and Optimization Factors. Besides normalization, other factors could influence the DNN behavior, especially concerning its robustness. We study two categories of factors: (a) structure category including network width, network depth, and ReLu variants; (b) optimization category including weight decay, initial learning rate, optimizer. The results are presented in Figure 7. We find that most of them have no significant influence on LIGS. Increasing network width and depth can increase or decrease the LIGS, respectively, but by a small margin. No visible difference between ReLU and Leaky ReLU can be observed, while SeLU leads to a slightly higher LIGS with lower clean accuracy. Some candidates from the optimization category are found to influence F robustness differently in the early and later stages of training. High weight decay leads to higher LIGS in the early stages of training and slightly lower in the end. A higher initial learning rate, such as 0.5, results in higher LIGS in the early training stage, but eventually leads to lower LIGS. For both weight decay and initial learning rate, an opposite trend of lower/higher in the early/later stage is observed with clean accuracy. SGD optimizer and ADAGrad show similar behavior on LIGS, ADAM leads to slightly higher LIGS. Their influence on clean accuracy is more significant.

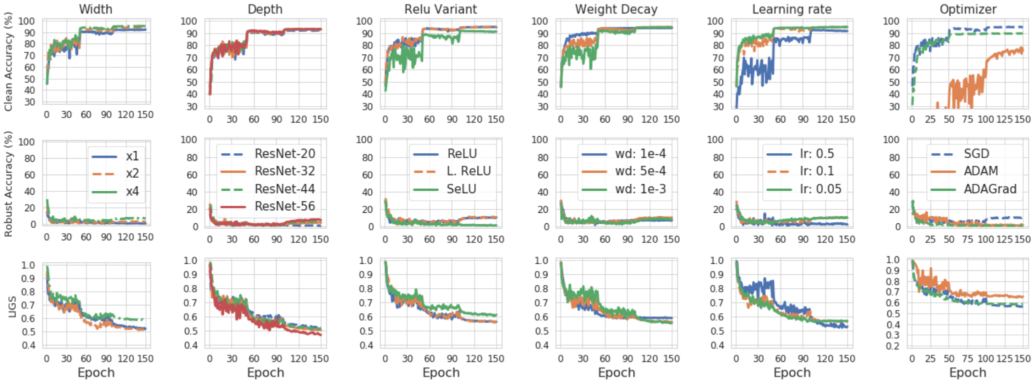

Figure 7: Influence of other factors on the behavior of DNN

E.1 Does stabilizing training lead to lower F robustness?

Given our observation that the model w/o BN cannot converge on a dataset with only NRFs and the wide belief that BN stabilizes training, we are wondering about the potential link between stability and F robustness. ResNet shortcut also stabilizes/accelerates training (He et al., 2016), thus we investigate whether it reduces F robustness. Fig. 8 shows that shortcut has trivial influence on LIGS with ResNet20. For a much deeper ResNet56, removing the shortcut has a significant influence on LIGS in the early stage of training, however, eventually, the influence also becomes marginal.

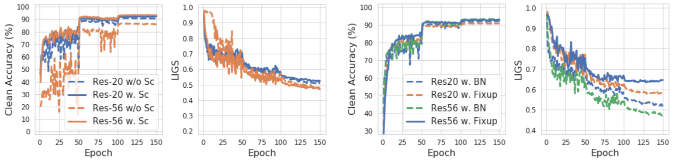

Figure 8: Effect of shortcut (left) and FixupIni (right) on model.

Fixup initialization (FixupIni) is introduced in (Zhang et al., 2019c) to replace the BN in ResNets. We compare their influence on the model and observe that their difference in clean accuracy is trivial, while BN leads to lower LIGS than FixupIni. Overall, it shows increasing training stability does not necessarily lead to lower F robustness. Santurkar et al. have shown that BN leads to more predictive and stable features. Following the procedure in (Santurkar et al., 2018), we visualize the gradient predictiveness and find BN leads to more stable and predictive gradients. We observe that shortcut has a trivial influence on the gradient stability, and FixupIni leads to lower gradient stability than BN. This seems to suggest a link between gradient stability and F robustness. However, IN/LN/GN also reduces the robustness while they do not lead to strong gradient stability as BN (See Appendix M). Our work, as the first step towards investigating what leads to lower F robustness, might inspire more future explorations.

F Discussion

BN on Corruption Robustness. One interesting observation from Fig. 3 is that BN is more vulnerable to all corruptions than other normalizations, *i.e.* IN, LN, GN. The reason can be attributed to the domain shift (Li et al., 2016) between clean images and corrupted images, which can be mitigated by rectifying the BN mean and variance with the corrupted images (Schneider et al., 2020; Benz et al., 2021a).

BN in Adversarial Training. With standard training, we find that BN increases adversarial vulnerability. To improve robustness, adversarial training is one of the most widely used methods. Xie & Yuille showed that BN might prevent networks from obtaining strong robustness in adversarial training. However, this is only true when clean images are utilized in the training and the reason is attributed to the two-domain hypothesis. For standard adversarial training (Madry et al., 2018) with only adversarial images, as shown in Fig. 1, BN is found to have no influence on LIGS as well as robust accuracy. This is reasonable because adversarial training explicitly discards NRFs.

BN with FGSM attack. Motivated by the (local) linearity assumption, Goodfellow et al. proposed one-step FGSM attack. FGSM efficiently attacks the model but is not as effective as PGD attack (Madry et al., 2018) 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, 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 Appendix L.

Regularization of LIGS. The above results show that there is a strong link between robust accuracy and LIGS. To further verify the link between them, we use LIGS as a regularizer during training. The results in Figure 9 confirm that increasing LIGS through regularization improves the robust accuracy by a large margin despite a small influence on clean accuracy.

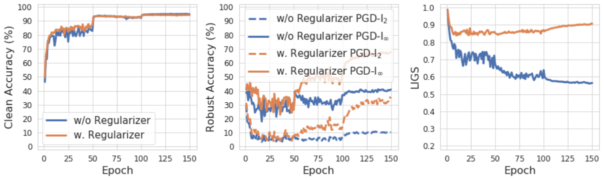

Figure 9: Effect of regularizing LIGS.

G Related Work

Normalization in DNNs. BN performs normalization along batch dimension to speed-up training (Ioffe & Szegedy, 2015). The stochasticity of the batch statistics also serves as a regularizer and improves generalization (Luo et al., 2019). However, the property of batch dependence makes BN not appropriate when a large batch size is impractical (Ba et al., 2016). Moreover, the pre-computed mean and variance from the training dataset might not be feasible to the test dataset especially when there are domain changes (Rebuffi et al., 2017). To avoid such an issue due to the batch dimension, several alternative normalization techniques have been proposed to exploit the channel dimension, such as layer normalization (LN) (Ba et al., 2016). Instance normalization (IN) (Ulyanov et al., 2016) has been explored for style transfer. Group normalization (GN) in (Wu & He, 2018) performs normalization for C/G channels, where C indicates the number of total channels of a certain layer and G indicates the number of groups. Inserting GN through non-linear shortcut (Zhang et al., 2019a) and dense shortcuts (Zhang et al., 2021a) can significantly improve the deep classifier performance. Different from BN, GN does not require a pre-computed mean and variance as BN. In essence, BN, LN, IN and GN are no more complex than performing a normalization on the intermediate layers, despite the difference of normalization dimension. Concurrent to (Galloway et al., 2019) showing BN increases adversarial vulnerability, our work finds that LN/IN/GN mirrors the same trend.

Adversarial Vulnerability. Adversarial examples have attracted significant attention in machine learning (Szegedy et al., 2013; Goodfellow et al., 2015), which raises concern for improving robust accuracy (Carlini & Wagner, 2017). The cause of adversarial vulnerability have been explored from different perspectives, such as local linearity (Goodfellow et al., 2015), input high-dimension (Gilmer et al., 2018; Shafahi et al., 2019; Mahloujifar et al., 2019), limited sample (Schmidt et al., 2018; Tanay & Griffin, 2016), boundary tilting (Tanay & Griffin, 2016), test error in noise (Fawzi et al., 2016; Gilmer et al., 2019; Cohen et al., 2019), etc. Some explanations might conflict with other explanations (Akhtar & Mian, 2018). Recently, the cause of adversarial vulnerability has also been explained from a feature perspective (Ilyas et al., 2019). A major line of works have also investigated the model vulnerability to universal adversarial perturbations (Moosavi-Dezfooli et al., 2017; Poursaeed et al., 2018; Zhang et al., 2020b; Benz et al., 2020a; Zhang et al., 2020a; Benz et al., 2020c; Zhang et al., 2021b; Benz et al., 2021b; Zhang et al., 2021c). In adversarial machine learning, both accuracy and robustness are important for evaluating the model performace (Carlini & Wagner, 2017). Despite many efforts to bridge their gap (Zhang et al., 2019b), it is widely recognized that there is a trade-off between them (Tsipras et al., 2019). Regardless of this trade-off, our work shows that robustness can be influenced by both clean accuracy and feature robustness, which provides insight into why normalization like BN decreases robust accuracy despite higher clean accuracy.

H Experimental Setup

H.1 Setup for Training Models in Sec. 2.2

The models for CIFAR10 and SVHN used in Sec. 2.2 were trained with SGD with the training parameters listed in Table 6. The ResNet50 models in Table 1 and Table 4 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 (He et al., 2016).

Table 6: Parameters to train a standard model on CIFAR10/SVHN

Parameter	Value
Learning rate	0.01
Batch size	128
Weight Decay	$5 \cdot 10^{-4}$
Epochs	300
Learning rate decay epochs	200
Learning rate decay factor	0.1

For robust models in Table 5 We follow the adversarial training strategy from (Madry et al., 2018) with l_2-PGD and we list the parameters in Table 7.

Table 7: Training parameters for adversarial training for CIFAR10/SVHN

Parameter	Value
Learning rate	0.01
Batch size	128
Weight Decay	$5 \cdot 10^{-4}$
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

H.2 PGD Attack for Evaluating Adversarial Robustness

In Table 1, we evaluate the robustness of models with the l_2 and l_∞ variants of the PGD-attack (Madry et al., 2018) and Carlini & Wagner (CW) attack (Carlini & Wagner, 2017). 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 $2.5\epsilon/$steps. For the CW-attack (Carlini & Wagner, 2017), we follow (Zhang & Wang, 2019) 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 3 are obtained with l_2-PGD as above, but for 10 iteration steps on 1000 evaluation samples (100 samples per class) to reduce computation cost.

H.3 Extracting Features as a Dataset from a Model

Inspired by (Ilyas et al., 2019; Benz et al., 2020b)To demonstrate feature transferability in Table 5, we extract features from standard and adversarially trained models as a dataset. We follow the procedure and hyperparameter choices in (Ilyas et al., 2019) and generate a dataset $\hat{\mathcal{D}}$, given a model C:

$$\mathbb{E}_{(x,y)\sim\hat{\mathcal{D}}}[y \cdot f(x)] = \begin{cases} \mathbb{E}_{(x,y)\sim\mathcal{D}}[y \cdot f(x)] & \text{if } f \in F_C \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

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:

$$\min_{\delta} ||g(x) - g(x' + \delta)||_2. \tag{3}$$

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.

H.4 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 5 the, ν is set to either Gaussian noise, uniform noise in the range of $[-0.01, 0, 01]$, FGSM wit $\epsilon = 0.01$ or l_∞-PGD with $\epsilon = 0.01$, 7 step iterations and a step size of 2.5ϵ/steps.

I INFLUENCE OF OTHER NORMALIZATION TECHNIQUES ON LIGS

In Fig. 4, we show the influence of BN on the robust accuracy and LIGS over the model training process. Additionally, Fig. 10 shows the results of repeating this experiment with IN, LN, and GN. Similar trends to those of BN are observed.

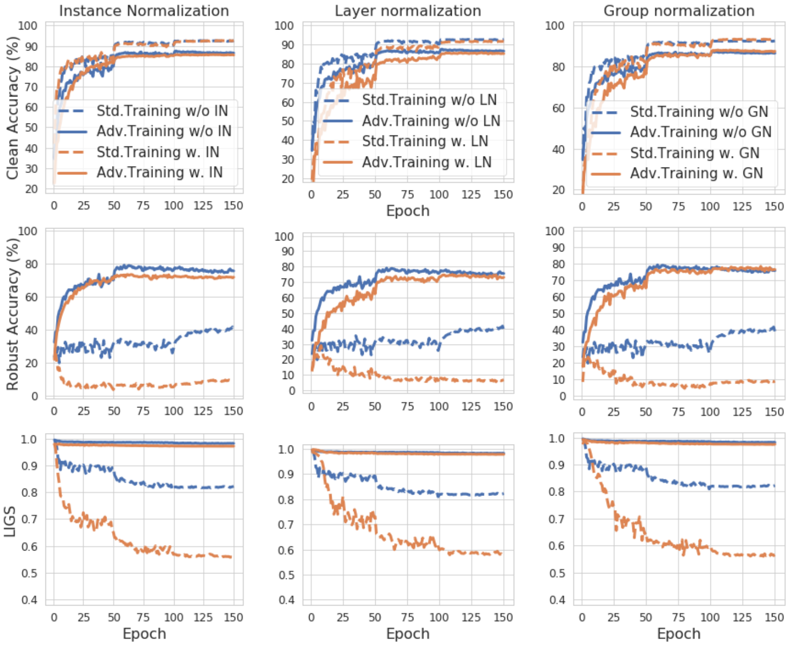

Figure 10: Trend of clean accuracy (top), robust accuracy (middle), LIGS (bottom) for ResNet18 on CIFAR10 with different normalization techniques (IN, LN, GN) applied.

J RESULTS ON IMAGENET WITH LIGS TREND

Fig. 11 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.

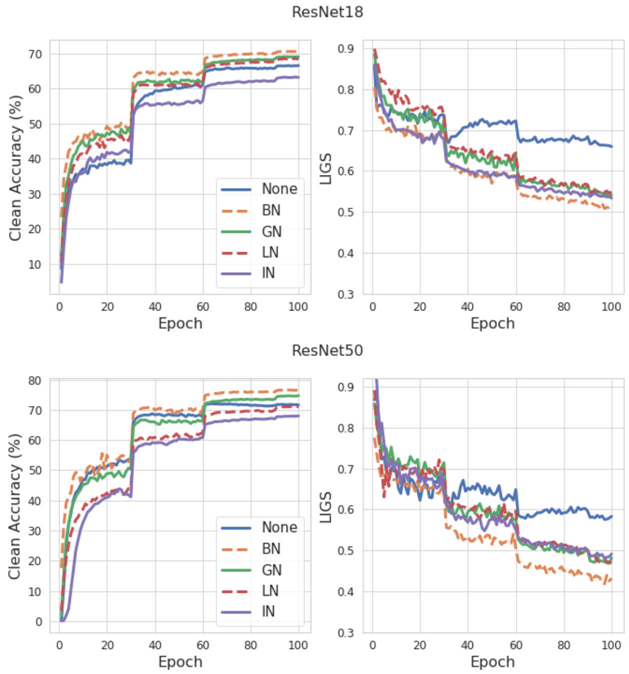

Figure 11: Comparison of different normalization techniques for ResNet18 (top) and ResNet50 (bottom) trained on ImageNet.

Table 8: 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	$5 \cdot 10^{-4}$

K DESCRIPTION OF $\widehat{\mathcal{D}}_R$/ $\widehat{\mathcal{D}}_{NR}$/ $\widehat{\mathcal{D}}_{rand}$/ $\widehat{\mathcal{D}}_{det}$

Ilyas et al. 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 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 (Ilyas et al., 2019) 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 8. 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. 12 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.

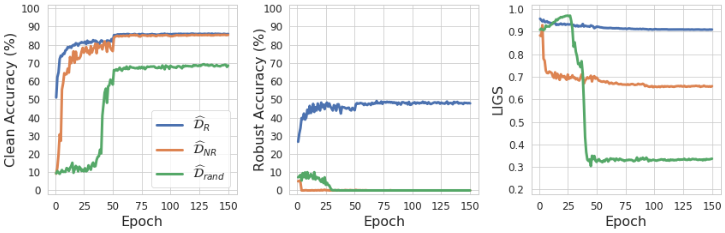

Figure 12: Comparison of $\widehat{\mathcal{D}}_R$, $\widehat{\mathcal{D}}_{NR}$ and $\widehat{\mathcal{D}}_{rand}$

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. 13. The results resemble those of $\widehat{\mathcal{D}}_{Conflict}$. However, we used $\widehat{\mathcal{D}}_{Conflict}$ to avoid the influence of the NRFs in $\mathcal{D}$.

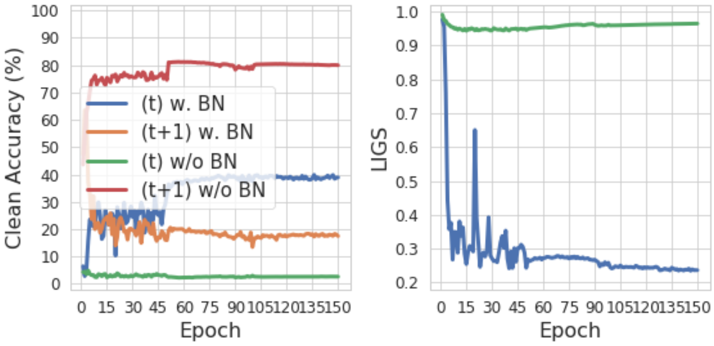

Figure 13: Result on $\widehat{\mathcal{D}}_{\text{det}}$

L Evaluating adversarial robustness with FGSM attack

As shown in Table. 9, we find that with FGSM attack, the model with BN has higher adversarial robustness than that w/o BN.

Table 9: 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

M Visualization of Optimization Landscape

Following (Santurkar et al., 2018), we visualize the optimization landscape. The results on ResNet18 and VGG16 are shown in Fig.14 and Fig.15, 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 of BN. The results demonstrating the influence of shortcuts are shown in Fig.16 where the shortcut is found to have a trivial influence on the gradient stability. The results comparing FixupIni and BN are shown in Fig.17, where FixupIni leads to a less stable gradient than BN.

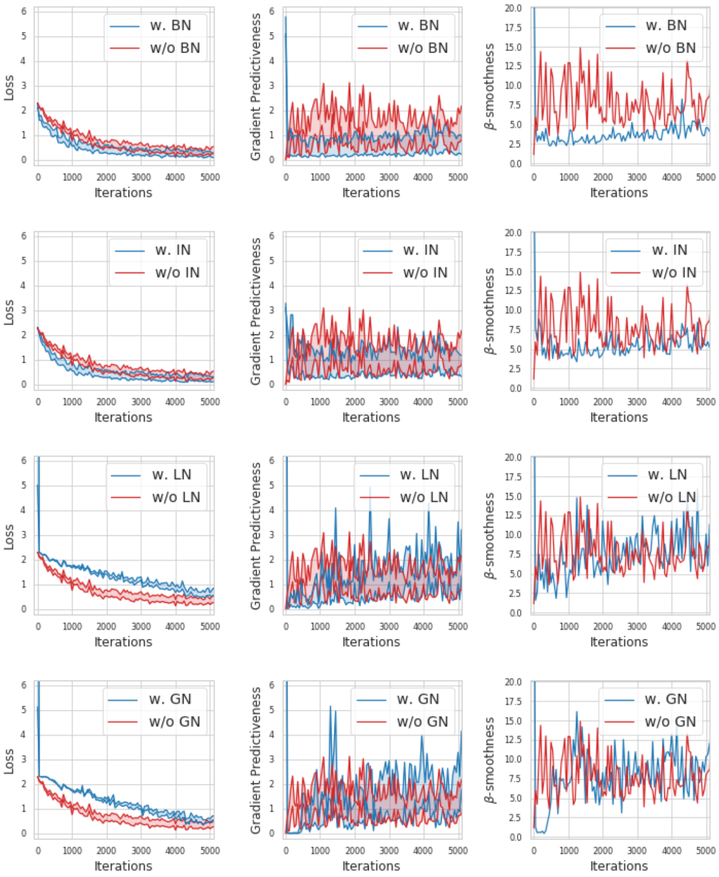

Figure 14: Optimization landscape of ResNet18 with and without normalization. Variation in loss (left); l_2 gradient change (center); β-smoothness (right).

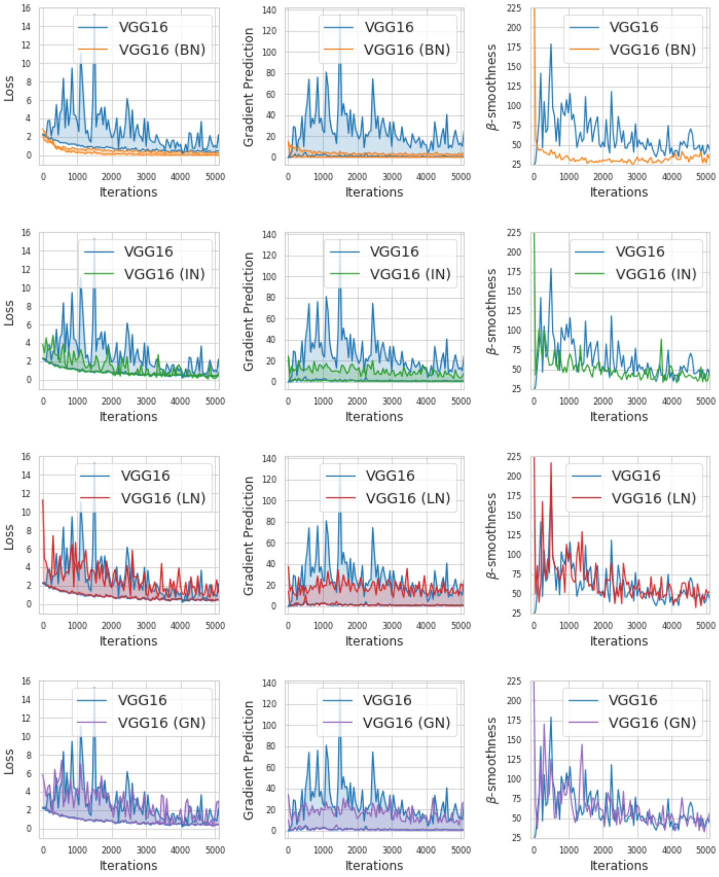

Figure 15: Optimization landscape of VGG16 with and without BN. Variation in loss (left); l_2 gradient change (center); β-smoothness (right).

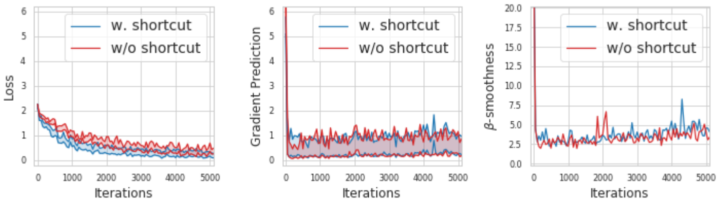

Figure 16: ResNet18 Shortcut comparison

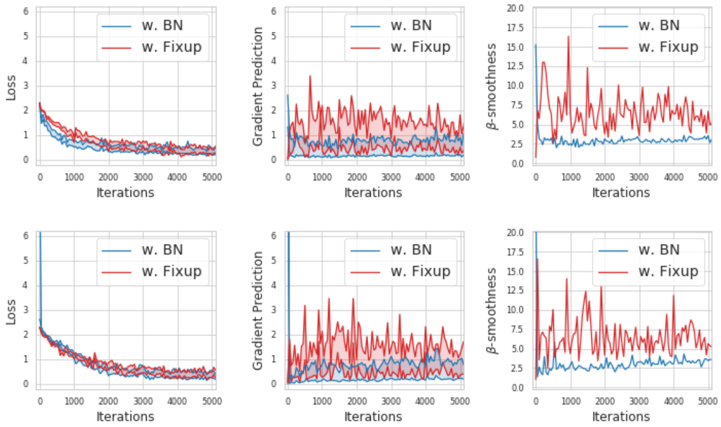

Figure 17: Comparison of BN and FixupIni on Resnet20 (top) and ResNet56 (bottom)