Understanding (Non-)Robust Feature Disentanglement and the Relationship Between Low- and High-Dimensional Adversarial Attacks

Zuowen Wang, Leo Horne
{wangzu, hornel}@ethz.ch

Abstract

Recent work has put forth the hypothesis that adversarial vulnerabilities in neural networks are due to them overusing "non-robust features" inherent in the training data. We show empirically that for PGD-attacks, there is a training stage where neural networks start heavily relying on non-robust features to boost natural accuracy. We also propose a mechanism reducing vulnerability to PGD-style attacks consisting of mixing in a certain amount of images containing mostly "robust features" into each training batch, and then show that robust accuracy is improved, while natural accuracy is not substantially hurt. We show that training on "robust features" provides boosts in robust accuracy across various architectures and for different attacks. Finally, we demonstrate empirically that these "robust features" do not induce spatial invariance.

1 Introduction

In recent years, deep neural networks (DNNs) have become the tool of choice for image classification tasks. State-of-the-art DNN models are able to achieve very high accuracies on standard datasets. However, the complexity of these neural networks has resulted in human beings being unable to fully understand their training and inference process, causing difficulties in interpreting certain counterintuitive phenomena. One such phenomenon is the existence of *adversarial examples* [3, 21], images that appear to humans to be obvious examples of a certain class y_{true} of objects (e.g. y_{true} = 'cat'), but that the model assigns to a completely different class y_{false} with very high confidence (e.g. y_{false} = 'dog' with 99% confidence). In some cases, even simple low-dimensional transformations such as translation or rotation can cause misclassification [8]. The existence of adversarial examples [8, 10, 17], as well as numerous defense schemes [5, 15, 19, 23, 25], has been well-studied. However, the mechanism underlying their existence is presently not known.

Recent work by Ilyas et al. [13] has put forth the hypothesis that the emergence of adversarial examples is due to the supervised learning paradigm currently employed, which solely aims to maximize categorical accuracy on natural (unperturbed) images. More specifically, they propose that each image class y may have two main kinds of features associated with it: *robust* features, features that are clearly indicative of class y_{true} both to humans and to neural networks (e.g. the existence of fur or pointed ears), and *non-robust* features, features that are strongly indicative of the class y_{true}, but are only meaningful to the neural network and not to humans, to whom they may appear to be random patterns. While non-robust features aid in boosting classification accuracy, they are *brittle*, meaning that slight perturbations to the image (imperceptible to humans) can completely transform the non-robust features to indicate a different class y_{false}. The core idea is that such perturbations to non-robust features are the root cause of the existence of adversarial examples, and the dominant supervised learning paradigm causes neural networks to make heavy use of non-robust features to maximize natural accuracy. Ilyas et al. claim that this is the reason adversarial attacks tend to transfer well among diverse neural network architectures.

Ilyas et al. constructed two special datasets to verify their hypothesis: from a base dataset D containing natural images, they created a dataset D_R containing images whose non-robust features cannot be relied on to indicate their class (i.e., their only useful features are robust) and a dataset D_{NR} containing images whose only useful features are non-robust. D_R is produced by using an adversarially-trained classifier $f_{\mathcal{A}}$ to distort a randomly selected image x' to the label y of a different image x. The intuition is that since $f_{\mathcal{A}}$ is adversarially trained, it will only distort the robust features of x' to the label y, leaving the non-robust features untouched and thus not relevant to the class y. D_{NR} is produced by constructing adversarial examples (whose non-robust features, by definition, point to a class $y_{false} \neq y_{true}$, but whose robust features point to y_{true}) and relabeling them to class y_{false}.

More research is required to fully understand if Ilyas et al.'s hypothesis holds, and if so, how and to what it applies. This project aims to clarify and extend some aspects of Ilyas et al.'s work by asking the following questions:

Question 1. *Does the current supervised training paradigm induce undesired utilization of non-robust features?*

We first ask whether it is possible that the current goal of maximizing a classifier's accuracy during training can lead to the classifier "overfitting" to non-robust features, i.e. primarily making use of robust features in the early stages of training but heavily relying on non-robust features later to bring classification accuracy to its peak level. We design an experiment that snapshots training progress at regular intervals, performs various adversarial attacks, and then calculates the attack success rate ($ASR = \frac{\text{number of adversarial examples wrongly classified}}{\text{number of corresponding natural examples correctly classified}}$). The intuition is that if neural networks first learn mainly robust features and only overfit to non-robust features in later training stages, the ASR should be lower in earlier stages of training.

Question 2. *Can we further validate the idea of natural images containing both robust and non-robust features?*

To answer this question, we design a dataset D_{mix} containing (1) natural images of various classes from the original dataset D and (2) images from D_R. We train various classifiers on this dataset and evaluate their robust accuracies for various different attacks as well as their natural accuracies. The intuition is that higher proportions of images from D_R will lead to higher robust accuracies. Also, to eliminate the

concern of defense information leakage to the constructed robust dataset [6], we evaluate the models trained on D_{mix} with different attack methods. We expect that the robustness induced by D_{mix} transfers well across different attack methods and architectures.

Question 3. *Can PGD-robust datasets induce spatial equivariance?*
Ilyas et al. only examine projected gradient descent (PGD) attacks [17] as a means of adversarial attack. We will therefore examine whether the "robust features" of [13] are also robust against spatial attacks (such as translation and rotation). In particular, if such features exist, they should (by definition) induce a certain degree of equivariance to rotation and translation attacks for the model.

To test this hypothesis, we train classifiers on D_R and perform spatial transformation attacks [8]. Since the adversarially-trained classifier $f_{\mathcal{A}}$ used to construct D_R is only known to be robust to PGD attacks [17], we also attempt to construct spatially robust and non-robust datasets D_R^S and D_{NR}^S using the same method as for D_R and D_{NR}, but using special equivariance-inducing classifiers as $f_{\mathcal{A}}$.

Finally, we train networks which claim to achieve a certain degree of spatial equivariance on D_R to evaluate whether training only on robust features causes these networks to achieve a higher degree of spatial equivariance compared to training on natural images. The architectures we use include the group equivariant network [4], spatial transformer network [14], and a special case of equivariance transformer models [9], the polar transformer network [22].

Contributions. Concretely, our contributions in this work are the following insights into the claims made in [13]:

- We show that the current training goal of maximizing accuracy on natural images is not in itself responsible for overfitting to non-robust features. In fact, we show that neural networks do not make a clear distinction between robust and non-robust features during training, utilizing both from the very beginning of training.
- Previous work [25] has shown that there is often a trade-off between natural accuracy and robust accuracy. We show empirically that we can achieve a boost in robust accuracy by adding only a small amount of images from D_R to the training dataset without perceptibly hurting natural accuracy. Furthermore, we show that the robustness of classifiers trained on D_R transfers well among attacks based on different high-dimensional perturbations as well as among different architectures. We can therefore consider such mixed datasets as a novel defense mechanism against adversarial attacks.
- We attempt to apply the theories from [13] to the case of spatial attacks, and find that their theories do not apply in the realm of low dimensional attacks. In particular, classifiers trained on D_R exhibit worse spatial robustness than classifiers trained on D, even in the case of spatially robust architectures.

2 Models and Methods

In this section, we go over the methods used to answer the Questions 1, 2, and 3.

2.1 Attacks

In our work, we mainly study two major classes of attacks (i.e. image perturbations): PGD attacks and spatial attacks. The attack settings largely mirror those in [6, 13]. We mainly use these attacks for evaluation. The accuracy of a model evaluated with adversarial examples generated by an attack scheme $\mathcal{A}$ is called *robust accuracy* under $\mathcal{A}$, while the original evaluation using only natural images is termed *natural accuracy*. We use various attack parameters for the specific purpose of evaluating degrees of robustness for the different models trained. The character of a model which enables it to resist PGD or spatial attacks is called *PGD robustness* or *spatial robustness* respectively.

PGD attacks Projected Gradient Descent (PGD) attacks are considered to be the strongest attacks which utilize the first-order information of the network, as studied in [17]. We use two types of PGD attacks (ℓ_2-PGD and ℓ_∞-PGD) which adopt different norms when bounding the attack space: ℓ_2-norm and ℓ_∞-norm. The adversarial examples are generated by perturbing the original images towards the gradient ascent direction, while the gradient is normalized by the constraint norm. The resulting perturbed image will be projected back to the allowed image space, with the norm of its difference from the original image constrained by a parameter ε. This attack step can be performed for multiple iterations.

For the ℓ_2-PGD attack, our attack parameters follow [13]. We choose $\varepsilon = 0.25$ or 0.5, step size $= 0.1$ and we run 100 iterations to ensure that it is possible for the attack step to reach the boundary. For the ℓ_∞-PGD attack, apart from following the attack settings in [17], we also set up experiments of smaller ℓ_∞boundary, i.e. weaker attacks, for our own purposes of demonstration. In total, we have $\varepsilon = \frac{1}{255}, \frac{2}{255}, \frac{4}{255}$, and $\frac{8}{255}$ with step size $= \frac{\varepsilon}{4}$ and 7 steps of attack.

Spatial attacks Spatial attacks are performed by generating rotated and/or translated images with the aim of causing misclassification. Previous work [8] has shown that a high number of spatial adversarial examples can be found through a *grid search*. There are infinitely many combinations of rotation and translation possible for a spatial attack, so we attempt to approximate this infinite number by discretizing the continuous space of rotations and translations into a mesh containing all possible combinations of the discretized attack parameters. In particular, we use the following grids in our work. (1) `grid775`, a grid with 5 values per translation direction (vertical or horizontal) and 31 values for rotation, yielding 775 images; (2) `grid135`, a grid with 3 values per translation direction and 15 values for rotation, yielding 135 images (for these two grids, we use spatial limits of at most 3px translation in either direction and 30° rotation in either direction); (3) `grid775,10°`, the same as `grid775` but with at most 10° of rotation in either direction; (4) `rot30`, the same as `grid775` but with no translation; and (5) `rot10`, the same as `rot30` but with at most 10° of rotation in either direction.

Attacks (2)–(5) serve as a weaker version of `grid775` to allow for the appropriate evaluation of the spatial robustness of certain models while avoiding being too strong to break any model claiming to achieve a certain degree of spatial

robustness.

2.2 Datasets

The base dataset D we use throughout this work is CIFAR-10 [16]. The datasets D_R and D_{NR} are derived from CIFAR-10 and are freely available as part of [13]. Additionally, we create a custom dataset D_{mix}. This dataset is designed such that during training, a certain proportion α of each batch is composed of randomly selected images from D_R, while the rest are randomly selected from D. One epoch is defined to be over after 50000 images (the size of D).

2.3 Architectures

Main architectures In this work we mainly conduct experiments with the ResNet architecture family [12], namely ResNet-34 and ResNet-50. The VGG-16 [20] model and the ResNet-18 model are used for the purpose of studying the transferability of robustness which is induced by D_R or D_{mix}, as the D_R we used are constructed using ResNet-50 in [13].

Spatial equivariant networks To detect the effect of the PGD-robust dataset D_R on special architecture designs aiming to gain equivariance against spatial transformations, we compare the spatial robust accuracy from spatial equivariant networks trained with the natural dataset D and the robust dataset D_R. Our scope of study includes: (a) G-ResNet18 and G-ResNet34 [4] which use $p4m$ convolutional layers to replace normal 2D convolution layers in the ResNet-18 and ResNet-34 architectures, resulting in feature maps inherently containing rotations in 90° increments. We port the code from [2] in PyTorch into our experiment framework. (b) Spatial Transformer Networks (STNs) [14], which incorporate a pose predictor module in a backbone ResNet architecture. We adapt the standard STN code from the PyTorch STN tutorial [11]. (c) Polar Transformer Networks (PTNs) [9], an instantiation of Equivariant Transformer Networks [22] which exploit the rotational equivariance of convolution operations under polar coordinates. We adapt the code offered in [22].

2.4 Training details

Hyperparameters For experiments about Question 1 we use batch size 64, a weight decay of $2 \cdot 10^{-4}$, and an initial learning rate of 0.1 which is divided by 10 after 40000 and 60000 iterations for the ResNet-34 model. We train for 100 epochs in total.

For experiments about Question 2 and 3 we use different hyperparameter settings than for Question 1. In Question 2, the ResNet-50, ResNet-18 and VGG-16 are trained using a batch size of 128, a weight decay of $5 \cdot 10^{-4}$, and an initial learning rate of 0.1 which is divided by 10 every 50 epochs. We train for 150 epochs in total. Due to constraints in computational resources, for Question 3 we tested two learning rates: 0.1 and 0.01 for the spatial equivariant networks.

Data augmentation We use two variants of data augmentation. We call the first variant `std`, which performs a translation of at most 4px in any direction (chosen uniformly at random), performs a random horizontal flip with probability 0.5, jitters the brightness, contrast, and saturation by at most 25% (chosen uniformly at random), and performs a rotation of at most 2° in either direction (chosen uniformly at random). The second variant, `std*`, is the same as `std`, but performs a translation of at most 3px in any direction and a rotation of up to 30° in either direction.

2.5 Methods

For the three questions proposed in Section 1 we designed a series of experiments to answer them. We describe the detailed methods in the following paragraphs.

Method for Question 1 We use a TensorFlow [1] framework [8, 24] to train a ResNet-34 and evaluate the ASRs for various attacks. In order to examine the claim proposed in [13] that aiming to achieve a high categorical accuracy introduces adversarial vulnerabilities, we design experiments to detect the correlation of model natural accuracies and corresponding ASRs. We train a ResNet-34 model on the CIFAR-10 training set with `std` data augmentation for 100 epochs. To observe how ASRs for both PGD attacks and spatial attacks evolve with model natural accuracy, we evaluate them every 4 epochs during the training process of the model on the full test set of CIFAR-10. We also evaluate every 625 steps (equals 0.8 epochs) additionally before reaching the fourth epoch, in order to capture the rapid changes in natural accuracy and ASR during the beginning of training. The reason why we use ASR instead of robust accuracy as an evaluation metric is that it reflects how effective an attack method is and it rules out the influence of low natural accuracy of the model during the early training phase by ruling out examples which the model can not even make a correct prediction for in their natural state.

Method for Question 2 We use our framework with `std` data augmentation to train the ResNet-50, VGG-16, and ResNet-18 on D_{mix} for values of $\alpha \in [0, 1]$ in 0.1-step increments and perform the following PGD attacks: (1) we attack all models with an ℓ_2-norm attack constrained by $\varepsilon = 0.25$ and 0.5 (with the learning rate and number of iterations described in Section 2.1); (2) we attack the ResNet-50 with an ℓ_∞-norm attack constrained by $\varepsilon = \frac{1}{255}, \frac{2}{255}, \frac{4}{255}$, and $\frac{8}{255}$ (with the step size and number of attack steps described in Section 2.1).

Method for Question 3 We use our framework to train the ResNet-50 and the G-ResNet, STN, and PTN (with as backbones ResNet-18 or ResNet-34). Each of the architectures is trained with `std` as well as `std*` data augmentation. We perform all the spatial attacks listed in Section 2.1 and log the best robust accuracies for each architecture by searching over the initial learning rates 0.1 and 0.01.

After careful consideration, we discovered that using the algorithm proposed in [13] to construct D_R cannot be used to construct a spatially robust dataset D_R^S due to the fact that doing so would essentially be assigning different labels to the same object in different poses, which would not help with training. We also did not construct a spatially non-robust dataset D_{NR}^S since after a careful grid search over several hyperparameters, we were unable to even achieve the results on D_{NR} listed in [13].

Our code framework for studying Question 2 and Question 3 is built on the PyTorch [18] `robustness` library [7].

3 Results

3.1 Answers to Question 1

Figures 1 and 2 show the results of our experiments to determine whether neural networks "overfit" to non-robust features during the later stages of training. We distinguish the case of spatial vs. PGD attacks.

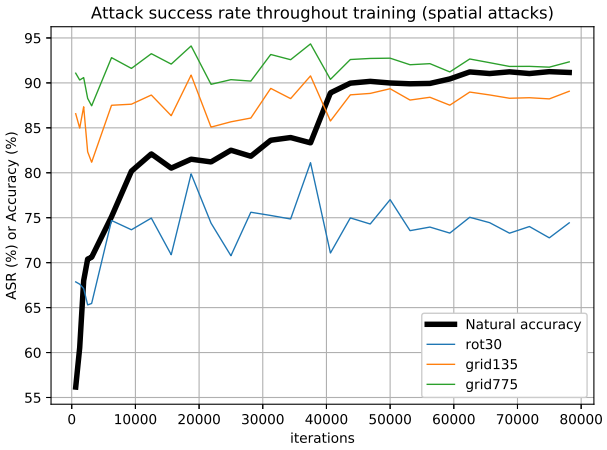

Figure 1: Attack success rates of spatial attacks at various training stages.

For spatial attacks For spatial attacks (Figure 1), we find that the attack success rate (ASR) fluctuates around a fairly constant rate over the training iterations of the neural network. The ASR for `rot` remains around 71-80%, the ASR for `grid135` remains around 84-90%, and the ASR for `grid775` remains around 89-95%. We conclude that in the case of spatial robustness, neural networks typically do not make a distinction between spatially robust and spatially non-robust features during training, instead using all available features from the very beginning of training.

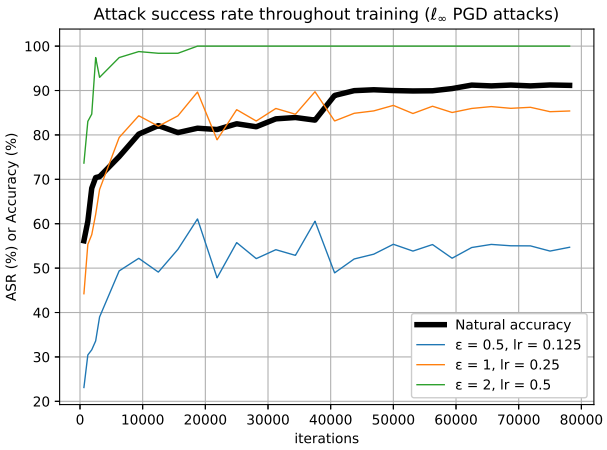

Figure 2: Attack success rates of ℓ_∞PGD attacks at various training stages.

For PGD attacks Contrary to the case of spatial attacks, we see a slight upward trend in the ASR when performing L_∞-norm PGD attacks as training progresses throughout the early stages of training (natural accuracies 60-80%), as shown in Figure 2. After reaching a natural accuracy of roughly 80%, ASR fluctuates around a constant level. We can conclude that at a certain stage of training, the neural network will start using non-robust features to increase natural accuracy.

One possible explanation for the difference in results between spatial and PGD attacks is that spatial attacks are a substantially different class of attacks than PGD attacks. In PGD attacks, there is a constraint based on some norm to limit the attack space, whereas in spatial attacks, the attack space is controlled by a maximum degree of rotation and translation.

In both the spatial and the PGD case, it is clear that neural networks do not specifically over-utilize non-robust features during the later stages of training, as ASR remains fairly constant. We can therefore give the following answer to Question 1: *increasing natural accuracy causes higher usage of PGD-non-robust features, shown by ASR increasing at the beginning of training. The current goal of optimizing accuracy to its peak level is therefore partially at fault for introducing adversarial vulnerabilities to PGD attacks caused by non-robust features. In the case of spatial robustness, we cannot draw any meaningful conclusion.*

3.2 Answers to Question 2

The results of our experiments on the dataset D_{mix} are summarized in Figure 3.

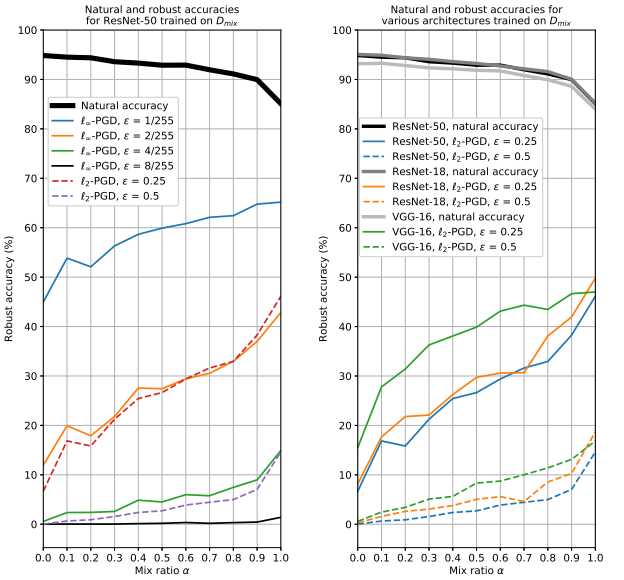

Figure 3: Left: Robust accuracies of ResNet-50 trained using D_{mix} for different values of the mix ration α, where α = 0 represents using only natural images and α = 1 represents using only robust images. Right: Comparison accuracies of the robust accuracies of ResNet-50, ResNet-18, and VGG-16 trained on D_{mix} for different values of α. For all attacks, we use the learning rates from Section 2.1.

Use of D_{mix} as a defense mechanism We observe a clear trend that as the mix ration α increases (i.e. the proportion of images from D_R in each training batch increases), the robust accuracy under all considered attacks increases. It is interesting to note that even adding a small amount of images from D_R (e.g. α = 0 vs. α = 0.1) greatly increases the robust accuracy, while removing all natural images (α = 0.9 vs. α = 1) greatly reduces the natural accuracy. However, even having 10% of natural images per batch only decreases the natural accuracy by around 4% compared to using 100% of natural images. Therefore, using a fixed proportion of images from D_R in each training batch can be considered a fairly effective defense mechanism.

Transferability among different attacks Ilyas et al. only consider ℓ_2-PGD attacks when evaluating their robust dataset

Table 1: Accuracies achieved from the experiments for Question 3. The first row of the table contains natural accuracies, while the remaining rows contain robust accuracies under the attack mentioned in the leftmost column. Each model has four columns of results: "D" refers to models trained with the natural dataset, while "D_R" refers to models trained with the robust dataset. The notation * indicates the model is trained with `std*` data augmentation. The entries reported in this table are the best ones across the backbone ResNet-18/-34 and learning rates. The raw experiment data is logged in Tables 3 and 4.

	ResNet-50				G-ResNet				STN				PTN[1]
Attack	D	D_R	D^*	D_R^*	D	D_R	D^*	D_R^*	D	D_R	D^*	D_R^*	–
None	94.84	84.75	93.9	83.53	**95.72**	85.46	94.74	85.11	94.05	84.39	93.12	83.72	–
`rot10`	80.97	67.78	83.24	69.67	83.56	69.22	**87.61**	73.34	84.07	69.52	87.67	73.55	–
`rot30`	36.77	24.94	58.75	42.62	41.74	26.95	75.38	51.97	41.01	26.4	**86.1**	70.86	–
`grid775,10°`	63.34	52.3	67.78	55.31	68.17	55.15	75.7	60.38	70.47	54.98	**77.01**	60.61	–
`grid135`	18.72	11.23	39.71	26.89	21.84	11.81	56.06	36.64	22.48	11.52	**69.47**	50.76	–
`grid775`	15.28	9.88	35.42	24.53	19.36	10.78	53.19	34.57	19.94	10.09	**67.47**	48.64	–

D_R. Figure 3 (left) shows that D_R is also robust to ℓ_∞-PGD attacks (provided the attack is not too strong, as is the case with $\varepsilon = \frac{8}{255}$). We conclude that the robustness induced by D_R transfers well among different variations of PGD attack.

Transferability among different architectures Ilyas et al. generate the robust dataset D_R using a ResNet-50. It is therefore natural to ask whether their generated dataset also has similar robustness-inducing effects on other architectures. Figure 3 (right) shows the results of two different ℓ_2-PGD attacks on three different architectures. In all three cases, we observe similar trends to ResNet-50 as α increases. We conclude that a robust dataset generated by one architecture can be used to induce robustness for another architecture.

3.3 Answers to Question 3

Table 1 summarizes our results for answering Question 3. The models are evaluated under various spatial attack methods. The natural accuracies match baselines in results of [4, 24]

PGD-robust features do not help boost spatial robustness Ilyas et al. state that training only with the PGD-robust dataset D_R helps the model to obtain a good degree of robustness in general, but our results clearly demonstrate that this robustness does not apply to the spatial case. In table 1 we can see that for all architectures, with the same data augmentation procedure, robust accuracy is much lower than its natural counterpart for every spatial attack setting.

Spatial equivariant networks induce limited spatial robustness with D_R It has been studied in [24] that several spatial equivariant designs help to boost spatial robustness of the models. However, the results we get from training these specially designed architectures both with D and D_R show that D_R does not further improve the robust accuracy, since all of them do not achieve higher accuracy than when simply trained with D.

4 Discussion

Our work reexamined several ideas proposed in [13] under a novel perspective. We try to verify the claimed drawback of the current supervised learning paradigm with a quantitative approach by snapshotting the evolution of the training procedure and measuring ASRs for various attacks. The results we get for Question 1 show a general trend, but fail to further disentangle the effects of robust and non-robust features on the training process. More experimental designs and a finer-grained control over the training procedure are needed for further study.

Furthermore, we proposed a new approach of constructing a training dataset by mixing natural and robust datasets, which can boost robust accuracy greatly almost without sacrificing natural accuracy. Although it could be considered "robustness for free", and it transfers well across attacks under the same class as well as different architectures, the robust accuracy obtained is still below that obtained through state-of-the-art defenses.

Finally, we empirically disproved the possibility that spatial vulnerabilities are caused by PGD-non-robust features, but the reason behind such attacks still remains unknown to us and is out of the scope of this study.

5 Conclusion

In this work, we have studied three questions we raised from the hypothesis of the existence of robust and non-robust features in training data. We first restricted the scope of the study into a smaller set of attacks, by ruling out spatial attacks through monitoring the evolution of ASR with model natural accuracy. We found that the goal of maximizing accuracy is partially responsible for inducing PGD vulnerabilities, but a similar conclusion cannot be drawn for low-dimensional attacks. Furthermore, we found that mixing a certain amount of training examples from the PGD-robust dataset can boost the PGD-robustness by a great degree without substantially hurting natural accuracy, which can be considered as a defense mechanism embedded in the training dataset. This defense transfers well across various first-order attack methods and architectures. Finally, we investigated the relationship between the PGD-robust dataset and spatial equivariant networks and found that it does not help enhance spatial robustness. We speculate that there are no "spatially robust features" or "spatially non-robust features". This also justifies the no-trade-off phenomenon from [24]. The spatial attack evaluation results, together with the conclusion for Question 1, indicates that we should reexamine certain theories which are presumed to hold for both domains.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, et al. TensorFlow: Large-Scale Machine Learning on Het-

[1] For the PTN architecture, due to the training instability for getting the natural accuracy above a meaningful baseline, we omit the results here and leave the results in the appendix.

erogeneous Systems, 2015. Software available from tensorflow.org.

[2] Adam Bielski. pytorch-gconv-experiments. Online: https://github.com/adambielski/pytorch-gconv-experiments, 2013. Accessed 2020-01-05.

[3] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8190 LNAI, pages 387–402, 2013.

[4] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *Proceedings of the International Conference on Machine Learning*, pages 2990–2999, 2016.

[5] Logan Engstrom, Andrew Ilyas, and Anish Athalye. Evaluating and understanding the robustness of adversarial logit pairing. *arXiv preprint arXiv:1807.10272*, 2018.

[6] Logan Engstrom, Andrew Ilyas, Aleksander Madry, Shibani Santurkar, Brandon Tran, and Dimitris Tsipras. A Discussion of 'Adversarial Examples Are Not Bugs, They Are Features': Discussion and Author Responses. *Distill*, 2019. Online: https://distill.pub/2019/advex-bugs-discussion/original-authors. Accessed 2020-01-05.

[7] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, and Dimitris Tsipras. Robustness (Python Library). Online: https://github.com/MadryLab/robustness, 2019. Accessed 2020-01-05.

[8] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the Landscape of Spatial Robustness. In *Proceedings of the International Conference on Machine Learning*, 2019.

[9] Carlos Esteves, Christine Allen-Blanchette, Xiaowei Zhou, and Kostas Daniilidis. Polar Transformer Networks. In *Proceedings of the International Conference on Learning Representations*, 2018.

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

[11] Ghassen Hamrouni. Spatial transformer networks tutorial. Online: https://pytorch.org/tutorials/intermediate/spatial_transformer_tutorial.html, 2018. Accessed 2020-01-05.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Patern Recognition*, pages 770–778, 2016.

[13] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial Examples Are Not Bugs, They Are Features. *arXiv preprint arXiv:1905.02175*, 2019.

[14] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial Transformer Networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.

[15] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial Logit Pairing. *arXiv preprint arXiv:1803.06373*, 2018.

[16] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 4, University of Toronto, 2009.

[17] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *Proceedings of the International Conference on Learning Representations*, 2018.

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'é Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[19] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified Defenses against Adversarial Examples. In *Proceedings of the International Conference on Learning Representations*, 2018.

[20] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations*, 2015.

[21] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *Proceedings of the International Conference on Learning Representations*, 2014.

[22] Kai Sheng Tai, Peter Bailis, and Gregory Valiant. Equivariant Transformer Networks. In *Proceedings of the International Conference on Machine Learning*, 2019.

[23] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5283–5292, 2018.

[24] Fanny Yang, Zuowen Wang, and Christina Heinze-Deml. Invariance-inducing regularization using worst-case transformations suffices to boost accuracy and spatial robustness. *arXiv preprint arXiv:1906.11235*, June 2019.

[25] 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 the International Conference on Machine Learning*, 2019.

Appendices

A Omitted experiment results

A.1 PTN-ResNet

Table 2: Natural accuracies for training PTN-ResNet18 and PTN-ResNet34 with initial learning rates (lr) = 0.01 or 0.005, trained with different datasets and data augmentations.Each model has four columns of results: "D" refers to models trained with the natural dataset, while "D_R" refers to models trained with the robust dataset. The notation * indicates the model is trained with `std`$\star$ data augmentation. We could not get satisfiable natural accuracy over a reasonable level. Thus, we do not evaluate PTN-ResNet under various spatial attacks and we omit obtained natural accuracy in the main text. We search over initial learning rates 0.01, 0.005 and we log the best results in this table.

	PTN-ResNet18				PTN-ResNet34			
	D	D_R	D^*	D_R^*	D	D_R	D^*	D_R^*
lr=0.01	77.73	63.81	74.03	61.53	73.99	62.25	71.51	58.70
lr=0.005	77.22	66.45	75.04	62.80	74.10	61.9	70.45	58.80

A.2 STN-ResNet raw experiment results

Table 3: Accuracies achieved with STN-ResNet18 and STN-ResNet34 architectures. The first row of the table contains natural accuracies, while the remaining rows contain robust accuracies under the attack mentioned in the leftmost column. Each model has four columns of results: "D" refers to models trained with the natural dataset, while "D_R" refers to models trained with the robust dataset. The notation * indicates the model is trained with `std`$\star$ data augmentation. We search over initial learning rates 0.01, 0.005 and we log the best results in this table.

	STN-ResNet-18				STN-ResNet-34			
Attack	D	D_R	D^*	D_R^*	D	D_R	D^*	D_R^*
None	93.55	83.75	92.83	83.05	94.05	84.39	93.12	83.72
`rot10`	83.54	68.66	86.46	73.08	84.07	69.52	87.67	73.55
`rot30`	38.66	24.98	84.72	68.93	41.01	26.4	86.1	70.86
`grid775,10°`	68.57	54.09	75.08	58.47	70.47	54.98	77.01	60.61
`grid135`	20.85	11.52	65.41	49.58	22.48	11.04	69.47	50.76
`grid775`	18.22	9.9	63.48	47.49	19.94	10.09	67.47	48.64

A.3 G-ResNet raw experiment results

Table 4: Accuracies achieved with G-ResNet18 and G-ResNet34 architectures. The first row of the table contains natural accuracies, while the remaining rows contain robust accuracies under the attack mentioned in the leftmost column. Each model has four columns of results: "D" refers to models trained with the natural dataset, while "D_R" refers to models trained with the robust dataset. The notation * indicates the model is trained with `std`$\star$ data augmentation. We search over initial learning rates 0.1, 0.01.

	G-ResNet-18				G-ResNet-34			
Attack	D	D_R	D^*	D_R^*	D	D_R	D^*	D_R^*
None	94.66	85.46	94.74	85.11	**95.72**	84.93	94.72	84.55
`rot10`	83.27	69.22	87.42	72.98	83.56	68.8	**87.61**	73.34
`rot30`	41.74	26.95	69.86	51.97	41.07	26.08	**75.38**	46.47
`grid775,10°`	68.17	54.24	**75.7**	58.96	67.98	55.15	**75.7**	60.38
`grid135`	21.84	11.81	52.51	36.64	20.99	10.99	**56.06**	31.52
`grid775`	19.36	10.78	49.69	34.57	17.82	9.89	**53.19**	29.78

B Training and validation curves while training on D_{mix}