Adversarial robustness and attacks

Dmytro Kuzmenko

Outline

- Overview
- Accuracy vs robustness trade-off
- Local Lipschitzness and r-separability
- State-of-the-art adversarial attacks and attack types
  - L-BFGS
  - JSMA
  - DeepFool
  - FGSM
  - Square and RayS
  - FAB
- Defenses
  - Hedge defense
  - Robust self-training / AT / TRADES
  - Other technique to combat adversary perturbations

Attacking Adversarial Attacks as A Defense

Table 2: Robust accuracy (%) on CIFAR100. All settings align with CIFAR10 in Table 1.

Model	Method	Nat-Acc.	PGD	C&W	Deep Fool	APGD CE	APGD T	FAB	Square	RayS	Auto Attack	Worst Case
WA+ SiLU Gowal et al. (2020)	-	69.15	40.84	39.46	40.74	40.32	37.33	37.65	42.97	43.15	37.29	37.26
	Random	69.02	40.97	43.61	52.22	41.35	38.75	54.59	51.08	55.09	38.79	37.33
	Hedge	68.67	**43.25**	**55.94**	**57.82**	**46.75**	**46.47**	**59.69**	**59.81**	**59.75**	**45.00**	**39.78**
AWP Wu et al. (2020b)	-	60.38	34.13	31.41	31.33	33.37	29.18	29.47	34.57	34.79	29.16	29.15
	Random	60.45	34.21	35.24	44.42	34.53	30.56	45.59	42.94	47.82	30.75	29.25
	Hedge	59.37	**36.69**	**46.01**	**48.13**	**40.18**	**38.10**	**49.49**	**49.82**	**50.34**	**37.75**	**32.29**
TRADES Zhang et al. (2019b)	-	57.34	25.19	32.83	31.97	34.51	37.24	52.74	**54.06**	30.01	33.94	19.61
	Random	55.60	25.30	33.34	33.97	34.11	36.73	51.41	52.36	55.59	33.78	22.00
	Hedge	56.04	**29.75**	**44.09**	**45.33**	**39.25**	**42.04**	**53.27**	54.01	**55.79**	**38.72**	**26.59**
AT Madry et al. (2018)	-	58.61	25.34	35.01	31.93	32.05	35.56	52.62	55.11	32.31	31.70	19.72
	Random	58.49	25.43	35.53	35.54	32.09	35.57	52.58	**55.00**	**58.72**	31.74	21.61
	Hedge	57.66	**28.23**	**43.73**	**45.02**	**35.79**	**40.08**	**53.24**	54.99	57.22	**35.54**	**24.80**

A Closer Look at Accuracy vs. Robustness

Robustness and Astuteness. Let $\mathbb{B}(\boldsymbol{x}, \varepsilon)$ denote a ball of radius $\varepsilon > 0$ around $\boldsymbol{x}$ in a metric space. We use $\mathbb{B}_\infty$ to denote the ℓ_∞ ball. A classifier g is *robust* at $\boldsymbol{x}$ with radius $\varepsilon > 0$ if for all $\boldsymbol{x}' \in \mathbb{B}(\boldsymbol{x}, \varepsilon)$, we have $g(\boldsymbol{x}') = g(\boldsymbol{x})$. Also, g is *astute* at $(\boldsymbol{x}, y)$ if $g(\boldsymbol{x}') = y$ for all $\boldsymbol{x}' \in \mathbb{B}(\boldsymbol{x}, \varepsilon)$. The *astuteness* of g at radius $\varepsilon > 0$ under a distribution μ is

$$\Pr_{(\boldsymbol{x},y)\sim\mu}[g(\boldsymbol{x}') = y \text{ for all } \boldsymbol{x}' \in \mathbb{B}(\boldsymbol{x}, \varepsilon)].$$

The goal of robust classification is to find a g with the highest astuteness [59]. We sometimes use *clean accuracy* to refer to standard test accuracy (no adversarial perturbation), in order to differentiate it from *robust accuracy* a.k.a. astuteness (with adversarial perturbation).

We take a closer look at the tradeoff between robustness and accuracy, aiming to identify properties of data and training methods that enable neural networks to achieve *both*. A plausible reason why robustness may lead to lower accuracy is that different classes are very close together or they may even overlap (which underlies the argument for an inevitable tradeoff [57]). We begin by testing if this is the case in real data through an empirical study of four image datasets. Perhaps surprisingly, we find that these datasets actually satisfy a natural separation property that we call r-separation: examples from different classes are at least distance $2r$ apart in pixel space. This r-separation holds for values of r that are higher than the perturbation radii used in adversarial example experiments.

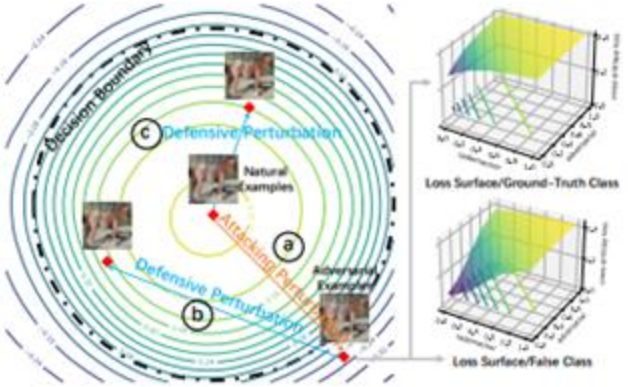


Figure 1: **Left:** The loss landscape Li et al. (2018) around a natural example. An attacking perturbation intends to push the example out of the decision boundary (dot-dashed line), while a defensive perturbation intends to pull it back. **Right:** For the adversarial example on the left, we plot its two loss surfaces for the class of the false prediction and the ground-truth class.

Local Lipschitzness and r-separability

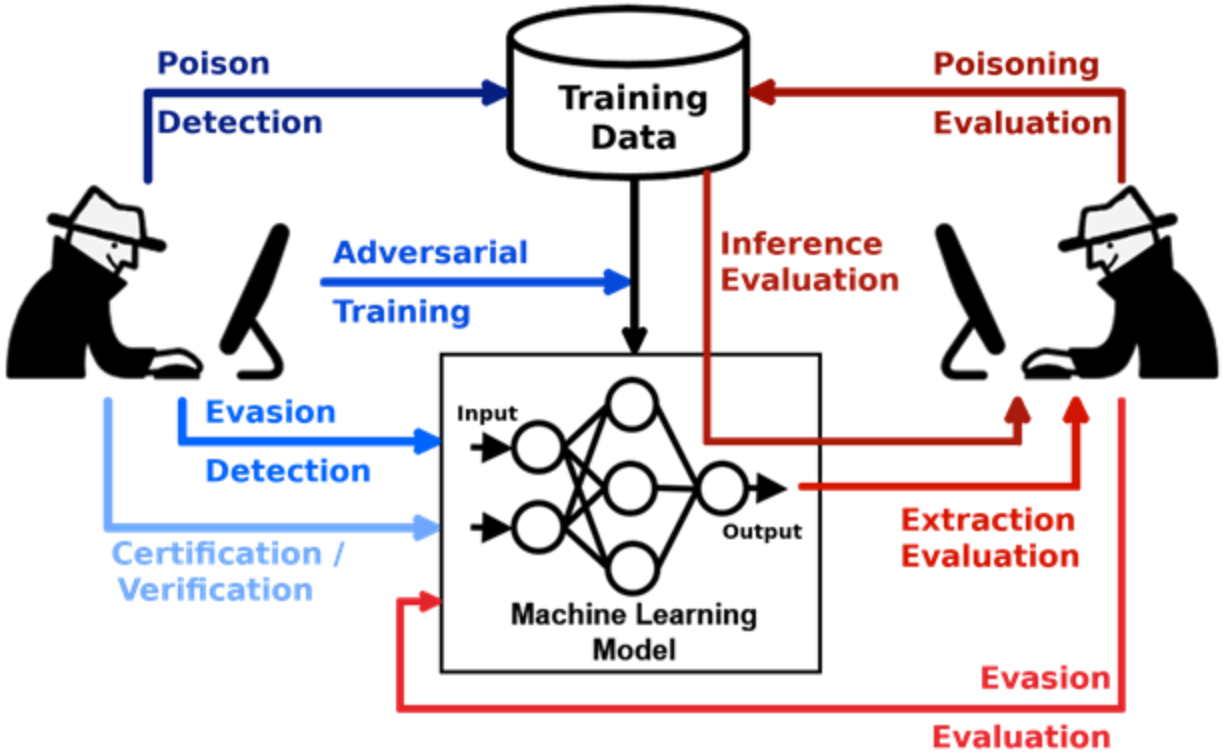
Definition 1. *Let* $(\mathcal{X}, \mathsf{dist})$ *be a metric space. A function* $f : \mathcal{X} \to \mathbb{R}^C$ *is* L*-locally Lipschitz at radius* r *if for each* $i \in [C]$*, we have* $|f(\boldsymbol{x})_i - f(\boldsymbol{x}')_i| \leq L \cdot \mathsf{dist}(\boldsymbol{x}, \boldsymbol{x}')$ *for all* $\boldsymbol{x}'$ *with* $\mathsf{dist}(\boldsymbol{x}, \boldsymbol{x}') \leq r$.

Separation. We formally define separated data distributions as follows. Let $\mathcal{X}$ contain C disjoint classes $\mathcal{X}^{(1)}, \ldots, \mathcal{X}^{(C)}$, where all points in $\mathcal{X}^{(i)}$ have label i for $i \in [C]$.

Definition 2 (r-separation)**.** *We say that a data distribution over* $\bigcup_{i \in [C]} \mathcal{X}^{(i)}$ *is* r*-separated if* $\mathsf{dist}(\mathcal{X}^{(i)}, \mathcal{X}^{(j)}) \geq 2r$ *for all* $i \neq j$*, where* $\mathsf{dist}(\mathcal{X}^{(i)}, \mathcal{X}^{(j)}) = \min_{\boldsymbol{x} \in \mathcal{X}^{(i)}, \boldsymbol{x}' \in \mathcal{X}^{(j)}} \mathsf{dist}(\boldsymbol{x}, \boldsymbol{x}')$.

In other words, the distance between any two examples from different classes is at least $2r$. One of our motivating observations is that many real classification tasks comprise of separated classes; for example, if dist is the ℓ_∞ norm, then images with different categories (e.g., dog, cat, panda, etc) will be r-separated for some value $r > 0$ depending on the image space. In the next section, we empirically verify that this property actually holds for a number of standard image datasets.

Adversarial attacks



Types of attacks

White-box attacks with PGD for a different number of of iterations and restarts, denoted by source A.

White-box attacks with PGD using the Carlini-Wagner (CW) loss function [6] (directly optimizing the difference between correct and incorrect logits). This is denoted as CW, where the corresponding attack with a high confidence parameter ($\kappa = 50$) is denoted as CW+.

Black-box attacks from an independently trained copy of the network, denoted A'.

Black-box attacks from a version of the same network trained only on natural examples, denoted A_{nat}.

Black-box attacks from a different convolution architecture, denoted B, described in Tramer et al. 2017 [29].

State-of-the-art attacks

- FGSM
- L-BFGS
- PGD
- Square
- RayS
- FAB (later Deepfool)
- C&W

L-BFGS

A. L-BFGS

Szegedy *et al.* [46] generated adversarial examples using box-constrained L-BFGS. Given an image x, their method finds a different image x' that is similar to x under L_2 distance, yet is labeled differently by the classifier. They model the problem as a constrained minimization problem:

$$\begin{aligned} \text{minimize} \quad & \|x - x'\|_2^2 \\ \text{such that} \quad & C(x') = l \\ & x' \in [0,1]^n \end{aligned}$$

This problem can be very difficult to solve, however, so Szegedy *et al.* instead solve the following problem:

$$\begin{aligned} \text{minimize} \quad & c \cdot \|x - x'\|_2^2 + \text{loss}_{F,l}(x') \\ \text{such that} \quad & x' \in [0,1]^n \end{aligned}$$

where $\text{loss}_{F,l}$ is a function mapping an image to a positive real number. One common loss function to use is cross-entropy. Line search is performed to find the constant $c > 0$ that yields an adversarial example of minimum distance: in other words, we repeatedly solve this optimization problem for multiple values of c, adaptively updating c using bisection search or any other method for one-dimensional optimization.

Jacobian-based Saliency Map Attack (JSMA)

At a high level, the attack is a greedy algorithm that picks pixels to modify one at a time, increasing the target classification on each iteration.

They use the gradient to compute a **saliency map**, which models the impact each pixel has on the resulting classification.

A large value indicates that changing it will significantly increase the likelihood of the model labeling the image as the target class *l*. Given the saliency map, it picks the most important pixel and modify it to increase the likelihood of class *l*.

This is repeated until either more than a set threshold of pixels are modified which makes the attack detectable, or it succeeds in changing the classification.

DeepFool (later evolves into FAB)

Deepfool is an untargeted attack technique optimized for the L2 distance metric. It is efficient and produces closer adversarial examples than the L-BFGS approach discussed earlier.

The authors construct Deepfool by imagining that the neural networks are totally linear, with a hyperplane separating each class from another.

From this, they analytically derive the optimal solution to this simplified problem, and construct the adversarial example.

Then, since neural networks are not actually linear, they take a step towards that solution, and repeat the process a second time. The search terminates when a true adversarial example is found.

DeepFool in a nutshell

Algorithm 1 DeepFool for binary classifiers

input: Image $\boldsymbol{x}$, classifier f.
output: Perturbation $\hat{\boldsymbol{r}}$.
Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
while $\text{sign}(f(\boldsymbol{x}_i)) = \text{sign}(f(\boldsymbol{x}_0))$ **do**
    $\boldsymbol{r}_i \leftarrow -\frac{f(\boldsymbol{x}_i)}{\|\nabla f(\boldsymbol{x}_i)\|_2^2} \nabla f(\boldsymbol{x}_i)$,
    $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$,
    $i \leftarrow i + 1$.
end while
return $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$.

FGSM

Fast gradient sign method

The fast gradient sign method works by using the gradients of the neural network to create an adversarial example. For an input image, the method uses the gradients of the loss with respect to the input image to create a new image that maximises the loss. This new image is called the adversarial image. This can be summarised using the following expression:

$$adv\_x = x + \epsilon * \text{sign}(\nabla_x J(\theta, x, y))$$

Square and RayS

Square and RayS are black-box attacks based on extensively querying deep networks.

They tend to stop queries once they find a successful adversarial example so that the computation budget can be saved.

Yet, such a design makes attacks stop at the adversarial examples that are closer to the unharmful example, which is the example one step earlier before the successful attacking query.

This explains why they are so vulnerable to defensive perturbation.

The black-box constraint of not knowing the gradient information of the model may also prevent them from finding more powerful adversarial examples that are far away from the decision boundary.

FAB

$$\textbf{FAB: } \mathbf{x}_{\text{fab}} = \underset{\mathbf{x}'}{\operatorname{argmin}} \|\mathbf{x}' - \mathbf{x}\|_p \quad s.t. \underset{c}{\operatorname{argmax}} f_c(\mathbf{x}') \neq y.$$

FAB can be considered as a direct improvement of DeepFool.

Unlike other white-box attacks that search adversarial examples within B(x, a), FAB tries to find a minimal perturbation for attacks (as shown in the picture).

However, this aim of finding smaller attacking perturbations also encourages them to find adversarial examples that are too close to the decision boundary and thus become very vulnerable to defensive perturbation.

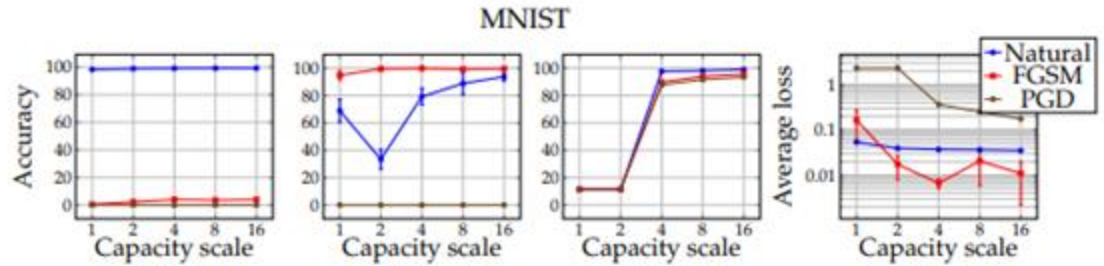
Thus, even on the undefended standard model where they generate smaller perturbations than on robust models, they can still be hugely nullified by random perturbation, and the robust accuracy increases from less than 1.0% to 74.57% and 84.37%.

PGD and C&W

Attacks like PGD and C&W show better resistance against perturbations. However, with Hedge Defense, the robust accuracy can still by improved by 3.0% to 11.0%.

Interestingly, several commonly believed more powerful attacks turn out to be weaker when facing perturbations.

Adversarial attacks in action



CIFAR10

	Simple	Wide
Natural	92.7%	95.2%
FGSM	27.5%	32.7%
PGD	0.8%	3.5%

(a) Standard training

	Simple	Wide
Natural	87.4%	90.3%
FGSM	90.9%	95.1%
PGD	0.0%	0.0%

(b) FGSM training

	Simple	Wide
Natural	79.4%	87.3%
FGSM	51.7%	56.1%
PGD	43.7%	45.8%

(c) PGD training

	Simple	Wide
Natural	0.00357	0.00371
FGSM	0.0115	0.00557
PGD	1.11	0.0218

(d) Training Loss

Figure 4: The effect of network capacity on the performance of the network. We trained MNIST and CIFAR10 networks of varying capacity on: (a) natural examples, (b) with FGSM-made adversarial examples, (c) with PGD-made adversarial examples. In the first three plots/tables of each dataset, we show how the standard and adversarial accuracy changes with respect to capacity for each training regime. In the final plot/table, we show the value of the cross-entropy loss on the adversarial examples the networks were trained on. This corresponds to the value of our saddle point formulation (2.1) for different sets of allowed perturbations.

Hedge defense

Algorithm 1 Hedge Defense

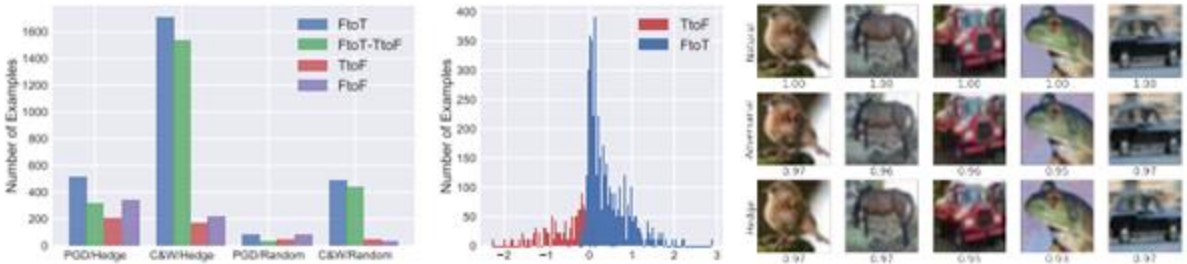
Input: the coming input $\mathbf{x}'$, the number of iterations K, the step size η, the deep network $f(\cdot)$, and the defensive radius ϵ_{d}.
// $\mathcal{U}(-\mathbf{1}, \mathbf{1})$ generates a uniform noise
Initialization: $\mathbf{x}''_0 \leftarrow \mathbf{x}' + \epsilon_{\mathrm{d}}\, \mathcal{U}(-\mathbf{1}, \mathbf{1})$.
for $k = 1 \ldots K$ **do**
    // Sum the losses for all the classes,
    // then update $\mathbf{x}''_k$ with gradient ascending.
    $\mathbf{x}''_k \leftarrow \mathbf{x}''_{k-1} +$
        $\eta \cdot \mathrm{sign}(\nabla_{\mathbf{x}''_{k-1}} \sum_{c=1}^{C} \mathcal{L}(f(\mathbf{x}''_{k-1}), c))$;
    // Π is the projection operator.
    $\mathbf{x}''_k \leftarrow \Pi_{\mathbb{B}(\mathbf{x}', \epsilon_{\mathrm{d}})}(\mathbf{x}''_k)$;
end for
Output: the safer prediction $f(\mathbf{x}''_K)$.

Hedge defense

Hedge Defense again achieves huge improvements. In particular, it improves the state-of-the-art model against AutoAttack from 37.29% to 45.00%.

Table 3: Top-1 robust accuracy (%) for Fast Adversarial Training on ImageNet.

Method	Model	Hedge	$\epsilon_a = 2/255$			$\epsilon_a = 4/255$		
			PGD-10	PGD-50	PGD-100	PGD-10	PGD-50	PGD-100
Fast AT Wong et al. (2020)	ResNet-50	-	43.44	43.40	43.38	30.81	30.17	30.13
		✓	**45.38**	**45.44**	**45.43**	**34.42**	**34.63**	**34.71**
	ResNet-101	-	44.69	44.62	44.60	34.02	33.26	33.18
		✓	**47.00**	**47.07**	**47.08**	**38.36**	**38.50**	**38.54**



(a) The Trade-off of Defensive Perturbation (b) Lipschitzness Difference (c) Visualization of Examples

Figure 2: Analytical Experiments: (a) The number of examples for FtoT, TtoF, FtoT-TtoF, and FtoF (Section 4.2.1). (b) The distribution of Lipschitzness difference for FtoT and TtoF (Section 4.2.2). (c) The visualization for natural, adversarial, and hedge examples with their SSIM scores (Section 4.2.3).

Defences against adversarial attacks

To better understand the theory-practice gap, several existing methods on a few image datasets with a special focus on their local Lipschitzness and generalization gaps were investigated.

There are 3 ways to train a model to be adversary robust:

- adversarial training (AT),
- robust self-training (RST)
- TRADES

Robust self-training

TRADES (Zhang et al., 2019) as the robust loss in the general RST formulation (10); we additionally evaluate RST with Projected Gradient Adversarial Training (AT) (Madry et al., 2018) as the robust loss. Carmon et al. (2019) considered ℓ_∞ and ℓ_2 perturbations. We study rotations and translations in addition to ℓ_∞ perturbations, and also study the effect of labeled training set size on standard and robust error. Table 1 presents the main results. More experiment details appear in Appendix D.3.

Both RST+AT and RST+TRADES have lower robust and standard error than their supervised counterparts AT and TRADES across all perturbation types. This mirrors the theoretical analysis of RST in linear regression (Theorem 2) where the RST estimator has small robust error while provably not sacrificing standard error, and never obtaining larger standard error than the standard estimator.

	Standard	Robust
Labeled	$(y - x^\top\theta)^2$ Noiseless targets	$\max_{x_{\text{adv}} \in T(x)} (x^\top\theta - x_{\text{adv}}^\top\theta)^2$ Consistent perturbations
Unlabeled	$(\tilde{y} - \tilde{x}^\top\theta)^2$ Imperfect pseudo-labels	$\max_{\tilde{x}_{\text{adv}} \in T(\tilde{x})} (\tilde{x}^\top\theta - \tilde{x}_{\text{adv}}^\top\theta)^2$ Consistent perturbations

Figure 5. Illustration shows the four components of the RST loss (Equation (10)) in the special case of linear regression (Eq. (11)). Green cells contain hard constraints where the optimal $\theta^\star$ obtains zero loss. The orange cell contains the soft constraint that is minimized while satisfying hard constraints to obtain the final linear RST estimator.

Other technique to combat adversary perturbations

- Projection - The development of generative models, such as auto-encoders and generative adversarial networks, gives rise to another line of research, which removes adversarial noise by fitting generative models on the training data.

  The input to the classifier is first fed into the generative model and then classified. Since the generative model is trained on natural examples, adversarial examples will be projected to the manifold learned by the generative model.

  Furthermore, “projecting” the adversarial examples onto the range of the generative model can have the desirable effect of reducing the adversarial perturbation.

- Detection (as simple as that) - training a plain binary classifier. The approach lacks in generalization, though.

Randomization as defense

Adversarial perturbation can be viewed as noise, and various methods have been proposed to improve the robustness of DNNs by incorporating random components into the model.

The randomness can be introduced to:

- the input; i.e., randomizing the input of a neural network to remove the potential adversarial perturbation (e.g., Xie et al., 2018; Cohen et al., 2019),
- the hidden layer output; i.e., adding Gaussian noise to the input and hidden output (e.g., Liu et al., 2018b) and introducing pruning methods to randomize the network output (e.g., Dhillon et al., 2018), or
- the parameters of the classifier; i.e., leveraging a Bayesian component to add randomness to the weights of the model (e.g., Liu et al., 2019).

More regarding adversarial training

Weak models may fail to learn non-trivial classifiers. In the case of small capacity networks, attempting to train against a strong adversary (PGD) prevents the network from learning anything meaningful.

The network converges to always predicting a fixed class, even though it could converge to an accurate classifier through standard training.

The small capacity of the network forces the training procedure to sacrifice performance on natural examples in order to provide any kind of robustness against adversarial inputs.