IEEE Symposium on Security & Privacy 2016

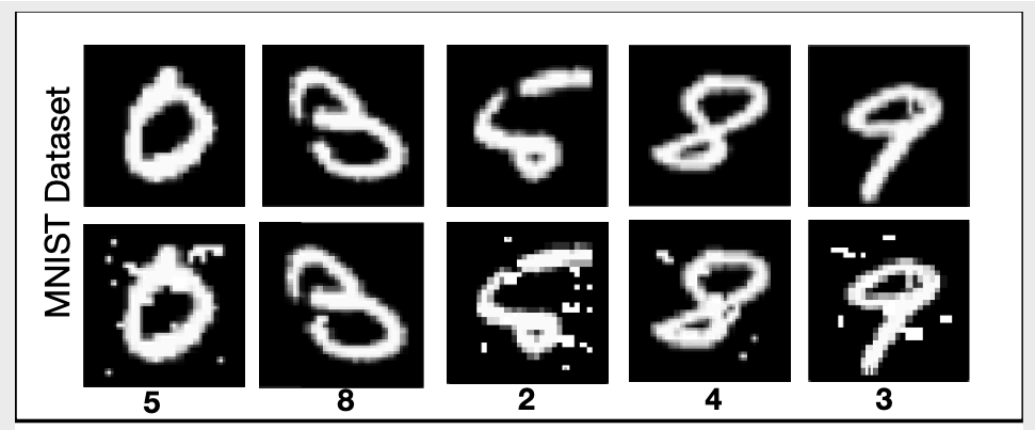
Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks

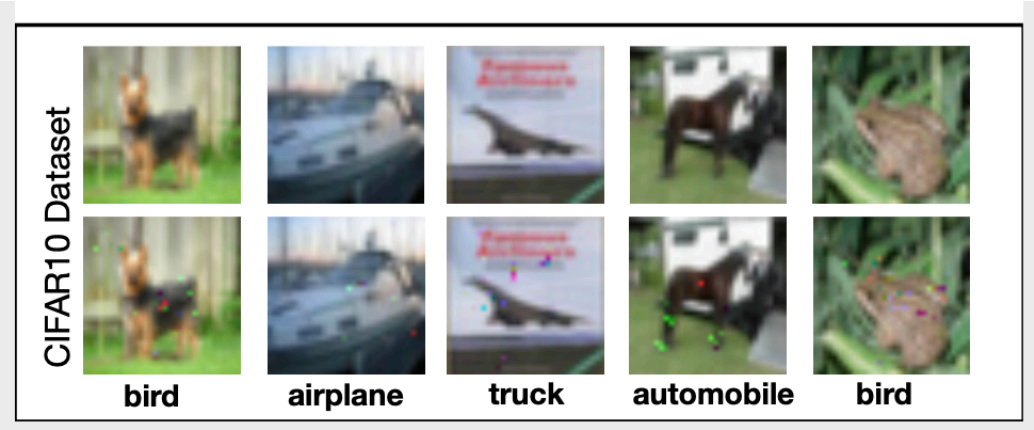
Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami
Penn State University; University of Wisconsin-Madison;
United States Army Research Laboratory

Slides compiled by Mengzhou Li

Introduction

- Deep learning is vulnerable to adversarial samples, and the attacks can seriously undermine the security of the system supported by the DNN.
- Distillation is popular in transferring knowledge from a complex DNN to a smaller one.
- The authors found a *defensive distillation* mechanism can effectively increase the resilience of DNNs to the adversarial attacks.



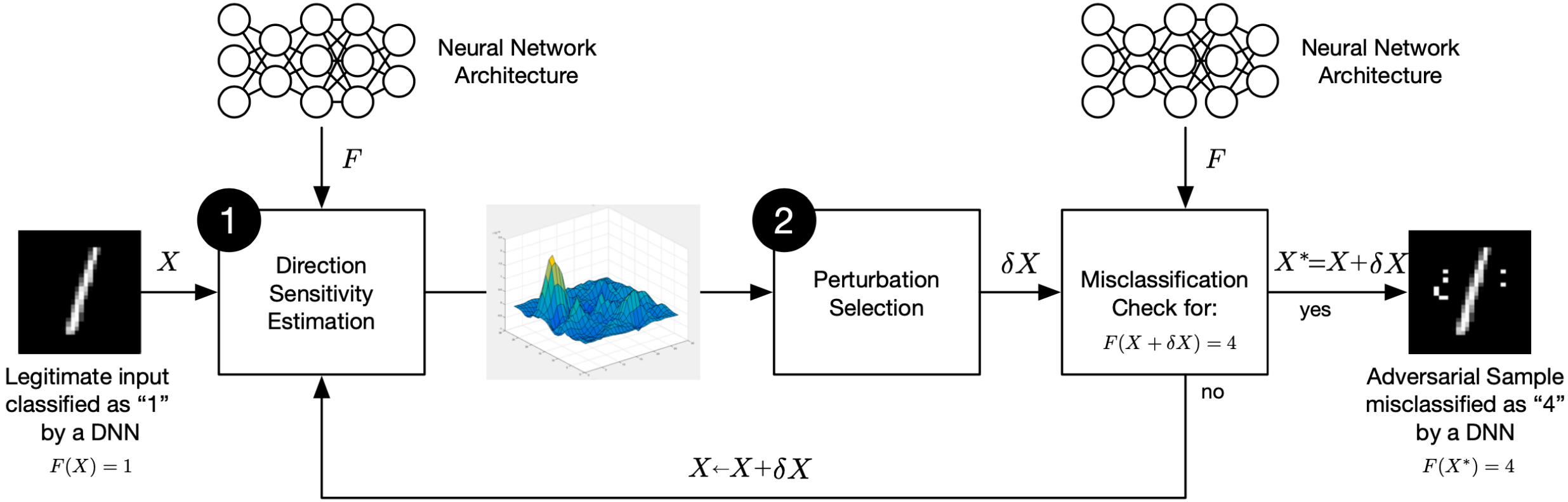


Contributions

- Introduce *defensive distillation* to improve the robustness of DNN-based classifier models
- Analytically investigate *defensive distillation* as a security countermeasure
- Experimentally demonstrate that *defensive distillation* reduces the success rate of adversarial sample crafting from 95% to 0.5%, and the sensitivity to input perturbations by a factor of 10^{30}

Adversarial Sample Crafting

The general framework is split into two folds: ***direction sensitivity estimation*** and ***perturbation selection***.



Adversarial Sample Crafting

The goal of attacks in this paper is to misclassify samples from a specific ***source class*** into a distinct ***target class***. (one of the strongest attacks)

$$\arg\min_{\delta X} \|\delta X\| \text{ s.t. } F(X+\delta X) = Y^*$$

Direction sensitivity Estimation:

(1) Jacobian of F, dF_i/dX_j, j = 1, 2, …, M (#pixels), i = 1, 2, …, N (#classes)

(2) Saliency Map, S(X, t) , M vector, to select the potential pixels to perturb

For each pixel j of X, target gradient dF_t/dX_j decides whether the perturb is in the right direction, positive is good;

the sum of the other gradients $\sum_{i\neq t} dF_i/dX_j$ indicates whether interferences exist, negative is good meaning no other interferences;

S(X, t)[j] = dF_t/dX_j | $\sum_{i\neq t} dF_i/dX_j$ | if the target gradient > 0 and no interferences

Adversarial Sample Crafting

Perturbation Selection:

(1) Select a limit number of input dimensions to perturb according to the Saliency Map (unnoticeable perturbations).

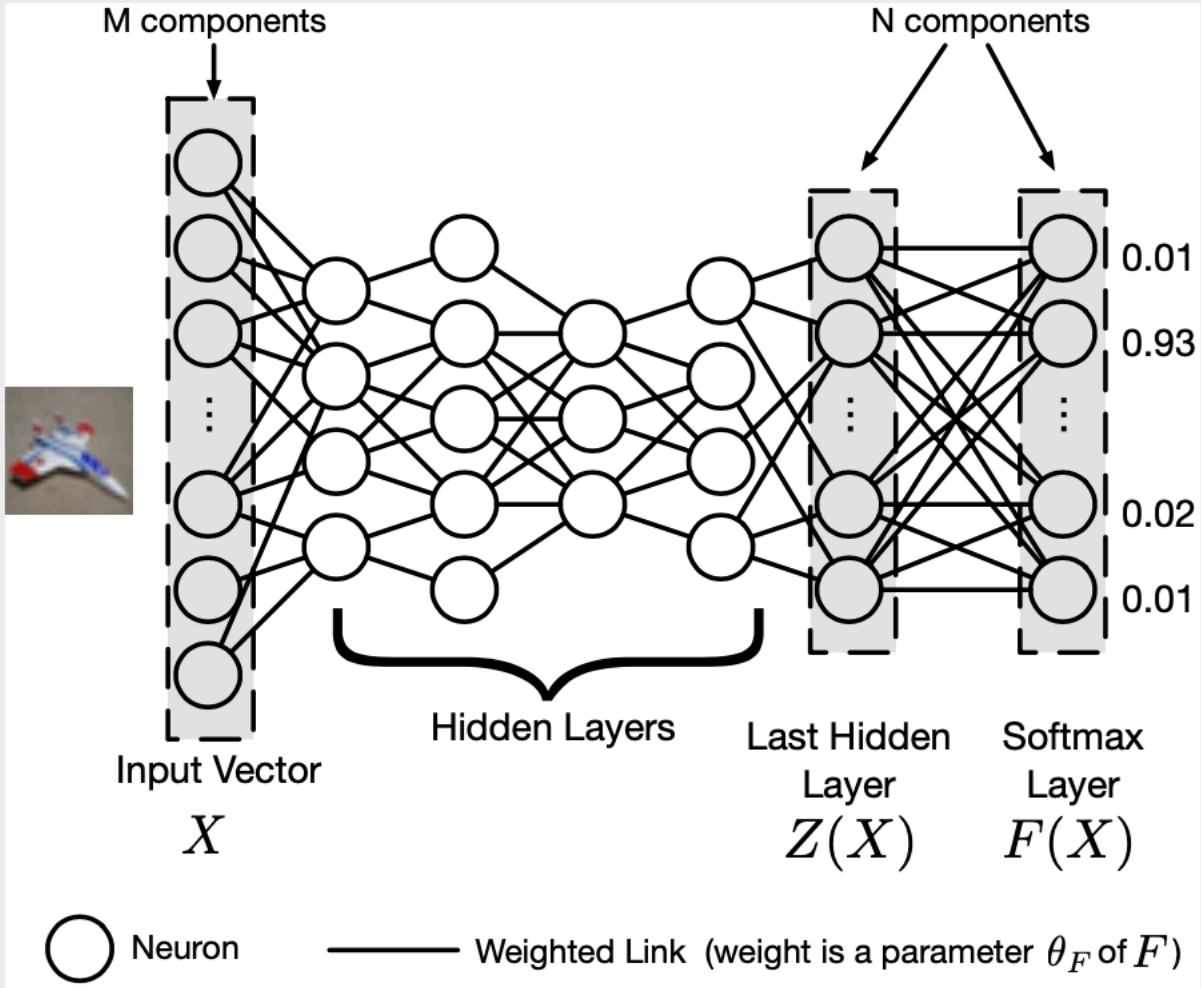
(2) The amplitude of the perturbation added to each pixel is fixed.

The two-step process can be iterative.

The success of attack is highly related to the computed gradients in direction sensitivity estimation, adversarial gradients.

High gradients => small perturbations will induce high output variations. In other words, smoothing the model can help defending the perturbations.

Network Distillation



The general intuition is to extract class probability vectors produced by a first DNN to train a second DNN of reduced complexity without loss of accuracy.

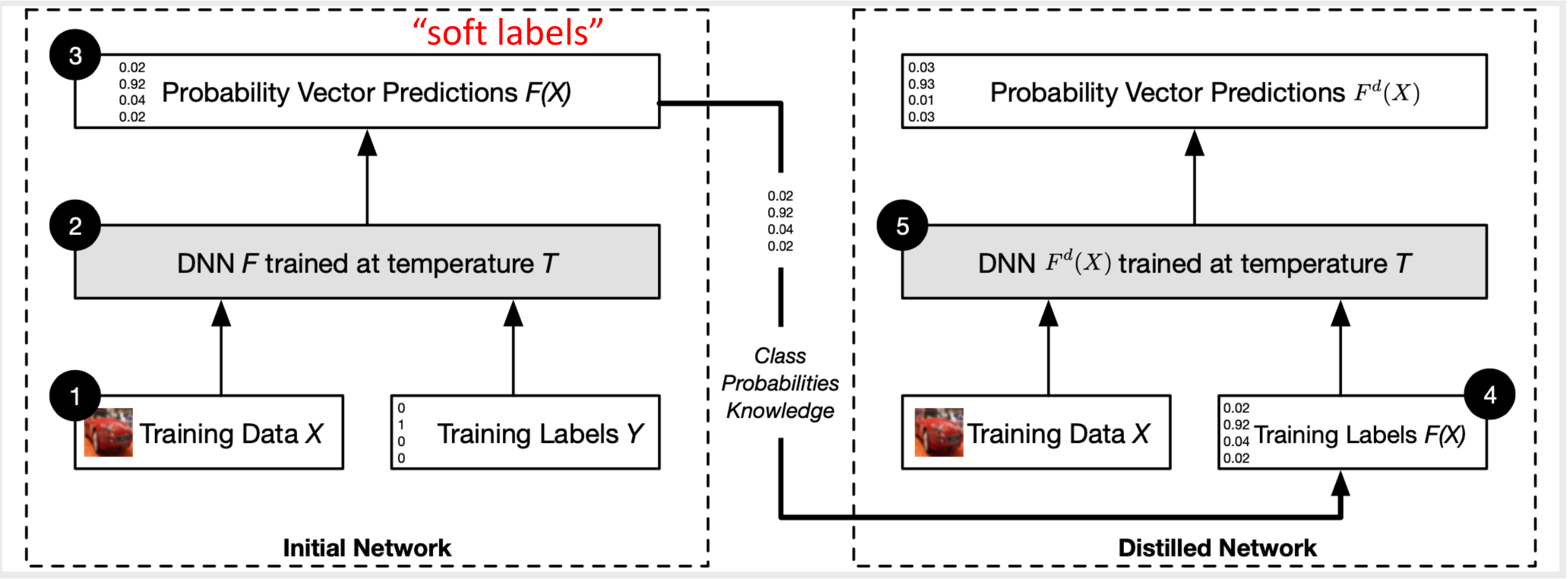
$$F(X) = \left[\frac{e^{z_i(X)/T}}{\sum_{l=0}^{N-1} e^{z_l(X)/T}} \right]_{i \in 0..N-1}$$

T is the distillation temperature.

High *T* => more spread probability distr.

Low *T* => concentrated probabilities

Network Distillation



This figure is actually defensive distillation, but here used for illustration of the general training process of distillation.

Defensive Distillation

The main difference compared to the original distillation is that the **same network architecture** is used to train both the original network as well as the distilled network. The target here is resilience instead of compression.

How this works:

(1) Distillation at high temperatures improves the smoothness of the distilled model F^d (reduce adversarial gradients or Jacobian).

$$\left.\frac{\partial F_i(X)}{\partial X_j}\right|_T = \frac{\partial}{\partial X_j}\left(\frac{e^{z_i(X)/T}}{\sum_{l=0}^{N-1} e^{z_l(X)/T}}\right)$$

$$g(X) = \sum_{l=0}^{N-1} e^{z_l(X)/T}$$

$$\Rightarrow = \frac{1}{T}\frac{e^{z_i/T}}{g^2(X)}\left(\sum_{l=0}^{N-1}\left(\frac{\partial z_i}{\partial X_j} - \frac{\partial z_l}{\partial X_j}\right)e^{z_l/T}\right)$$

$$\begin{aligned}
\left.\frac{\partial F_i(X)}{\partial X_j}\right|_T &= \frac{\partial}{\partial X_j}\left(\frac{e^{z_i/T}}{\sum_{l=0}^{N-1} e^{z_l/T}}\right) \\
&= \frac{1}{g^2(X)}\left(\frac{\partial e^{z_i(X)/T}}{\partial X_j} g(X) - e^{z_i(X)/T}\frac{\partial g(X)}{\partial X_j}\right) \\
&= \frac{1}{g^2(X)}\frac{e^{z_i/T}}{T}\left(\sum_{l=0}^{N-1}\frac{\partial z_i}{\partial X_j}e^{z_l/T} - \sum_{l=0}^{N-1}\frac{\partial z_l}{\partial X_j}e^{z_l/T}\right) \\
&= \frac{1}{T}\frac{e^{z_i/T}}{g^2(X)}\left(\sum_{l=0}^{N-1}\left(\frac{\partial z_i}{\partial X_j} - \frac{\partial z_l}{\partial X_j}\right)e^{z_l/T}\right)
\end{aligned}$$

Defensive Distillation

How this works:

(2) One hot map labels forces the DNN to make overly confident predictions, and “this is a fundamental lack of precision during training as most of the architecture remains unconstrained as weights are updated”.

Entropy loss: $\arg\min_{\theta_F} -\frac{1}{|\mathcal{X}|}\sum_{X\in\mathcal{X}}\sum_{i\in 0..N} Y_i(X)\log F_i(X)$

With one-hot map labels: **=>** $\arg\min_{\theta_F} -\frac{1}{|\mathcal{X}|}\sum_{X\in\mathcal{X}} \log F_{t(X)}(X)$

With “soft labels” of probabilities: $\arg\min_{\theta_F} -\frac{1}{|\mathcal{X}|}\sum_{X\in\mathcal{X}}\sum_{i\in 0..N} F_i(X)\log F_i^d(X)$

Defensive Distillation

How this works:

(2) One hot map labels forces the DNN to make overly confident predictions, and “this is a fundamental lack of precision during training as most of the architecture remains unconstrained as weights are updated”.

The relative information about classes **prevents** the distilled model F^d **fitting too tightly** to the data, and contributes to a **better generalization** around training points (more robust).

e.g., digits 8 and 3 share the same structure “3”, hence, when given an input 3, it is more natural that the output probability for 8 has a small value rather than totally 0.

Experiments

- Dataset: MNIST and CIFAR10
- Architectures and training parameters

Layer Type	MNIST Architecture	CIFAR10 Architecture
Relu Convolutional	32 filters (3x3)	64 filters (3x3)
Relu Convolutional	32 filters (3x3)	64 filters (3x3)
Max Pooling	2x2	2x2
Relu Convolutional	64 filters (3x3)	128 filters (3x3)
Relu Convolutional	64 filters (3x3)	128 filters (3x3)
Max Pooling	2x2	2x2
Relu Fully Connect.	200 units	256 units
Relu Fully Connect.	200 units	256 units
Softmax	10 units	10 units

Parameter	MNIST Architecture	CIFAR10 Architecture
Learning Rate	0.1	0.01 (decay 0.5)
Momentum	0.5	0.9 (decay 0.5)
Decay Delay	-	10 epochs
Dropout Rate (Fully Connected Layers)	0.5	0.5
Batch Size	128	128
Epochs	50	50

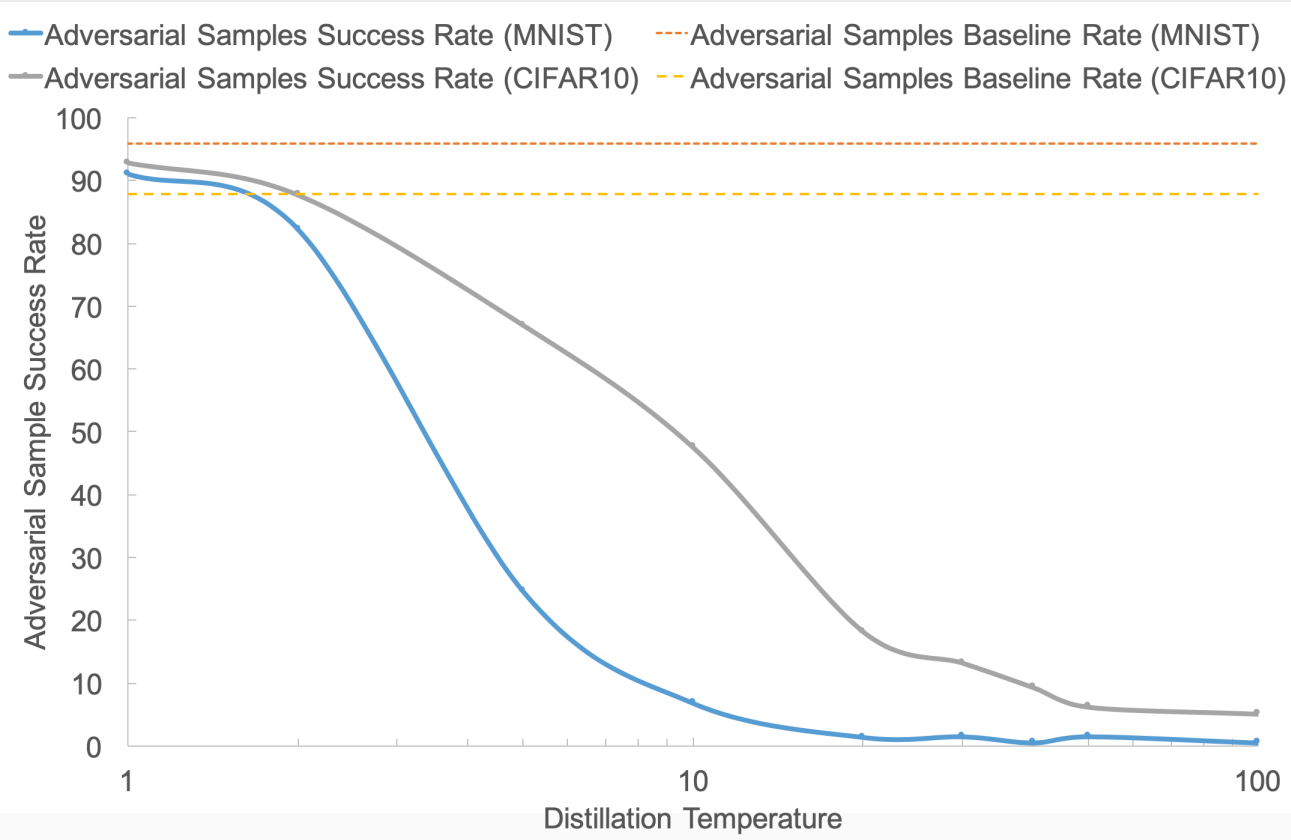
Impact on Adversarial Crafting

- If the # pixels perturbed is larger than 112, stop and the attack is considered failed.
- 100 samples randomly selected, 9 adversarial samples generated for each considered sample. Totally, 900 samples for each model generated.
- Distillation temperature T = 20,
  Success attack rate for MNIST: original **95.89% => 1.34%**
  Success attack rate for CIFAR10: original **89.9% => 16.76%**

Impact on Adversarial Crafting

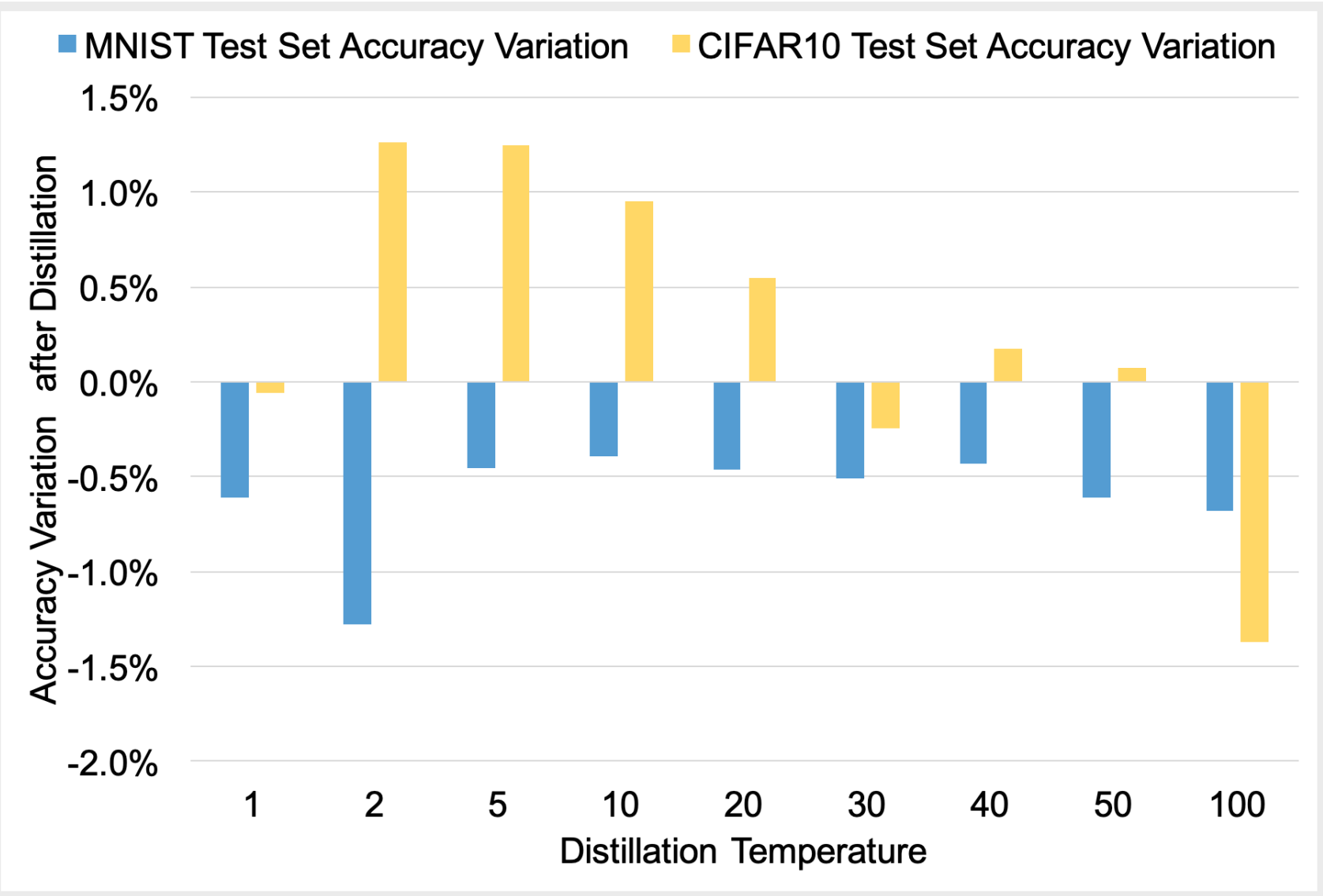
- Temperature

  (1) Higher *T* => Harder to attack; (2) elbow points after which the rate remains const.



Distillation Temperature	MNIST Adversarial Samples Success Rate (%)	CIFAR10 Adversarial Samples Success Rate (%)
1	91	92.78
2	82.23	87.67
5	24.67	67
10	6.78	47.56
20	1.34	18.23
30	1.44	13.23
40	0.45	9.34
50	1.45	6.23
100	0.45	5.11
No distillation	95.89	87.89

Impact on Accuracy

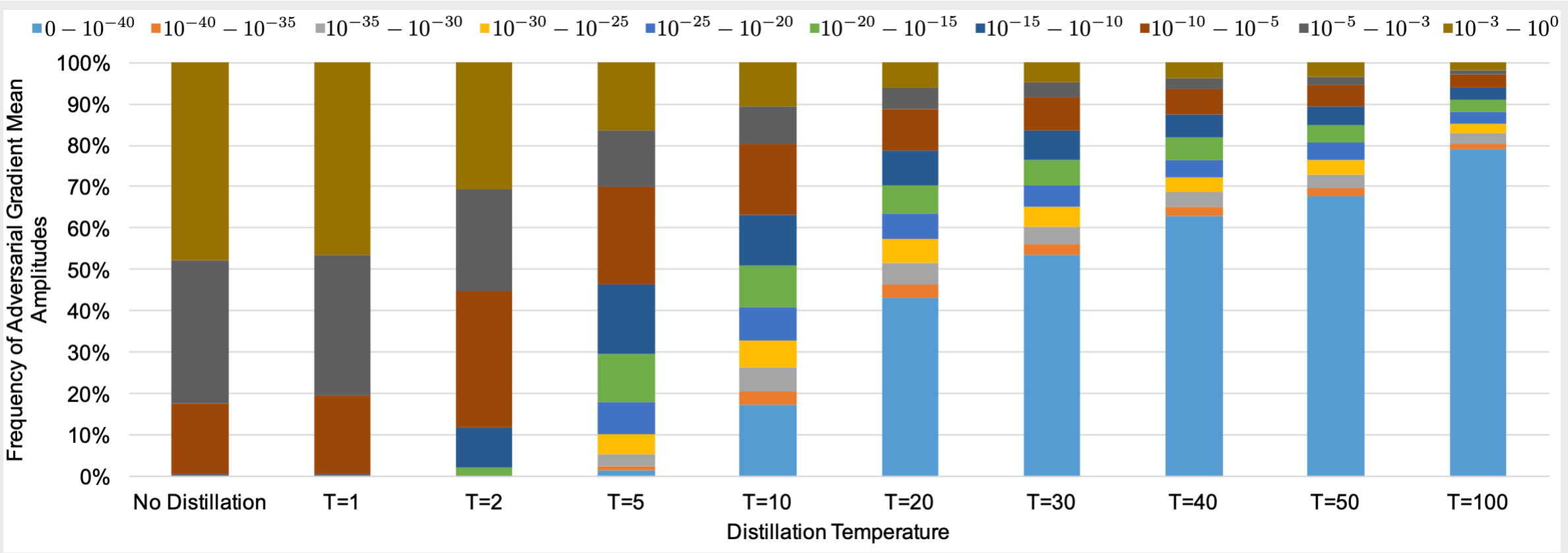


Baselines:
99.51% for MNIST
80.95% for CIFAR10

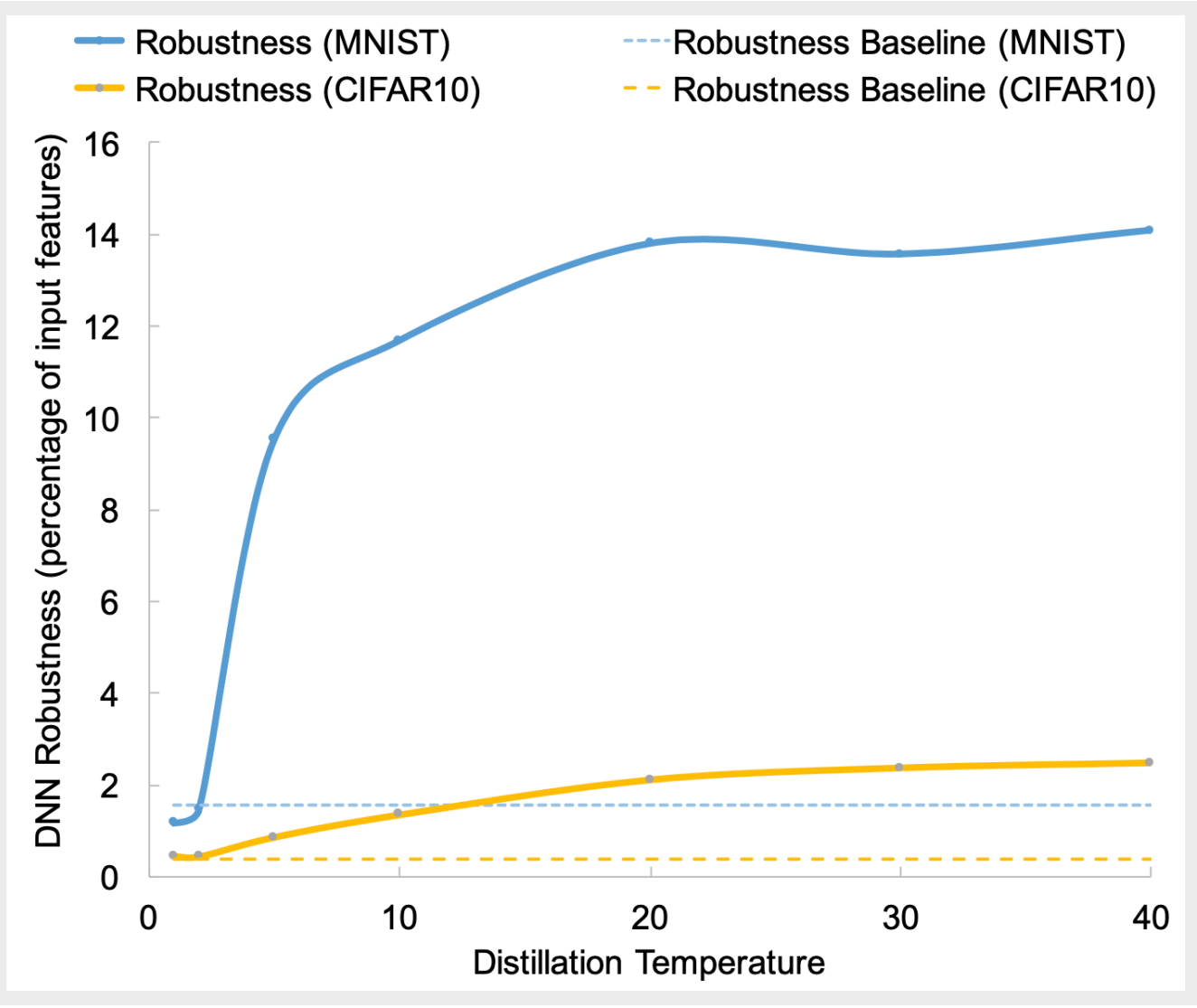
Negligible degradation for MNIST (<=1.28%) and CIFAR10 (<=1.37%, potential improvements)

Impact on Sensitivity

Mean amplitude of the adversarial gradients calculated for each sample in the CIFAR10 test set, and the values are classified into bins.
The majority is decreasing from >0.001 to $<10^{-40}$ along the increase of *T*.



Impact on Robustness



Robustness defined as the expectation of the minimum perturbation required for misclassification.

Here use the average minimum #pixels required for misclassification as an approximation.

$$\rho_{adv}(F) \simeq \frac{1}{|\mathcal{X}|} \sum_{X \in \mathcal{X}} \min_{\delta X} \|\delta X\|$$

Higher T => More perturbation required for misclassification => More robust

Take-homes

- “Soft labels” prevent **too tight fitting to data**, hence improve generalization.
- High distillation temperature reduces the **adversarial gradients**, hence improves resilience to perturbations.

- One limitation is that it only suits DNNs that produce an energy-based probability distribution, for which a temperature can be defined.
- This method could still be vulnerable to other type attacks, e.g., L-BFGS, fast gradient sign, and genetic algorithms.

Thanks for your attention!