Robustness

Deep Learning Theory

Pu Yang

School of Mathematical Science, Peking University

Table of contents

Transfer

Adversarial sample transferability

Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples

Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, 2016

Adversarial sample transferability:

- Adversarial examples that affect one model often affect another model(, even if their architectures greatly differ)
- An attacker may therefore train their own *substitute* model, craft adversarial examples against the substitute, and transfer them to a victim model

Contributions

1. develop and validate a generalized algorithm – *reservoir sampling* to greatly enhance the efficiency of the training procedure for the substitute model.
2. explore *transferability* within and between different classes of machine learning classifier algorithms (DNN, LR, SVM, decision tree, kNN, Ensembles).
3. demonstrate attacks on two commercial machine learning classification systems

Two definitions:

- **Intra-technique Transferability**: models A and B both using the same machine learning technique, but different initialization and dataset
- **Cross-technique Transferability**: models A and B trained using different machine learning techniques

Hypothesis and experiment

Hypothesis 1: intra-technique and cross-technique adversarial sample transferability are strong phenomena across the machine learning space.

Experiment setting1 – Intra-technique Transferability:

- MNIST, 50,000 training samples, 10,000 validation samples, and 10,000 test samples
- split the training set in disjoint subsets A,B,C,D,E of 10,000 samples each
- Models: DNN, LR, SVM, DT, kNN (totally get 25 models)

Result

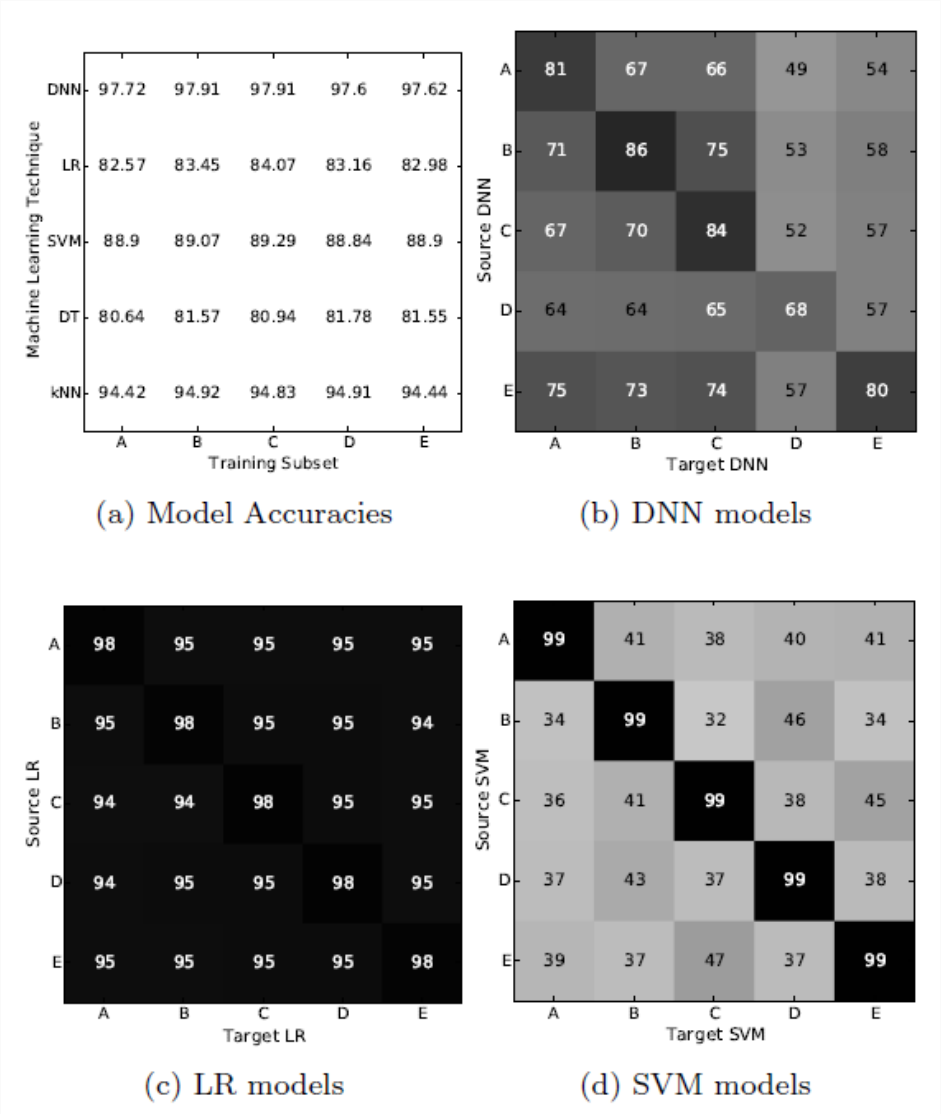


Figure 1: (a)Model accuracy; (b)(c)(d) intra-technique transferability rates for different models

Result

Experiment setting2 – Cross-technique Transferability:

- train one model per machine learning technique on the full MNIST training set of 50,000 samples

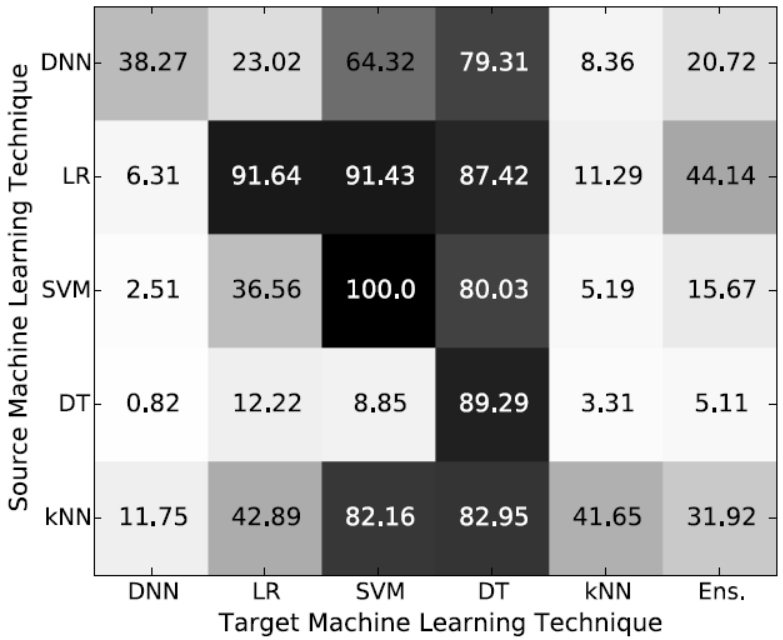


Figure 3: cross-technique Transferability matrix: cell (i, j) is the percentage of adversarial samples crafted to mislead a classifier learned using machine learning technique i that are misclassified by a classifier trained with technique j.

Dataset Augmentation for Substitutes

- Jacobian-based dataset augmentation

  ref: **Practical black-box attacks against deep learning systems using adversarial examples**

$$S_{\rho+1} = \{\vec{x} + \lambda_\rho \cdot \text{sgn}(J_f[\tilde{O}(\vec{x})] : \vec{x} \in S_\rho)\} \cup S_\rho \qquad (4)$$

where S_ρ and $S_{\rho+1}$ are the previous and new training sets, λ_ρ a parameter fine-tuning the augmentation step size, J_f the Jacobian matrix of substitute f, and $\tilde{O}(\vec{x})$ the oracle's label for sample $\vec{x}$. We train a new instance f of the substitute with the augmented training set $S_{\rho+1}$, which we can label simply by querying oracle $\tilde{O}$

- Periodical Step Size

we introduce an iteration period τ after which the step size is multiplied by -1. Thus, the step size λ_ρ is defined as:

$$\lambda_\rho = \lambda \cdot (-1)^{\lfloor \frac{\rho}{\tau} \rfloor} \qquad (5)$$

Reservoir Sampling

Algorithm 1 Jacobian-based augmentation with Reservoir Sampling: sets are considered as arrays for ease of notation.

Input: $S_{\rho-1}$, κ, J_f, λ_ρ

$N \leftarrow |S_{\rho-1}|$
Initialize S_ρ as array of $N + \kappa$ items
$S_\rho[0 : N-1] \leftarrow S_{\rho-1}$
for $i \in 0..\kappa - 1$ **do**
    $S_\rho[N+i] \leftarrow S_{\rho-1}[i] + \lambda_\rho \cdot \mathrm{sgn}(J_f[\tilde{O}(S_{\rho-1}[i])])$
end for
for $i \in \kappa..N-1$ **do**
    $r \leftarrow$ random integer between 0 and i
    if $r < \kappa$ **then**
        $S_\rho[N+r] \leftarrow S_{\rho-1}[i] + \lambda_\rho \cdot \mathrm{sgn}(J_f[\tilde{O}(S_{\rho-1}[i])])$
    end if
end for
return S_ρ

SKIP CONNECTIONS MATTER: ON THE TRANSFERABILITY OF ADVERSARIAL EXAMPLES GENERATED WITH RESNETS

ICLR 2020

whether or not the DNN architecture itself can expose more transferability of adversarial attacks is an unexplored problem.

An observation

At each of the last 3 skip connections and residual modules of ResNet-18, we illustrate the **success rate of attacks** crafted using gradients back-propagate through either the skip connection or the residual module

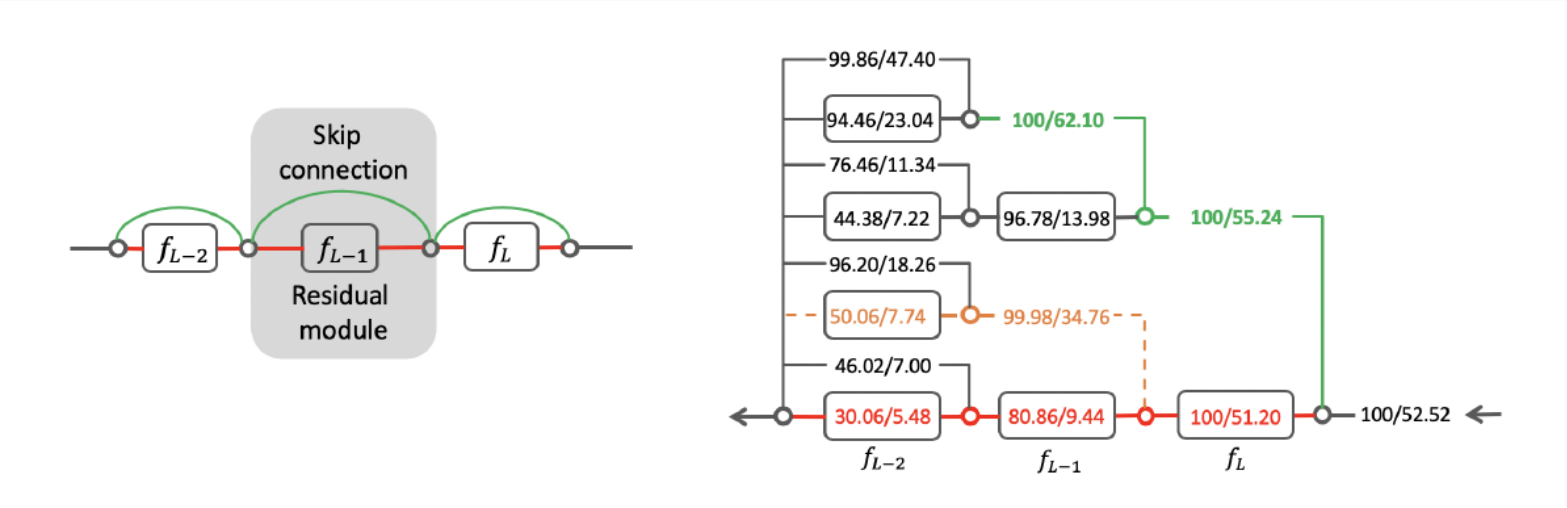


Figure 2: The success rate (in the form of "white-box/black-box") of adversarial attacks crafted using gradients flowing through either a skip connection (going upwards) or a residual module (going leftwards) at each junction point (circle).

SKIP GRADIENT METHOD (SGM)

- Residual module: $z_{i+1} = z_i + f_{i+1}(z_i)$
- gradient decomposition of skip connection

$$\frac{\partial \ell}{\partial \boldsymbol{x}} = \frac{\partial \ell}{\partial \boldsymbol{z}_L} \prod_{i=l}^{L-1} \left(\frac{\partial f_{i+1}}{\partial \boldsymbol{z}_i} + 1 \right) \frac{\partial \boldsymbol{z}_l}{\partial \boldsymbol{x}}$$

- 'skip' gradient:

$$\nabla_{\boldsymbol{x}} = \frac{\partial \ell}{\partial \boldsymbol{z}_L} \prod_{i=l}^{L-1} \left(\gamma \frac{\partial f_{i+1}}{\partial \boldsymbol{z}_i} + 1 \right) \frac{\partial \boldsymbol{z}_l}{\partial \boldsymbol{x}}$$

  use more gradient from the skip connections

PROPOSED SKIP GRADIENT ATTACK

$$\boldsymbol{x}_{adv}^{t+1} = \Pi_{\epsilon}\left(\boldsymbol{x}_{adv}^{t} + \alpha \cdot \text{sign}\left(\frac{\partial \ell}{\partial \boldsymbol{z}_L}\prod_{i=0}^{L-1}\left(\gamma\frac{\partial f_{i+1}}{\partial \boldsymbol{z}_i} + 1\right)\frac{\partial \boldsymbol{z}_0}{\partial \boldsymbol{x}}\right)\right)$$

Table 1: The success rates (%±std over 5 random runs) of black-box attacks (untargeted) crafted by PGD and its "skip gradient" (SGM) version, on different source models against a Inception V3 target model. The best results are in **bold**.

	RN18	RN34	RN50	RN101	RN152	DN121	DN169	DN201
PGD	23.23±0.69	24.38±0.41	22.80±0.55	22.98±0.83	26.56±0.75	30.71±0.60	30.90±0.31	36.01±0.59
SGM	**28.92±0.45**	**43.43±0.32**	**36.71±0.55**	**38.38±0.53**	**44.84±0.14**	**57.38±0.14**	**60.45±0.42**	**65.48±0.23**

ADVERSARIALLY ROBUST TRANSFER LEARNING

ICLR 2020

Goal:

- transfer not only performance but also robustness from a source model to a target domain
- produce accurate and robust models with little data, and without the cost of adversarial training

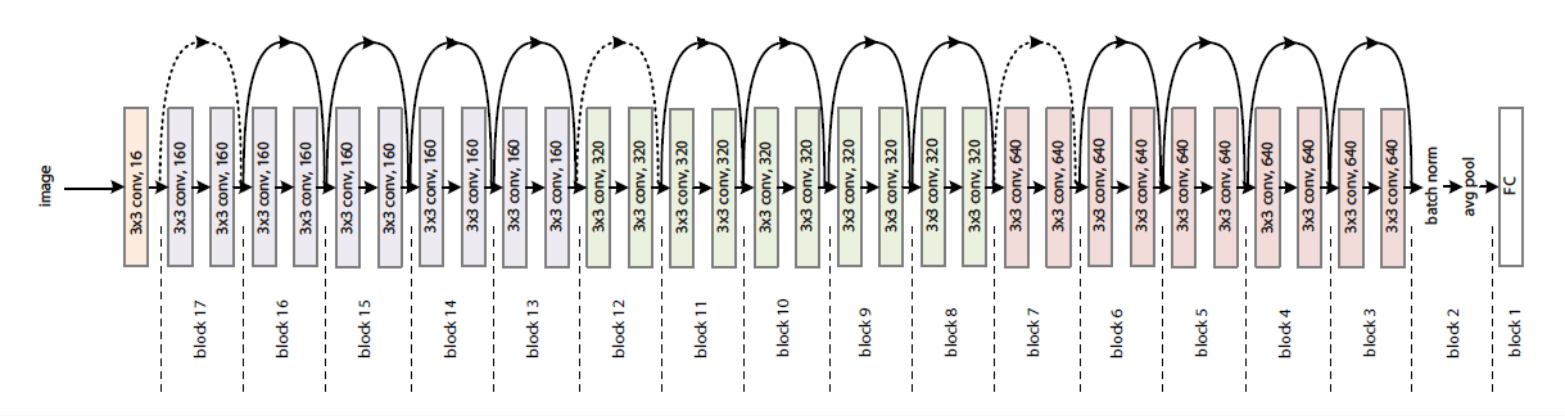
Ways:

1. observing that robust networks contain robust feature extractors
2. produce new models that inherit the robustness of their parent networks
3. consider the case of "fine tuning" a network by re-training end-to-end in the target domain

The robustness of different network layers

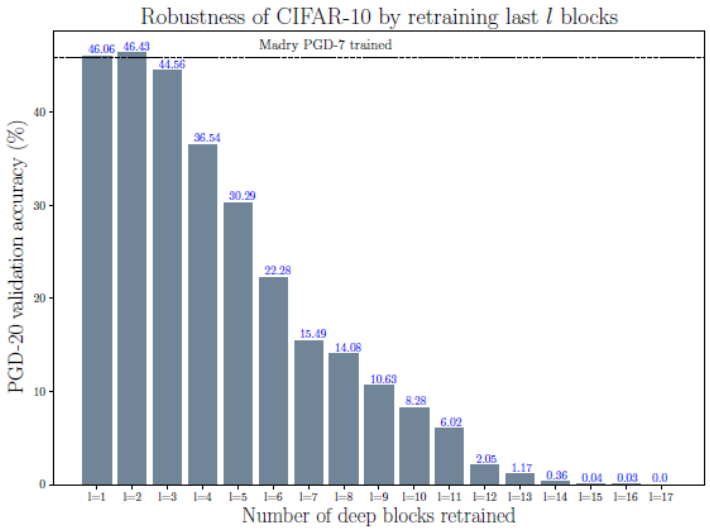
Do a 3-step training:

1. adversarially train CIFAR10/CIFAR100 Wide-ResNet 32-10
2. break the WRN 32-10 model into 17 blocks
3. re-initialize the k deepest blocks (blocks 1 through k) and then train the parameters of those blocks on natural images

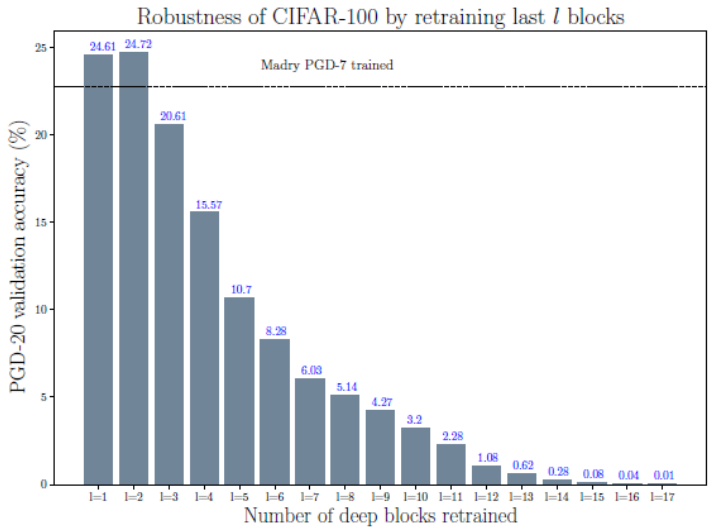


The robustness of different network layers

Dataset	model	validation accuracy	accuracy on PGD-20	accuracy on CW-20
CIFAR-10+	natural	95.01%	0.00%	0.00%
	robust	87.25%	45.84%	46.96%
CIFAR-100+	natural	78.84%	0.00%	0.00%
	robust	59.87%	22.76%	23.16%



(a) CIFAR-10 PGD-20 accuracy



(b) CIFAR-100 PGD-20 accuracy

It shows that a hardened network's robustness is mainly due to **robust deep feature representations**.

Approach 1: only the last layer is re-trained

Source Dataset	Target Dataset	Source Model	val.	PGD-20	CW-20
CIFAR-100	CIFAR-10	natural	83.05%	0.00%	0.00%
		robust	72.05%	17.70%	17.43%
CIFAR-100 (50% of classes)	CIFAR-100 (other 50% of classes)	natural	71.44%	0.00%	0.00%
		robust	58.48%	15.86%	15.30%
CIFAR-100 (50% of classes)	CIFAR-100 (same 50% of classes)	natural	80.20%	0.00%	0.00%
		robust	64.96%	25.16%	25.56%
CIFAR-10	CIFAR-100	natural	49.66%	0.00%	0.00%
		robust	41.59%	11.63%	9.68%

Architecture and Source Dataset	Target Dataset	Source Model	val.	PGD-20	CW-20
ResNet-50 ImageNet	CIFAR-10+	natural	90.49%	0.01%	0.00%
		robust ($\epsilon = 5$)	88.33%	22.66%	26.01%
	CIFAR-100+	natural	72.84%	0.05%	0.00%
		robust ($\epsilon = 5$)	68.88%	15.21%	18.34%
Robust u-ResNet-50 ($\epsilon = 5$) for CIFAR-10+			82.00%	53.11%	
Robust u-ResNet-50 ($\epsilon = 5$) for CIFAR-100+			59.90%	29.54%	

Training deeper networks on top of robust feature extractors

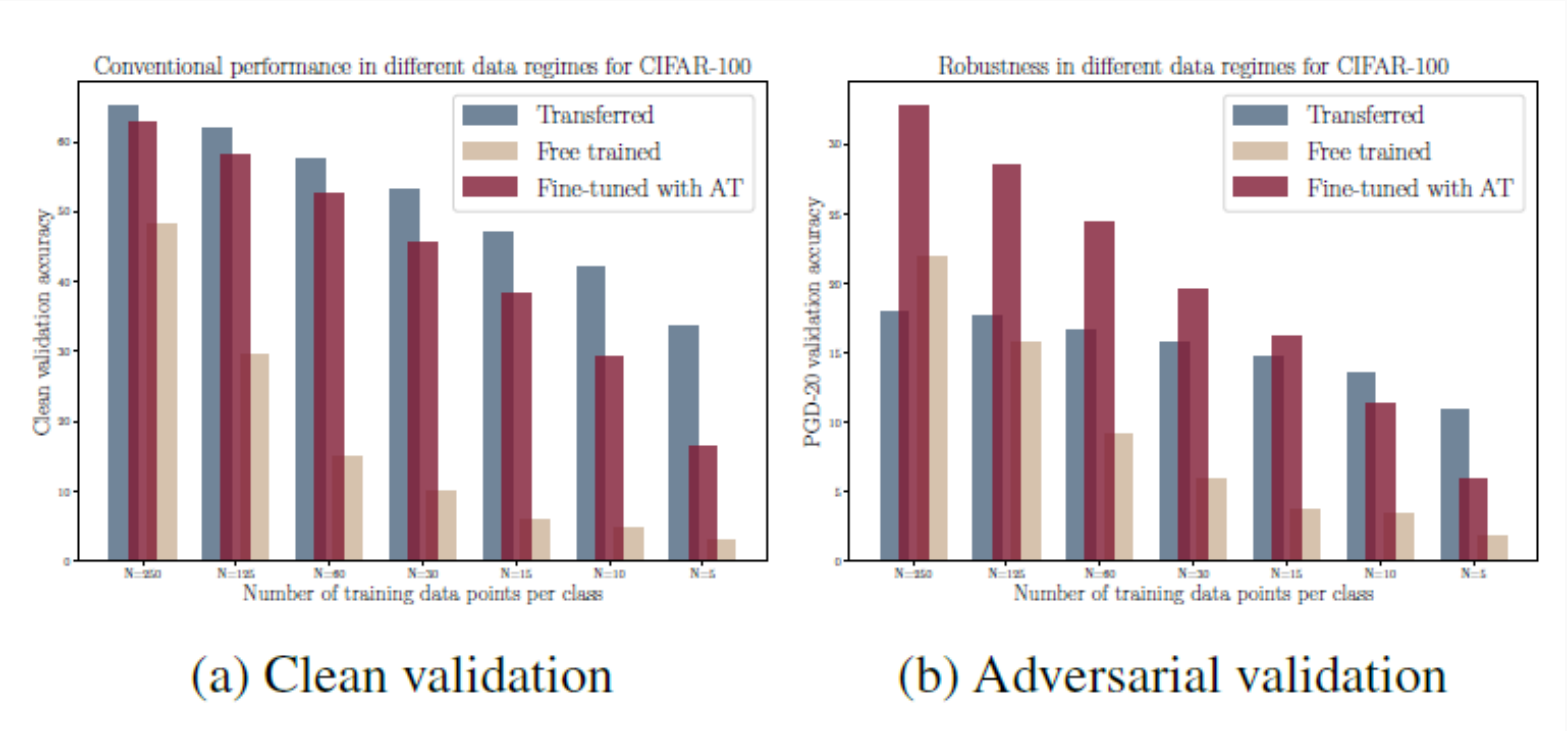
A disadvantage:

- only re-train one layer for the new task $\rightarrow$ small number of trainable parameters left $\rightarrow$
- not capable of completely fitting the training data $\rightarrow$
- bad natural training accuracy

Improvement: train a **multi-layer perceptron** (MLP) network on top of the robust feature extractor with Dropout and Batch Normalization to prevent overfitting

Low-data regime

- transfer learning as above
- free training
- starting from a pre-trained robust ImageNet model and fine-tuning on adversarial examples



(a) Clean validation

(b) Adversarial validation

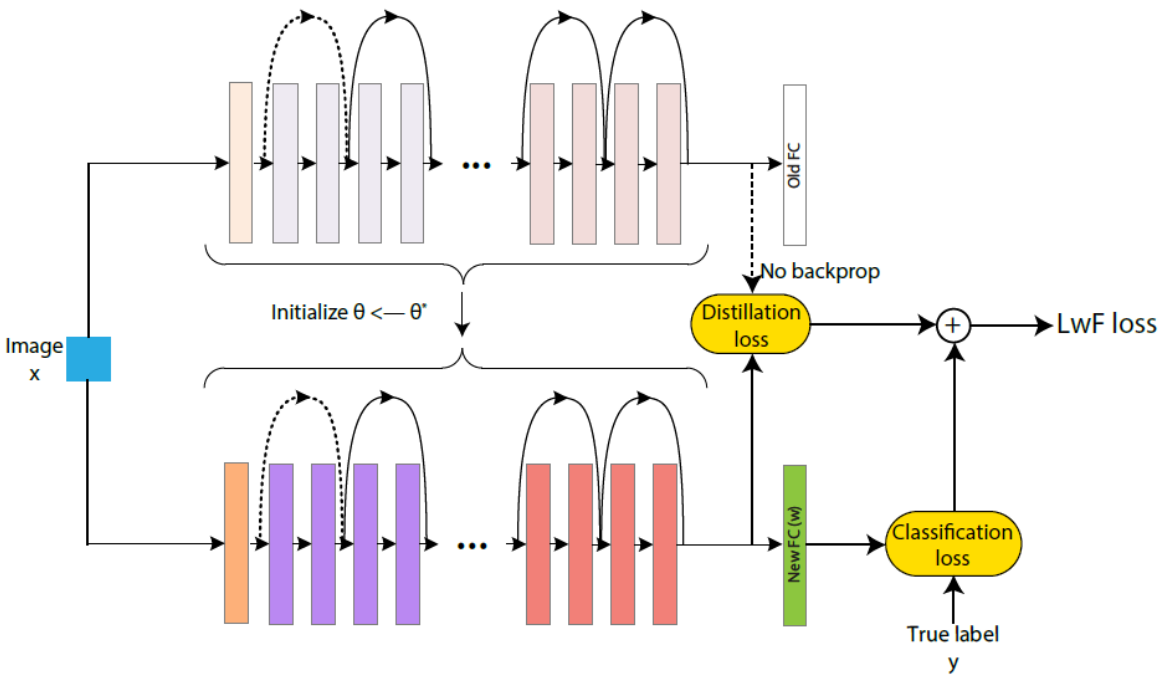
Approach2: end-to-end training without forgetting

fine tune the feature extractor parameters

Main question:

to perform well on the target dataset without catastrophically forgetting the robustness of the source model

$$\min_{w,\theta} \quad l(z(x,\theta), y, w) + \lambda_d \cdot d(z(x,\theta), z_0(x, \theta_r^*))$$



Understanding

Adversarial Examples Are Not Bugs, They Are Features

NeurIPS 2019

Main claim:

Adversarial vulnerability is a direct result of our models' sensitivity to well-generalizing features in the data.

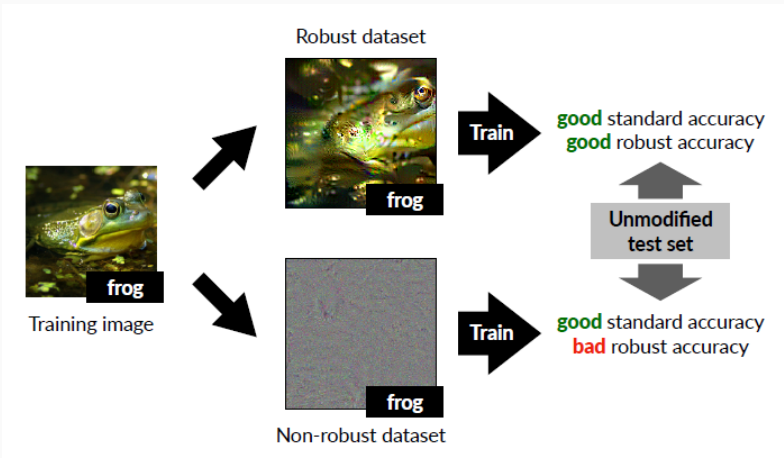
Experiment 1: Disentangling robust and non-robust features

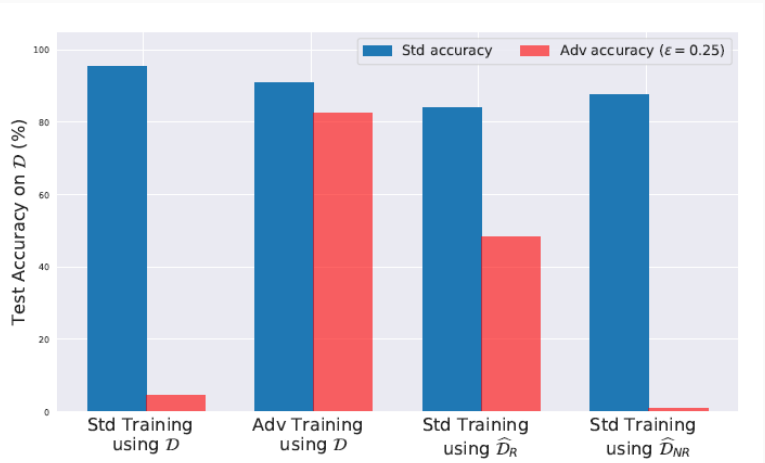
How to disentangle:

1. given a robust model f
2. for each sample (x, y), do the following optimization:

$$\min_{x_r} ||g(x_r) - g(x)||_2$$

where g is the mapping from x to the representation layer of f



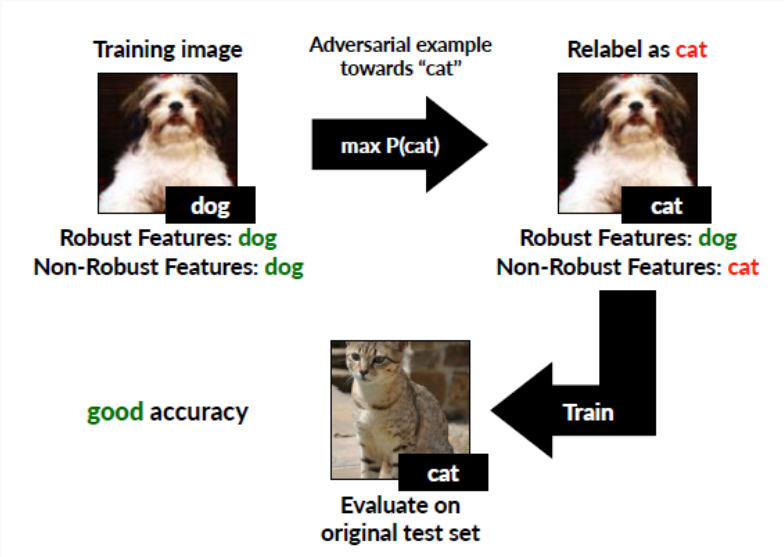


Experiment 2: Non-robust features suffice for standard classification

1. given a standard model f
2. for each sample (x, y), do the following optimization:

   $$x_{\text{adv}} = \arg \min_{||x'-x|| \leq \epsilon} \text{loss}(f(x'), y') \quad \text{or} \quad x_{\text{adv}} = \arg \min_{||x'-x|| \leq \epsilon} \text{loss}(f(x'), y+1)$$

   where y' is a random label, and y+1 = 0 if y is the final label. So get dataset D_{rand} and D_{det}.



Source Dataset	Dataset	
	CIFAR-10	ImageNet_R
$\mathcal{D}$	95.3%	96.6%
$\widehat{\mathcal{D}}_{rand}$	63.3%	87.9%
$\widehat{\mathcal{D}}_{det}$	43.7%	64.4%

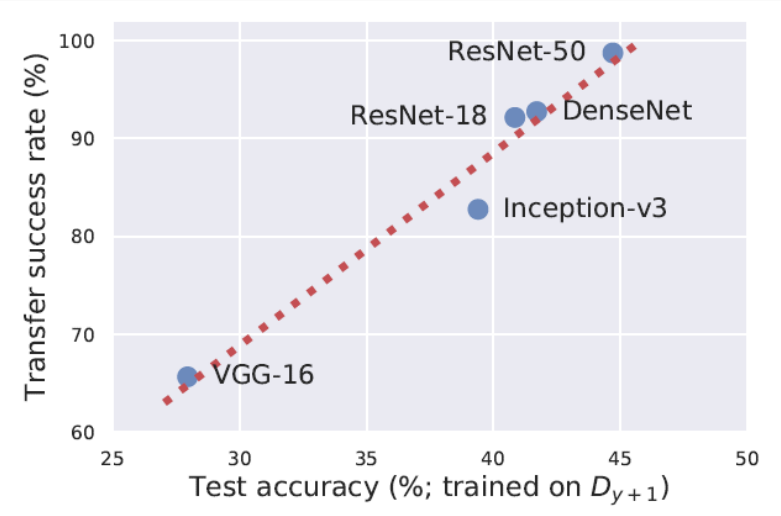
Experiment 3: Transferability can arise from non-robust features

Prior work claims:

adversarial examples **transfer** *across models with different architectures and independently sampled training sets*

Explain by this work:

1. get dataset D_{y+1} as above for a standard ResNet-50
2. train five different architectures on the dataset D_{y+1}
3. test their accuracy on origin dataset and the transfer success rate



Trade-off between standard and robust

Robustness May Be at Odds with Accuracy

ICLR 2019

Main problem:

Why does there seem to be a trade-off between standard and adversarially robust accuracy?

Theory

Theoretically establish that robust and standard models might depend on very different sets of features on a simple task. In other words, robust model lead to a decrease in standard accuracy.

Our binary classification task. Our data model consists of input-label pairs (x, y) sampled from a distribution $\mathcal{D}$ as follows:

$$y \overset{u.a.r}{\sim} \{-1, +1\}, \qquad x_1 = \begin{cases} +y, & \text{w.p. } p \\ -y, & \text{w.p. } 1-p \end{cases}, \qquad x_2, \ldots, x_{d+1} \overset{i.i.d}{\sim} \mathcal{N}(\eta y, 1), \tag{3}$$

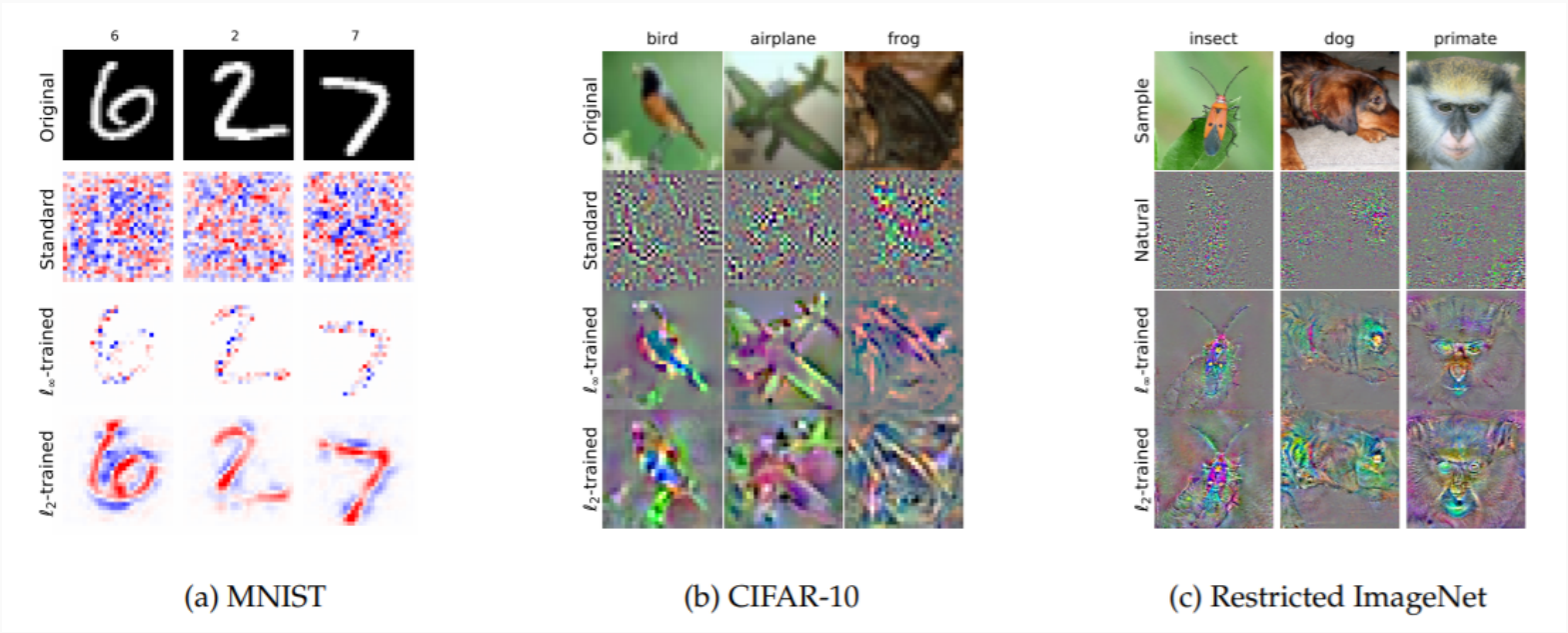
- a simple task, which does not work in the field of deep learning
- The previous paper has illustrated this conclusion empirically!

Unexpected benefits of adversarial robustness

Main claims:

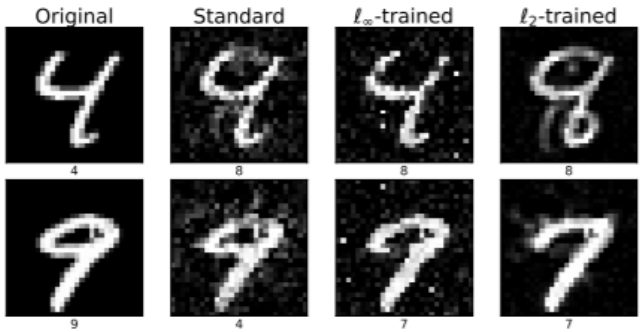
robust models will be more aligned with human vision than standard models

Visualization of the loss gradient with respect to input pixels

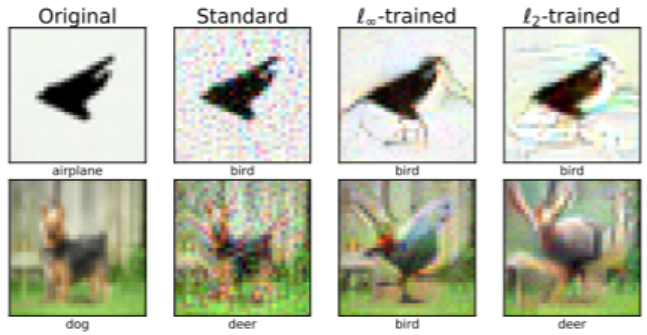


(a) MNIST (b) CIFAR-10 (c) Restricted ImageNet

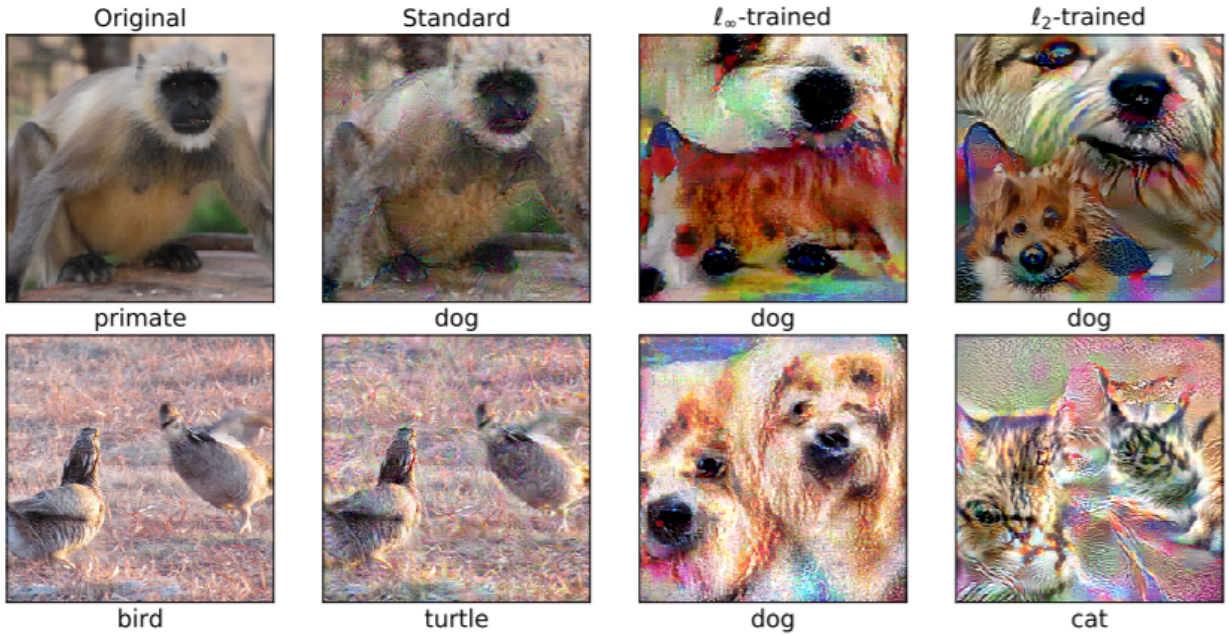
View the adversarial example of standard and robust model



(a) MNIST



(b) CIFAR-10

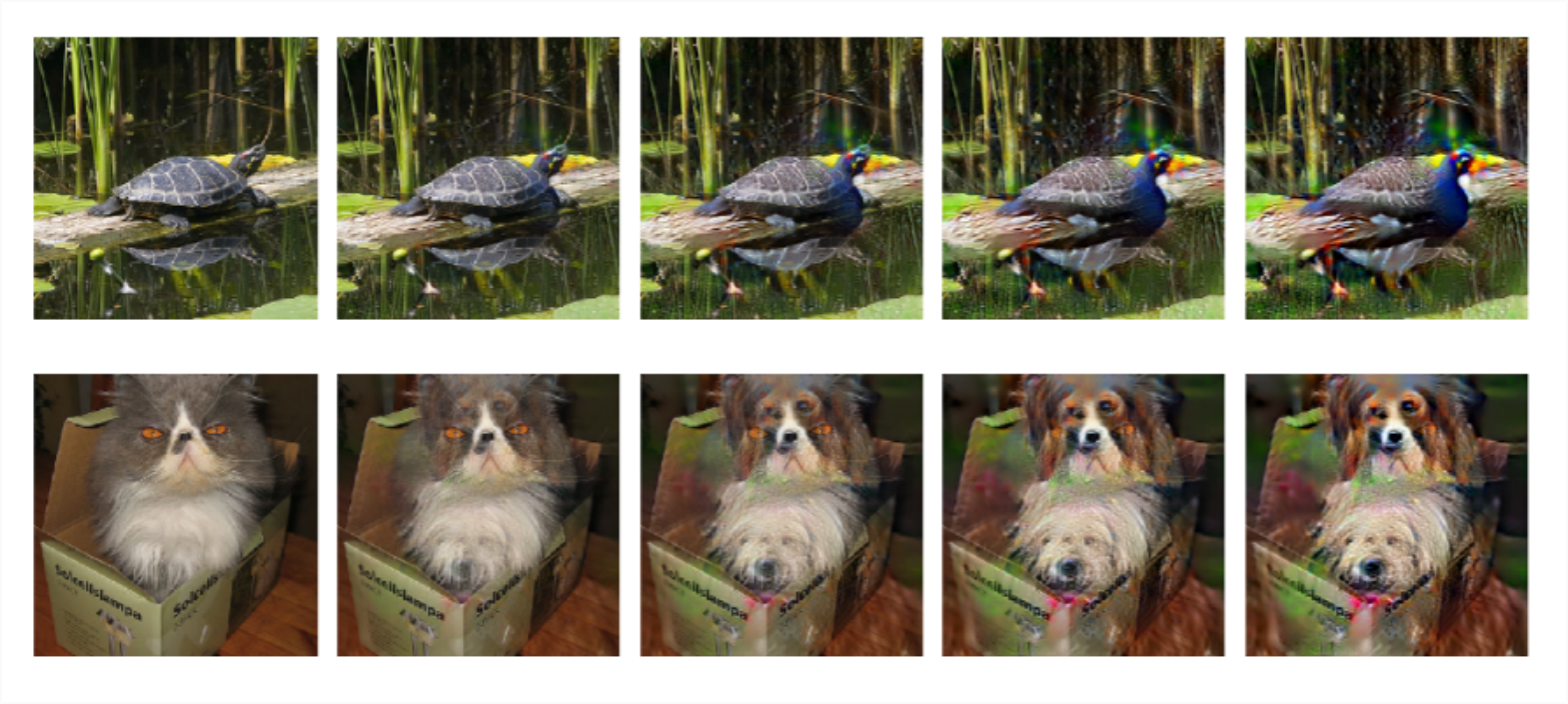


(c) Restricted ImageNet

adversarial perturbations for robust models tend to produce salient

Smooth cross-class interpolations via gradient descent

By linearly interpolating between the original image and the image produced by PGD we can produce a smooth



these inter-class trajectories is similar to the GAN interpolations (future work)

Trade-off between robustness and accuracy

Theoretically Principled Trade-off between Robustness and Accuracy

arxiv: Learning, 2019

Contribution:

1. Theoretically, characterize the trade-off between accuracy and robustness for classification problems
2. Algorithmically, propose a new formulation of adversarial defense, TRADES
3. Experimentally, the algorithm outperforms state-of-the-art methods

Basic settings

- Robust error

$$\mathcal{R}_{\mathrm{rob}}(f) := \mathbb{E}_{(X,Y)\sim\mathcal{D}} \mathbf{1}\left\{\exists X' \in \mathbb{B}(\boldsymbol{X}, \epsilon) \text{ s.t. } f\left(\boldsymbol{X}'\right) Y \leq 0\right\}$$

- Boundary error

$$\mathcal{R}_{\mathrm{bdy}}(f) := \mathbb{E}_{(X,Y)\sim\mathcal{D}} \mathbf{1}\{\boldsymbol{X} \in \mathbb{B}(\mathrm{DB}(f), \epsilon), f(\boldsymbol{X}) Y > 0\}$$

  so that

$$\mathcal{R}_{\mathrm{rob}}(f) = \mathcal{R}_{\mathrm{nat}}(f) + \mathcal{R}_{\mathrm{bdy}}(f)$$

- surrogate loss

$$\mathcal{R}_{\phi}(f) := \mathbb{E}_{(\boldsymbol{X},Y)\sim\mathcal{D}} \phi(f(\boldsymbol{X}) Y)$$

Theoretically

Under a week assumption, there are two following theories:

- upper bound

Theorem 3.1. *Let* $\mathcal{R}_\phi(f) := \mathbb{E}\phi(f(\boldsymbol{X})Y)$ *and* $\mathcal{R}_\phi^* := \min_f \mathcal{R}_\phi(f)$. *Under Assumption 1, for any non-negative loss function* ϕ *such that* $\phi(0) \geq 1$, *any measurable* $f : \mathcal{X} \to \mathbb{R}$, *any probability distribution on* $\mathcal{X} \times \{\pm 1\}$, *and any* $\lambda > 0$, *we have*[1]

$$\begin{aligned}\mathcal{R}_{\mathrm{rob}}(f) - \mathcal{R}_{\mathrm{nat}}^* &\leq \psi^{-1}(\mathcal{R}_\phi(f) - \mathcal{R}_\phi^*) + \Pr[\boldsymbol{X} \in \mathbb{B}(\mathrm{DB}(f), \epsilon), f(\boldsymbol{X})Y > 0] \\ &\leq \psi^{-1}(\mathcal{R}_\phi(f) - \mathcal{R}_\phi^*) + \mathbb{E} \max_{\boldsymbol{X}' \in \mathbb{B}(\boldsymbol{X}, \epsilon)} \phi(f(\boldsymbol{X}')f(\boldsymbol{X})/\lambda).\end{aligned}$$

- lower bound

Theorem 3.2. *Suppose that* $|\mathcal{X}| \geq 2$. *Under Assumption 1, for any non-negative loss function* ϕ *such that* $\phi(x) \to 0$ *as* $x \to +\infty$, *any* $\xi > 0$, *and any* $\theta \in [0, 1]$, *there exists a probability distribution on* $\mathcal{X} \times \{\pm 1\}$, *a function* $f : \mathbb{R}^d \to \mathbb{R}$, *and a regularization parameter* $\lambda > 0$ *such that* $\mathcal{R}_{\mathrm{rob}}(f) - \mathcal{R}_{\mathrm{nat}}^* = \theta$ *and*

$$\psi\left(\theta - \mathbb{E} \max_{\boldsymbol{X}' \in \mathbb{B}(\boldsymbol{X}, \epsilon)} \phi(f(\boldsymbol{X}')f(\boldsymbol{X})/\lambda)\right) \leq \mathcal{R}_\phi(f) - \mathcal{R}_\phi^* \leq \psi\left(\theta - \mathbb{E} \max_{\boldsymbol{X}' \in \mathbb{B}(\boldsymbol{X}, \epsilon)} \phi(f(\boldsymbol{X}')f(\boldsymbol{X})/\lambda)\right) + \xi.$$

the upper bound is tight

Algorithm: TRADES

In order to minimize $\mathcal{R}_{\text{rob}}(f) - \mathcal{R}^*_{\text{nat}}$, the theorems suggest minimizing:

$$\min_f \mathbb{E}\{\underbrace{\phi(f(\boldsymbol{X})Y)}_{\text{for accuracy}} + \underbrace{\max_{\boldsymbol{X}' \in \mathbb{B}(\boldsymbol{X},\epsilon)} \phi\left(f(\boldsymbol{X}) f\left(\boldsymbol{X}'\right)/\lambda\right)}_{\text{regularization for robustness}}\}$$

So they design TRADES:

Algorithm 1 Adversarial training by TRADES

Input: Step sizes η_1 and η_2, batch size m, number of iterations K in inner optimization, network architecture parametrized by θ
Output: Robust network f_θ
Randomly initialize network f_θ, or initialize network with pre-trained configuration
repeat
  Read mini-batch $B = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_m\}$ from training set
  for $i = 1, ..., m$ (in parallel) **do**
    $\boldsymbol{x}'_i \leftarrow \boldsymbol{x}_i + 0.001 \cdot \mathcal{N}(\mathbf{0}, \mathbf{I})$, where $\mathcal{N}(\mathbf{0}, \mathbf{I})$ is the Gaussian distribution with zero mean and identity variance
    for $k = 1, ..., K$ **do**
      $\boldsymbol{x}'_i \leftarrow \Pi_{\mathbb{B}(\boldsymbol{x}_i,\epsilon)}(\eta_1 \text{sign}(\nabla_{\boldsymbol{x}'_i} \mathcal{L}(f_\theta(\boldsymbol{x}_i), f_\theta(\boldsymbol{x}'_i))) + \boldsymbol{x}'_i)$, where Π is the projection operator
    end for
  end for
  $\theta \leftarrow \theta - \eta_2 \sum_{i=1}^m \nabla_\theta [\mathcal{L}(f_\theta(\boldsymbol{x}_i), \boldsymbol{y}_i) + \mathcal{L}(f_\theta(\boldsymbol{x}_i), f_\theta(\boldsymbol{x}'_i))/\lambda]/m$
until training converged

Understanding Black-box Predictions via Influence Functions

ICML 2017, best paper

trace a model's prediction through the learning algorithm and back to its **training data**, thereby identifying training points most responsible for a given prediction

Approach

- Upweighting a training point

$$\hat{\theta}_{\epsilon,z} = \text{argmin}_\theta \frac{1}{n}\Sigma_{i=1}^{n} L\left(z_i, \theta\right) + \epsilon L(z, \theta)$$

$$\mathcal{I}_{\text{up,params}}\left(z\right) \stackrel{\text{def}}{=} \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} = -H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta})$$

where $H_{\hat{\theta}} \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta^2 L\left(z_i, \hat{\theta}\right)$ is the Hessian and is positive definite

$$\begin{aligned}
\mathcal{I}_{\text{up , loss}}\left(z, z_{\text{test}}\right) &\stackrel{\text{def}}{=} \left.\frac{dL\left(z_{\text{test}}, \hat{\theta}_{\epsilon,z}\right)}{d\epsilon}\right|_{\epsilon=0} \\
&= \nabla_\theta L\left(z_{\text{test}}, \hat{\theta}\right)^\top \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} \\
&= -\nabla_\theta L\left(z_{\text{test}}, \hat{\theta}\right)^\top H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta})
\end{aligned}$$

Approach

- Perturbing a training input
  define $z_\delta \stackrel{\text{def}}{=} (x+\delta, y)$ and
  $\hat{\theta}_{\epsilon, z_\delta, -z} \stackrel{\text{def}}{=} \arg\min_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^{n} L(z_i,\theta) + \epsilon L(z_\delta,\theta) - \epsilon L(z,\theta)$, so that:

$$\left.\frac{d\hat{\theta}_{\epsilon,z_\delta,-z}}{d\epsilon}\right|_{\epsilon=0} = \mathcal{I}_{\text{up.params}}(z_\delta) - \mathcal{I}_{\text{up.params}}(z)$$

$$= -H_{\hat{\theta}}^{-1}\left(\nabla_\theta L\left(z_\delta,\hat{\theta}\right) - \nabla_\theta L(z,\hat{\theta})\right)$$

$$\approx -H_{\hat{\theta}}^{-1}\left[\nabla_x \nabla_\theta L(z,\hat{\theta})\right]\delta$$

$$\mathcal{I}_{\text{pert , loss}}(z, z_{\text{test}})^\top \stackrel{\text{def}}{=} \nabla_\delta L\left(z_{\text{test}}, \hat{\theta}_{z_\delta,-z}\right)^\top\Big|_{\delta=0}$$

$$= -\nabla_\theta L\left(z_{\text{test}},\hat{\theta}\right)^\top H_{\hat{\theta}}^{-1}\nabla_x\nabla_\theta L(z,\hat{\theta})$$

Approach

- Relation to Euclidean distance
  To find the training points most relevant to a test point, use Influence function to replace Euclidean distance. Take a logistic regression model for example:

$$\mathcal{I}_{up,loss}(z, z_{test}) = -y_{test} y \cdot \sigma\left(-y_{test}\theta^T x_{test}\right) \cdot \sigma\left(-y\theta^T x\right) \cdot x_{test}^T H_{\hat{\theta}}^{-1} x$$

  compare with $x \cdot x_{\text{test}}$

  1. $\sigma\left(-y\theta^T x\right)$ gives points with high training loss more influence, revealing that outliers can dominate the model parameters
  2. $H_{\hat{\theta}}^{-1}$ measures the "resistance" of the other training points to the removal of z

Efficiently Calculating Influence

Influence function:

$$\mathcal{I}_{up,loss}\left(z, z_{test}\right) = -\nabla_\theta L\left(z_{test}, \hat{\theta}\right)^T H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta})$$

Two calculating problems:

1. $H_{\hat{\theta}}$, $O(np^2 + p^3)$
2. Every training sample need to be calculated

Solution:

Hessian-vector products – directly calculate $s_{test} = H_{\hat{\theta}}^{-1} \nabla_\theta L\left(z_{test}, \hat{\theta}\right)$, so that

$$\mathcal{I}_{up,loss}\left(z, z_{test}\right) = -s_{test} \nabla_\theta L(z, \hat{\theta})$$

Application 1: Understanding model behavior

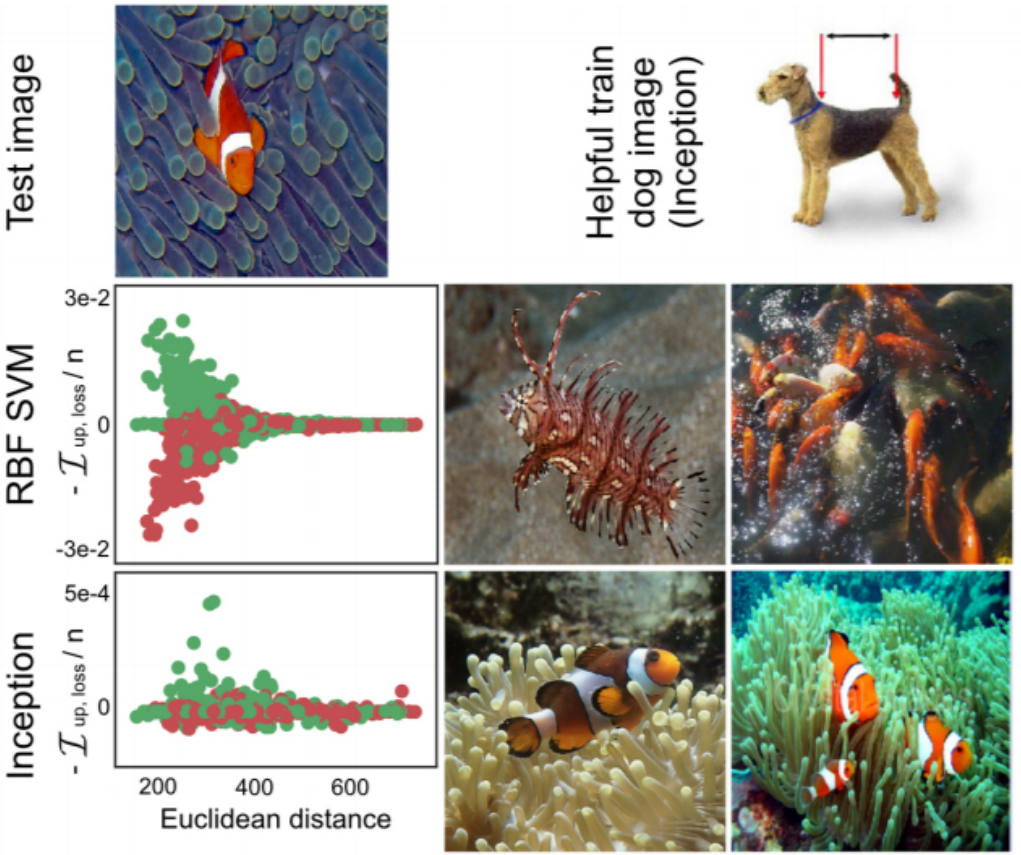


Figure 4. **Inception vs. RBF SVM. Bottom left:** $-\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}})$ vs. $\|z - z_{\text{test}}\|_2^2$. Green dots are fish and red dots are dogs. **Bottom right:** The two most helpful training images, for each model, on the test. **Top right:** An image of a dog in the training set that helped the Inception model correctly classify the test image as a fish.

Application 2: Adversarial training examples

for a target test image z_{test}, construct $\tilde{z}_i$, an adversarial version of a training image z_i by:

1. initializing $\tilde{z}_i = z_i$
2. $\tilde{z}_i = \Pi(\tilde{z}_i + \alpha \,\text{sign}\,(\mathcal{I}_{\text{pert , loss}}\ (\tilde{z}_i, z_{\text{test}})))$
3. retraining the model
4. repeating steps 2 and 3

By increasing the average loss of multiple test images:

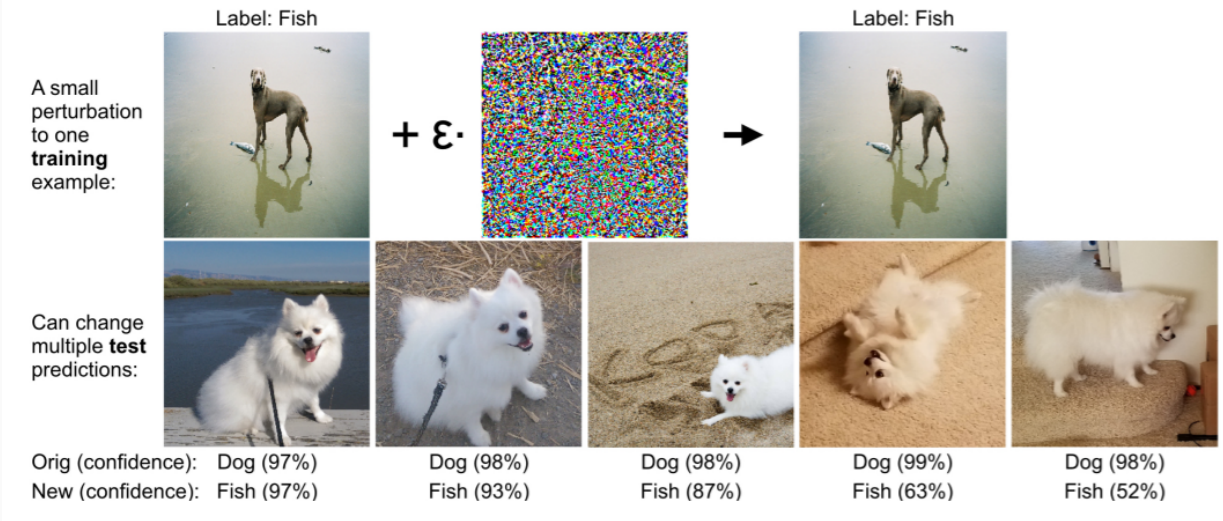


Figure 5. **Training-set attacks.** We targeted a set of 30 test images featuring the first author's dog in a variety of poses and backgrounds. By maximizing the average loss over these 30 images, we created a visually-imperceptible change to the particular training image (shown on top) that flipped predictions on 16 test images.

Application 3: Debugging domain mismatch

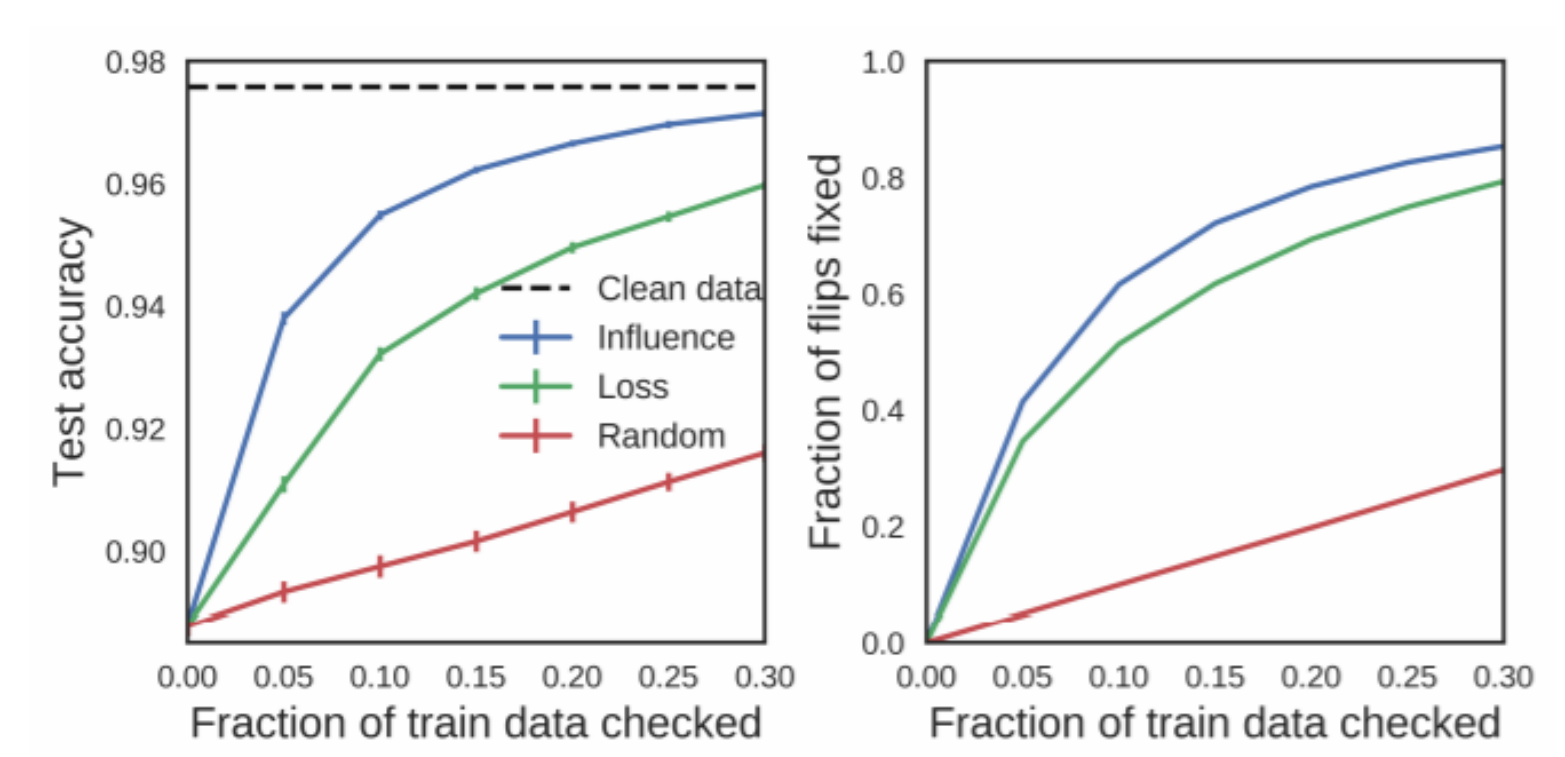
in biomedical data

Find the training example that maximizes test error

Fixing mislabeled examples

measure the influence of z_i with $\mathcal{I}_{\text{up,loss}}(z_i, z_i)$, which approximates the error incurred on z_i

Experiment is for spam classification. They flipped the labels of a random 10% of the training data and then simulated manually inspecting a fraction of the training points, correcting them if they had been flipped.



Unlabeled data

Unlabeled Data Improves Adversarial Robustness

NeurIPS 2019

Contributions:

1. Theoretically, revisit a simple Gaussian model
2. Empirically, augment CIFAR-10 with 500K unlabeled images sourced from 80 Million Tiny Images

Theory

Claim: a simple semisupervised learning procedure (self-training) achieves high robust accuracy using the same number of labels required for achieving high standard accuracy

- Error metrics

$$\text{err}_{\text{standard}}\left(f_\theta\right) := \mathbb{P}_{(x,y)\sim P_{\mathrm{x},y}}\left(f_\theta(x) \neq y\right)$$

$$\text{err}^{p,\epsilon}_{\text{robust}}\left(f_\theta\right) := \mathbb{P}_{(x,y)\sim P_{\mathrm{x},y}}\left(\exists x' \in \mathcal{B}^p_\epsilon(x), f_\theta\left(x'\right) \neq y\right)$$

- Self-training
  1. obtain an intermediate model $\hat{\theta}_{\text{intermediate}}$ via supervised learning on (X, Y)
  2. generate **pseudo-labels** $\tilde{y}_i = f_{\hat{\theta}_{\text{intermediate}}}\left(\tilde{x}_i\right)$
  3. combine the data and pseudo-labels to obtain a final model
- Gaussian model
  binary classification task, $y = \pm 1$, $x|y \sim \mathcal{N}\left(y\mu, \sigma^2 I\right)$

In Advances in Neural Information Processing Systems

NeurIPS 2018

Learning a simple linear classifier. We consider linear classifiers of the form $f_\theta = \mathrm{sign}(\theta^\top x)$. Given n labeled data $(x_1, y_1), \ldots, (x_n, y_n) \overset{\text{iid}}{\sim} P_{\mathsf{x},\mathsf{y}}$, we form the following simple classifier

$$\hat{\theta}_n := \frac{1}{n} \sum_{i=1}^{n} y_i x_i. \tag{4}$$

Proposition 1. *There exists a universal constant r such that for all $\epsilon^2 \sqrt{d/n_0} \geq r$,*

$$n \geq n_0 \;\Rightarrow\; \mathbb{E}_{\hat{\theta}_n} \mathsf{err}_{\text{standard}} \left(f_{\hat{\theta}_n} \right) \leq \frac{1}{3} \quad \textit{and} \quad n \geq n_0 \cdot 4\epsilon^2 \sqrt{\frac{d}{n_0}} \;\Rightarrow\; \mathbb{E}_{\hat{\theta}_n} \mathsf{err}^{\infty,\epsilon}_{\text{robust}} \left(f_{\hat{\theta}_n} \right) \leq 10^{-3}.$$

Theorem 1 (Schmidt et al. [41])**.** *Let A_n be any learning rule mapping a dataset $S \in (\mathcal{X} \times \mathcal{Y})^n$ to classifier $\mathsf{A}_n[S]$. Then,*

$$n \leq n_0 \frac{\epsilon^2 \sqrt{d/n_0}}{8 \log d} \;\Rightarrow\; \mathbb{E}\, \mathsf{err}^{\infty,\epsilon}_{\text{robust}}(\mathsf{A}_n[S]) \geq \frac{1}{2}(1 - d^{-1}), \tag{5}$$

where the expectation is with respect to the random draw of $S \sim P^n_{\mathsf{x},\mathsf{y}}$ as well as possible randomization in A_n.

Semi-supervised learning in the Gaussian model

- Self-learning
  1. intermediate classifier: $\hat{\theta}_{\text{intermediate}} := \hat{\theta}_n = \frac{1}{n}\sum_{i=1}^{n} y_i x_i$
  2. generate **pseudo-labels**: $\tilde{y}_i := f_{\hat{\theta}_{\text{intermediate}}}(\tilde{x}_i) = \text{sign}\left(\tilde{x}_i^\top \hat{\theta}_{\text{intermediate}}\right)$ for $i = 1, \ldots, \tilde{n}$
  3. final semisupervised classifier: $\hat{\theta}_{\text{final}} := \frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}} \tilde{y}_i \tilde{x}_i$

Theorem 2. *There exists a universal constant $\tilde{r}$ such that for $\epsilon^2\sqrt{d/n_0} \geq \tilde{r}$, $n \geq n_0$ labeled data and additional $\tilde{n}$ unlabeled data,*

$$\tilde{n} \geq n_0 \cdot 288\epsilon^2\sqrt{\frac{d}{n_0}} \Rightarrow \mathbb{E}_{\hat{\theta}_{\text{final}}} \text{err}_{\text{robust}}^{\infty,\epsilon}\left(f_{\hat{\theta}_{\text{final}}}\right) \leq 10^{-3}.$$

Semi-supervised learning with irrelevant unlabeled data

Settings:

- only $\alpha\tilde{n}$ of the unlabeled data are relevant to the task
- the irrelevant data: y is uniform on $\{-1, 1\}$ and $x \sim \mathcal{N}\left(0, \sigma^2 I\right)$ independent of y

Conclusion:

for any fixed α, high robust accuracy is still possible, but the required number of relevant examples grows by a factor of $1/\alpha$

Semi-supervised learning of robust neural networks

Meta-Algorithm 1 Robust self-training

Input: Labeled data $(x_1, y_1, \ldots, x_n, y_n)$ and unlabeled data $(\tilde{x}_1, \ldots, \tilde{x}_{\tilde{n}})$

Parameters: Standard loss L_{standard}, robust loss L_{robust} and unlabeled weight w

1: Learn $\hat{\theta}_{\text{intermediate}}$ by minimizing $\sum_{i=1}^{n} L_{\text{standard}}(\theta, x_i, y_i)$

2: Generate pseudo-labels $\tilde{y}_i = f_{\hat{\theta}_{\text{intermediate}}}(\tilde{x}_i)$ for $i = 1, 2, \ldots \tilde{n}$

3: Learn $\hat{\theta}_{\text{final}}$ by minimizing $\sum_{i=1}^{n} L_{\text{robust}}(\theta, x_i, y_i) + w \sum_{i=1}^{\tilde{n}} L_{\text{robust}}(\theta, \tilde{x}_i, \tilde{y}_i)$

- **Standard loss**: multi-class logarithmic loss

$$L_{\text{standard}}(\theta, x, y) = -\log p_\theta(y|x)$$

- **Robust loss**: add robustness-promoting regularization term

$$L_{\text{robust}}(\theta, x, y) = L_{\text{standard}}(\theta, x, y) + \beta L_{\text{reg}}(\theta, x)$$
$$\text{where } L_{\text{reg}}(\theta, x) := \max_{x' \in \mathcal{B}_\epsilon^p(x)} D_{\text{KL}}\left(p_\theta(\cdot|x) \| p_\theta(\cdot|x')\right)$$

approximations for the maximization in L_{reg}

1. **Adversarial training**: a heuristic defense via approximate maximization

We focus on ℓ_∞ perturbations and use the projected gradient method to approximate the regularization term of (6),

$$L_{\text{reg}}^{\text{adv}}(\theta, x) := D_{\text{KL}}(p_\theta(\cdot \mid x) \parallel p_\theta(\cdot \mid x'_{\texttt{PG}}[x])), \tag{7}$$

where $x'_{\texttt{PG}}[x]$ is obtained via projected gradient ascent on $r(x') = D_{\text{KL}}(p_\theta(\cdot \mid x) \parallel p_\theta(\cdot \mid x'))$.

2. **Stability training**: a certified ℓ_2 defense via randomized smoothing

Alternatively, we consider stability training [57, 26], where we replace maximization over small perturbations with much larger additive random noise drawn from $\mathcal{N}(0, \sigma^2 I)$,

$$L_{\text{reg}}^{\text{stab}}(\theta, x) := \mathbb{E}_{x' \sim \mathcal{N}(x, \sigma^2 I)} D_{\text{KL}}(p_\theta(\cdot \mid x) \parallel p_\theta(\cdot \mid x')). \tag{8}$$

Let f_θ be the classifier obtained by minimizing $L_{\text{standard}} + \beta L_{\text{robust}}^{\text{stab}}$. At test time, we use the following *smoothed* classifier.

$$g_\theta(x) := \operatorname*{argmax}_{y \in \mathcal{Y}} \; q_\theta(y \mid x), \quad \text{where} \quad q_\theta(y \mid x) := \mathbb{P}_{x' \sim \mathcal{N}(x, \sigma^2 I)}(f_\theta(x') = y). \tag{9}$$

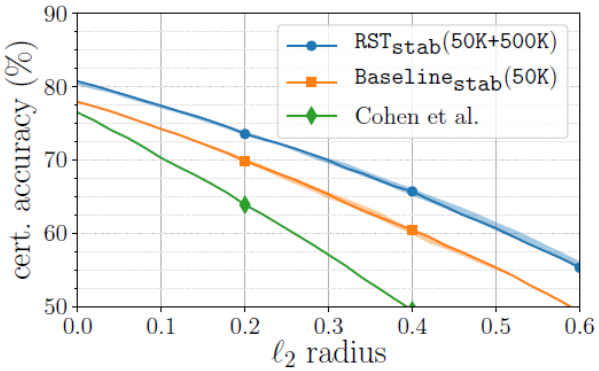
Experiments and Results

50K CIFAR-10 samples + 500K Tiny Images samples from 80M

1. Robustness of $RST_{adv}(50K + 500K)$ against strong attacks

Model	PG_{Madry}	$\text{PG}_{\text{TRADES}}$	PG_{Ours}	CW [7]	Best attack	No attack
RST_{adv}(50K+500K)	63.1	63.1	62.5	64.9	**62.5** ±0.1	**89.7** ±0.1
TRADES [56]	55.8	56.6	55.4	65.0	55.4	84.9
Adv. pre-training [18]	57.4	58.2	57.7	-	$57.4^{\dagger}$	87.1
Madry et al. [29]	45.8	-	-	47.8	45.8	87.3
Standard self-training	-	0.3	0	-	0	96.4

2. Certified robustness of $RST_{adv}(50K + 500K)$



(a)

Model	ℓ_∞ acc. at $\epsilon = \frac{2}{255}$	Standard acc.
RST_{stab}(50K+500K)	**63.8** ± 0.5	**80.7** ± 0.3
$\text{Baseline}_{\text{stab}}$(50K)	58.6 ± 0.4	77.9 ± 0.1
Wong et al. (single) [50]	53.9	68.3
Wong et al. (ensemble) [50]	**63.6**	64.1
IBP [17]	50.0	70.2

(b)