MART: Misclassification Aware adveRsarial Training

Sai Manasa Pappu, Siddharth Maurya

IIT Hyderabad

May 2021

Background

Deep neural networks (DNNs) are vulnerable to adversarial examples crafted by imperceptible perturbations. A range of defense techniques have been proposed to improve DNN robustness to adversarial examples, among which adversarial training has been demonstrated to be the most effective.

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \max_{||x_i^{'} - x_i||_p \leq \epsilon} \ell(\mathrm{h}_\theta(x_i^{'}), y_i) \qquad \text{(Madry et al, 2018)}$$

Adversarial training is often formulated as a min-max optimization problem, with the inner maximization for generating adversarial examples (above expression).

Background continued

However, the adversarial examples are defined on the correctly classified examples (natural examples),but inevitably, some (natural) examples will be misclassified during training.

The paper we are presenting is based around this question : Are the adversarial examples generated from i) misclassified and ii) correctly classified examples, equally important for adversarial robustness? If not, how can one make better use of the difference to improve robustness?

The authors then propose a new defense algorithm called Misclassification Aware adveRsarial Training (MART), which explicitly differentiates the misclassified and correctly classified examples during the training

Previous defense methods against adversarial examples

- Feature squeezing
- Input denoising
- Adversarial Detection
- Defensive Distillation
- Gradient regularization
- Model compression
- Activation Pruning
- Adversarial Training

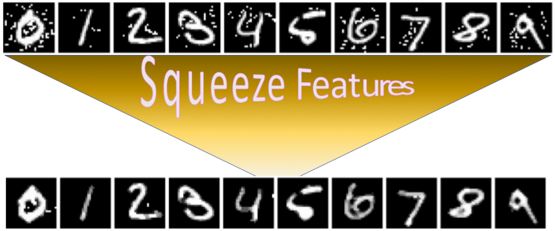
Above all the above mentioned methods, adversarial training is the most effective.

Adversarial detection

The main idea is to investigate model confidence on adversarial samples by looking at Bayesian uncertainty estimates, available in dropout neural networks, and by performing density estimation in the subspace of deep features learned by the model. The result is a method for implicit adversarial detection that is oblivious to the attack algorithm. (Feinman et al 2018)

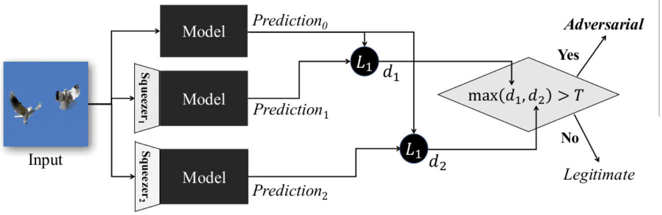
Feature Squeezing

- Feature squeezing is a way to harden DNN systems against adversarial examples by detecting adversarial inputs.
- Feature input spaces are often unnecessarily large, and this vast input space provides extensive opportunities for an adversary to construct adversarial examples.



Feature Squeezing

- Feature squeezing reduces the degrees of freedom available to an adversary by squeezing out unnecessary input features, thus reducing the search space available to an adversary.
- After squeezing, if the original and squeezed inputs produce significantly different outputs from, the input is likely to be adversarial and hence rejected.



Defensive distillation

- In Defensive distillation, the knowledge extracted from a DNN is used to improve its own resilience to adversarial samples.
- If adversarial gradients are high, crafting adversarial samples becomes easier because small perturbations will induce high DNN output variations. Therefore we must reduce variations around the input, and consequently the amplitude of adversarial gradients.
- Defensive distillation is used to smooth the model learned by a DNN thus enabling the model to generalize better to test data as well.

Athalye et al.

Their work showed that adversarial training is the most effective against adversarial attack.

They found that most of the defenses rely on obfuscated gradients, which in reality do not provide security. The authors took 9 models out of which 7 were using obfuscated gradients . Then they were successful in circumventing the defense in 6 completely and 1 partially by their method.

Notable points

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \max_{||x_i' - x_i||_p \leq \epsilon} \ell(\mathrm{h}_\theta(x_i'), y_i)$$

- n- Number of training examples
- l(.) - Classification loss,such as the commonly used cross-entropy (CE) loss.
- The inner maximization generates adversarial examples that can be used by the outer minimization to train robust DNNs.
- (Athalye et al , 2017) : adversarial training with adversarial examples generated by Projected Gradient Descent (PGD) has been demonstrated to be the only method that can train moderately robust DNNs without being fully attacked.
- However, there is still a significant gap between adversarial robustness (test accuracy on adversarial examples) and natural accuracy (test accuracy on natural examples), even for simple image datasets like CIFAR-10

Notable points continued

- Nakkiran et al 2019 - Simple models can have high natural accuracy but are less likely to be robust.
- Sample complexity of adversarial training can be significantly higher than that of natural training, that is, training robust DNNs tends to require more data.

Adversarial Training formulation

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \max_{||x_i^{'} - x_i||_p \le \epsilon} \ell(\mathrm{h}_\theta(x_i^{'}), y_i)$$

This is the expression we will use in this paper.

It comes from the paper by Madry et al,2018 : Towards Deep Learning Models Resistant to Adversarial Training.

Their objective was to formulate a model on which they can study the adversarial robustness by properly defining the attack and defense objective.

$\min_\theta \rho(\theta)$ where

$$\rho(\theta) = E_{(x,y)\sim D}[\max_{\delta \epsilon S} L(\theta, x + \delta, y)]$$

Here L is the adversarial risk, that can be viewed as commonly used loss functions like CE. S is the set of all possible perturbations δ

Adversarial Training formulation contd.

If we see closely, the formula used in our paper is a soft version of that defined in Madry et al.

PGD

- Projected Gradient Descent(PGD) is a white box attack in which the attacker has access to model weights/ gradients.
- Given an input sample X, start from a random perturbation in the L_p ball around X(all points that are at ϵ distance from X), and keep moving in the direction of greatest loss till convergence.
- If the new point goes outside the ϵ-ball, project it back i.e, new point will be the one in the ϵ-ball that is closest to the previous point.

Now we return to our main question:

Are the adversarial examples generated from i) misclassified and ii) correctly classified examples, equally important for adversarial robustness? If not, how can one make better use of the difference to improve robustness?

Author's setup

We conduct a proof-of-concept experiment on CIFAR- 10 in a white-box setting with L_∞ maximum perturbation $\epsilon = 8/255$.

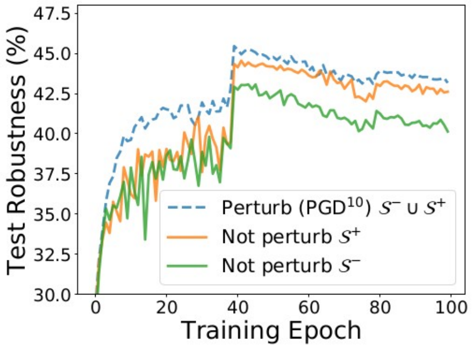
We first train an 8-layerConvolutional Neural Network (CNN) using standard adversarial training with 10-step PGD (PGD^{10}) and step size $\epsilon/4$, then use this network (87 % training accuracy) to select two subsets of natural training examples to investigate

- a subset of misclassified examples S^- (13 % of training data) and
- a subset of correctly classified examples S^+ (13 % of training data)

Note that $|S^-| = |S^+|$

Using these two subsets, we explore different ways to re-train the same network, and evaluate its robustness against white-box PGD 20 (step size $\epsilon/10$) attacks on the test dataset.

Observation

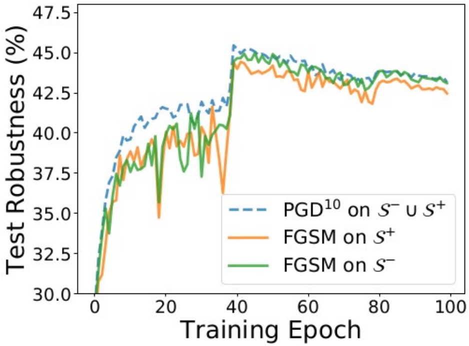


(a) Not perturb

y axis denotes robustness of model against pgd attack.

We can see that if we are perturbing only the correctly classified samples during adversarial training , there is significant drop in robustness.

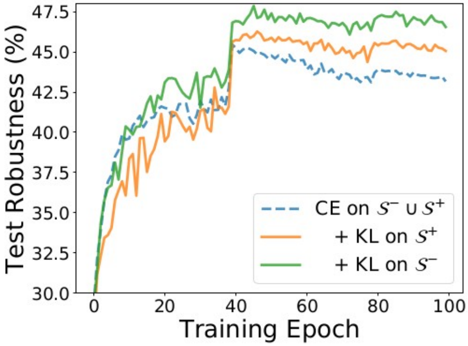
Observation



(b) Inner maximization

Here , we apply FGSM for finding adversarial examples (which is the objective of inner maximization of the previously mentioned adversarial training formula). There is no significant change in robustness in all 3 methods.

Observation



(c) Outer minimization

Here we apply different minimization techniques for minimizing the loss (given in the adversarial training formula). We see that the green curve (where we regularize the loss on misclassified examoles using KL divergence) has significantly higher robustness as compared to others.

Adversarial risk

$R(h_\theta) = \sum_{i=1}^{n} \max_{x_i' \epsilon B_\epsilon(x_i)} 1(h_\theta(x_i') \neq y_i)$

Here, $1(.)$ is the indicator function

$B_\epsilon(x_i) = \{x : ||x - x_i||_p \leq \epsilon\}$ denotes the L_p-norm ball centered at x_i with radius ϵ.

This papers deal with ∞-norm ball as it was proved in the recent works to be best suited for adversarial perturbations.

Misclassification aware regularization

- In the third graph, we saw that regularization on misclassified samples increases the robustness. We will exploit this property to obtain the final goal of this paper.
- With respect to current model h_θ, we define two subsets of natural examples ,one subset of correctly classified examples ($S^+_{h_\theta}$) and one subset of misclassified examples ($S^-_{h_\theta}$) :
- $S^+_{h_\theta} = \{i : i \epsilon [n], h_\theta(x_i) = y_i\}$ and $S^-_{h_\theta} = \{i : i \epsilon [n], h_\theta(x_i) \neq y_i\}$.
- Now we define adversarial risk separately for correctly classified and misclassified examples. We use the first point of this slide in order to define modified adversarial risk.

Adversarial risk for misclassified examples

$$R^{-}(\ h_\theta, x_i)\ := \mathbb{1}(h_\theta(\hat{x}_i^{'}) \neq y_i) + \mathbb{1}(h_\theta(x_i) \neq h_\theta(\hat{x}_i^{'})) \tag{1}$$

where the adversarial example $\hat{x}_i^{'}$ is generated by solving

$$\hat{x}_i^{'} = \arg \max_{x_i^{'} \epsilon B_\epsilon(x_i)} \mathbb{1}(h_\theta(x_i^{'}) \neq y_i) \tag{2}$$

Here the second term in RHS of equation (1) correspond to the standard adversarial risk and the regularization term $\mathbb{1}(h_\theta(x_i) \neq h_\theta(\hat{x}_i^{'}))$ aims to encourage the output of neural network to be stable against misclassified adversarial examples (used in previous works).

Adversarial risk for correctly classified examples

For correctly classified examples, $h_\theta(x_i) = y_i$ which implies

$1(h_\theta(x_i) \neq h_\theta(\hat{x}_i^{'})) = 1(h_\theta(\hat{x}_i^{'}) \neq y_i)$

This means that the regularizer has exactly same form as the adversarial risk.

Formally speaking,

$$R^{+}(\ h_\theta, x_i) \ := \max_{x_i^{'} \epsilon B_\epsilon(x_i)} 1(h_\theta(x_i^{'}) \neq y_i) = 1(h_\theta(\hat{x}_i^{'}) \neq y_i) \tag{3}$$

where the adversarial example $\hat{x}_i^{'}$ is defined in (2)

Overall adversarial risk

$$\min_{\theta} R_{misc}(h_\theta) := \frac{1}{n}\Big(\sum_{i\epsilon S^{+}_{h_\theta}} R^{+}(\ h_\theta, x_i) \ + \sum_{i\epsilon S^{-}_{h_\theta}} R^{-}(\ h_\theta, x_i)\ \Big) \tag{4}$$

$$= \frac{1}{n}\textstyle\sum_{i=1}^{n}\{1(h_\theta(\hat{x}_i') \neq y_i) + 1(h_\theta(x_i) \neq h_\theta(\hat{x}_i').1(h_\theta(x_i) \neq y_i)\} \tag{5}$$

where $\hat{x}^{'}_{i}$is given by (2).

The first term in (5) is the adversarial risk for correctly classified examples $(h_\theta(x_i) = y_i)$ for which the second term of (5) will be 0.

For misclassifed examples , $(h_\theta(x_i) \neq y_i)$, hence $1(h_\theta(x_i) \neq y_i)) = 1$ and so we will be left with expression for $R^{-}(\ h_\theta, x_i)$, which is desired.

Hence the equality from (4) to (5) follows

The 2nd term of (5) , $\frac{1}{n}\sum_{i=1}^{n} 1(h_\theta(x_i) \neq h_\theta(\hat{x}_i^{'}).1(h_\theta(x_i) \neq y_i)$, is the required misclassification aware regularization.

Replacing 0-1 loss with appropriate loss functions

Using 0-1 loss function in training is not tractable as derivatives can't be evaluated which is essential in back propagation.

That is the reason we need to use different loss functions .

Surrogate loss for outer minimization

The authors have used Boosted Cross Entropy (BCE) for 1st indicator function $1(h_\theta(\hat{x}_i^{'}) \neq y_i)$. But any other similar loss will also do the job.

Proposed BCE loss:

$$BCE(p(\hat{x}_i^{'}, \theta)) = -log(p_{y_i}(\hat{x}_i^{'}, \theta)) - log(1 - \max_{k \neq y_i} p_k(\hat{x}_i^{'}, \theta)) \tag{6}$$

Here, $p_k(\hat{x}_i^{'}), \theta)$ is the k-th logit . The first term in (6) is the commonly used CE loss and the second term is a margin term used to improve the decision margin of the classifier.

Surrogate loss for outer minimization contd..

For second indicator function $1(h_\theta(x_i) \neq h_\theta(\hat{x}_i^{'}))$, the authors used KL divergence, since $1(h_\theta(x_i) \neq h_\theta(\hat{x}_i^{'}))$ implies that adversarial examples have different output distributions to that of natural examples.

Hence minimizing KL divergence will imply that we want to bring the distribution of natural and adversarial examples closer. (7)

Surrogate loss for outer minimization contd..

For third indicator function, $1(h_\theta(x_i) \neq y_i)$, we notice that it emphasizes learning on misclassified examples.

We use soft decision scheme by replacing $1(h_\theta(x_i) \neq y_i)$ with the output probability 1-$p_{y_i}(x_i, \theta)$.

Surrogate loss for inner maximization

Goal of inner maximization is to find adversarial examples by solving equation (2).

Authors have used the commonly used cross entropy loss CE.

$$\hat{x}_i^{'} = \arg \max_{x'_i \epsilon B_\epsilon(x_i)} CE(p(x_i^{'}, \theta), y_i) \tag{8}$$

Overall Objective

From above discussion, it follows that:
$L^{MART}(\theta) = \frac{1}{n}\sum_{i=1}^{n} L(x_i, y_i, \theta)$ where

$$L(x_i, y_i, \theta) := BCE(p(\hat{x}_i^i, \theta), y_i) + \lambda . KL(p(x_i, \theta) || p(\hat{x}_i^{\prime}, \theta)).(1 - p_{y_i}(x_i, \theta)). \quad (9)$$

This is the desired loss function.

References

https://evademl.org/docs/featuresqueezing.pdf
https://arxiv.org/pdf/1511.04508.pdf
https://towardsdatascience.com/know-your-enemy-7f7c5038bdf3