Ensembling as a Defense Against Adversarial Examples

Liu, Evan
SUNetID: evanliu

Go, Brendon
SUNetID: bgo

December 15, 2016

1 Introduction

As more and more machine learning systems, such as spam detection, malware detection, and object recognition in self-driving cars, are being applied in the real world, the necessity to make these systems robust to adversaries grows.

Adversarial attacks on machine learning systems take two main flavors. First, there are training-time attacks, which involve compromising the training data that the system is trained on. Unsurprisingly, machines can misclassify examples, if they are trained on malicious data. Second, there are test-time attacks, which involve crafting an adversarial example, which a human would easily classify as some class A, whereas the system erroneously classifies it as some different class B [1]. For example, both images in Figure 1 are easily classified by humans to be a stop sign. In the adversarial setting, some classifier may misclassify the image on the right to be a speed limit 45 sign instead.

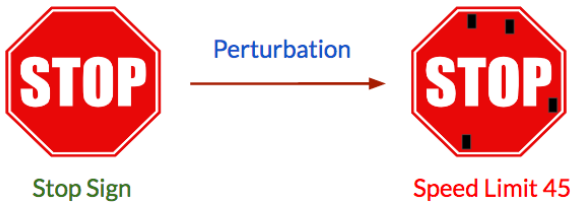

Figure 1: An adversarial example

Intuitively, these examples exist because the decision boundaries of machine learning classifiers differ from human decision boundaries. As a result, it is often possible to affect a machine learning system's classification by simply perturbing several pixels.

2 Attack Setting

In general, attackers do not have access to the model internals of the target system, including the exact architecture of the model and training dataset. One approach to attack such "black-box" systems is for attackers to build their own "substitute" system, which will attempt to approximate the "black-box" system [2]. Then, adversarial examples are generated against the "substitute" system, whose internals attackers have access to.

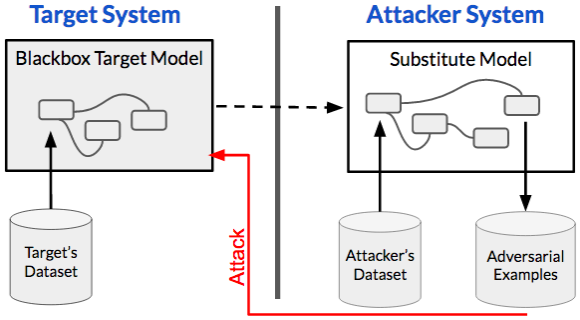

Figure 2: Attack Setting

Adversarial examples generated against the "substitute" system may still fool the real "black-box" system with high probability, even if the architecture and training data of the systems differ [6]. While this result is discouraging from the defense perspective, the amount of transferability is dependent on the "black-box" model type. For example, adversarial examples generated against any model type tend to also fool SVMs, and generally, adversarial examples generalize best against the same model type as the model on which they were generated.

3 Goal

Our goal is to provide a defense against test-time adversarial attacks without sacrificing accuracy. It has been shown that attacks generated on kNN have low transferability to neural models and vice versa [6]. We attempt to achieve lower transferability using an ensemble model of a kNN and a neural model. We will evaluate the ensemble model by measuring the transferability of attacks trained on our three different "substitute" models (kNN, CNN and Ensemble) and by comparing the test accuracy of the ensemble model against the test accuracy of the component models.

4 Approach

We perform all of our experiments on the MNIST dataset [5]. MNIST is a set of labeled grey-scaled images of handwritten digits from 0 to 9, expressed as an array of 784 pixel intensities. The MNIST train dataset has 55000 examples and we split this into three groups: 27000 to train our "black-box" models, 27000 to train our "substitute" models, and 1000 to create adversarial examples. The validation data is similarly split.

Both the "substitute" and "black-box" models are evaluated on the full MNIST test dataset. The "substitute" and "black-box" models have different architectures to emulate the fact that the attacker doesn't know the exact architecture of the "black-box" models in our setting.

5 Models

5.1 k-Nearest Neighbors

Both the "substitute" and the "black-box" kNN models use l_2 distance. The "substitute" model uses $k = 3$, and the "black-box" model uses $k = 5$ where k is the number of nearest neighbors that the model considers. Generating adversarial examples requires taking gradients of the scoring function. Since the kNN scoring function is not differentiable, when calculating the gradient we approximate kNN using soft-min [6], as:

$$s_k(x) = \frac{\sum_{z \in class_k(X)} e^{-\|z-x\|^2}}{\sum_{z \in X} e^{-\|z-x\|^2}}$$

where X is the set of training data, $class_k(X)$ is the set of all training data that has label k, and $s_k(x)$, the scoring function, is the weight that the model places on label k.

5.2 Convolutional Neural Network

The "substitute" CNN uses two convolutional layers followed by two fully-connected layers. The "black-box" CNN uses two max-pool layers followed by two fully-connected layers, and is trained using dropout in order to further increase its robustness to adversarial examples [3].

5.3 Ensemble Model

The Ensemble model ensembles kNN and CNN using the following scoring function, where α is trained using batched-gradient descent:

$$s(x) = \alpha s_{kNN}(x) + (1 - \alpha) s_{CNN}(x)$$

where s_{kNN} is the scoring function for its kNN component and s_{CNN} is the scoring function for its CNN component.

6 Adversarial Generator

We have also implemented an adversarial generator, which takes a model with scoring function $s_k(x)$, an example x, and a target label $l_{adversarial}$, and concocts adversarial examples by perturbing the given example using the model's scoring function. In order to ensure that the generator does not perturb the given example beyond human recognition, we ensure that the output adversarial example x^* is within an $\epsilon-$ball of x. If our generator is only able to create an adversarial example by leaving this $\epsilon-$ball, we throw out the example. We use $\epsilon = 0.15$ as in previous works [2]. The algorithm takes gradient steps toward the desired class with a step size of

η. Specifically, it uses the following algorithm [4]:

$x^* := x$;
while $model(x^*) \neq l_{adversarial}$ **do**
  $k := l_{adversarial}$;
  $x^* := clip(x + \eta sign(\nabla_x s_k(x)))$;
  if $x^* \notin B_\epsilon(x)$ **then**
    Fail;
  end
end

Algorithm 1: Adversarial Example Generation

We choose this algorithm instead of the traditional *Fast Gradient Sign* method [4], because it generates better adversarial examples. Using this algorithm, we generate an adversarial dataset for each "substitute" model, by attempting to perturb each of the 1000 examples to each label 0 to 9 excluding the original label. We then test the "black-box" models on this dataset.

7 Results

Figure 3 displays some adversarial examples created using our "substitute" models. It is clear that a human would have no difficulty in labeling these digits correctly as 9, 8, 5, 2, and 3. However the "black-box" models classify these as 1, 2, 3, 4, and 5 respectively. We can see that the perturbations are very minimal and would not arouse suspicion even with human visual inspection.

Figure 3: Adversarial Examples from left to right are classified as 1, 2, 3, 4, and 5

Our adversarial generator successfully generated about 2700 adversarial examples for each model. We measured transferability of an attack by testing what percentage of adversarial examples created using the "substitute" model are adversarial against the target "black-box" model. We count an example as an adversarial success if the target model classifies the example as our desired target class $l_{adversarial}$. We count an example to be a partial success if the target model misclassifies the adversarial example.

Substitute \ Target	kNN	CNN	Ens
kNN	18.7%/ 30.1%	12.0%/ 24.3%	11.4%/ 19.9%
CNN	4.2%/ 12.4%	11.2%/ 18.4%	9.6%/ 15.6%
Ens	4.7%/ 12.9%	14.0%/ 22.8%	8.8%/ 15.2%

Figure 4: Adversarial Example Transferability (Adversarial Success % / Partial Success %)

Figure 4 shows that the ensemble model is relatively robust to attacks generated with a kNN or CNN "substitute" model. Against adversarial examples generated on a kNN, the ensemble model achieves the lowest transferability. Against adversarial examples generated on a CNN, the ensemble model achieves a transferability rate between kNN and CNN. Encouragingly, the ensemble model achieves lower transferability with other ensemble models than CNN to CNN and kNN to kNN. This means that in the setting where the attacker knows what type of model is being deployed, if the attacker constructs his own "substitute" model of the same type, the ensemble model will be the most robust.

One result that surprised us is that the "black-box" ensemble model was more vulnerable to adversarial examples crafted for the "substitute" kNN and those crafted for the "substitute" CNN, than adversarial examples crafted for the "substitute" ensemble model. This may be because an adversarial example crafted for an ensemble might just partially fool both the underlying CNN and the underlying kNN, and therefore may not take advantage of the CNN to CNN or kNN to kNN transferability, whereas an adversarial example crafted just for a CNN or kNN

may better take advantage of this transferability.

Importantly, we achieve these results without sacrificing test accuracy. Figure 5 shows the test accuracy of our models. The ensemble model performs about as well as our other models do.

Our results suggest that ensembling helps as a defense against adversarial examples. However, while ensemble models may be more robust against attacks, they are still vulnerable to a non-negligible degree, so it is not a sufficient defense on its own.

	Substitute	Target
kNN	96.3%	96.1%
CNN	96.3%	97.7%
Ens	96.1%	97.7%

Figure 5: Test Accuracy

8 Future Work

In the future we would like to investigate whether ensembling works as a defense on more complex datasets like CIFAR10. We are also interested in ensembling other models together, such as ensembling multiple CNNs. We are also currently working on a different way to combine models. Distillation is a technique, where a deep neural network labels a dataset using soft-labels, which are just the soft-maxed logits. Next, a shallower neural network is trained on the soft-labels [3]. This has been shown to be an effective partial defense. We are working on using distillation, where a kNN labels a dataset using its scores, and then a CNN is trained on this. The CNN will be biased to approximate the kNN, achieving a different way of combining the CNN and kNN.

References

[1] N. Papernot, P. McDaniel, and al. The limitations of deep learning in adversarial settings. In Proceedings of the 1st IEEE European Symposium on Security and Privacy. IEEE, 2016.

[2] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, and al. Practical black-box attacks against deep learning systems using adversarial examples. arXiv preprint arXiv:1602.02697, 2016.

[3] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In Proceedings of the 37th IEEE Symposium on Security and Privacy. IEEE, 2016.

[4] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In Proceedings of the 2015 International Conference on Learning Representations. Computational and Biological Learning Society, 2015.

[5] Y. LeCun and C. Cortes. The mnist database of handwritten digits, 1998.

[6] Papernot, N., McDaniel, P., and Goodfellow, I. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples. ArXiv e-prints, May 2016b. URL http://arxiv.org/abs/1605.07277.

9 Acknowledgements

We would like to thank Bahman Bahmani for having a conversation with us about what the security community is thinking about with respect to adversarial machine learning.