Security and Privacy of Machine Learning - Homework 1

Wu-Jun Pei
National Taiwan University
b06902029@ntu.edu.tw

Abstract

In this homework, I implemented adversarial attacks against CIFAR-10 image classifiers under a gray-box setting. I tried several attacking methods and consider multiple attacking parameters. Furthermore, I also implemented some possible preprocessing-based defenses so that I can evaluate in a more approporiate setting. Based on the experiments I made, I finally selected the final adversarial examples that could tear down xxx% on our evaluation set.

1 Introduction

Powerful as neural networks are, they are also vulnerable to malicious adversarial examples. The perturbations are imperceptible to human; however, they are likely to make the model output the wrong answer with high confidence.

In this homework, I implemented adversarial attacks against CIFAR-10 image classifiers under a gray-box setting. The models are chosen from imgcls[1], and the adversarial perturbation is $l_\infty = 8/256$.

Code is available at: `https://github.com/jimpei8989/SPML-HW1`.

2 Attacking Details

In this section, I'll explain the attacking methods and attacking techniques I use. Moreover, some experiments are conducted to achieve the final adversarial examples. During the experiments, the attack accuracies are not the only metric I consider, transferbility is also taken into account since it's a gray-box setting. Intuitively, I use the standard deviation of the accuracies among different models to examine transferbility. The greater the value is, the less transferbility it is.

2.1 Attacking Mechanisms

To find adversarial examples, that is, to find a perturbation δ such that

$$\delta = \underset{x' \in \mathrm{Adv}(x)}{\arg\min} \, (x', y; \theta) \tag{1}$$

The simplest method is Fast Gradient Sign Method[2]. To achieve better attack performance, we also tried Iterative Fast Gradient Sign Method (I-FGSM)[3], Momentum Iterative Fast Gradient Sign Method (MI-FGSM)[4] and Projective Gradient Descent[5].

The experiment results are shown on table 2.1. In I-FGSM, MI-FGSM and PGD, the number of iterations is selected to be 4 based on some random experiments. The magic number selected on my own, considering both attack accuracy and transferbility.

Table 1: Experiment results. The evaluation models A-F are `nin`, `resnet20`, `resnext29-32x4d`, `seresnet20`, `pyramidnet110-a48`, `densenet40-k12`, respectively. Bold texts indicate the proxy model is the same as the evaluation model.

Settings		Evaluation models							
Mechanisms	Proxy Model	A	B	C	D	E	F	Avg.	Std. Dev.
None	None	0.94	0.96	1.00	0.97	1.00	0.98	0.9750	0.0214
FGSM	ResNet20	0.64	**0.36**	0.54	0.53	0.56	0.53	0.5267	0.0836
	ResNext20	0.81	0.73	**0.45**	0.70	0.63	0.72	0.6733	0.1129
	DenseNet40	0.73	0.67	0.58	0.71	0.65	**0.56**	0.6500	0.0624
	Average								
I-FGSM # iterations = 4	ResNet20	0.77	**0.03**	0.74	0.41	0.50	0.42	0.4783	0.2460
	ResNext20	0.90	0.80	**0.06**	0.77	0.58	0.67	0.6300	0.2740
	DenseNet40	0.86	0.62	0.71	0.56	0.55	**0.10**	0.5667	0.2336
	Average								
MI-FGSM # iterations = 4	ResNet20	0.67	**0.03**	0.69	0.38	0.44	0.40	0.4350	0.2193
	ResNext20	0.89	0.79	**0.04**	0.74	0.58	0.66	0.6167	0.2756
	DenseNet40	0.82	0.62	0.64	0.53	0.52	**0.10**	0.5383	0.2194
	Average								
PGD # iterations = 4	ResNet20	0.63	**0.00**	0.55	0.31	0.41	0.31	0.3683	0.2024
	ResNext20	0.86	0.73	**0.03**	0.66	0.43	0.55	0.5433	0.2662
	DenseNet40	0.77	0.54	0.51	0.60	0.43	**0.04**	0.4817	0.2233
	Average								

Table 2: Experiment results. The evaluation models A-G are `nin`, `resnet20`, `resnext29-32x4d`, `seresnet20`, `pyramidnet110-a48`, `densenet40-k12`, `resnet1001`, respectively. Bold texts indicate the proxy model is the same as the evaluation model.

Proxy Model	A	B	C	D	E	F	G	Avg.	Std. Dev.

From the table, we can tell that FGSM is simple and already performances well. On the other hand, the iterative methods, I-FGSM, MI-FGSM and PGD, lead to an ever better attack accuracy. However, they have little transferbility as the standard deviations are much larger. Also, we can observe that those non-proxy models have similar accuracies among row 1, row 5, row 9 and row 13. I think all these methods focus only on the proxy model so that the adversarial examples have limited effect on other models.

As for the selection of proxy model, it's relatively better when we choose ResNet20. Thus, I'll choose it as the experiment proxy model in the next sections. Also, I'll consider PGD as the major attack mechanism.

2.2 Proxy Model Selection

Selection of proxy models plays an important role in black / gray box attacks. A better selection may lead to better transferbility and thus make the attack more successful without knowing the exact model. In this part, we want to discuss the selection in three manners, *one weak model*, *one strong model*, and *ensemble of multiple models*.

First of all, let's discuss first two first. The *weak* models and *strong* models are mainly examined by the number of layers, parameters and FLOPs. In this experiment, I'll take `ResNet20` and `ResNet1001` as the *weak* and *strong* models, respectively.

Table 3: Experiment results. For the first four rows, they are tested on the benign examples with different defense methods.

Settings		Evaluation models							
Attack Method	Defense Method	A	B	C	D	E	F	Avg.	Std. Dev.
None	None	0.94	0.96	1.00	0.97	1.00	0.98	0.9750	0.0214
	Gaussian (radius=1)	0.63	0.57	0.38	0.49	0.53	0.50	0.5167	0.0770
	JPEG (quality=80)	0.86	0.83	0.92	0.85	0.85	0.81	0.8533	0.0340
	JPEG (quality=60)	0.85	0.77	0.83	0.77	0.79	0.79	0.8000	0.0300

2.3 Possible Defenses

To evaluate our attack more precisely, we add two possible preprocessing-based defenses in the evaluation phase.

- **Gaussian Blur**: We tried to add some gaussian blur in the evaluation phase. However, even we set the radius to 1 have a terrible accuracy. Since the image size is 32x32, a 3x3 filter might be so big for it that the classifiers are not able to classify the blur images even they haven't been perturbed.
- **JPEG Compression** [6]: We can adopt the systematic compression method to reduce adversarial noise in the preprocessing phase. In this experiment, we tried two quality parameters, 60 and 80, which are both able to

Experiments

2.4 Some Miscellious Techiniques Used

2.5 Final Decision

3 Conclusion

References

[1] "Convolutional neural networks for computer vision." `https://github.com/osmr/imgclsmob`.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[3] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[4] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9185–9193, 2018.

[5] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[6] N. Das, M. Shanbhogue, S.-T. Chen, F. Hohman, L. Chen, M. E. Kounavis, and D. H. Chau, "Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression," *arXiv preprint arXiv:1705.02900*, 2017.