Security and Privacy of Machine Learning - Homework 1

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

Abstract

In this homework, we[1] implemented adversarial attacks against CIFAR-10 image classifiers under a gray-box setting. We've tried several attacking methods and consider multiple attacking parameters. Furthermore, we also implemented some possible preprocessing-based defenses so that we can evaluate in a more approporiate setting. Based on the experiments we made, we finally selected the final adversarial examples that could perfectly take down the model. Our attack is also effective on models with strong preprocessing-based defense.

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 classifiers output the wrong label with high confidence.

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

Our code is available at: `https://github.com/jimpei8989/SPML-HW1`[2].

2 Attacking Details

In this section, we'll explain the attacking methods and attacking techniques we 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 and we have no knowledge about the models used. Intuitively, we use the standard deviation of the accuracies among different models to examine transferbility. The greater the value is, the less transferbility it has as models have different performance on our adversarial examples.

2.1 Attacking Mechanisms

To find adversarial examples[2], that is, to find a perturbation δ such that

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

The simplest method is Fast Gradient Sign Method[3]. To achieve better attack performance, we also tried Iterative Fast Gradient Sign Method (I-FGSM)[4], Momentum Iterative Fast Gradient Sign Method (MI-FGSM)[5] and Projective Gradient Descent[6].

[1]The subject is "we" although I do the homework on my own only :)

[2]It is made public after the deadline.

Table 1: Experiment results on different attack methods. The evaluation models A-F are `resnet20`, `sepreresnet20`, `densenet40-k12`, `nin`, `resnext29-32x4d`, `pyramidnet110-a48`, 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`	**0.36**	0.53	0.53	0.64	0.54	0.56	0.5267	0.0836
	`sepreresnet20`	0.58	**0.49**	0.60	0.71	0.61	0.65	0.6067	0.0670
	`densenet40-k12`	0.67	0.67	**0.56**	0.73	0.58	0.65	0.6433	0.0576
	Average	0.54	0.56	0.56	0.69	0.58	0.62	0.5922	0.0694
I-FGSM # iterations = 4	`resnet20`	**0.03**	0.37	0.42	0.77	0.74	0.50	0.4717	0.2483
	`sepreresnet20`	0.52	**0.20**	0.57	0.78	0.85	0.66	0.5967	0.2104
	`densenet40-k12`	0.62	0.65	**0.10**	0.86	0.71	0.55	0.5817	0.2356
	Average	0.39	0.41	0.36	0.80	0.77	0.57	0.5500	0.2314
MI-FGSM # iterations = 4	`resnet20`	**0.03**	0.36	0.40	0.67	0.69	0.44	0.4317	0.2203
	`sepreresnet20`	0.45	**0.19**	0.51	0.72	0.69	0.59	0.5250	0.1768
	`densenet40-k12`	0.62	0.58	**0.10**	0.82	0.64	0.52	0.5467	0.2199
	Average	0.37	0.38	0.34	0.74	0.67	0.52	0.5011	0.2057
PGD # iterations = 4	`resnet20`	**0.00**	0.28	0.31	0.63	0.55	0.41	0.3633	0.2041
	`sepreresnet20`	0.33	**0.06**	0.42	0.74	0.61	0.41	0.4283	0.2144
	`densenet40-k12`	0.54	0.56	**0.04**	0.77	0.51	0.43	0.4750	0.2202
	Average	0.29	0.30	0.26	0.71	0.56	0.42	0.4222	0.2129

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.

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 rate. However, they have worse transferbility as the standard deviations are significantly larger. We can also observe that the attack is only effective on those proxy models, but have slight effect on other models.

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. From the previous experiment, we can observe that different proxy models have different attack rate. In this part, we want to discuss the selection in the following three manners, *one weak model*, *one strong model*, and *ensemble of multiple models*.

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`, `sepreresnet20` and `densenet40-k12` as *weak* models, `resnet1001`, `sepreresnet542bn` and `densenet100-k24` as *strong* models. As for the *ensemble*, I average the model outputs (logits) and take the whole process as an ensemble model. Same attacking mechanism can be directly applied on the ensemble models.

The experiment results are shown in table 2.2. To our surprise, if we choose strong models as proxy models, the outcome standard deviation is unignorably smaller, indicating that stronger proxy models yield better transferbility. The result is contrast to what we thought at first[3], that strong models with greater capacity will make the adversarial examples "*overfit*" to them. So far, we haven't had a better explanation on it.

At last, it's clear to see that *ensemble of several models* does improve the transferbility a lot. Interestingly, when we take ensemble of model A-F as proxy model, meaning that we have no knowledge

[3]Discussion with Andy Chen (B06902001)

Table 2: The evaluation models A-G are `resnet20`, `resnet1001`, `sepreresnet20`, `sepreresnet542bn`, `densenet40-k12`, `densenet100-k24`, `pyramidnet110-a48`, respectively. Bold texts indicate the proxy model is the same as the evaluation model. The attacking mechanism is PGD, with number of iterations 8.

Category	Proxy Model(s)	A	B	C	D	E	F	G	Avg.	Std. Dev.
Weak	`resnet20`	**0.00**	0.75	0.21	0.51	0.27	0.62	0.38	0.3914	0.2378
	`seresnet20`	0.30	0.70	**0.02**	0.54	0.38	0.62	0.43	0.4271	0.2099
	`densenet40-k12`	0.55	0.89	0.55	0.66	**0.00**	0.54	0.41	0.5143	0.2510
	Average	0.28	0.78	0.26	0.57	0.21	0.59	0.40	0.4443	0.2329
Strong	`resnet1001`	0.49	**0.18**	0.48	0.59	0.43	0.61	0.42	0.4571	0.1318
	`seresnet542bn`	0.55	0.87	0.60	**0.16**	0.43	0.59	0.48	0.5257	0.1978
	`densenet100-k24`	0.61	0.84	0.65	0.58	0.45	**0.09**	0.41	0.5186	0.2181
	Average	0.55	0.63	0.57	0.44	0.43	0.43	0.43	0.5005	0.1826
Ensemble	A, C, E	0.00	0.50	0.00	0.20	0.00	0.32	0.13	0.1643	0.1774
	B, D, F	0.39	0.20	0.37	0.16	0.27	0.15	0.20	0.2486	0.0906
	A-F	0.00	0.06	0.00	0.01	0.00	0.01	0.04	0.0171	0.0219

Table 3: Experiment results. For the first four rows, they are tested on the benign examples with different defense methods. The adversarial examples are generated using PGD, with 8 iterations.

Settings		Evaluation models							
Attack Method	Defense Method	A	B	C	D	E	F	Avg.	Std. Dev.
None	None	0.96	0.98	0.98	0.94	1.00	1.00	0.9767	0.0213
	Gaussian (radius=1)	0.57	0.55	0.50	0.63	0.38	0.53	0.5267	0.0767
	JPEG (quality=80)	0.83	0.82	0.81	0.86	0.92	0.85	0.8483	0.0362
	JPEG (quality=60)	0.77	0.81	0.79	0.85	0.83	0.79	0.8067	0.0269
Ensemble PGD-8	None	0.00	0.00	0.00	0.55	0.08	0.04	0.1117	0.1982
	JPEG (quality=80)	0.52	0.46	0.48	0.70	0.60	0.56	0.5533	0.0806
	JPEG (quality=60)	0.66	0.59	0.68	0.77	0.72	0.62	0.6733	0.0599

about G during the attacking phase, the adversarial images have a better attack rate on G. This shows that ensemble of several models could help a lot us on the transferbility.

2.3 Possible Defenses

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

- **Gaussian Blur**: A heuristic way to add some noise to the image to reduce adversarial noise.
- **JPEG Compression** [7]: A systematic compression that can reduce adversarial noise.

The experiment results are shown on table 2.3. Unluckily, it seems not to be a good idea to add gaussian noise because it also affect the benign dataset a lot. As for JPEG compression method, the performance on benign dataset is much more acceptable (original accuracy > 0.8), and the defense on the adversarial dataset is also successful (adversarial accuracy > 0.5).

2.4 Iterations

As a chinese saying goes:

Constant dropping wears the stone.

Table 4: Experiment results of increasing number of iterations. Here we use an ensemble of several models, as listed in section 2.5. The adversarial examples are generated using PGD, and the defense we use here is JPEG Compression with quality 80.

	w/o defense		w/ defense	
# iterations	Avg.	Std. Dev.	Avg.	Std. Dev.
2	0.3023	0.0963	0.6438	0.0383
8	0.0715	0.0764	0.5508	0.0639
32	0.0438	0.0697	0.5438	0.0697
128	0.0400	0.0695	0.5631	0.0678
512	0.0400	0.0732	0.5515	0.0747

We want to examine whether it's good if we increase the number of iterations in iterative methods. The settings are the same with section 2.5. In the experiments below, we use a ensemble of several models, and the adversarial images will be tested with and without defenses. The defense we use here is *JPEG-80*.

The experiment results are shown on table 2.4. As we increase the number of iterations, the attack rate, both with and without defense applied on evaluation models, improve only by a slight amount. Thus,

2.5 Final Decision

In previous sections, we discussed about several methods used in this homework. Those experiments gave us useful insights and helped us to come up with the final selection to generate adversarial examples. We have the following settings for our adversarial examples:

- Attach mechanism: **PGD**, with **32** iterations.
- Proxy model: **Ensemble of several models**, including `resnet20`, `resnet1001`, `sepreresnet20`, `sepreresnet542bn`, `densenet40-k12`, `densenet100-k24`, `pyramidnet110-a48`, `resnext29-32x4d` and `nin`.

The attack rate is **4.38%** on the models without any defense, and **54.38%** on the models with strong preprocessing-based deense.

3 Conclusion

In this homework, we've tried several methods and techniques to achieve better attack rate. For attacking method, we've tried FGSM and several iterative methods, and we found PGD to be the most desirable. Another important issue is the selection of proxy models, we've made some experiments on weak models, strong models and ensemble of models, and we find ensemble to be the most powerful choice. Moreover, we also applied some preprocessing-based defense and evaluate our works with such defense. At last, we show that our adversarial examples can attack the models well, even they are equipped with strong preprocessing-based defenses.

References

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

[2] "Course slides of security and privacy of machine learning." `https://www.csie.ntu.edu.tw/~stchen/`.

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

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

[5] 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.

[6] 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.

[7] 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.