Security and Privacy of Machine Learning Homework 1

Bo-Wei Huang
National Taiwan University
r10922007@csie.ntu.edu.tw

Abstract

In this homework, we[1] implement adversarial attacks against classifiers trained on CIFAR-100 under black-box (Phase One) and gray-box (Phase Two) settings. For black-box attack, we've tried four attacking methods without targeted classes and considered multiple combinations of proxy models for better transferability. Further, for grey-box attack, we've tried different hyperparameters, implemented some popular preprocessing-based defenses and included more proxy models. By conducting several experiments, we finally select the best settings for both black-box and grey-box attacks, believing they have relatively good ability to crash down target models during evaluation.

1 Introduction

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

In this homework, we implement untargeted adversarial attacks against CIFAR-100 classifiers under black-box (Phase One) and gray-box (Phase Two) settings. For the former, there is nothing we are informed of the target models; for the latter, we can upload our adversarial images to a query system, receiving top-5 logits predicted by the target model. The adversarial perturbation δ is always restricted to $||\delta||_\infty \leq 8$.

For all experiments, average accuracy of 500 images from TA is measured. To reproduce the results, please refer to `https://github.com/joe0123/SPML2021`[2].

2 Phase One - Black-box Attack

In this section, we'll explain the untargeted attacking methods we utilize for black-box attack. Moreover, some experiments are carried out for the final selection. Since we have no knowledge about target models, we use ensemble of models to enhance transferability. In addition, we set learning rate and number of iterations constantly to 1 and 100 regardless of attacking algorithms and proxy models.

2.1 Untargeted Attacking Algorithm

According to the definition of untargeted adversarial attack [1], given a classifier θ, an example x and its ground-truth label y, one can find the best perturbation δ, s.t. $||\delta||_\infty \leq 8$, without targeted class by

[1]The subject is "we" although I do the homework by myself

[2]It is made public after the deadline

optimizing the following equation:

$$\delta = \arg\max_{\delta} \ l(x + \delta, y; \theta)$$

One of the most efficient and effective approximation method is Fast Gradient Sign Method (FGSM) [2]. To achieve better attack performance, we also take Iterative Fast Gradient Sign Method (I-FGSM) [3], Projected Gradient Descent (PGD) [4] and Optimization-based method[3] (OPT) into accounts.

2.2 Experiments

In this part, several experimental evidences are provided to help find relatively transferable attacking settings. First, we'll discuss the selection criterion of proxy models from imgcls [5], and the transferability of attacking algorithms FGSM, I-FGSM, PGD and OPT. Moreover, we'll empirically show that involving models trained with different source codes, i.e. weiaicunzai[4] [6], can effectively mitigate possible overfitting on training details of proxy models, achieving better transferability of adversarial attacks.

2.2.1 Selection of Attacking Algorithms

Before discovering ideal attacking algorithm, we will first explain how we comprise the set of proxy models. Since architectures differ greatly between families of neural networks, like ResNet and DenseNet, we decided to ensemble almost every family for adversarial attacks. However, the representatives for each family should be carefully selected because internal transferability are not guaranteed. Specifically, as shown in Table 1, transferability declines as capacities of proxy model and target model deviate. After considering effectiveness and efficiency, we finally select one small and one large model for each family, which can empirically achieve acceptable transferability.

All trained models for selection of attacking algorithms are provided by imgcls. When generating adversaries, we ensemble 14 models, i.e. `resnet20`, `resnet1001`, `preresnet20`, `preresnet1001`, `seresnet20`, `seresnet272bn`, `densenet40_k12`, `densenet250_k24_bc`, `pyramidnet110_a84`, `pyramidnet272`, `resnext29_32x4d`, `wrn28`, `nin` and `ror3_164`.

As for evaluation, we consider three situations:

- **In Model**: Target models are identical to proxy models.
- **In Family**: Families of target models are included by those of proxy models, but target models are not proxy models. Selected target models are `resnet56`, `preresnet164_bn`, `densenet100_k12` and `pyramid236`.
- **Out**: Target models are totally different from proxy models. Selected target models are `sepreresnet56`, `shakeshakeresnet`, `diaresnet110`, `rir` and `xdensenet_40`.

The experiment results are listed in Table 2. Roughly speaking, with ensemble of proxy models, iterative methods, such as I-FGSM and OPT, outperform single-step FGSM. Particularly, I-FGSM can perfectly attack almost every proxy model without losing transferability. On the other hand, PGD, though being iterative, gets surprisingly poor but stable performance across the three evaluation situations. It suggests that PGD sacrifices too much for generalization. Maybe more sophisticated tuning for hyperparameters is required to boost the performance, but we leave it as future work.

2.2.2 More diverse proxy models

Intuitively, models trained by the same scripts might inherit some common properties affected by training details, such as the setting of hyperparameters, the choice of learning rate scheduler or early stopping criterion, and so on. If we neglect diversifying those configurations, the variety of proxy models might be limited and the transferability hurts. As shown in Table 3, attack with proxy models from imgcls can only defeat those from the same training source. When facing models trained with other scripts, performance declines to some degree.

[3] By using Adam optimizer on δ, we maximize $l(x + \delta, y; \theta)$ directly

[4] We use the owner name to identify the github repository

Table 1: Experiment results on ResNet family, where I-FGSM with learning rate 1 and number of iterations 100 is considered.

Proxy	Target					
	20	56	110	164	272	1001
None	0.82	0.95	0.99	0.99	1.00	1.00
20	0.00	**0.22**	0.29	0.37	0.46	0.65
56	**0.15**	0.00	**0.27**	0.33	0.40	0.65
110	0.16	0.24	0.01	0.32	0.38	0.68
164	0.23	0.25	0.32	0.00	**0.29**	0.66
272	0.22	0.29	0.34	**0.24**	0.00	**0.61**
1001	0.19	0.28	0.32	0.29	0.33	0.00

Table 2: Experiment results on different attacking algorithms. The evaluation models A-I are `resnet56`, `preresnet164_bn`, `densenet100_k12`, `pyramid236`, `sepreresnet56`, `shakeshakeresnet`, `diaresnet110`, `rir`, `xdensenet_40`.

Algor.	In Model		In Family				Out				
	Avg.	Std.	A	B	C	D	E	F	G	H	I
None	0.94	0.08	0.95	1.00	1.00	1.00	0.97	0.98	0.99	1.00	0.97
FGSM	0.26	0.12	0.20	0.18	0.18	0.40	0.28	0.25	0.23	0.31	0.14
I-FGSM	**0.01**	0.03	**0.02**	**0.01**	**0.01**	**0.05**	**0.03**	**0.02**	**0.04**	**0.02**	**0.01**
PGD	0.38	0.02	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.39
OPT	0.06	0.04	0.06	0.05	0.04	0.12	0.11	0.07	0.09	0.07	0.03

In this regard, we decided to include different training sources to expand our proxy models; that is, add `vgg16` and `googlenet` from weiaicunzai to the set of models selected in Section 2.2.1. For evaluation, we select the same models from imgcls as in Table 2 (Out), and `seresnet50`, `mobilenetv2`, `nasnet`, `xception` from weiaicunzai. Additionally, to be fair, we consider `resnet32` and `shufflenetv2_x1_5` from the third model hub chenyaofo [7], which is independent of proxy models.

The experiment results are shown in Table 3, where I-FGSM, the winner in Section 2.2.1, is used as attacking algorithm. After taking weiaicunzai into accounts, the effect of attacks improves significantly, even in the case of chenyaofo. It is empirically proved that attacking algorithm might overfit on training details of proxy models, and involving models trained with different source codes can effectively enhance transferability.

2.3 Final Selection

Since all information remains unknown in black-box setting, transferability should be highlighted. From the experiment results above, we ultimately select I-FGSM with learning rate 1 and number of iterations 100 as our attacking algorithm. As for proxy models, we choose 14 models from imgcls and 2 models from weiaicunzai, as mentioned in the previous section.

Table 3: Experiment results on different sources of proxy models. The evaluation models A-F are `seresnet50`, `mobilenetv2`, `nasnet`, `xception`, `resnet32` , `shufflenetv2_x1_5`.

Source	imgcls (Out)		weiaicunzai				chenyaofo	
	Avg.	Std.	A	B	C	D	E	F
None	0.94	0.08	1.00	0.88	1.00	1.00	0.99	1.00
imgcls	**0.02**	0.01	0.29	0.17	0.20	0.35	0.07	0.17
+ weiaicunzai	**0.02**	0.01	**0.17**	**0.14**	**0.11**	**0.19**	**0.04**	**0.11**

Table 4: Experiment results on different number of iterations. Learning rate is fixed to 1 like Phase One.

Max iter	Top1	Top3	Top5
100	0.34	**0.68**	0.81
10	0.34	0.68	0.82
20	**0.33**	0.69	0.80
50	0.36	**0.68**	**0.80**

Table 5: Experiment results on different learning rates. Number of iterations is fixed to 20 since it's the winner in Table 4.

LR	Top1	Top3	Top5
1.0	0.33	0.69	0.80
0.1	0.28	0.64	0.75
0.2	**0.25**	**0.62**	**0.74**
0.5	0.28	0.63	0.76

3 Phase Two - Grey-box Attack

In this section, predictions of target model can be obtained via query system. With such knowledge, we can adjust our attacking details to achieve better performance. Following the results in Phase One, we fix our attacking algorithm to I-FGSM with ensemble of proxy models, which is the most transferable approach among our attempts.

3.1 Experiments

In this part, we conduct experiments to tune hyperparameters of I-FGSM, measure the effect of attacks considering different popular defenses, and re-select our proxy models delicately. In Section 3.1.1 and 3.1.2, we follow the selection of proxy models in Phase One but replace 14 models from imgcls to 8 models (`resnet110`, `preresnet110`, `seresnet110`, `densenet100_k12`, `pyramidnet110_a84`, `resnext29_32x4d`, `nin` and `ror3_164`) for computational efficiency. In Section 3.1.3, imgcls, weiaicunzai and chenyaofo are all considered, and the details will be provided then.

3.1.1 Tuning Hyperparameters

Configurations of hyperparameters are crucial for attacking algorithm. In black-box attack (Phase One), we fix number of iterations to 100 and learning rate to 1. However, according to query system, they are not optimal choices. With the results shown in Table 4 and 5, we adjust our number of iterations to 20 and learning rate to 0.2 for all the rest of experiments in Phase Two. Such settings can reduce Top-1 accuracy from 0.34 to 0.25, indicating better transferability.

3.1.2 Attack Against Defenses

Preprocessing-based defenses can suppress noises in input images, improving model robustness against adversarial attacks to some extent. There are three popular defenses considered in this part:

- **JPEG Compression** [8]: A systematic compression that can reduce ignorable noise.
- **Spatial Smoothing** [9]: Local smoothing method to reduce differences among pixels.
- **Gaussian Blur**: Blurring images with Gaussian function.

One heuristic way to breach these defenses is including proxy models trained with certain preprocessed data. However, since there is no time re-training models, we directly forward preprocessed images into the models from imgcls, weiaicunzai or chenyaofo when generating adversaries, which might certainly interfere model predictions on clean images, making proxy models unreliable. Therefore, intensity of each defense must be set moderately. In order not to sacrifice too much accuracy,

Table 6: Experiment results of models selcted by ourselves on different popular preprocessing-based defenses. In Model is identical to proxy models, and Out is the same as mentioned in Section 2.2.1. Moreover, ✓(*) indicates the preprocessing method used before proxy models, and ✓(None) means no preprocessing is considered when generating adversaries.

Defense	Attack	In Model		Out	
		Avg.	Std.	Avg.	Std.
None	✗	0.98	0.04	0.98	0.01
None	✓(None)	0.01	0.01	0.01	0.00
JPEG	✗	0.88	0.05	0.88	0.06
JPEG	✓(None)	0.12	0.03	0.16	0.04
JPEG	✓(JPEG)	0.03	0.02	0.05	0.01
Spatial	✗	0.87	0.05	0.87	0.05
Spatial	✓(None)	0.12	0.06	0.16	0.03
Spatial	✓(Spatial)	0.10	0.07	0.15	0.03
Blur	✗	0.91	0.07	0.92	0.04
Blur	✓(None)	0.04	0.05	0.06	0.02
Blur	✓(Blur)	0.01	0.03	0.02	0.01

Table 7: Experiment results from query system on different popular preprocessing-based defenses.

Attack	Top1	Top3	Top5
✓(None)	0.25	0.62	0.74
✓(JPEG)	0.32	0.65	0.76
✓(Spatial)	0.57	0.82	0.88
✓(Blur)	**0.22**	**0.55**	**0.69**

quality of JPEG compression is set to 90, window size of spatial smoothing is set to 2, and radius of Gaussian blur is set to 0.6.

We first validate the defenses and attacks by ourselves. As shown in Table 6, when facing naive adversarial attacks, i.e. ✓(None), all defenses can slightly prevent models from being attacked, increasing accuracy from 0.01 to a range between 0.04 and 0.16. On the other hand, by comparing ✓(None) and ✓(*), adversarial attacks with preprocessing techniques can break through their corresponding defenses to some degree, indicating our heuristics works in some way.

Finally, we resort to query system for the selection of preprocessing methods. In Table 7, Gaussian blur outperforms naive adversarial attacks surprisingly; in contrast, JPEG compression and spatial smoothing receive incredibly poor accuracy.

Although we cannot assert that the target model utilizes Gaussian blur defense, we still believe such preprocessing can help promote our attacks in the face of the target model.

3.1.3 Review on proxy models

As we can receive partial knowledge about the target model, re-selecting proxy models is a reasonable try to improve attacking performance.

Following the conclusion of Section 2.2.2, multiple sources of proxy models are considered. For this experiment, selections from imgcls, weiaicunzai and chenyaofo are listed below:

- **imgcls**: Eight models are selected, i.e. `resnet110`, `preresnet110`, `seresnet110`, `densenet100_k12`, `pyramidnet110_a84`, `resnext29_32x4d`, `nin` and `ror3_164`.
- **weiaicunzai**: Five models are selected, i.e. `vgg16`, `googlenet`, `seresnet50`, `xception` and `inceptionv4`.
- **chenyaofo**: Five models are selected, i.e. `resnet32`, `vgg16`, `mobilenetv2_x1_0`, `shufflenetv2_x1_5`, `repvgg_a1`.

Table 8: Experiment results on different sources of proxy models.

Source	Top1	Top3	Top5
imgcls	0.32	0.66	0.77
weiaicunzai	0.39	0.74	0.84
chenyaofo	0.40	0.77	0.87
all	**0.20**	**0.58**	**0.72**

Table 9: Experiment results on final selection in different phases. Public set consists of 500 images from TA, while private set is composed of 500 images sampled by ourselves.

Phase	Public			Private		
	Top1	Top3	Top5	Top1	Top3	Top5
Blackbox	0.34	0.68	0.81	-	-	-
Greybox (w/o blur)	0.17	0.59	0.71	0.18	0.58	0.70
Greybox (w/ blur)	**0.14**	**0.48**	**0.63**	**0.17**	**0.51**	**0.64**

As shown in Table 8, adversaries generated from single source of proxy models cannot successfully deceive the target model like those generated from all sources do. The results are actually unsurprising since they are aligned with the concept in Section 2.2.2.

3.2 Final Selection

Based on predictions of the target model, we ultimately select I-FGSM with learning rate 0.2 and number ofiterations 20 as our attacking algorithm in grey-box setting. In addition, we implement Gaussian blur preprocessing ahead of proxy model. As for proxy models, we choose the models mentioned in Section 3.1.3 and adjust the set of imgcls back to 14-model version as in black-box setting.

The final results are listed in Table 9. To re-verify our selection, especially the choice of preprocessing mechanism, we sample another 500 images from CIFAR-100 train set for evaluation. As expected, considering Gaussian blur is beneficial to adversarial attacks when facing the target model.

4 Conclusion

In this homework, we've tried several methods and techniques to achieve better performance. For attacking algorithm, we've tried FGSM and other iterative methods, and we finally select I-FGSM. Furthermore, we've empirically proven that attacking algorithm might overfit on training details of proxy models; therefore, multiple sources of proxy models are applied. Last but not least, with query system in Phase Two, we've tuned hyperparamters, considered popular defenses and re-selected our proxy models. Our selection receives 30% error-rate improvement from black-box setting to grey-box setting.

References

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

[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] 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.

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

[6] "Pytorch-cifar100." https://github.com/weiaicunzai/pytorch-cifar100.

[7] "Pytorch cifar models." https://github.com/chenyaofo/pytorch-cifar-models.

[8] 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.

[9] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *Proceedings 2018 Network and Distributed System Security Symposium*, 2018.