A Brief Report: Implementation of Different Adversarial Attacks

Haowei Lin 1900017808
Class of Artificial General Intelligence, Yuanpei College
Peking University
linhaowei@pku.edu.cn

Abstract

This report is a brief description of my implementation for several adversarial attacks. As required, I implemented FGSM (Goodfellow et al. [2]), PGD(Madry et al. [5]). I also implemented MI-FGSM(Dong et al. [1]) for white-box attack. For black-box attack, I exploited MI-FGSM as transfer-based attack. On top of that, I explored the impact of different factors in attacks, including random initialization, step size, model architecture and loss function, to name a few. Codes and report are available in `https://github.com/linhaowei1/Introduction-to-AI/tree/main/Attack`.

1. White-box Attacks

In white box attacks, there is no restriction in the level of information to which the attacker can access. As a consequence the adversary knows model parameters, dataset, or any other information regarding the model. Under such assumption, given a model $f(.)$, an input (x, y), the main objective is to produce x', which is within certain distance from the original x and maximizes the loss $\mathcal{L}(f(x+\delta, y)$.

$$\max_{\delta \in \Delta} \mathcal{L}(f(x+\delta), y)$$

As required, we consider the $L\infty$ distance for δ. For MNIST we set $||\delta||_\infty \leq 0.3$, and $||\delta||_\infty \leq 0.031$ for CIFAR10.

1.1. Fast Sign Gradient Method

FGSM proposed by Goodfellow et al. [2] introduced a one step adversarial attack framework. The attack image, x' is obtained by a simple additive disturb:

$$x' = x + \delta$$

Note that we only consider untargeted attack. We obtain the perturbation from:

$$\delta := \epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(f(x), y))$$

As it is a one-step algorithm, it is hard to get a high ASR(Attack Success Rate) but is very fast implementation. I investigated in various factors that affects ASR for FGSM exhaustively and reach high ASR (compared to the requirements for homework). Details will be talked in 2.

1.2. Projected Gradient Descend

PGD, Known as the basic iterative method, was initially proposed by Madry et al. [5]. It is based on the FGSM, but instead a single step of the projected gradient descend, it iterates through more steps, as in:

$$\delta := P(\delta + \alpha \text{sign}(\nabla_\delta \mathcal{L}(f(x+\delta), y)))$$

in which P denotes the projection over the ball of interest. With such formulation, the PGD requires more fine-tuning in choosing the step size α. Additionally, Madry et al. [5] proposed an iterative method with a random initialization for δ, which was also studied in my report. See section 2.

1.3. Momentum Iterative FGSM

MI-FGSM(Dong et al. [1]) is an advanced version of PGD using momentum for better transferability. I also explored it in white-box attack section for better understanding when using in black-box scenario. It also iterates through many steps, as in:

$$\delta := P(\delta + \alpha \cdot \text{sign}(g))$$

$$g := \mu g + \frac{\nabla_x \mathcal{L}(f(x), y)}{||\nabla_x \mathcal{L}(f(x), y)||_1}$$

Also, random initialization, step size, loss function etc. will also be studied in 2.

1.4. Experimental Setup

In MNIST attack, the perturbation limits in ϵ equals 0.3 and 0.031 for CIFAR10. The inner iteration for PGD and MI-FGSM is 10. We set the step size 0.1 for MNIST and

Dataset	Method	ASR%	ASR bar%
MNIST	FGSM	87.54	65.00
	PGD	100.00	98.00
	MI-FGSM	100.00	-
CIFAR10	FGSM	92.77	80.00
	PGD	100.00	100.00
	MI-FGSM	100.00	-

Table 1. Main Results. My Implementation of these methods achieved the requirements and some of them are greatly better than the base requirements.

0.03 for CIFAR10. The momentum decay μ is set 1.0 in MI-FGSM. We use NLL loss for all tasks. To random initialize δ, it is randomly sampled from $\mathcal{U}(-\epsilon, \epsilon)$.

1.5. Results

The relevant results are showed in table 1. It is clear that the inner iteration can help to find a better δ satisfying:

$$\delta \approx \arg \max_{||\delta||_\infty \leq \epsilon} \mathcal{L}(f(x), y)$$

which is our goal in white-box scenario. The algorithm upon one step, FGSM, may be too coarse while more steps methods, PGD and MI-FGSM, lead to better estimation.

Also we could see that MI-FGSM won't hurt the attack much to some extent. Since iterative methods have many hyperparameters to fine-tune, The results have more randomness. Generally, we don't need to try hard and easily get 100% ASR.

2. Ablation Study

2.1. Loss Function

In this section, we explore the relationship between different Loss function $\mathcal{L}$ and ASR.Usually, the conventional pipeline of training a classification model is as

$$x \rightarrow \text{model}(x) \rightarrow \text{logits} \rightarrow \text{CrossEntropyLoss}(\text{logits}, \text{y})$$

in which the CrossEntropyLoss combines a logsoftmax unit and a NLLLoss unit. The NLLLoss is designed to make our predicition similar to the training set distribution while softmax is a sort of regularization to some extent in order to enhace generalization. However, our goal in adversarial attack doen't need generalization, thus if we remove the softmax unit and only use NLLLoss, we can somehow boost the attacking ability. Empirically, softmax provides softer labels by sharing probability with other classes, without which we can get more accurate(sharp) gradient. I do some expriments to verify the hypothesis. See table 2.

Dataset	Loss	Noise	ASR%	ASR bar%
MNIST	CE	+	69.05	65.00
		-	46.75	65.00
	NLL	+	87.54	65.00
		-	80.09	65.00
CIFAR10	CE	+	91.06	80.00
		-	86.56	80.00
	NLL	+	92.77	80.00
		-	89.37	80.00

Table 2. The results of FGSM attack using different loss and random initialization(denoted by the 'Noise' column). Still, we follow the basic setup in section 1.4. We can see using NLLLoss and random initialization greatly improves the performance.

2.2. Step Size

In the baseline experiments we set the PGD step size α to 0.1 for MNIST and 0.03 for CIFAR10. In this section, I explore the relationship between the step size α nad ASR. Denote our vanilla α as $\hat{\alpha}$, I test $\alpha = 1/16\hat{\alpha}, 1/8\hat{\alpha}, 1/4\hat{\alpha}, 1/2\hat{\alpha}, \hat{\alpha}, 2\hat{\alpha}$ for 2 datasets. Other settings are the same as section 1.4.

It is obvious that if we set α very small, for example, $\alpha < \epsilon/T$, where ϵ is the perturbation limitation and T is the inner iteration number, then projection in PGD is of no use. On the contrary, if the α is very large, the projection will happen during every iteration. A proper α should be around ϵ/T, which is adopted in our baseline experiment and reached 100% ASR. The relationship between step size and ASR can be seen in Fig 1.

2.3. Random Initialization

The random initialization was first proposed by Madry et al. [5] as a trick to boost PGD attack performance. As shown in table 2, it can stably improve the ASR using FGSM. I also do some interesting experiments to see how the random initialization affects the performance. The vanilla experiments sampled the initial δ from $\mathcal{U}(-\epsilon, \epsilon)$, and I tried $\mathcal{N}(0, \epsilon)$. For CIFAR10 attack, using $\mathcal{N}(0, \epsilon)$ can reach 94.22% ASR, outperforming the 92.77% in $\mathcal{U}(-\epsilon, \epsilon)$. In addition, I tried different mean and variance only to find that $\mathcal{N}(0, \epsilon)$ is the best. I do not find very promising explanation. This could be set for future study.

3. Black-box Attacks

Under Black box restriction, models are different from the white box, with respect to the information the attacker has access to. In this section, I will focus on transfer-based black-box attack.

The fact that adversarial examples generated for a model can transfer to others, regardless of architecture, was first

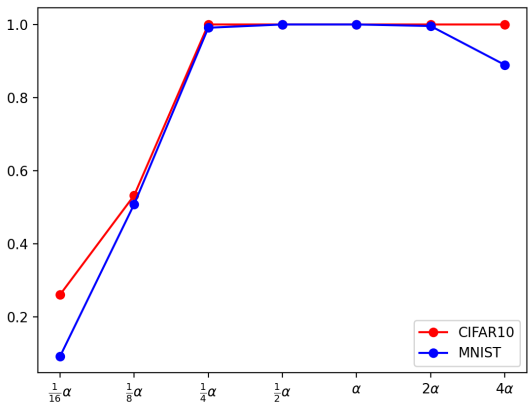

Figure 1. The relationship between ASR and stepsize. The y axis denotes the ASR and the x axis denotes the step size. We set α equals 0.1 in MNIST and 0.003 in CIFAR10 task.

observed by Szegedy et al. [6]. The black-box attack I implemented is momentum based one. Dong et al. [1] proposed to boost the iterative version of FGSM using gradient momentum, which was discussed in white box section. In this section, I will show the transferability of adversarial examples with the three methods described in white-box attack section.

3.1. Experimental Setup

For the part of generating adversarial examples, most of the setting is just the same as the white-box attack in section 1.4. A small exception is that the loss used this time is CrossEntropyLoss. More details about designing training algorithms will be discussed in section 4. Since I am implementing transfer-based attack, I choose two source models and two target models. The source models for MNIST and CIFAR10 are small CNN and preActResnet18 which were used in section 1.4, and the target model for MNIST is LeNet5 proposed by LeCun et al. [4] and Resnet18 proposed by He et al. [3] for CIFAR10. All of the models are pretrained on training datasets.

3.2. Results

Main results are shown in table 3. Using MI-FGSM, I meet the required ASR bar. I also find that adversarial examples generated by FGSM and PGD do have some transferability. However, MI-FGSM can generate more transferable examples. As in the case of gradient descent, momentum can stabilize the update directions and help escape poor local minimum/maxima by accumulating a velocity vector in the gradient direction. Setting the velocity vector to 0 is equivalent to the normal FGSM and PGD attack.

Dataset	Method	ASR%	ASR bar%
MNIST	FGSM	35.88	40.00
	PGD	48.43	40.00
	MI-FGSM	54.19	40.00
CIFAR10	FGSM	51.12	70.00
	PGD	71.33	70.00
	MI-FGSM	79.42	70.00

Table 3. Main Results for black-box attack. We use source models(smallCNN for MNIST, preActResnet18 for CIFAR10) to generate adversarial examples and attack target models(LeNet5 for MNIST, Resnet18 for CIFAR10). The loss used to calculate gradient is CrossEntropyLoss, and the perturbation δ is random initialized by $\mathcal{U}(-\epsilon, \epsilon)$, where ϵ equals 0.0031 for CIFAR10 and 0.03 for MNIST.

Dataset	Loss	Noise	ASR%	ASR bar%
MNIST	CE	+	54.19	40.00
		-	50.75	40.00
	NLL	+	49.54	40.00
		-	49.09	40.00
CIFAR10	CE	+	79.42	70.00
		-	77.61	70.00
	NLL	+	75.77	70.00
		-	75.32	70.00

Table 4. The results of MI-FGSM attack using different loss and random initialization(denoted by the 'Noise' column). We can see using CrossEntropyLoss and random initialization improve the performance slightly.

4. Ablation Study

4.1. Loss Function and Random Initialization

Remember that I explored the relationship between different loss function and ASR in white-box attack. We have found that NLLLoss can boost performance in FGSM attack because it leads to sharper gradient. However, from table 4 we can see CrossEntropyLoss is better in black-box scenario. From my perspective, MI-FGSM is an iteration method thus the one step mistake can be erased through many iterations, so using different loss or random initialization doesn't make great impact on the results. In spite of this, CrossEntropyLoss and random initialization do have better generalization ability, or trasferability in black-box attack. We could draw the conclusion from the experiment results too.

4.2. Model Architecture

In transfer-based attack, model architecture is also a key factor. We can have a very naïve idea that if the target model's architecture is similar to the source model, then

Dataset	Method	Target Model	
		small CNN	LeNet5
MNIST	FGSM	62.21	35.88
	PGD	95.31	48.43
	MI-FGSM	96.02	54.19
		preActResnet18	Resnet18
CIFAR	FGSM	57.57	51.12
	PGD	88.31	71.33
	MI-FGSM	90.45	79.42

Table 5. The results of Mi-FGSM attack towards different target models. I use small CNN and preActResnet18 as our source model. Then I pretrain another 4 target models.

the transfer-based attack could be more possible to succeed. I pretrain different target model and verify the hypothesis. See table 5. The reason why the ASR can be boosted may comes from trasferability itself. Since DNN shares some common feature in learning some representations for tasks, similar model architecture may share similar feature space. If we use the same model architecture for target model and source model, then we could get adversarial examples with pretty high transferability. When the model architecture is changed, the transferability drops apparently.

5. Conclusion

In my project, I implemented 3 classical attack algorithm, FGSM, PGD and MI-FGSM. By applying tricks like random initialization, proper loss function and step size, my methods achieve high performance in white-box attack. On top of that, I applied these three algorithms into black-box attack and investigated the transferability of adversarial examples. The factors like loss function and random initailzation are also discussed in black-box section. To make it comprehensive, the model architecture similarity in transfer-based attack is included in discussion, too.

References

[1] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

[2] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[5] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[6] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.