GAP++: Learning to generate target-conditioned adversarial examples

Xiaofeng Mao∗, Yuefeng Chen, Yuhong Li, Yuan He and Hui Xue
Alibaba Group
{mxf164419, yuefeng.chenyf, daniel.lyh, heyuan.hy, hui.xueh}@alibaba-inc.com

Abstract

Adversarial examples are perturbed inputs which can cause a serious threat for machine learning models. Finding these perturbations is such a hard task that we can only use the iterative methods to traverse. For computational efficiency, recent works use adversarial generative networks to model the distribution of both the universal or image-dependent perturbations directly. However, these methods generate perturbations only rely on input images. In this work, we propose a more general-purpose framework which infers target-conditioned perturbations dependent on both input image and target label. Different from previous single-target attack models, our model can conduct target-conditioned attacks by learning the relations of attack target and the semantics in image. Using extensive experiments on the datasets of MNIST and CIFAR10, we show that our method achieves superior performance with single target attack models and obtains high fooling rates with small perturbation norms.

1 Introduction

Deep Neural Networks (DNNs) have been extensively studied and seen in most daily applications. However, recent works have demonstrated that DNNs are vulnerable to adversarial perturbations on various tasks, e.g., image recognition [Szegedy et al., 2013; Goodfellow et al., 2014b], object detection [Thys et al., 2019] or semantic segmentation [Pour-saeeed et al., 2018]. These carefully crafted perturbations are imperceptible to humans but can mislead a well trained model after adding to the original inputs.

Many attack algorithms have been proposed for generating such adversarial perturbations. Among them, model-based methods use feed-forward networks to generate adversarial perturbations. Prior to optimization-based methods, they need not to iteratively search the optimal perturbations in the pixel space, and improves the attack efficiency greatly. Additionally, model-based methods can produce adversarial examples without requiring access to the victim network once the attacker network is trained.

However, model-based methods are not practical enough currently for two reasons: 1) model training is a time-consuming process, and particularly, we need to retrain the model when the type of the attack (targeted or nontargeted) or the attack targets are changed. 2) adversarial examples generated by model-based methods have a poor transferability.

In this work, we aim at the first drawback mentioned above and propose an improved framework to generate target-conditioned perturbations. Our method is motivated by [Mao et al., 2019], which uses conditional generative adversarial networks (cGAN) to generate the output conditioned on both given image and text. We think a good attacker should learn the relations of attack target and the semantics in image. Thus, it can generate not just the target-specific perturbations dependent on input images, but the image-specific perturbations conditioned on attack targets. For the latter, we named target-conditioned perturbations. Different from previous methods which train one model for single target attack, our approach can generate all types of targeted perturbations using only one trained model. Furthermore, by treating the nontarget attack as a peculiar target attack, we use trained model for nontarget attack as well. Although the model deals with more attack tasks, the comparative experiment shows it achieves high fooling rates and superior performance with single target attack models.

Our contributions can be summarized as follows:

• We present an improved model-based attack method which can generate target-conditioned perturbations under $L_0$, $L_2$ and $L_{\infty}$ norm. It greatly reduces the training cost and facilitates the attack process.

• Our method can employ a trained targeted attack model for nontarget attack. It also achieves comparatively high fooling rates.

• The extensive experiments on MNIST and CIFAR10 illustrate that our method gains superior performance with single target attack models.

2 Related Work

We introduce the recent work on adversarial examples and model-based attack in this section.
Adversarial Examples Various techniques for crafting adversarial examples have been proposed recently. Traditional methods of adversarial attack can be divided into two classes: gradient-based approaches and optimization-based approaches. In gradient-based approaches such as the classic Fast Gradient Sign Method (FGSM) [Goodfellow et al., 2014b] and their targeted variants [Kurakin et al., 2016b], adversarial examples are generated by superimposing small perturbations along the gradient direction (with respect to the input image). FGSM uses the linear approximation of the gradient in deep networks whereas such models are commonly thought to be highly non-linear. It leads to suboptimal results. To settle this problem, [Kurakin et al., 2016a; Madry et al., 2017] iteratively take multiple small steps while adjusting the gradient direction after each step and get better results. In optimization based approaches, adversarial perturbation is optimized for targeted attacks while satisfying certain constraints [Carlini and Wagner, 2017; Liu et al., 2016]. However, the optimization process is slow and can only optimize perturbation for one specific instance each time. Therefore, more and more works use feed-forward network to approximate optimal perturbations.

Model-based attack methods Model-based attack methods use image generation [Goodfellow et al., 2014a] or image-to-image translation [Isola et al., 2017] techniques to create adversarial examples. This idea is first proposed by [Baluja and Fischer, 2017]. The created adversarial examples can be universal [Hayes and Danezis, 2018; Poursaeed et al., 2018] or image-dependent [Xiao et al., 2018]. Model-based attack methods ensure the imperceptibility of perturbations in different ways. [Baluja and Fischer, 2017] and [Xiao et al., 2018] minimize the $L_2$ distance between the original input and adversarial examples. [Li et al., 2019; Poursaeed et al., 2018] embed normalization module in network to constrain the output directly. Another branch of the methods does not guarantee the imperceptibility of perturbations. They create adversarial examples closed in latent space instead of pixel space [Zhao et al., 2017], or train a class-conditional GAN to generate adversarial examples in the same class [Tsai, 2018; Song et al., 2018]. Thus, more unrestricted adversarial examples can be constructed in this case.

3 Proposed method
Suppose $\mathcal{X} \subseteq \mathbb{R}^n$ be the pixel space, $\mathcal{Y} = \{1, 2, ..., C\}$ be the label space. Given original input $x \in \mathcal{X}$, a classifier $f(x): x \in \mathcal{X} \rightarrow y \in \mathcal{Y}$ and a target label $y^* \in \mathcal{Y} \{f(x)\}$, the problem of generating targeted adversarial examples can be expressed as the optimization: $\text{arg min}_{x} L(x, x^*)$ s.t. $f(x^*) = y^*$, where $L(\cdot)$ is a distance metric between examples from the input space (e.g., the $L_2$ norm). Similarly, untargeted attack can also be defined as $\text{arg min}_{x} L(x, x^*)$ s.t. $f(x^*) \in \mathcal{Y} \{f(x)\}$. We will go through the proposed scheme of both targeted and untargeted attacks in the rest of this section.

3.1 Targeted attacks
We firstly overview the general workflow of model-based adversarial attacks. The overall idea of model-based attacks is to train a neural network $g$ that transforms an input $x$ into an adversarial example $x^*$, which fools the classifier $f$ to output target label $y^*$. Formally, suppose that the neural network parameterized by $\theta$ is $g(\cdot)$, we have the following objective:

$$L_{\text{targeted}} = \sum_{x \in \mathcal{X}} \mathcal{H}(f(g_0(x)), y^*) + L(x, g_0(x))$$

where $\mathcal{H}$ is the cross-entropy loss.

There are many improvements based on this framework. For example, GAP [Poursaeed et al., 2018] trains $g_0$ to approximate the difference of input and adversarial images for better controlling over the perturbation magnitude. In other words, it generates adversarial examples $x^*$ by adding $\Delta x = g_0(x)$ to input $x$. In equation 1, $L(\cdot)$ term makes the perturbation imperceptible. More advanced metrics such as soft hinge loss [Xiao et al., 2018], perceptual similarity loss [Reddy Mopuri et al., 2018] had been proposed. But optimizing $L(\cdot)$ cannot guarantee that the model outputs adversarial examples which conform to the norm condition in the test phase. Therefore, another improvement is to normalize the model output instead of adding an objective term in the training stage. Our work adopts these modification in the framework for getting preferable results.

Figure 1 illustrates the overall architecture of the proposed model, which mainly consists of two parts: an attacker network $g$ and a victim network $f$. The attacker network takes the target label $y^*$ and original instance $x$ as its input. To add $y^*$ as conditional information, the attacker network can produce a distribution of perturbations, i.e. $P(\Delta x | x, y^*)$, which is conditioned on target labels. Then the output of the attacker is scaled to constrain its fixed norm. Specifically, if we have $\Delta x$, the normalized perturbations can be obtained by multiplying $\Delta x$ with $\min \left(1, \frac{\epsilon}{\|\Delta x\|_p} \right)$, where $\epsilon$ is the maximum permissible $L_p$ norm. Three types of normalization term are used in our model. $L_\infty$ and $L_2$ have been proposed in [Poursaeed et al., 2018]. Besides, we put forward the new $L_0$ norm for model-based attack. In our method, $L_0$ norm is implemented by simply selecting top-$k$ pixel values in $\Delta x$, where $k$ is given by users, similar to $\epsilon$. Next we add the normalized perturbations $\Delta x_{norm}$ and original input $x$ together, i.e. $x^* = \Delta x_{norm} + x$. In last stage, $x^*$ is clipped for ensuring it in the valid range of images and final result is the generated adversarial example.

Overall, the objective function of targeted attack is:

$$L_{\text{targeted}} = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y} \{f(x)\}} \mathcal{H}(f(g_0(x), y^*)), y^*)$$

normalizing and clipping operations are omitted here for simplicity.

3.2 Untargeted attacks
In this section, we show how to employ our method to conduct untargeted attacks. Traditional methods using fooling loss to optimize attack algorithms. Fooling loss aims to maximize the cross-entropy between fooled prediction and original prediction:

$$L_{\text{untargeted}} = -\sum_{x \in \mathcal{X}} \mathcal{H}(f(g_0(x)), f(x))$$
This maximize objective is incompatible with $\mathcal{L}_{\text{targeted}}$, which has a minimize objective. They cannot be optimized together. Alternatively, as proposed in [Kurakin et al., 2016b] and [Kurakin et al., 2016a], we consider least-likely class $f_{ll}(x) = \arg \min f(x)$ as the target to train the model. In our architecture, every input image needs corresponding target label as condition information. However, no conditional target label exists in untargeted case. For this problem, as shown in Figure 1, we simply use zero vector for untargeted training. Because it does not affect the learning of internal representations by concatenating zero tensors in the model. The untargeted loss can be formulated as:

$$\mathcal{L}_{\text{untargeted}} = \sum_{x \in X} \mathcal{H}(g_\theta(x), \tilde{y}); f_{ll}(x)) \tag{4}$$

By viewing least-likely untargeted attack as a special targeted attack. We adopt $\mathcal{L}_{\text{untargeted}}$ as the one training objective in the model. Now, we have the overall training objective as follows:

$$\mathcal{L} = \mathcal{L}_{\text{targeted}} + \alpha \mathcal{L}_{\text{untargeted}} \tag{5}$$

where $\alpha$ is a balance coefficient. We set $\alpha$ to 1 as default in the rest of the paper.

Conventional model-based method needs to train models for attacks of different types or different targets. It leads to a great cost of time and computing resources. However, by using our approach, we can generate all types of adversarial examples just by training one model. In this aspect, our method is a lighter version which can be used for many attack tasks. Additionally, although the model deals with more attack tasks, our experiment shows it still has good attack performance.

4 Experiments

In this section, we first evaluate our model for both targeted and non-targeted white-box attack settings on the benchmarks of MNIST and CIFAR10. For each dataset, we simply use the training set to train the generative model and the validation set for evaluation.

We will provide details of the network architecture and the training procedure. The generation of perturbations has many shared properties with the image translation task, so our generative model mainly follows the architecture design of [Mao et al., 2019]. Target labels as the conditioning information, are concatenated with the output of residual blocks before the deconvolution layers. The training time is partly determined by the class number of the recognition task. It takes half an hour to converge that training attack models on MNIST and CIFAR10 task. The algorithm is running on two Tesla P100 GPUs and optimized by Adam with the learning rate of 0.0002.

4.1 Experiment on MNIST and CIFAR10

We first train three networks on the standard MNIST and CIFAR10 classification tasks. For MNIST, we use 3-full-connected-layer MLP network, ResNet18 and LeNet as the basic classifier. For more complicated CIFAR10 object recognition, three stronger model: SENet18, ResNet18 and VGG11 are used instead. Both the MNIST and CIFAR10 images are rescaled to 32×32 size with 3 RGB channels for fitting the input size of the basic networks. To restrict the generated adversarial examples, we normalize the network output on different metrics. $L_0, L_2$ and $L_{inf}$ are used in the experiment (detailed analysis of different norm type can be seen in last paragraph of this section).

We compare the performance of our algorithm with FGSM and GAP. FGSM is the most representative gradient-based method which query the victim model at once and get the gradient of the loss function with respect to pixels, and moves a single step based on the sign of the gradient. GAP is a model-based method which use generative model to produce universal or image-dependent perturbations. Both of compared methods are fast and efficient. In order to ensure the fairness, we do not contrast with iterative optimized methods, such as Carlini and Wagner’s methods (CW). They attack slowly by querying many times to the target model, which are not practical in real-world scenario.

Performance of non-target attack. We show the success rate of non-target attack in Table 1. All the methods are evalu-
Table 1: Comparison of the non-targeted attack performance. We show the AV% metric (Accuracy on the Validation set) and AA% metric (Accuracy on the Adversary set) on three different attack methods: FGSM, GAP, and GAP++.

| Datasets | Attack Type | Models | AV% | AA%<sub>=0.05</sub> | AA%<sub>=0.15</sub> | AA%<sub>=0.2</sub> |
|----------|-------------|--------|-----|---------------------|---------------------|---------------------|
| CIFAR10  | FGSM        | ResNet18 | 99.4% | 62.5% | 55.4% | 30.9% | 29.3% |
| MNIST    | GAP         | ResNet18 | 99.4% | 78.5% | 62.1% | 43.9% | 30.0% |
|          | LeNet       |         | 99.2% | 94.5% | 85.1% | 68.3% | 43.1% |
|          | MLP         |         | 97.5% | 72.4% | 44.7% | 36.9% | 28.0% |
|          | GAP++       | ResNet18 | 99.4% | 76.8% | 58.3% | 40.5% | 30.7% |
|          |             | LeNet   | 99.2% | 85.2% | 68.2% | 57.5% | 46.1% |

Table 2: Comparison of the targeted attack performance. We pick 5 different target labels to conduct the targeted attack on MNIST dataset using $\epsilon=0.2$ as the $L_\infty$ norm. GAP cannot generate various targeted perturbations, so we train the model for each target respectively. AA% is calculated by averaging the accuracy on total 5 targeted adversaries.

| Attack Type | AA%<sub>=0.05</sub> | Target-1 | Target-2 | Target-3 | Target-4 | Target-5 |
|-------------|---------------------|----------|----------|----------|----------|----------|
| FGSM        | 93.4%               | 11.4%    | 10.3%    | 11.1%    | 11.0%    | 11.4%    |
| GAP-1       | 75.9%               | 12.5%    | 1.2%     | 1.2%     | 1.2%     | 0.9%     |
| GAP-2       | 75.3%               | 1.9%     | 12.5%    | 1.4%     | 1.4%     | 1.4%     |
| GAP-3       | 76.3%               | 1.2%     | 1.7%     | 12.3%    | 1.7%     | 1.7%     |
| GAP-4       | 76.2%               | 1.0%     | 1.8%     | 16.5%    | 2.0%     | 2.0%     |
| GAP-5       | 77.7%               | 1.3%     | 1.1%     | 2.0%     | 2.3%     | 11.5%    |
| GAP++       | 50.5%               | 13.3%    | 12.8%    | 12.8%    | 12.9%    | 11.6%    |

Performance of target attack. We make a comparative analysis of our method with FGSM and GAP. Table 2 shows the experimental results, we use the accuracy of target class prediction as attack success rate. In the experiment, we choose 5 classes and calculate the prediction accuracy of each class. FGSM can generate arbitrary targeted attacks by calculating the gradient that reduces the loss of the target class, but it is not very accurate. Contrary to FGSM, GAP attacks target model with higher accuracy but it is unable to generate arbitrary targeted perturbations. So, for each target, we train a GAP model respectively (the model is tagged as GAP-1 if target of the attack is target-1). The result suggests each GAP model only focuses only on generating single target attacks. It signifies we must train 5 models to generate 5 types of target attacks. The proposed GAP++ overcomes this disadvantage of traditional model-based attack. By modeling the distribution of multi-targeted perturbations, we deal with all types of target attacks using one GAP++ model. Additionally, from the table 2 we found even GAP++ generates more various perturbations, it gains superior performance with single targeted attack models. In terms of fooling rate, GAP++ achieves higher attack success rate than GAP because it modifies more samples to cheat the model.

Norm constraint analysis. We analyze three normalization term named $L_\infty$, $L_2$ and $L_0$ in GAP++. $L_\infty$ is a norm term which limits the maximum change of single pixel in adversarial samples. While $L_2$ constrains the sum of total pixel variations. $L_0$ introduces the sparsity to variations and aims to cheat the model by altering as few pixels as possible. We establish relation among these three norm terms by defining total variation metric, which represents the maximum variation of all pixels in the image for each norm term. As shown in Figure 2, $L_\infty$ achieves the best effect on both MNIST and CIFAR10 dataset, while $L_2$ and $L_0$ suffer a great performance loss. The result indicates $L_2$ and $L_0$ are still hard to applied on model-based adversarial attacks. By using $L_\infty$ norm, we found it is easy for model to learn the pattern or structure of perturbations instead of predicting the perturbation values...
directly. Although $L_2$ and $L_0$ normalize the output as well, however, the model tries hard to pick the right perturbation values from a wide range (0-128). Therefore, GAP++ trained on $L_2$ and $L_0$ norm can be more prone to fall into local optimum and output the sub-optimal results. We visualize the reconstructed adversary and its corresponding perturbations in Figure 3. $L_\infty$ and $L_2$ norm makes the model to generate smooth perturbations. $L_0$ norm lets the model output salt-and-pepper perturbations, which can be clearly distinguished. From the perspective of perception, we can draw a conclusion that $L_\infty$ is a preferable normalization method.

![Figure 3: Visualization of targeted adversarial examples when total variation is 160 (corresponding $L_\infty = 20$, $L_2 = 25$ and $L_0 = 160$). For each image grid, rows represent different targets and columns represent different original inputs.](image)

5 Conclusion

In this paper, we propose a novel model-based adversarial attack method named GAP++, which can be used to generate target-conditioned perturbations. GAP++ borrows the network architecture and normalization trick from original GAP, but achieves superior performance and lighter than GAP. Using extensive experiments on the datasets of MNIST and CIFAR10, we show that our method achieves high fooling rates with small perturbation norms. However, Our approach still behaves weakly in transferability, which is a problem to be solved in the future work.

References

[Baluja and Fischer, 2017] Shumeet Baluja and Ian Fischer. Adversarial transformation networks: Learning to generate adversarial examples. arXiv preprint arXiv:1703.09387, 2017.

[Carlini and Wagner, 2017] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In 2017 IEEE Symposium on Security and Privacy (SP), pages 39–57. IEEE, 2017.

[Goodfellow et al., 2014a] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014.

[Goodfellow et al., 2014b] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In ICLR, 2014.

[Hayes and Danezis, 2018] Jamie Hayes and George Danezis. Learning universal adversarial perturbations with generative models. In 2018 IEEE Security and Privacy Workshops (SPW), pages 43–49. IEEE, 2018.

[Isola et al., 2017] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1125–1134, 2017.

[Kurakin et al., 2016a] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533, 2016.

[Kurakin et al., 2016b] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. arXiv preprint arXiv:1611.01236, 2016.

[Li et al., 2019] Yingwei Li, Song Bai, Cihang Xie, Zhenyu Liao, Xiaohui Shen, and Alan L Yuille. Regional homogeneity: Towards learning transferable universal adversarial perturbations against defenses. arXiv preprint arXiv:1904.00979, 2019.

[Liu et al., 2016] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In ICLR, 2016.

[Madry et al., 2017] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In ICLR, 2017.

[Mao et al., 2019] X. Mao, Y. Chen, Y. Li, T. Xiong, Y. He, and H. Xue. Bilinear representation for language-based image editing using conditional generative adversarial networks. In ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 2047–2051, May 2019.

[Poursaeed et al., 2018] Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. Generative adversarial perturbations. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4422–4431, 2018.

[Reddy Mopuri et al., 2018] Konda Reddy Mopuri, Utkarsh Ojha, Utsav Garg, and R Venkatesh Babu. Nag: Network for adversary generation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 742–751, 2018.

[Song et al., 2018] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing unrestricted adversarial examples with generative models. In Advances in Neural Information Processing Systems, pages 8312–8323, 2018.
[Szegedy et al., 2013] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In ICLR, 2013.

[Thys et al., 2019] Simen Thys, Wiebe Van Ranst, and Toon Goedemé. Fooling automated surveillance cameras: adversarial patches to attack person detection. arXiv preprint arXiv:1904.08653, 2019.

[Tsai, 2018] Shih-hong Tsai. Customizing an adversarial example generator with class-conditional gans. arXiv preprint arXiv:1806.10496, 2018.

[Xiao et al., 2018] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. In Proceedings of the 27th International Joint Conference on Artificial Intelligence, pages 3905–3911. AAAI Press, 2018.

[Zhao et al., 2017] Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. arXiv preprint arXiv:1710.11342, 2017.