LEVERAGE OPEN-SOURCE INFORMATION TO MAKE AN EFFECTIVE ADVERSARIAL ATTACK AGAINST DEEP LEARNING MODEL

Li Chen, Xiaojin Jiao, Wei Li, Yang Li, Chuan Yang

Northwest Security

Abstract

In the 2018 CAAD, our team, Northwest Security (NWSec), won the 3rd place in the targeted attack competition and 5th place in the non-targeted attack competition by leveraging open-source information to make effective attacks against deep learning model. This team is also the 2nd place winner of CTF2018 in Las Vegas with the same methodology. In this report, the engineering strategies used by NWSec and technical details of attack methods have been elaborated.

Introduction

Discovery of adversarial examples unveiled the vulnerability of neural-network based image classifier against adversarial attacks[1,2]. Since then, tremendous efforts have been made to explore this vulnerability and to improve the robustness of neural network[3-11]. On the other hand, competition has been proved to be one effective way to boost the learning on security-related topics. Since 2017 NIPS hosted the first online competition on adversarial attacks[12], more and more experts from both academia and industry have started to utilize competitions as another effective approach to tackle this adversarial issue as well as demonstrate their novel findings and results. In 2018 GeekPwn hosted the online competition on adversarial attacks and defenses (CAAD), and CAAD Capture-the_Flag (CTF) which is the first ever adversarial CTF at DefCon 27.

In the 2018 CAAD, team NWSec won the 3rd place in the targeted attack and 5th place in the non-targeted attack for the online competitions. NWSec is also the 2nd place winner of CTF2018 in Las Vegas, who accommodated the same method and concept as one of the core method for the CTF attack. This paper will summarize engineering strategies and technical details of attack methods used in CAAD.

Info evaluation and dataset

The main strategy of NWSec was to build effective adversarial attacks and defenses through engineering approach by leveraging the pulic available info including technical documents and open-source codes.

To make the info collection and evaluation more efficient, NWSec first determined three key factors of powerful models according to the CTF/CAAD rule: speed, strength and transferability. It should be noted that CTF and CAAD rules are different, which will change the priority of these factors. This paper will mainly focus on CAAD. The strength of attack methods is benchmarked by classification accuracy in non-targeted attack and the hit-target rate in targeted attack. Hit-target rate is the percentage of images which has been misclassified as target classes. Good transferability means the attack has consistent superior performance against various defend methods. Out of three factors, transferability is the most important, which will be discussed in detail in following section.

In addition, building a baseline benchmark is also crucial for model evaluation, since it will provide the reference for model comparison. NWSec utilized two open-source libraries for the benchmark task. For attack, CleverHans[13] a Python library of a collection of adversarial attacks to benchmark vulnerability of machine learning classifier against adversarial examples has been used. For defense, adversarial pre-trained models[14] including two popular NN architectures: inception v3 and Inception Reset V2 provided by Google has been used. The adversarial training means the model has been trained with both original and adversarial images. Since they are public available resources, we assume their performance should represent the lower boundary of the more advanced models used in CAAD.

The test dataset provided by CAAD committee includes 1000 fresh images, which have been classified into 1000 imagenet labels. The coverage of the 1000 classes has been determined to be representative for attack model evaluation.

Method and Result

NWSec attack strategy can be briefed as using white-box attack plus good transferability to achieve black-box attack. Therefore, good transferability is critical for effective attack.

	Inception V3	Adv_Inception V3	Ens_adv_Ires V2
BIM_inception	88.50%	0	0
BIM_adv_inception	0	75.70%	0
BIM_ens_adv_Ires	0	0	85.30%
BIM_ens3	93.70%	76.80%	81.50%

Table 1 Attack models evaluation against the open source defend models.

Table 1 summarizes the hit-target rate of four attacks against three defenses. All four attacks implement the same gradient-based Basic Iterative Method (BIM)[3], but with different methods for gradient calculation. Actually the first three attacks use the gradient of the models corresponding to those three defenses, including base inception V3, adversarially trained inception V3 and ensemble adversarially trained inception-resnet v2. Basically, each defense has its corresponding white-box attack counterpart to break it effectively. BIM based on inception v3 can have 88% hit target rate against Inception V3. For other two defenses, even if they are adversarially trained, they are still vulnerable to the white-box attacks: attacks can still have hit-target rate around 80%. But the white-box attack is only effective against the corresponding defense model, which means poor transferability. Fortunately, researchers have demonstrated that good transferability can be achieved by multi-model ensembling[3]. The fourth attack in the table demonstrates that transferability can be indeed improved by ensembling three models into one attack method. Simply speaking, three adversarial images have been calculated based on these three models, then the final image is the average of those three. With this attack, the hit-target rate is consistently high against all three defenses.

After extensive evaluation of public available methods, NWSec adopted non-targeted and targeted attack methods from Team toshi_k[15], which was the 5th place winner of 2017 NIPS competition. On top of that, NWSec optimized the code by model selection and hyperparameter tuning. In addition, the key idea proposed by team Sanxia[16] has also been fused into the adversarial attack to further boost the performance.

Both non-targeted and targeted attack methods were based on BIM with multiple models ensembling[3]. The only difference is the definition of objective function for gradient calculation. Non-targeted attack tries to push the perturbed label further away from the true label, so the objective function is defined as distance to the original label, and the goal of optimization is to increase the distance. In contrast, targeted attack tries to pull the perturbed label closer to the target label, so the objective function is defined as distance to the target label, and the goal of optimization is to reduce the distance. Due to the fact that the two methods are fundamentally the same, the tricks used to boost the attack performance can be shared between both of them.

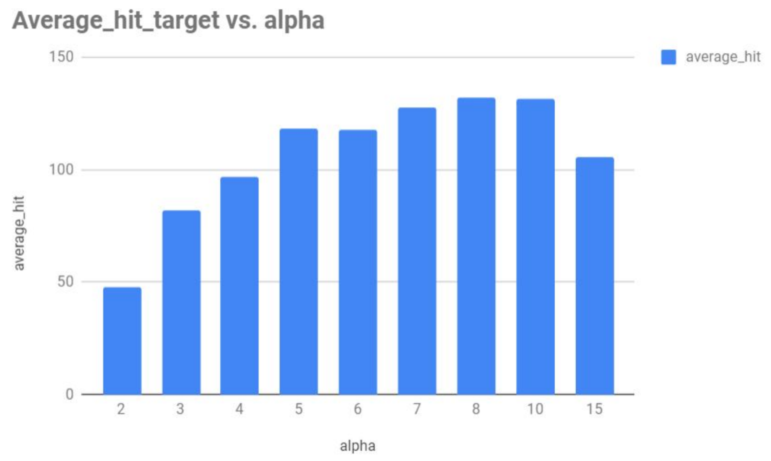
Adversarially trained defense models usually have unsmooth gradients, which means there are many local minima acting like gradient traps. Both Toshi_K and Sangxia methods put additional efforts to get away from local minima during the calculation. As shown in the pseudocode below, Toshi_k method applied a 2D Gaussian smoothing over gradient in each iteration, which can effectively remove the local minima. On the other hand, sangxia method added a random perturbation to the calculated adversarial image, which increases the chance of calculation “jumping” out of the local gradient traps.

Furthermore, Toshi_k method ensembled three models (inception_v3, adv_inception_v3 and ens_adv_inception_resnet_v2) as the targeted defenses in order to improve transferability. The perturbation was calculated based on the average of ensembled model gradients. The adversarially trained model was included to further improve the performance against adversarially trained defenses. Overall object for the picture perturbation is to either increase the loss of correct classification for non-targeted attack or decrease loss of the targeted (wrong) classification for targeted attack.

```
x_adv = original_image
For each iteration:
    loss = calculate the loss through loss function
    gradient = calculate the gradient of loss w.r.t. x_adv

    # 2d Gaussian smoothing
    gradient = 2D_Gaussian_smoothing(gradient)

    # calculate adversarial image x_adv
    x_adv = x_adv - alpha * sign(gradient)

    # random perturbation
    x_adv = x_adv + random_number
```

After confirming the methods, NWSec carried out design of experiment for extensive hyperparameter optimization. Impact of step size was first studied (alpha in pseudocode). The step size controls how aggressive the perturbation is calculated. The step size was increased for the purpose of acceleration the picture change to the targeted classification, which means for each iteration it has more chance to get closer to the target. In the meantime, the step can not be too large to cause oscillation around the targerd class. The optimal step size was identified upon the 1000 given pictures and public available defender including NWSec developed state-of-art defenders. It should be noted that the optimal step size could be different based on the pictures and the defender selection, which is also feature of adversarial: no best offender or defender without the context of opponent. Based on our experiment on the 1000 pictures, if the step size is too small, output picture remain almost unchanged and only micro-patterns start to show. While if the step size is too large, perturbation is mainly clipped and output pictures present rainbow color stripes. In both case, the attack performance is poor.

The following is the discussion of the hyperparameter selection. Since our attack got high scores among all the public available adversarial defenders, we introduced most powerful defender, guided denoise[17] in the 2017 NIPS adversarial competition as another classifier to get scores of hyperparameter-tuned attack models seperated a little bit more to justify the hyperparameter selection. We identified the optimal step size to be 8 times of the original Toshi-K step size based on the high average hit target rate based as shown in the bar plot as well as the best coverage towards different defenders as shown in the radar plot in Fig. 1. In the radar plot, the defenders we tried to attack include some guided denoise based methods with various hyperparameters selection and they can be quite effective to defend the various attacks. Among the radar plot the step size of 8 times alpha encloses the rest which determined with the best coverage.



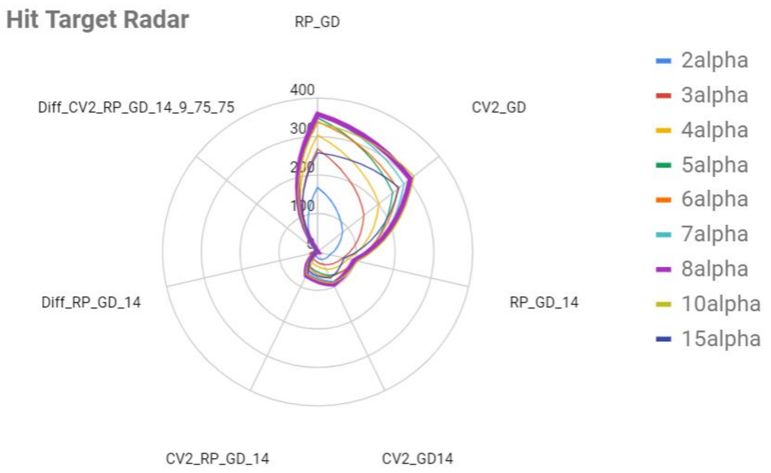


Fig 1 Radar plot for various step sizes upon various adversarial defenders and bar plot for average hit target on 1000 images for various step sizes.

NWSec also carried out the experiment on the parameter tuning for gradient smooth and the iterations. Taken into account the attack-speed required by CAAD, NWSec determined the iteration numbers to make most use of the time given while leaving some margin for the cloud instance speed variation which we also experienced in the CTF contest. Iteration number of 15 has been determined as the optimal value for targeted and non-targeted attacks.

All experiments have been run on local PC with GeForce GTX 10 series (GTX 1060/1070/1080) GPU to evaluate the time of Tesla P100 on the Google Cloud as well as from the CAAD test run feedback .

Conclusion

To sum it up, NWSec evaluated and selected effective attack method from public resources of the state-of-art adversarial attack/defend codes. Using them as starting point, NWSec creatively injected boosting methods and designs through engineering approaches. Furthermore, NWSec carried out design of experiment for hyperparameter fine tuning as well as meeting the contest speed requirement with some margin to achieve the competitive result for this black-box attack contest .

Reference

[1] Szegedy, Christian, et al. "Intriguing properties of neural networks." *arXiv preprint arXiv:1312.6199* (2013).
[2] Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples.” arXiv preprint arXiv:1412.6572 (2015).
[3] Tramèr, Florian, et al. "Ensemble adversarial training: Attacks and defenses." *arXiv preprint arXiv:1705.07204* (2017)
[4] Evtimov, Ivan, et al. "Robust physical-world attacks on machine learning models." *arXiv preprint arXiv:1707.08945*(2017).
[5] Kurakin, Alexey, Ian Goodfellow, and Samy Bengio. "Adversarial examples in the physical world." *arXiv preprint arXiv:1607.02533* (2016).
[6] Papernot, Nicolas, Patrick McDaniel, and Ian Goodfellow. "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples." *arXiv preprint arXiv:1605.07277* (2016).
[7] Athalye, Anish, and Ilya Sutskever. "Synthesizing robust adversarial examples." *arXiv preprint arXiv:1707.07397*(2017).
[8] Carlini, Nicholas, and David Wagner. "Towards evaluating the robustness of neural networks." *arXiv preprint arXiv:1608.04644* (2016).
[9] Madry, Aleksander, et al. "Towards deep learning models resistant to adversarial attacks." *arXiv preprint arXiv:1706.06083* (2017).
[10] Athalye, Anish, Nicholas Carlini, and David Wagner. "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples." *arXiv preprint arXiv:1802.00420* (2018).
[11] Akhtar, Naveed, and Ajmal Mian. "Threat of adversarial attacks on deep learning in computer vision: A survey." *arXiv preprint arXiv:1801.00553* (2018).
[12] Kurakin, Alexey, et al. "Adversarial attacks and defences competition." *arXiv preprint arXiv:1804.00097* (2018).
[13] https://github.com/tensorflow/cleverhans
[14] https://github.com/tensorflow/models/tree/master/research/adv_imagenet_models
[15] https://github.com/toshi-k/kaggle-nips-2017-adversarial-attack
[16] https://github.com/sangxia/nips-2017-adversarial
[17] Liao, Fangzhou, et al. "Defense against adversarial attacks using high-level representation guided denoiser." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.