In this supplementary document, we provide the following details to support the main text:

Section A: descriptions of the 13 defense methods studied in the experiments,

Section B: architecture of the regression neural network for initializing our $\mathcal{N}$ATTACK algorithm, and

Section C: run-time analysis about $\mathcal{N}$ATTACK and BPDA (Athalye et al., 2018).

A. More Details of the 13 Defense Methods

- **Thermometer encoding (THERM).** To break the hypothesized linearity behavior of DNNs (Goodfellow et al., 2014a), Buckman et al. (2018) proposed to transform the input by non-differentiable and non-linear thermometer encoding, followed by a slight change to the input layer of conventional DNNs.

- **ADV-TRAIN & THERM-ADV.** Madry et al. (2018) proposed a defense using adversarial training (ADV-TRAIN). Specially, the training procedure alternates between seeking an “optimal” adversarial example for each input by projected gradient descent (PGD) and minimizing the classification loss under the PGD attack. Furthermore, Athalye et al. (2018) find that the adversarial robust training (Madry et al., 2018) can significantly improve the defense strength of THERM (THERM-ADV). Compared with ADV-TRAIN, the adversarial examples are produced by the logit-space projected gradient ascent in the training.

- **Cascade adversarial training (CAS-ADV).** Na et al. (2018) reduced the computation cost of the adversarial training (Goodfellow et al., 2014b; Kurakin et al., 2016) in a cascade manner. A model is trained from the clean data and one-step adversarial examples first. The second model is trained from the original data, one-step adversarial examples, as well as iterative adversarial examples generated against the first model. Additionally, a regularization is introduced to the unified embeddings of the clean and adversarial examples.

- **Adversarially trained Bayesian neural network (ADV-BNN).** Liu et al. (2019) proposed to model the randomness added to DNNs in a Bayesian framework in order to defend against adversarial attack. Besides, they incorporated the adversarial training, which has been shown effective in the previous works, into the framework.

- **Adversarial training with adversarial examples generated from GAN (ADV-GAN).** Wang & Yu (2019) proposed to model the adversarial perturbation with a generative network, and they learned it jointly with the defensive DNN as a discriminator.

- **Stochastic activation pruning (SAP).** Dhillon et al. (2018) randomly dropped some neurons of each layer with the probabilities in proportion to their absolute values.

- **RANDOMIZATION.** (Xie et al., 2018) added a randomization layer between inputs and a DNN classifier. This layer consists of resizing an image to a random resolution, zero-padding, and randomly selecting one from many resulting images as the actual input to the classifier.

- **Input transformation (INPUT-TRANS).** By a similar idea as above, Guo et al. (2018) explored several combinations of input transformations coupled with adversarial training, such as image cropping and rescaling, bit-depth reduction, JPEG compression.

- **PIXEL DEFLECTION.** Prakash et al. (2018) randomly sample a pixel from an image and then replace it with another pixel randomly sampled from the former’s neighborhood. Discrete wavelet transform is also employed to filter out adversarial perturbations to the input.
Table 1. Average run time to find an adversarial example (N\textsc{Attack}-R stands for N\textsc{Attack} initialized with the regression net).

<table>
<thead>
<tr>
<th>Defense</th>
<th>Dataset</th>
<th>BPDA (Athalye et al., 2018)</th>
<th>N\textsc{Attack}</th>
<th>N\textsc{Attack}-R</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP (Dhillon et al., 2018)</td>
<td>CIFAR-10 ($L_{\infty}$)</td>
<td>33.3s</td>
<td>29.4s</td>
<td>–</td>
</tr>
<tr>
<td>RANDOMIZATION (Xie et al., 2018)</td>
<td>ImageNet ($L_{\infty}$)</td>
<td>3.51s</td>
<td>70.77s</td>
<td>48.22s</td>
</tr>
</tbody>
</table>

- **Guided denoiser.** Liao et al. (2018) use a denoising network architecture to estimate the additive adversarial perturbation to an input.

- **Random self-ensemble (RSE).** Liu et al. (2018) combine the ideas of randomness and ensemble using the same underlying neural network. Given an input, it generates an ensemble of predictions by adding distinct noises to the network multiple times.

B. Architecture of the Regression Network

We construct our regression neural network by using the fully convolutional network (FCN) architecture (Shelhamer et al., 2016). In particular, we adapt the FCN model pre-trained on PASCAL VOC segmentation challenge (Everingham et al., 2010) to our work by changing its last two layers, such that the network outputs an adversarial perturbation of the size $32 \times 32 \times 3$. We train this network by a mean square loss.

C. Run Time Comparison

Compared with the white-box attack approach BPDA (Athalye et al., 2018), N\textsc{Attack} may take longer time since BPDA can find the local optimal solution quickly being guided by the approximate gradients. However, N\textsc{Attack} can be executed in parallel in each episode. We leave implement the parallel version of our algorithm to the future work and compare its sing-thread version with BPDA below.

We attack 100 samples on one machine with four TITAN-XP graphic cards and calculate the average run time for reaching an adversarial example. As shown in Table 1, N\textsc{Attack} can succeed even faster than the white-box BPDA on CIFAR-10, yet runs slower on ImageNet. The main reason is that when the image size is as small as CIFAR10 ($3^1 \times 32^1 \times 32^3$), the search space is moderate. However, the run time could be lengthy for high resolution images like ImageNet ($3^1 \times 299^1 \times 299^3$) especially for some hard cases (we can find the adversarial examples for nearly 90% test images but it could take about 60 minutes for a hard case).

We use a regression net to approximate a good initialization of $\mu_0$, and we name N\textsc{Attack} initialized with the regression net as N\textsc{Attack}-R. We run N\textsc{Attack} and N\textsc{Attack}-R on ImageNet with the mini-batch size $b = 40$. The success rate for N\textsc{Attack} with random initialization is 82% and for N\textsc{Attack}-R is 91.9%, verifying the efficacy of the regression net. The run time shown in Table 1 is calculated on the images with successful attacks. The results demonstrate that N\textsc{Attack}-R can reduce by 22.5s attack time per image compared with the random initialization.

References


