Supplementary Material: Improving Query Efficiency of Black-box Adversarial Attack

Yang Bai[1,4,⋆], Yuyuan Zeng[2,4,⋆],
Yong Jiang[1,2,4,†], Yisen Wang[3,†], Shu-Tao Xia[2,4], and Weiwei Guo[5]

[1] Tsinghua Berkeley Shenzhen Institute, Tsinghua University
[2] Tsinghua Shenzhen International Graduate School, Tsinghua University
[3] Shanghai Jiao Tong University
[4] PCL Research Center of Networks and Communications, Peng Cheng Laboratory
[5] vivo AI Lab

A The Structure of ANP in NP-Attack

Here, we show the detailed structure of the pre-trained ANP [15, 27] model on CIFAR10 [17].

B More Optimization Options of NP-Attack

Here we show the details of the other two optimization options, NP-Attack-Z and NP-Attack-RZ.

NP-Attack-Z In this case, we fix r and optimize on the distribution of z, in order to sample z with adversary. Under the guidance of score outputs in each query, to be simplified, we just optimize the predictive mean μ, keeping σ fixed. The optimization of our NP-Attack is inherited from NES [31], which could enhance query efficiency compared with those vector-wise gradient estimation strategies. Omitting the fixed r while inheriting *Decoder* g from the pre-trained NP, the estimated loss function in our NP-Attack could be considered as $L = \mathbb{E}_{\mathcal{N}(z|\mu,\sigma^2)} l(g(z))$. The optimization of μ is implemented by adding some random Gaussian noises and further using NES to estimate the gradient. Those added Gaussian noises are of the same variance with z in pre-trained NP. The reason comes as that a same variance could keep z of a simpler representation for variance after adjustment in each iteration, which benefits the optimization later. The optimization function could be computed as: $\mu_{t+1} \leftarrow \mu_t - \eta \nabla_\mu L \mid_{\mu_t}$, where η denotes a learning rate. To guarantee such optimization being more accurate, in practice the optimization is over a mini-batch of sample size b, which means

⋆ Equal contribution. ({y-bai17, zengyy19}@mails.tsinghua.edu.cn)
† Corresponding authors: Yisen Wang (eewangyisen@gmail.com) and Yong Jiang (jiangy@sz.tsinghua.edu.cn).

Table 1. Structure of the deterministic part of encoder h. (The input of Linear1 includes the pixel position: 3072 ($32 \times 32 \times 3$) $\times$ 3, the pixel value in RGB respectively: 3072 $\times$ 1, thus with a size of 3072 $\times$ 4 in total.)

Layer	Input	Output	Activation function
Linear1	3072×4	3072×128	ReLU
Linear2	3072×128	3072×128	ReLU
Linear3	3072×128	3072×128	ReLU
SelfAtt	3072×128	3072×128	-
CrossAtt	3072×128	3072×128	-
Linear4	3072×128	3072×128	-

Table 2. Structure of the sampled latent part of encoder h.

Layer	Input	Output	Activation function
Linear1	3072×4	3072×128	ReLU
Linear2	3072×128	3072×128	ReLU
Linear3	3072×128	3072×128	ReLU
SelfAtt	3072×128	3072×128	-
Mean	3072×128	3072×128	-
Linear4	3072×128	3072×128	-

Table 3. Structure of the latent part of decoder g. (The input of Linear1 includes the deterministic path: 3072 $\times$ 128, the sampled latent path: 3072 $\times$ 128 and the target pixel position: 3072 $\times$ 3, thus with a size of 3072 $\times$ 259 in total.)

Layer	Input	Output	Activation function
Linear1	3072×259	3072×128	ReLU
Linear2	3072×128	3072×128	ReLU
Linear3	3072×128	3072×128	ReLU
Linear4	3072×128	3072×1	ReLU

we add b random perturbations independently during each iteration and operate on those outputs. So the updating operation is on the average values instead:

$$\mu_{t+1} \leftarrow \mu_t - \frac{\eta}{b} \sum_{i=1}^{b} l(g(z_i)) \nabla_{\mu_t} \log \mathcal{N}(z_i \mid \mu_i, \sigma^2). \tag{1}$$

Given that $\mu_i = \mu_t + p_i\sigma, z_i = \mu_i + p_i\sigma = \mu_t + 2p_i\sigma$, where p_i is sampled from the standard Gaussian distribution $\mathcal{N}(0, I)$, and σ is multiplied to keep a simple representation for variance, $\nabla_{\mu_t} \log \mathcal{N}(z_i \mid \mu_i, \sigma^2) \propto \sigma^{-1} p_i$.

NP-Attack-RZ Similarly, we propose NP-Attack-RZ as a third choice by optimizing both r and z simultaneously. The operation is the same with NP-Attack-R/Z as shown in Algorithm 1.

C The Visual Comparison of Adversarial Examples by Different NP-Attack Versions.

We show the adversarial examples generated by NP-Attack-R/Z/RZ in Figure 1. We observe that adversarial perturbations generated by NP-Attack-R are centered around the main digits while perturbations generated by NP-Attack-Z are scattered on the background. Furthermore, the location of adversarial perturbations in NP-Attack-RZ is somehow in a moderate degree between NP-Attack-R and NP-Attack-Z.

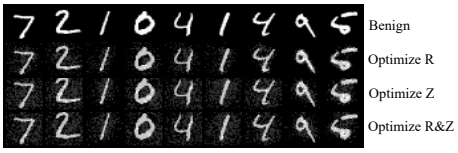

Fig. 1. Adversarial examples of different optimization methods (NP-Attack-R/Z/RZ).

D Evaluation on Additional Architectures

We additionally do experiments under $\epsilon = 0.031$ on CIFAR-10 on various architectures including ResNet18 [12], VGG16 [24] and WideResNet [35]. We compare NP-Attack-R with the state-of-the-art $\mathcal{N}$Attack here and report the attack success rate (ASR) and average queries in Table 4. We conduct untargeted attacks with the same experimental setting of Section 4.2. The experimental results demonstrated that our method still outperforms towards various architectures.

Table 4. Adversarial evaluation of untargeted black-box attacks on CIFAR-10 on various architectures.

Attack Method	ASR			Avg Queries		
	ResNet18	VGG16	WRN	ResNet18	VGG16	WRN
$\mathcal{N}$Attack	96.79%	**100%**	99.89%	311	430	335
NP-Attack-R(Ours)	**100%**	**100%**	**100%**	**185**	**294**	**196**

E The Generated Adversarial Examples on ImageNet by NP-Attack

We show the adversarial examples generated by our NP-Attack-R here. It is demonstrated that though we perturb each 32×32×3 patch in the original images, the perturbation generated by our method is still imperceptible.

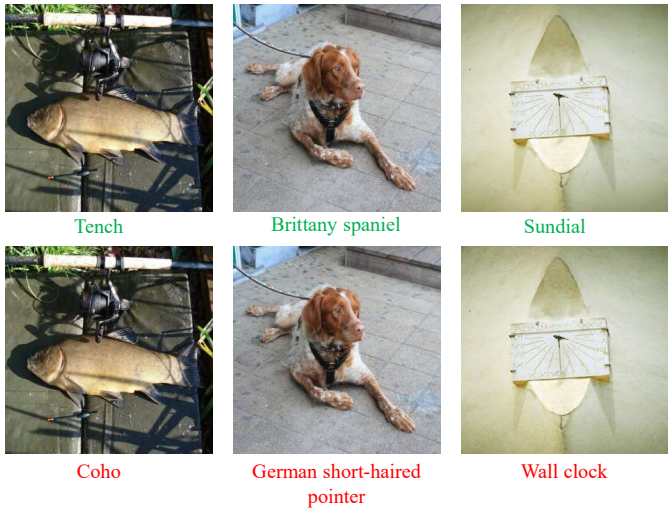

Fig. 2. Adversarial example of ImageNet generated by NP-Attack-R on untargeted attack bounded by L_∞=0.05. Top row shows the original images and the true labels while the bottom row are the adversarial examples with predicted label.