A New Ensemble Adversarial Attack Powered by Long-term Gradient Memories

Zhaohui Che[1], Ali Borji[2], Guangtao Zhai[1]*, Suiyi Ling[3], Jing Li[4], Patrick Le Callet[3]
[1]Shanghai Jiao Tong University, Shanghai, China
[2]MarkableAI Inc., Brooklyn, NY 11201 USA
[3]Université de Nantes, Nantes, France
[4]Alibaba Group, Hangzhou, China

Abstract

Deep neural networks are vulnerable to adversarial attacks. More importantly, some adversarial examples crafted against an ensemble of pre-trained source models can transfer to other new target models, thus pose a security threat to *black-box* applications (when the attackers have no access to the target models). Despite adopting diverse architectures and parameters, source and target models often share similar decision boundaries. Therefore, if an adversary is capable of fooling several source models concurrently, it can potentially capture intrinsic transferable adversarial information that may allow it to fool a broad class of other *black-box* target models. Current ensemble attacks, however, only consider a limited number of source models to craft an adversary, and obtain poor transferability. In this paper, we propose a novel *black-box* attack, dubbed *Serial-Mini-Batch-Ensemble-Attack* (*SMBEA*). *SMBEA* divides a large number of pre-trained source models into several mini-batches. For each single batch, we design 3 new ensemble strategies to improve the intra-batch transferability. Besides, we propose a new algorithm that recursively accumulates the "long-term" gradient memories of the previous batch to the following batch. This way, the learned adversarial information can be preserved and the inter-batch transferability can be improved. Experiments indicate that our method outperforms state-of-the-art ensemble attacks over multiple pixel-to-pixel vision tasks including image translation and salient region prediction. Our method successfully fools two online *black-box* saliency prediction systems including DeepGaze-II (Kummerer 2017) and SALICON (Huang et al. 2017). Finally, we also contribute a new repository to promote the research on adversarial attack and defense over pixel-to-pixel tasks: https://github.com/CZHQuality/AAA-Pix2pix.

Introduction

Deep neural networks, despite their great success in various vision tasks, are susceptible to adversarial attacks (Szegedy et al. 2014; Goodfellow et al. 2015). The adversarial attacks add some quasi-imperceptible perturbations to the original input, to significantly change the model output. More importantly, some well-designed adversarial examples can transfer across different models. That is, the adversary crafted against some pre-trained source models can transfer to other new target models. Despite the source and target models adopting diverse architectures and parameters, they may share similar decision boundaries. Thus, if an adversary can fool several source models, it can capture the intrinsic transferable adversarial information that allows it to fool a broad class of other *black-box* target models. The transferability of adversarial examples provides a potential chance to launch *black-box* attacks without having access to the target model. In contrary, *white-box* attacks require all information of target model, thus they are not practical in real world.

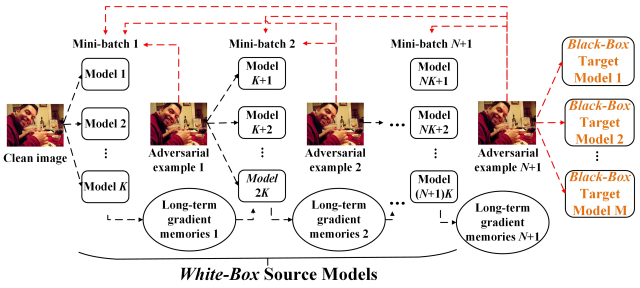

Figure 1: The general idea of the proposed attack. Our attack divides a large number of pre-trained source models into several mini-batches. For each batch, we craft an adversary that fools multiple intra-batch source models. We also recursively accumulate the "long-term" gradient memories of previous batch to the following batch, in order to preserve the learned adversarial information and to improve inter-batch transferability. The red dashed lines denote that the crafted adversarial example can fool the previous source models, and also successfully fools the *black-box* target models.

Particularly, adversarial attack serves as an efficient surrogate to evaluate the robustness of deep networks before they are deployed in real world, especially for security-related applications, *e.g.* autonomous driving (Yang and Hsu 2017; Alletto et al. 2016) and face verification (Sharif et al. 2016; Dong et al. 2019). Therefore, exploring adversarial attacks, especially the transferable *black-box* ones, is critical to demystifying the fragility of deep neural networks.

Current approaches for crafting transferable adversarial examples fall into two major categories: (1) *Ensemble at-*

*Corresponding Author.

tacks (Dong et al. 2018; Liu et al. 2017; Xie et al. 2017) craft transferable adversarial examples via fooling multiple *white-box* source models in parallel. (2) *Generative methods* (Zhao, Dheeru, and Sameer 2018; Wei et al. 2019) rely on an extra generative adversarial network (GAN). Specifically, a generator is trained to produce the adversaries that aim to fool the target model, while a discriminator is trained to distinguish the synthetic adversaries from original clean images for minimizing the perceptibility.

However, current methods have some drawbacks: (1) Normal *ensemble attacks* only consider a limited number of source models. To the best of our knowledge, the state-of-the-art *ensemble attacks* (Liu et al. 2017; Dong et al. 2018) adopt less than 8 source models to craft the adversaries. In their implementations, all of the source models are combined in parallel. As a result, the number of source models is limited by the GPU memory. (2) Although the parallel computing technique enables concurrent attacks against a large number of models, it brings new optimization challenges, because computing and back-propagating the gradients of cost function w.r.t a large number of models become slow and difficult. (3) *Generative methods* rely on an extra GAN network which is not easy to train, and also require a lot of training samples with expensive labels.

For solving these problems, we propose a novel *Serial-Mini-Batch-Ensemble-Attack* (*SMBEA*). Before elaborating our method, we first introduce two empirical observations that inspire our method: (1) *Crafting an adversarial example is analogous to training a model, and the transferability of the adversarial example is analogous to the generalizability of the model* (Dong et al. 2018). Thus, it is expected to increase the transferability of adversary via fooling diverse source models as much as possible, (2) *Compared to the magnitude of the perturbation, the spatial structure of the adversarial perturbation has stronger impact on the final fooling ability* (Xie et al. 2017). Thus, we focus on preserving the learned adversarial structure information to optimize the fooling ability and transferability, while mitigating the magnitude of perturbation to minimize the perceptibility.

Inspired by the aforementioned empirical observations, our method mimics classical deep network training procedures to craft transferable adversaries, as shown in Fig. 1. Specifically, we divide a large number of pre-trained source models into several mini-batches, and each single batch contains K (K is the batch-size) individual source models. For each batch, we introduce 3 new ensemble strategies to combine these individual models, in order to improve intra-batch transferability. For the inter-batch case, we propose a new algorithm that recursively accumulates the "long-term" gradient memories of previous batch to the following batch. This way, the learned adversarial information can be preserved and the inter-batch transferability can be improved. As shown in Fig. 1, we start from a clean image, then recursively update the adversary across different batches, and finally obtain an adversary that not only fools all previous source models, but also fools new *black-box* target models.

We summarize our contributions as follows:

- **A new black-box attack approach:** We propose a novel *black-box* attack, where we introduce 3 new ensemble strategies for improving intra-batch transferability, and propose a new algorithm that preserves "long-term" gradient memories for improving inter-batch transferability.
- **Generality:** Our method can attack multiple pixel-to-pixel vision tasks, *e.g.* image translation and saliency prediction. Besides, our method successfully fools two online *black-box* saliency prediction systems in the real world: *i.e.* DeepGaze-II and SALICON.
- **A new repository:** We provide a *software repository* including 13 common attack methods and our proposed attack, and 16 pre-trained source models. This repository aims to boost adversarial attack and defense research in pixel-to-pixel tasks. It also serves as a complement to *CleverHans* repository (Papernot et al. 2016a).

Related works

In 2014, Szegedy *et al.* verified the existence of adversarial examples for the first time (Szegedy et al. 2014).

Goodfellow *et al.* (Goodfellow et al. 2015) introduced the fast gradient sign method (FGSM) to craft *white-box* adversarial examples by one-step gradient update along the direction of the sign of gradient at each pixel.

Kurakin *et al.* (Kurakin et al. 2016) proposed the basic iterative version of FGSM, *i.e.* I-FGSM. I-FGSM utilizes a small step to update adversarial example for multiple iterations by vanilla Stochastic Gradient Descent (*SGD*) optimization. However, *SGD* has some drawbacks, such as slow convergence and always drops into poor local minima.

Papernot *et al.* (Papernot 2017) proposed a *black-box* attack against image classifiers. Specifically, they trained a surrogate model to mimic the target *black-box* model.

Dong *et al.* (Dong et al. 2018) introduced the Momentum based Iterative Method (MIM), which utilizes Momentum based Stochastic Gradient Descent (*MSGD*) (Qian 1999) optimizer to craft adversaries. MIM accumulates the 1^{st} momentum in gradient descent direction to reduce poor local minima and to avoid "over-fitting" one specific model, thus demonstrating better transferability in *black-box* setting.

Madry *et al.* (Madry et al. 2018) proposed the Projective Gradient Descent (PGD) attack, which extends the I-FGSM method to a universal first-order adversary by introducing a random start state. PGD also uses *SGD* to update the adversary iteratively. It serves as a strong *white-box* attack.

Carlini *et al.* (Carlini and Wagner 2017) introduced an efficient *white-box* attack, dubbed C&W's attack, which breaks defensive distillation (Papernot et al. 2016b). C&W's attack utilizes *Adam* optimization due to its fast convergence and high fooling ability.

Liu *et al.* (Liu et al. 2017) proposed *targeted* and *non-targeted* ensemble attacks that successfully fool *black-box* classification system *i.e. Clarifai.com*. Wei *et al.* (Wei et al. 2019) also trained a generative network to craft transferable adversaries against image and video detection models.

Xie *et al.* (Xie et al. 2017) proposed Dense Adversary Generation (DAG) to attack segmentation and object detection models. Mopuri *et al.* (Mopuri, Ganeshan, and Radhakrishnan 2018) introduced a general objective function that produces image-agnostic adversaries from latent space.

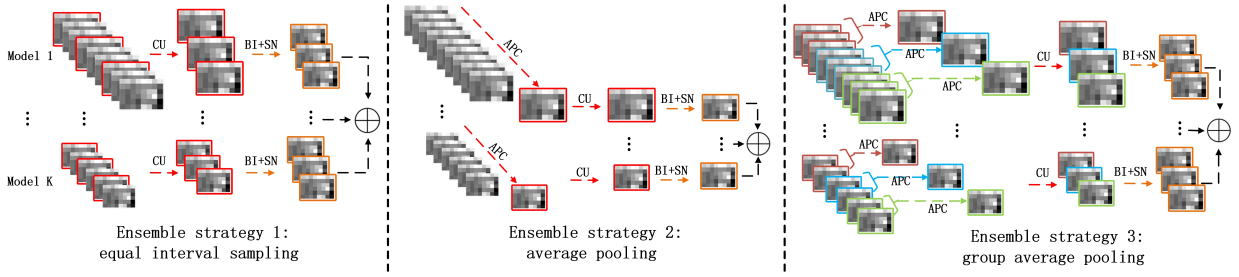

Figure 2: Visualizations of three ensemble strategies in *feature space*. The raw feature maps are processed by the batch normalization and ReLU activation function. **CU**: Channel amount Unification, **BI**: Bilinear Interpolation, **SN**: Softmax Normalization, **APC**: Average Pooling in Channel direction. $\oplus$ represents an elementwise weighted summation.

Proposed method

In this section, we introduce a novel *black-box* adversarial attack dubbed *Serial-Mini-Batch-Ensemble-Attack* (*SMBEA*). We first introduce the intra-batch algorithm, then we elaborate the inter-batch algorithm.

Intra-batch ensemble strategies

In our implementation, each single mini-batch includes K=4 [1] *white-box* source models that are pre-trained over the same pixel-to-pixel task. Thus, these models have similar decision boundaries, despite adopting diverse architectures and parameters. Similar decision boundaries across models increase the chance of crafting an adversary that fools all these models. At the same time the diversity across different models serves as the regularization and alleviates "over-fitting" to a specific model, which in turn results in high intra-batch transferability.

In this work, we only consider the *targeted attack*. The *non-targeted attack* is a straightforward extension. We formulate the *targeted* ensemble attack in pixel-to-pixel tasks as a constrained optimization problem. For simplicity, we first introduce a basic ensemble strategy, where multiple models are fused in *output space*, *i.e.* the optimization objective is computed by an element-wise weighted summation of the final predictions of multiple source models:

$$\begin{cases} \min \mathcal{L}_o = \mathcal{L}_1[\sum_{n=1}^{K} \sigma_n \cdot \mathbb{F}_n(I^*),\ \mathcal{F}(G)] + \lambda_1 \cdot \mathcal{L}_2(I,\ I^*), \\ s.t. \quad \mathcal{L}_2(I,\ I^*) \leq \mathcal{T}_1. \end{cases} \tag{1}$$

where I and I^* are original clean image and adversarial example, respectively. G represents the guide image, while $\mathcal{F}(G)$ is the ground-truth output of G. For the *targeted attack*, the goal is to change the models' ensemble prediction of I^* towards the prediction of the guide image. $\mathbb{F}_n$ is the n_{th} source model within the mini-batch, and $\mathbb{F}_n(I^*)$ represents the final prediction of the n_{th} model on the crafted adversary I^*. σ_n is ensemble weight, $\sum_{n=1}^{K} \sigma_n = 1$. $\mathcal{L}_1$ is the loss function that is minimized when $\sum_{n=1}^{K} \sigma_n \cdot \mathbb{F}_n(I^*) = \mathcal{F}(G)$. $\mathcal{L}_2$ is the perceptual constraint, *e.g.* $\mathbf{L_0}$, $\mathbf{L_1}$, or $\mathbf{L}_\infty$ norms, which is minimized to guarantee that the crafted adversary I^* looks (perceptually) similar to the original clean image I. $\mathcal{T}_1$ is the maximum perceptual constraint for single batch. λ_1 is a hyper-parameter to balance the fooling ability loss $\mathcal{L}_1$ and the perceptual constraint $\mathcal{L}_2$.

However, the basic ensemble strategy in *Eq.* 1 only fools the final predictions in the *output space*. Here we further dig into the *feature space* to explore other efficient ensemble strategies. This is motivated by the consideration that combining *output space* and *feature space* ensembles provides a deep supervision for crafting strong adversary that not only fools the final predictions, but also fools the intermediate feature maps. This way, the objective $\mathcal{L}_{o,f}$ is rewritten as:

$$\begin{cases} \min \mathcal{L}_{o,f} = \mathcal{L}_o + \lambda_2 \cdot \mathcal{L}_3[\sum_{n=1}^{K} \omega_n \cdot \mathbb{D}_n(I^*),\ \sum_{n=1}^{K} \omega_n \cdot \mathbb{D}_n(G)], \\ s.t.\ \ \mathcal{L}_2(I,\ I^*) \leq \mathcal{T}_1. \end{cases} \tag{2}$$

where $\mathbb{D}_n(I^*)$ and $\mathbb{D}_n(G)$ represent the feature maps of the n_{th} source model on the crafted adversary I^* and guide image G. ω_n is the *feature space* ensemble weight, $\sum_{n=1}^{K} \omega_n = 1$. $\mathcal{L}_3$ is the loss function that aims to minimize the *feature space* distance between I^* and G. λ_2 is a hyper-parameter to balance *feature space* fooling loss $\mathcal{L}_3$, together with *output space* fooling loss and perceptual constraint.

Different models utilize different network architectures, so their feature maps have different resolutions and channels. For solving this, we introduce 3 different *feature space* ensemble strategies, as shown in Fig. 2. More details regarding the feature layer selection are provided in the experiments section. Here we focus on explaining ensemble strategies.

The first ensemble strategy evenly samples p feature maps from each model, and the sampling interval $\bar{p}_n$ of the n_{th} model can be computed as $\bar{p}_n = P_n/p$, where P_n is the total number of feature channels of one selected feature layer from the n_{th} model. We set p and P_n as the powers of 2, to make sure that $\bar{p}_n$ is an integer. This way, we obtain the same number of feature maps (channels) from different models. Next, we use bilinear interpolation to resize the selected feature maps of different models to the same resolution (*i.e.* height×weight). Then, we adopt *softmax* function to normalize these feature maps. Finally, we obtain the *feature-space* ensemble result by an element-wise weighted summation of different feature maps.

The other two ensembles have similar pipelines, except for the first step, which is explained below. For each model, the second ensemble computes the average pooling of the

[1] Batch-size K is a tunable hyper-parameter. Here we adopt 4 as the batch-size because it achieves a good tradeoff between the transferability and GPU memory cost.

P_n feature maps in the channel direction, and obtains a 2D one-channel feature map from each model. The third ensemble divides P_n feature maps into p groups, then computes the channel direction average of each group, and obtains p candidate feature maps from each model.

Algorithm 1 : Intra-batch update rules of *SMBEA*. This algorithm is applicable to the first mini-batch. m_t represents the 1^{st} gradient momentum vector, while v_t represents the 2^{nd} gradient momentum vector. $\odot$ is an element-wise product.

Require:

Original clean image I, guide image G (randomly selected);
Intra-batch source models: $\mathbb{F}_1, \mathbb{F}_2, ..., \mathbb{F}_K$;
Decay factors of short-term gradient momentums: μ_1, μ_2;
Smoothing term: ϵ;
Maximum iterations *X* for single mini-batch;
Maximum perceptual constraint $\mathcal{T}_1$ for the first mini-batch;
Step size of iterative gradient descent α;

Ensure:

An adversarial example I_X^*; The ultimate 1^{st} momentum m_X, and the ultimate 2^{nd} momentum v_X.

Initialization: $I_0^* \leftarrow I, m_0 \leftarrow \mathbf{0}^d, v_0 \leftarrow \mathbf{0}^d, t \leftarrow 0$
while $(0 \leq t < X$ and $\|I, \ I_t^*\|_1 \leq \mathcal{T}_1)$ **do**
  $t \leftarrow t + 1$; (update the iteration epoch)
  $g_t \leftarrow \nabla_{I_{t-1}^*} \mathcal{L}_{o,f}$; ($\mathcal{L}_{o,f}$ is defined in *Eq.* 2)
  $\hat{g_t} \leftarrow \frac{g_t}{\|g_t\|_1}$; (gradient normalization)
  $m_t \leftarrow \mu_1 \cdot m_{t-1} + (1 - \mu_1) \cdot \hat{g_t}$; (update the m_t)
  $v_t \leftarrow \mu_2 \cdot v_{t-1} + (1 - \mu_2) \cdot \hat{g_t}^2$; (update the v_t)
  $\hat{m_t} \leftarrow m_t / (1 - \mu_1^t)$; (bias correction)
  $\hat{v_t} \leftarrow v_t / (1 - \mu_2^t)$; (bias correction)
  $I_t^* \leftarrow \text{Clip}(I_{t-1}^* - \alpha \cdot \frac{1}{\sqrt{\hat{v_t}} + \epsilon} \odot \hat{m_t})$; (update the adversary)
end while
return $I_X^* \leftarrow I_t^*, \ m_X \leftarrow m_t, \ v_X \leftarrow v_t$.

Intra-batch update rules

For solving the constrained optimization problem in *Eq*. 2, we exhaustively test 5 common gradient descent optimization methods, *i.e.* stochastic gradient descent (*SGD*), momentum based gradient descent (*MSGD*) (Qian 1999), *Adagrad* (Duchi, Hazan, and Singer 2011), *RMSProp* (Tieleman and Hinton 2012), and *Adam* (Kingma and Ba 2015). The major differences between these gradient descent methods are two gradient momentums, explained below.

The 1^{st} gradient momentum accumulates the gradients of previous iterations to stabilize the gradient descent direction, and helps to reduce poor local minima.

The 2^{nd} gradient momentum adapts the learning rates to different parameters. In pixel-to-pixel attacking tasks, we aim to update the image pixels of the crafted adversarial example. However, in the attacking process, a small fraction of pixels are frequently updated, while the remaining pixels are occasionally updated. The intensities of frequent pixels grow rapidly and reach the bound of perceptual constraint quickly, while the infrequent pixels are far from convergence at this moment. This issue limits the tradeoff between fooling ability and perceptibility. For mitigating this limitation, the 2^{nd} gradient momentum was proposed, which assigns a small update step-size for frequent pixels, while assigning a big update step-size for infrequent pixels.

In our tasks, *Adam* achieves the best tradeoff between transferability, perceptibility, and convergence speed. The update rules of the intra-batch algorithm based on *Adam* optimizer are given in the Algorithm 1. Specifically, *Adam* optimizer utilizes the 1^{st} momentum to avoid local poor minima and to prevent "over-fitting", thus improving transferability. Besides, it also uses the 2^{nd} momentum to adapt the learning rates for different pixels, thus enhancing the tradeoff between fooling ability and perceptibility.

In our implementation, the original image I and the guide image G are normalized to be in the range [0, 1]. The default decay factors are set as $\mu_1 = 0.9$ and $\mu_2 = 0.99$. $\epsilon = 1 \times 10^{-8}$ is a smoothing term to avoid division by zero. The maximum number of iterations *X* for a single batch is 20. The iterative gradient descent step size $\alpha = 2 \times 10^{-4}$. We adopt $\mathbf{L_1}$ norm as the perceptual constraint. Finally, we clip the crafted adversary I_t^* into the range [0, 1] to make sure I_t^* is a valid image. This way, we obtain an adversary I^* that is able to fool multiple intra-batch source models.

Inter-batch update rules

The intra-batch algorithm only guarantees that the crafted adversary can fool a limited number of source models. For breaking this limitation, we propose a novel inter-batch algorithm that recursively accumulates the "long-term" gradient memories of the previous batches to the following batches. These "long-term" gradient memories preserve the learned adversarial information, and also serve as the regularization to prevent "over-fitting" on a specific batch, thus increasing the inter-batch transferability.

The proposed inter-batch update rules are presented in the Algorithm 2. The main differences between the two algorithms are the initialization and the bias corrections.

We first introduce the initialization. We adopt four variables of the previous batch to recursively initialize the variables of the following batch, explained below.

- $I_0^{*\,(i)} \leftarrow I_X^{*\,(i-1)}$: we adopt the adversarial example of the previous batch $I_X^{*\,(i-1)}$ as the initial state of the current batch, because $I_X^{*\,(i-1)}$ has learned some adversarial information against multiple models of previous batch.
- $m_0^{(i)} \leftarrow \beta_1 \cdot m_X^{(i-1)}$ *and* $v_0^{(i)} \leftarrow \beta_2 \cdot v_X^{(i-1)}$: we utilize the 1^{st} and 2^{nd} momentums of the previous batch to initialize the momentums of the current batch. These "long-term" gradient momentums preserve the learned adversarial information, and also serve as the regularization to prevent "over-fitting" on the following batch, thus boosting the inter-batch generalizability of the crafted adversary.
- $\mathcal{T}_1^{(i)} \leftarrow \mathcal{T}_1^{(i-1)} + \beta_3^i \cdot \mathcal{T}_1^{(1)}$: we recursively update the maximum perceptual constraint of the current batch (*i.e.* $\mathcal{T}_1^{(i)}$) by adding a loose factor $\beta_3^i \cdot \mathcal{T}_1^{(1)}$ to the perceptual constraint of the previous batch (*i.e.* $\mathcal{T}_1^{(i-1)}$), in order to prevent premature convergence that causes "under-fitting". Besides, by increasing the number of batches, the adversarial example tends to be converged, so we reduce the loose factor via a decay rate β_3^i, where β_3^i denotes the $\beta_3 \in [0, 1]$ to the power i (i is the batch number).

Algorithm 2 : Inter-batch update rules of *SMBEA*. This algorithm is applicable to all mini-batches, except for the first one, *i.e.* $i > 1$. Notice that the superscript in brackets denotes the batch number, *e.g.* ${I_X^*}^{(i)}$ is the adversary of the i_{th} batch, while the superscript w/o brackets denotes the pow, *e.g.* β_3^i denotes the β_3 to the power i.

Require:

The adversarial example of the previous batch ${I_X^*}^{(i-1)}$, the guide image G;
The 1^{st} gradient momentum of the previous batch $m_X^{(i-1)}$;
The 2^{nd} gradient momentum of the previous batch $v_X^{(i-1)}$;
Maximum perceptual constraint $\mathcal{T}_1^{(i-1)}$ of the previous batch;
Maximum perceptual constraint $\mathcal{T}_1^{(1)}$ of the first batch;
Maximum batch number N;
Intra-batch models of the current batch: $\mathbb{F}_1^{(i)}, \mathbb{F}_2^{(i)}, ..., \mathbb{F}_K^{(i)}$;
Decay factors of short-term gradient momentums: μ_1, μ_2;
Weights of long-term gradient momentums: $\beta_1, \beta_2 \in [0, 1]$;
Decay factor of perceptual constraint: $\beta_3 \in [0, 1]$;

Ensure:

An adversarial example of current batch ${I_X^*}^{(i)}$; The ultimate 1^{st} momentum $m_X^{(i)}$, and ultimate 2^{nd} momentum $v_X^{(i)}$ of the current batch.

1: Initialization: ${I_0^*}^{(i)} \leftarrow {I_X^*}^{(i-1)}$, $m_0^{(i)} \leftarrow \beta_1 \cdot m_X^{(i-1)}$, $v_0^{(i)} \leftarrow \beta_2 \cdot v_X^{(i-1)}$, $\mathcal{T}_1^{(i)} \leftarrow \mathcal{T}_1^{(i-1)} + \beta_3^i \cdot \mathcal{T}_1^{(1)}$, $t \leftarrow 0$
2: **while** $(i \leq N$ and $0 \leq t < X$ and $\|I,\ I_t^*\|_1 \leq \mathcal{T}_1^{(i)})$ **do**
3: do Step.3 - Step.7 of the Algorithm. 1.
4: $\hat{m_t}^{(i)} \leftarrow \frac{m_t^{(i)}}{(1-\mu_1^t)} + \beta_1 \cdot m_X^{(i-1)}$; (bias correction)
5: $\hat{v_t}^{(i)} \leftarrow \frac{v_t^{(i)}}{(1-\mu_2^t)} + \beta_2 \cdot v_X^{(i-1)}$; (bias correction)
6: ${I_t^*}^{(i)} \leftarrow \text{Clip}({I_{t-1}^*}^{(i)} - \alpha \cdot \frac{1}{\sqrt{\hat{v_t}^{(i)}}+\epsilon} \odot \hat{m_t}^{(i)})$;
7: **end while**
8: **return** ${I_X^*}^{(i)} \leftarrow {I_t^*}^{(i)}, m_X^{(i)} \leftarrow m_t^{(i)}, v_X^{(i)} \leftarrow v_t^{(i)}, i \leftarrow i+1$.

The proposed inter-batch algorithm inherits the good properties of the classical *Adam* method, explained below.

Property 1: *The effective step-size of inter-batch update rules is invariant to the scale transform of the gradient.*

Proof 1: As shown in Step.6 of Algorithm. 2, assuming $\epsilon = 0$, the effective step-size of the adversarial example at iteration t is $\Delta_t^{(i)} = \alpha \cdot \frac{1}{\sqrt{\hat{v_t}^{(i)}}} \odot \hat{m}_t^{(i)}$. The effective step-size $\Delta_t^{(i)}$ is invariant to the scale transform of gradient, because scaling raw gradient g_t with factor c will be normalized by $\mathbf{L_1}$ norm, *i.e.* $\frac{g_t}{\|g_t\|_1} = \frac{c \cdot g_t}{\|c \cdot g_t\|_1}$. Thus, $\hat{m_t}^{(i)}$, $\hat{v_t}^{(i)}$, $\Delta_t^{(i)}$ are invariant to the scale transform of the gradient.

Property 2: *The proposed inter-batch bias corrections can correct for the discrepancy between the expected value of the exponential moving averages (*i.e. $\mathbb{E}[m_t^{(i)}]$ *or* $\mathbb{E}[v_t^{(i)}]$*) and the true expected gradients (*i.e. $\mathbb{E}[\hat{g}_t]$ *or* $\mathbb{E}[{\hat{g}_t}^2]$*).*

Proof 2: The proposed inter-batch bias corrections are shown in Steps.4-5 of the Algorithm. 2. Here, we derive the bias correction for the 2^{nd} momentum estimate, and the derivation for the 1^{st} momentum is completely analogous.

Let $\hat{g}_t$ be the normalized gradient at iteration t, and we wish to estimate its 2^{nd} momentum $\hat{v}_t^{(i)}$ using an exponential moving average of the true squared gradient. In the inter-batch case, the raw 2^{nd} momentum is initialized as $v_0^{(i)} \leftarrow \beta_2 \cdot v_X^{(i-1)}$. The recursive update equation of raw momentum $v_t^{(i)} = \mu_2 \cdot v_{t-1}^{(i)} + (1-\mu_2) \cdot {\hat{g}_t}^2$ can be rewritten as:

$$v_t^{(i)} = \mu_2^t \cdot v_0^{(i)} + (1-\mu_2) \sum_{k=1}^{t} \mu_2^{t-k} \cdot {\hat{g}_k}^2, \tag{3}$$

We wish to know how $\mathbb{E}[v_t^{(i)}]$, the expected value of the exponential moving average at iteration t, relates to the true expected squared gradient $\mathbb{E}[{\hat{g}_t}^2]$, so we can correct for the discrepancy between them. We take expectations of the left and right sides of *Eq*. 3

$$\begin{cases} \mathbb{E}[v_t^{(i)}] = \mathbb{E}[\mu_2^t \cdot v_0^{(i)}] + \mathbb{E}[(1-\mu_2)\sum_{k=1}^{t} \mu_2^{t-k} \cdot {\hat{g}_k}^2] \\ \qquad\quad = \mu_2^t \cdot v_0^{(i)} + \mathbb{E}[{\hat{g}_t}^2] \cdot (1-\mu_2)\sum_{k=1}^{t} \mu_2^{t-k} + \zeta \\ \qquad\quad = \mu_2^t \cdot v_0^{(i)} + \mathbb{E}[{\hat{g}_t}^2] \cdot (1-\mu_2^t) + \zeta, \end{cases} \tag{4}$$

where $\zeta = 0$ if the true 2^{nd} momentum $\mathbb{E}[{\hat{g}_t}^2]$ is stationary, according to *Adam* (Kingma and Ba 2015). We suppose $\mathbb{E}[{\hat{g}_t}^2]$ is stationary, and divide the left and right sides of *Eq*. 4 by $(1-\mu_2^t)$:

$$\frac{\mathbb{E}[v_t^{(i)}]}{1-\mu_2^t} = \frac{\mu_2^t \cdot v_0^{(i)}}{1-\mu_2^t} + \mathbb{E}[{\hat{g}_t}^2], \tag{5}$$

where $\frac{\mu_2^t \cdot v_0^{(i)}}{1-\mu_2^t} = \frac{\mu_2^t}{1-\mu_2^t} \cdot \beta_2 \cdot v_X^{(i-1)}$ ($v_0^{(i)}$ is initialized as $\beta_2 \cdot v_X^{(i-1)}$) is the "long-term" momentum from the previous batch. This "long-term" momentum decreases rapidly with the increase of iteration t due to the decay rate $\frac{\mu_2^t}{1-\mu_2^t}$. However, in our tasks, we wish to assign a smooth decay rate to the "long-term" momentum in subsequent iterations, in order to preserve the learned adversarial information as much as possible. Thus, we modify the decay weight as $\frac{1}{1-\mu_2^t}$, which obtains a slower decay rate than $\frac{\mu_2^t}{1-\mu_2^t}$. We first subtract $\frac{\mu_2^t \cdot v_0^{(i)}}{1-\mu_2^t}$, then add $\frac{1 \cdot v_0^{(i)}}{1-\mu_2^t}$ to the left and right sides of *Eq*. 5

$$\frac{\mathbb{E}[v_t^{(i)}]}{(1-\mu_2^t)} - \frac{\mu_2^t \cdot v_0^{(i)}}{1-\mu_2^t} + \frac{1 \cdot v_0^{(i)}}{1-\mu_2^t} = \frac{1 \cdot v_0^{(i)}}{1-\mu_2^t} + \mathbb{E}[{\hat{g}_t}^2], \tag{6}$$

Next, we plug $v_0^{(i)} = \beta_2 \cdot v_X^{(i-1)}$ into *Eq*. 6, and obtain

$$\frac{\mathbb{E}[v_t^{(i)}]}{(1-\mu_2^t)} + \beta_2 \cdot v_X^{(i-1)} = \frac{1}{1-\mu_2^t} \cdot \beta_2 \cdot v_X^{(i-1)} + \mathbb{E}[{\hat{g}_t}^2]. \tag{7}$$

This way, we obtain the corrected 2^{nd} momentum, as shown in Step.5 of Algorithm. 2 (left side of *Eq*.7), which is composed of two parts (right side of *Eq*.7): the "long-term" gradient momentum with a smooth decay rate $\frac{1}{1-\mu_2^t} \cdot \beta_2 \cdot v_X^{(i-1)}$, and the true expected squared gradient $\mathbb{E}[{\hat{g}_t}^2]$.

We utilize 3 new hyper-parameters in the inter-batch algorithm, *i.e.* β_1, β_2, and β_3, where β_1 and β_2 control the weights of "long-term" momentums from previous batches, and β_3 decides the decay rate of the perceptual constraint. In our implementation, the default settings are $\beta_1 = 0.10$, $\beta_2 = 0.01$, $\beta_3 = 0.60$. For selecting the good settings, we test these hyper-parameters by line-searching on 2 validation datasets, *i.e.* Cityspaces and LSUN'17.

Table 1: Comparison under *black-box* setting. Fooling ability is measured by performance drop: *CC*↑ for LSUN'17, *MSE*↓ for other datasets.

Datasets (original performance)	Cityspaces (*MSE*=0.0139)			Facades (*MSE*=0.0521)			Google Satellite (*MSE*=0.0255)			LSUN'17 (*CC*=0.7748)		
Target *black-box* model	pix2pix U-Net			Global pix2pixHD			Local pix2pixHD			SALICON		
Number of mini-batches	1	3	5	1	3	5	1	3	5	1	3	5
Percep. cons. ($\mathbf{L_1}$ norm)	$1.0e^{-2}$	$2.0e^{-2}$	$2.4e^{-2}$	$1.2e^{-2}$	$2.4e^{-2}$	$2.8e^{-2}$	$1.0e^{-2}$	$2.0e^{-2}$	$2.4e^{-2}$	$3.4e^{-2}$	$6.6e^{-2}$	$7.8e^{-2}$
Random noise	+0.0002	+0.0008	+0.0011	+0.0003	+0.0010	+0.0013	+0.0002	+0.0006	+0.0011	-0.0002	-0.0002	-0.0003
Ensemble Attack using PGD	+0.0108	+0.0169	+0.0174	+0.0104	+0.0133	+0.0166	+0.0074	+0.0083	+0.0099	-0.0022	-0.0264	-0.0511
Ensemble Attack using C&W	+0.0113	+0.0170	+0.0173	+0.0097	+0.0131	+0.0168	+0.0068	+0.0087	+0.0096	-0.0763	-0.2117	-0.2452
Ensemble Attack using MIM	+0.0116	+0.0193	+0.0227	+0.0125	+0.0162	+0.0178	+0.0093	+0.0099	+0.0113	-0.0771	-0.2417	-0.2880
Liu's Ensemble Attack	+0.0118	+0.0194	+0.0230	+0.0129	+0.0165	+0.0184	+0.0098	+0.0105	+0.0116	-0.0780	-0.2533	-0.2941
Proposed *SMBEA*	**+0.0148**	**+0.0213**	**+0.0264**	**+0.0155**	**+0.0230**	**+0.0275**	**+0.0108**	**+0.0137**	**+0.0145**	**-0.0871**	**-0.3017**	**-0.4180**

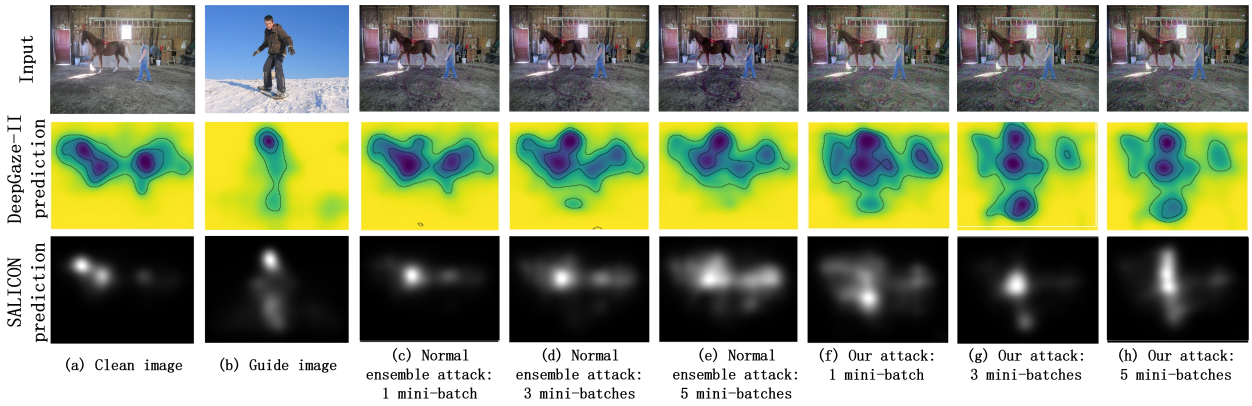

Figure 3: Attacking real-world applications. With the increase of batch number, our attack fools two online *black-box* saliency prediction systems, *i.e.* DeepGaze-II and SALICON. However, normal ensemble attack based on PGD (Madry et al. 2018) fails to fool these models.

Experiments

The selection of loss functions

In *Eqs*.1-2, we provide a general paradigm for computing objective cost functions. For different tasks, we select different task-specific loss metrics to reach a better attack performance. Specifically, for image-translation, we use a linear combination of *Mean Absolute Error* (*MAE*), negative *Pearson's Linear Correlation Coefficient* (*CC*), and VGG loss (Dosovitskiy and Brox 2016) as the *output space* fooling ability loss $\mathcal{L}_1$. For saliency prediction, we use a linear combination of *Kullback-Leibler divergence* (*KL*), *MAE*, and negative *CC* as $\mathcal{L}_1$. We use *KL* as *feature-space* fooling ability loss $\mathcal{L}_3$, because the intermediate feature maps of source models are normalized by *softmax* as the distributions. The averaged $\mathbf{L_1}$ norm serves as perceptual loss $\mathcal{L}_2$.

Datasets and evaluation protocol

To explore the generalization ability of *SMBEA*, we conduct experiments on 4 pixel-to-pixel vision datasets, *i.e.* Cityspaces (Cordts and Omran 2016), Facades (Tylecek 2013), Google Satellites, and LSUN'17 (Jiang et al. 2015). For Cityspaces, we select 1000 "*Semantic label* & *Real photo*" pairs as test set. For Facades, we select 400 "*Architectural label* & *Real photo*" pairs. For Google Satellites, we select 1000 "*Google Map* & *Satellite Image*" pairs. For LSUN'17, we select 1000 "*Real Photo* & *Saliency Map*" pairs.

For fair comparison, we adopt the performance drop to measure the fooling ability, *i.e.* the difference between the performances on clean images and on adversarial examples. The stronger the attack, the bigger the performance drop. For image translation tasks, we use the *Mean Squared Error* (*MSE*) to measure the performance drop. For the saliency prediction task, we use the *Pearson's Linear Correlation Coefficient* (*CC*) to measure the performance drop. For measuring the perceptibility, we use the averaged $\mathbf{L_1}$ norm. In this paper, the images are normalized to be in the range [0, 1], and the $\mathbf{L_1}$ norm is averaged by the number of pixels.

Source models

For the saliency prediction task, we adopt 16 state-of-the-art deep saliency models as the raw source models. However, we wish to use additional source models to explore the upper-bound of our attack. Thus, we design an augmentation strategy to enlarge the number of models. Specifically, we replace the standard convolution (adopted in most of the current CNN models) in the original architecture with two new convolutions, *i.e.* deformable convolution kernel (Dai et al. 2017) and dilated convolution kernel (Yu and Koltun 2016). By doing so, for each raw source model, we obtain two new variants that have diverse architectures, without causing obvious performance drop. This way, we obtain 48 source models in total. We use the same model augmentation strategy for other tasks.

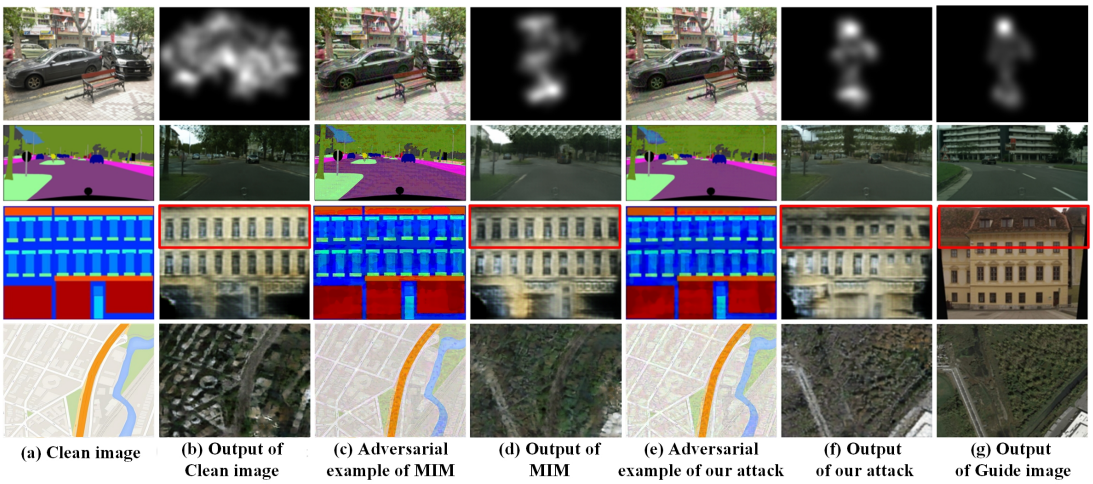

Figure 4: Qualitative results of *black-box* MIM (Dong et al. 2018) ensemble attack and our attack over multiple pixel-to-pixel vision tasks.

Table 2: Evading defense: attack performance comparison against the adversarially trained *black-box* models. LSUN'17 is the test set.

Target model	Attack	No. of Batch: 3	No. of Batch: 5	No. of Batch: 7
GazeGAN	I-FGSM	-0.0125	-0.0177	-0.0206
GazeGAN	MIM	-0.1315	-0.1582	-0.2063
GazeGAN	*SMBEA*	**-0.2190**	**-0.2731**	**-0.3255**
SAM-ResNet	I-FGSM	-0.0164	-0.0238	-0.0295
SAM-ResNet	MIM	-0.1622	-0.1900	-0.2258
SAM-ResNet	*SMBEA*	**-0.2317**	**-0.2996**	**-0.3484**

Comparison

In Table. 1, we compare our method with other ensemble attacks in the *black-box* setting. These ensemble attacks adopt state-of-the-art gradient back-propagation attack algorithms. For fair comparison, we use the same perceptual constraint for different competing methods. We can see that our attack achieves the best performance over different datasets.

It was verified that injecting adversarial examples into training set will increase the robustness of deep networks against attacks (Goodfellow et al. 2015). Currently, this adversarial training strategy is the most efficient defense method. In Table. 2, we compare our method with other attacks against the adversarially trained *black-box* models. Specifically, we keep two adversarially trained models as the hold-out target models, and use the rest source models to craft the adversarial examples. We can see that the adversarially trained models can not defend our attack effectively.

We further compare our attack with the normal ensemble attack based on PGD (Madry et al. 2018) algorithm against two online *black-box* saliency prediction models, as shown in Fig. 3. We notice that, by increasing the number of batches, our method misleads the model prediction towards the guide image, while the normal attack fails to fool these models. Besides, we also compare our method with the *black-box* ensemble attack based on MIM (Dong et al. 2018) algorithm from a qualitative perspective, as shown in Fig. 4. For fair comparison, we apply the same perceptual constraint to different competing methods. We observe that, the outputs of our attack are more similar to the outputs of the guide images, demonstrating better fooling ability.

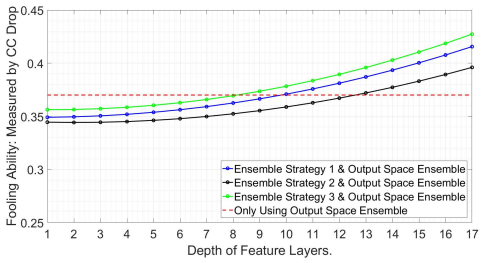

Figure 5: The relationship between fooling ability (against the source models) and the depth of feature layers. We compare different ensemble strategies when fusing 4 source models including GazeGAN (Che et al. 2019), Globalpix2pix (Wang et al. 2018), SAM-ResNet (Cornia et al. 2018), and SalGAN (Pan et al. 2017).

Ablation studies

In Fig. 5, we explore the relationship between the fooling ability and the depth of feature layers. We introduce 3 new *feature space* ensemble strategies, as shown in Fig. 2. Here we explain how to select good feature layers to conduct an efficient attack. Experiments indicate that the deeper layers obtain better fooling ability. Besides, the proposed *feature space* ensemble strategies further improve the performance, and the 3_{rd} ensemble strategy obtains the best performance.

Next, we explore the contributions of intra-batch momentums and inter-batch "long-term" momentums, as shown in Fig. 6. We notice that, both intra-batch "short-term" momentums and inter-batch "long-term" gradient memories increase the transferability in the *black-box* setting.

Conclusion

In this paper, we propose a novel *black-box* attack. Our attack divides a large number of pre-trained source models in-

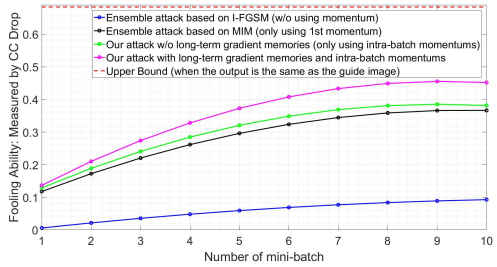

Figure 6: Ablation study of our attack. We use the online *black-box* SALICON model as the target model. With the increase of batch number, the perceptual constraint increases (as shown in Step.1 of Algorithm.2). For fair comparison, in the same batch, different competing attacks adopt the same perceptual constraint.

to several batches. For each batch, we introduce 3 *feature-space* ensemble strategies for improving intra-batch transferability. Besides, we propose a new algorithm that utilizes the "long-term" gradient memories. The long-term gradient memories preserve the learned adversarial information and improve inter-batch transferability. Our attack achieves the best performance over multiple pixel-to-pixel datasets, and fools two online *black-box* applications in the real world. We share our code with the community to promote the research on adversarial attack and defense over pixel-to-pixel tasks.

Acknowledgement

This work was supported in part by the National Science Foundation of China under Grant 61831015, Grant 61771305, and Grant 61927809.

References

Alletto, S.; Palazzi, A.; Solera, F.; Calderara, S.; and Cucchiara, R. 2016. Dr(eye)ve: a dataset for attention-based tasks with applications to autonomous and assisted driving. In *CVPRw*.

Carlini, N., and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In *SP*.

Che, Z.; Borji, A.; Zhai, G.; Min, X.; Guo, G.; and Callet, P. 2019. How is gaze influenced by image transformations? dataset and model. *TIP*.

Cordts, M., and Omran, M. 2016. The cityscapes dataset for semantic urban scene understanding. In *CVPR*.

Cornia, M.; Baraldi, L.; Serra, G.; and et al. 2018. Predicting human eye fixations via an lstm-based saliency attentive model. *TIP*.

Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; and Wei, Y. 2017. Deformable convolutional networks. In *ICCV*.

Dong, Y.; Liao, F.; Pang, T.; Su, H.; Zhu, J.; Hu, X.; and Li, J. 2018. Boosting adversarial attacks with momentum. In *CVPR*.

Dong, Y.; Su, H.; Wu, B.; Li, Z.; Liu, W.; and Zhang, T. 2019. Efficient decision-based black-box adversarial attacks on face recognition. In *CVPR*.

Dosovitskiy, A., and Brox, T. 2016. Generating images with perceptual similarity metrics based on deep networks. In *NeurIPS*.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*.

Goodfellow, I.; Shlens, J.; Szegedy, C.; and Goodfellow, I. 2015. Explaining and harnessing adversarial examples. In *ICLR*.

Huang, X.; Shen, C.; Boix, X.; and Zhao, Q. 2017. Online saliency prediction system SALICON. http://salicon.net/demo/.

Jiang, M.; Huang, S.; Duan, J.; and Zhao, Q. 2015. Salicon: Saliency in context. In *CVPR*.

Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Kummerer, M. 2017. Online saliency prediction system Deepgaze-II. https://deepgaze.bethgelab.org/.

Kurakin, A.; Goodfellow, I.; Bengio, S.; and Bengio, S. 2016. Adversarial examples in the physical world. In *ICLRw*.

Liu, Y.; Chen, X.; Liu, C.; and Song, D. 2017. Delving into transferable adversarial examples and black-box attacks. In *ICLR*.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards deep learning models resistant to adversarial attacks. In *ICLR*.

Mopuri, K. R.; Ganeshan, A.; and Radhakrishnan, V. B. 2018. Generalizable data-free objective for crafting universal adversarial perturbations. *TPAMI*.

Pan, J.; Canton, C.; McGuinness, K.; Connor, N.; Torres, J.; Sayrol, E.; and Nieto, X. 2017. Salgan: Visual saliency prediction with generative adversarial networks. In *CoRR:1701.01081*.

Papernot, N.; Goodfollow, I.; Sheatsley, R.; Feinman, R.; and McDaniel, P. 2016a. cleverhans v2. 0.0: an adversarial machine learning library. In *arXiv preprint*.

Papernot, N.; McDaniel, P.; Wu, X.; Jha, S.; and Swami, A. 2016b. Distillation as a defense to adversarial perturbations against deep neural networks. In *SP*.

Papernot, N. 2017. Practical black-box attacks against machine learning. In *ACM ACCCS*.

Qian, N. 1999. On the momentum term in gradient descent learning algorithms. *Neural networks*.

Sharif, M.; Bhagavatula, S.; Bauer, L.; and Reiter, M. K. 2016. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *ACM SIGSAC*.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I. J.; and Fergus, R. 2014. Intriguing properties of neural networks. In *ICLR*.

Tieleman, T., and Hinton, G. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*.

Tylecek, R. 2013. Spatial pattern templates for recognition of objects with regular structure. In *GCPR*.

Wang, T. C.; Liu, M. Y.; Zhu, J. Y.; Tao, A.; Kautz, J.; and Catanzaro, B. 2018. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*.

Wei, X.; Liang, S.; Chen, N.; and Cao, X. 2019. Transferable adversarial attacks for image and video object detection. In *IJCAI*.

Xie, C.; Wang, J.; Zhang, Z.; Zhou, Y.; and an A. Yuille, L. X. 2017. Adversarial examples for semantic segmentation and object detection. In *ICCV*.

Yang, S., and Hsu, Y. 2017. Full speed region sensorless drive of permanent-magnet machine combining saliency-based and back-emf-based drive. *TIE*.

Yu, F., and Koltun, V. 2016. Multi-scale context aggregation by dilated convolutions. In *ICLR*.

Zhao, Z.; Dheeru, D.; and Sameer, S. 2018. Generating natural adversarial examples. In *ICLR*.