ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks without Training Substitute Models

Pin-Yu Chen*
AI Foundations Group
IBM T. J. Watson Research Center
Yorktown Heights, NY
pin-yu.chen@ibm.com

Huan Zhang*†
University of California, Davis
Davis, CA
ecezhang@ucdavis.edu

Yash Sharma
IBM T. J. Watson Research Center
Yorktown Heights, NY
Yash.Sharma3@ibm.com

Jinfeng Yi
AI Foundations Group
IBM T. J. Watson Research Center
Yorktown Heights, NY
jinfengy@us.ibm.com

Cho-Jui Hsieh
University of California, Davis
Davis, CA
chohsieh@ucdavis.edu

ABSTRACT

Deep neural networks (DNNs) are one of the most prominent technologies of our time, as they achieve state-of-the-art performance in many machine learning tasks, including but not limited to image classification, text mining, and speech processing. However, recent research on DNNs has indicated ever-increasing concern on the robustness to adversarial examples, especially for security-critical tasks such as traffic sign identification for autonomous driving. Studies have unveiled the vulnerability of a well-trained DNN by demonstrating the ability of generating barely noticeable (to both human and machines) adversarial images that lead to misclassification. Furthermore, researchers have shown that these adversarial images are highly transferable by simply training and attacking a substitute model built upon the target model, known as a black-box attack to DNNs.

Similar to the setting of training substitute models, in this paper we propose an effective black-box attack that also only has access to the input (images) and the output (confidence scores) of a targeted DNN. However, different from leveraging attack transferability from substitute models, we propose **z**eroth **o**rder **o**ptimization (*ZOO*) based attacks to directly estimate the gradients of the targeted DNN for generating adversarial examples. We use zeroth order stochastic coordinate descent along with dimension reduction, hierarchical attack and importance sampling techniques to efficiently attack black-box models. By exploiting zeroth order optimization, improved attacks to the targeted DNN can be accomplished, sparing the need for training substitute models and avoiding the loss in attack transferability. Experimental results on MNIST, CIFAR10 and ImageNet show that the proposed ZOO attack is as effective as the state-of-the-art white-box attack (e.g., Carlini and Wagner's attack) and significantly outperforms existing black-box attacks via substitute models.

KEYWORDS

adversarial learning; black-box attack; deep learning; neural network; substitute model

1 INTRODUCTION

The renaissance of artificial intelligence (AI) in the past few years roots in the advancement of deep neural networks (DNNs). Recently, DNNs have become an essential element and a core technique for existing and emerging AI research, as DNNs have achieved state-of-the-art performance and demonstrated fundamental breakthroughs in many machine learning tasks that were once believed to be challenging [24]. Examples include computer vision, image classification, machine translation, and speech processing, to name a few.

Despite the success of DNNs, recent studies have identified that DNNs can be vulnerable to adversarial examples - a slightly modified image can be easily generated and fool a well-trained image classifier based on DNNs with high confidence [12, 40]. Consequently, the inherent weakness of lacking robustness to adversarial examples for DNNs brings out security concerns, especially for mission-critical applications which require strong reliability, including traffic sign identification for autonomous driving and malware prevention [9, 16, 17], among others.

Preliminary studies on the robustness of DNNs focused on an "open-box" (white-box) setting - they assume model transparency that allows full control and access to a targeted DNN for sensitivity analysis. By granting the ability of performing back propagation, a technique that enables gradient computation of the output with respect to the input of the targeted DNN, many standard algorithms

such as gradient descent can be used to attack the DNN. In image classification, back propagation specifies the effect of changing pixel values on the confidence scores for image label prediction. Unfortunately, most real world systems do not release their internal configurations (including network structure and weights), so open-box attacks cannot be used in practice.

Throughout this paper, we consider a practical "black-box" attack setting where one can access the input and output of a DNN but not the internal configurations. In particular, we focus on the use case where a targeted DNN is an image classifier trained by a convolutional neural network (CNN), which takes an image as an input and produces a confidence score for each class as an output. Due to application popularity and security implications, image classification based on CNNs is currently a major focus and a critical use case for studying the robustness of DNNs.

We consider two types of black-box attacks in this paper. Given a benign example with correct labeling, an *untargeted* attack refers to crafting an adversarial example leading to misclassification, whereas a *targeted* attack refers to modifying the example in order to be classified as a desired class. The effectiveness of our proposed black-box attack (which we call *ZOO*) is illustrated in Figure 1. The crafted adversarial examples from our attack not only successfully mislead the targeted DNN but also deceive human perception as the injected adversarial noise is barely noticeable. In an attacker's foothold, an adversarial example should be made as indistinguishable from the original example as possible in order to deceive a targeted DNN (and sometimes human perception as well). However, the best metric for evaluating the similarity between a benign example and a corresponding adversarial example is still an open question and may vary in different contexts.

In what follows, we summarize recent works on generating and defending adversarial examples for DNNs, and specify the "black-box" setting of training substitute models for adversarial attacks.

1.1 Adversarial attacks and transferability

We summarize four principal open-box methods developed for attacking image classification trained on DNNs as follows.
• **Fast gradient sign method (FGSM) [12]:** Originated from an L_∞ constraint on the maximal distortion, FGSM uses the sign of the gradient from the back propagation on a targeted DNN to generate admissible adversarial examples. FGSM has become a popular baseline algorithm for improved adversarial example generation [21, 22], and it can be viewed as an attack framework based on first-order projected gradient descent [27].
• **Jacobian-based Saliency Map Attack (JSMA) [36]:** By constructing a Jacobian-based saliency map for characterizing the input-output relation of a targeted DNN, JSMA can be viewed as a greedy attack algorithm that iteratively modifies the most significant pixel based on the saliency map for crafting adversarial examples. Each iteration, JSMA recomputes the saliency map and uses the derivative of the DNN with respect to the input image as an indicator of modification for adversarial attacks. In addition to image classification, JSMA has been applied to other machine learning tasks such as malware classification [14], and other DNN architectures such as recurrent neural networks (RNNs) [37].
• **DeepFool [31]:** Inspired from linear classification models and

(a) a ZOO black-box targeted attack example
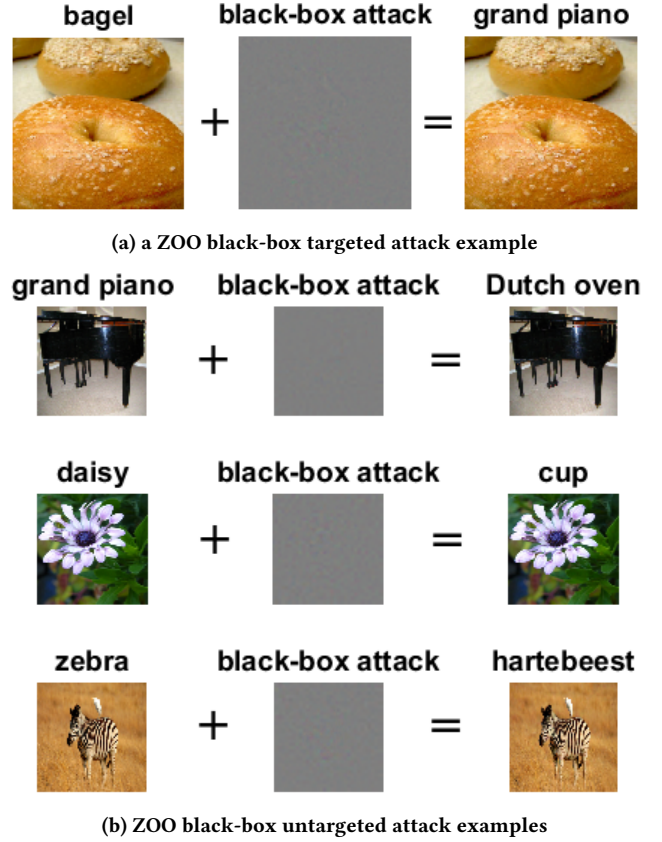

(b) ZOO black-box untargeted attack examples

Figure 1: Visual illustration of our proposed black-box attack (ZOO) to sampled images from ImageNet. The columns from left to right are original images with correct labels, additive adversarial noises from our attack, and crafted adversarial images with misclassified labels.

the fact that the corresponding separating hyperplanes indicate the decision boundaries of each class, DeepFool is an untargeted attack algorithm that aims to find the least distortion (in the sense of Euclidean distance) leading to misclassification, by projecting an image to the closest separating hyperplane. In particular, an approximate attack algorithm is proposed for DNNs in order to tackle the inherit nonlinearity for classification [40].
• **Carlini & Wagner (C&W) Attack [8]:** The adversarial attack proposed by Carlini and Wagner is by far one of the strongest attacks. They formulate targeted adversarial attacks as an optimization problem, take advantage of the internal configurations of a targeted DNN for attack guidance, and use the L_2 norm (i.e., Euclidean distance) to quantify the difference between the adversarial and the original examples. In particular, the representation in the logit layer (the layer prior to the final fully connected layer as illustrated in Figure 2) is used as an indicator of attack effectiveness. Consequently, the C&W attack can be viewed as a gradient-descent based targeted adversarial attack driven by the representation of the logit layer of a targeted DNN and the L_2 distortion. The formulation of the C&W attack will be discussed in detail in Section 3. Furthermore, Carlini and Wagner also showed that their attack can
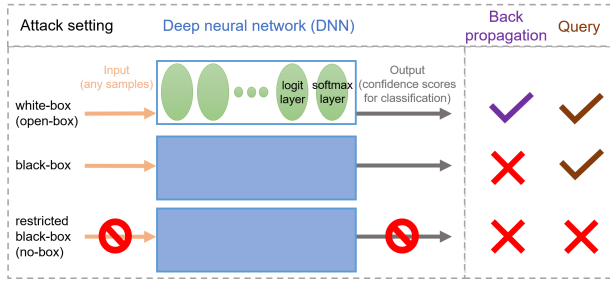
Figure 2: Taxonomy of adversarial attacks to deep neural networks (DNNs). "Back propagation" means an attacker can access the internal configurations in DNNs (e.g., performing gradient descent), and "Query" means an attacker can input any sample and observe the corresponding output.

successfully bypass 10 different detections methods designed for detecting adversarial examples [7].

• **Transferability:** In the context of adversarial attacks, *transferability* means that the adversarial examples generated from one model are also very likely to be misclassified by another model. In particular, the aforementioned adversarial attacks have demonstrated that their adversarial examples are highly transferable from one DNN at hand to the targeted DNN. One possible explanation of inherent attack transferability for DNNs lies in the findings that DNNs commonly have overwhelming generalization power and local linearity for feature extraction [40]. Notably, the transferability of adversarial attacks brings about security concerns for machine learning applications based on DNNs, as malicious examples may be easily crafted even when the exact parameters of a targeted DNN are absent. More interestingly, the authors in [29] have shown that a carefully crafted universal perturbation to a set of natural images can lead to misclassification of all considered images with high probability, suggesting the possibility of attack transferability from one image to another. Further analysis and justification of a universal perturbation is given in [30].

1.2 Black-box attacks and substitute models

While the definition of an open-box (white-box) attack to DNNs is clear and precise - having complete knowledge and allowing full access to a targeted DNN, the definition of a "black-box" attack to DNNs may vary in terms of the capability of an attacker. In an attacker's perspective, a black-box attack may refer to the most challenging case where only benign images and their class labels are given, but the targeted DNN is completely unknown, and one is prohibited from querying any information from the targeted classifier for adversarial attacks. This restricted setting, which we call a "no-box" attack setting, excludes the principal adversarial attacks introduced in Section 1.1, as they all require certain knowledge and back propagation from the targeted DNN. Consequently, under this no-box setting the research focus is mainly on the attack transferability from one self-trained DNN to a targeted but completely access-prohibited DNN.

On the other hand, in many scenarios an attacker does have the privilege to query a targeted DNN in order to obtain useful information for crafting adversarial examples. For instance, a mobile app or

a computer software featuring image classification (mostly likely trained by DNNs) allows an attacker to input any image at will and acquire classification results, such as the confidence scores or ranking for classification. An attacker can then leverage the acquired classification results to design more effective adversarial examples to fool the targeted classifier. In this setting, back propagation for gradient computation of the targeted DNN is still prohibited, as back propagation requires the knowledge of internal configurations of a DNN that are not available in the black-box setting. However, the adversarial query process can be iterated multiple times until an attacker finds a satisfactory adversarial example. For instance, the authors in [26] have demonstrated a successful black-box adversarial attack to Clarifai.com, which is a black-box image classification system.

Due to its feasibility, the case where an attacker can have free access to the input and output of a targeted DNN while still being prohibited from performing back propagation on the targeted DNN has been called a practical black-box attack setting for DNNs [8, 16, 17, 26, 34, 35]. For the rest of this paper, we also refer a black-box adversarial attack to this setting. For illustration, the attack settings and their limitations are summarized in Figure 2. It is worth noting that under this black-box setting, existing attacking approaches tend to make use of the power of free query to train a *substitute model* [17, 34, 35], which is a representative substitute of the targeted DNN. The substitute model can then be attacked using any white-box attack techniques, and the generated adversarial images are used to attack the target DNN. The primary advantage of training a substitute model is its total transparency to an attacker, and hence essential attack procedures for DNNs, such as back propagation for gradient computation, can be implemented on the substitute model for crafting adversarial examples. Moreover, since the substitute model is representative of a targeted DNN in terms of its classification rules, adversarial attacks to a substitute model are expected to be similar to attacking the corresponding targeted DNN. In other words, adversarial examples crafted from a substitute model can be highly transferable to the targeted DNN given the ability of querying the targeted DNN at will.

1.3 Defending adversarial attacks

One common observation from the development of security-related research is that attack and defense often come hand-in-hand, and one's improvement depends on the other's progress. Similarly, in the context of robustness of DNNs, more effective adversarial attacks are often driven by improved defenses, and vice versa. There has been a vast amount of literature on enhancing the robustness of DNNs. Here we focus on the defense methods that have been shown to be effective in tackling (a subset of) the adversarial attacks introduced in Section 1.1 while maintaining similar classification performance for the benign examples. Based on the defense techniques, we categorize the defense methods proposed for enhancing the robustness of DNNs to adversarial examples as follows.

• **Detection-based defense:** Detection-based approaches aim to differentiate an adversarial example from a set of benign examples using statistical tests or out-of-sample analysis. Interested readers can refer to recent works in [10, 13, 18, 28, 42, 43] and references therein for details. In particular, *feature squeezing* is

shown to be effective in detecting adversarial examples by projecting an image to a confined subspace (e.g., reducing color depth of a pixel) to alleviate the exploits from adversarial attacks [42, 43]. The success of detection-based approaches heavily relies on the assumption that the distributions of adversarial and benign examples are fundamentally distinct. However, Carlini and Wagner recently demonstrated that their attack (C&W attack) can bypass 10 different detection algorithms designed for detecting adversarial examples [7], which challenges the fundamental assumption of detection-based approaches as the results suggest that the distributions of their adversarial examples and the benign examples are nearly indistinguishable.

• **Gradient and representation masking:** As the use of gradients via back propagation on DNNs has been shown to be crucial to crafting adversarial examples, one natural defense mechanism is to hide the gradient information while training a DNN, known as *gradient masking*. A typical example of gradient masking is the defense distillation proposed in [38]. The authors proposed to retrain a DNN using distillation [15] based on the original confidence scores for classification (also known as soft labels) and introduced the concept of "temperature" in the softmax step for gradient masking. An extended version has been proposed to enhance its defense performance by incorporating model-agnostic uncertainty into retraining [33]. Although the C&W attack has shown to be able to break defensive distillation [8], it is still considered as a baseline model for defending adversarial attacks. Another defense technique, which we call *representation masking*, is inspired by the finding that in the C&W attack the logit layer representation in DNNs is useful for adversarial attacks. As a result, representation masking aims to replace the internal representations in DNNs (usually the last few layers) with robust representations to alleviate adversarial attacks. For example, the authors in [6] proposed to integrate DNNs with Gaussian processes and RBF kernels for enhanced robustness.

• **Adversarial training:** The rationale behind adversarial training is that DNNs are expected to be less sensitive to perturbations (either adversarial or random) to the examples if these adversarial examples are jointly used to stabilize training, known as the *data augmentation* method. Different data augmentation methods have been proposed to improve the robustness of DNNs. Interested readers can refer to the recent works in [19, 27, 41, 44, 45] and the references therein. Notably, the defense model proposed in [27] showed promising results against adversarial attacks, including the FGSM and the C&W attack. The authors formulated defense in DNNs as a robust optimization problem, where the robustness is improved by iterative adversarial data augmentation and retraining. The results suggest that a DNN can be made robust at the price of increased network capacity (i.e., more model parameters), in order to stabilize training and alleviate the effect of adversarial examples.

1.4 Black-box attack using zeroth order optimization: benefits and challenges

Zeroth order methods are derivative-free optimization methods, where only the zeroth order oracle (the objective function value $f(\mathbf{x})$ at any $\mathbf{x}$) is needed during optimization process. By evaluating the objective function values at two very close points $f(\mathbf{x} + h\mathbf{v})$ and $f(\mathbf{x} - h\mathbf{v})$ with a small h, a proper gradient along the direction

vector $\mathbf{v}$ can be estimated. Then, classical optimization algorithms like gradient descent or coordinate descent can be applied using the estimated gradients. The convergence of these zeroth order methods has been proved in optimization literature [11, 25, 32], and under mild assumptions (smoothness and Lipschitzian gradient) they can converge to a stationary point with an extra error term which is related to gradient estimation and vanishes when $h \to 0$.

Our proposed black-box attack to DNNs in Section 3 is cast as an optimization problem. It exploits the techniques from zeroth order optimization and therefore spares the need of training a substitute model for deploying adversarial attacks. Although it is intuitive to use zeroth order methods to attack a black-box DNN model, applying it naively can be impractical for large models. For example, the Inception-v3 network [39] takes input images with a size of $299 \times 299 \times 3$, and thus has $p = 268, 203$ variables (pixels) to optimize. To evaluate the estimated gradient of each pixel, we need to evaluate the model twice. To just obtain the estimated gradients of all pixels, $2p = 536, 406$ evaluations are needed. For a model as large as Inception-v3, each evaluation can take tens of milliseconds on a single GPU, thus it is very expensive to even evaluate all gradients once. For targeted attacks, sometimes we need to run an iterative gradient descent with hundreds of iterations to generate an adversarial image, and it can be forbiddingly expensive to use zeroth order method in this case.

In the scenario of attacking black-box DNNs, especially when the image size is large (the variable to be optimized has a large number of coordinates), a single step of gradient descent can be very slow and inefficient, because it requires estimating the gradients of all coordinates to make a single update. Instead, we propose to use a coordinate descent method to iteratively optimize each coordinate (or a small batch of coordinates). By doing so, we can accelerate the attack process by efficiently updating coordinates after only a few gradient evaluations. This idea is similar to DNN training for large datasets, where we usually apply stochastic gradient descent using only a small subset of training examples for efficient updates, instead of computing the full gradient using all examples to make a single update. Using coordinate descent, we update coordinates by small batches, instead of updating all coordinates in a single update as in gradient descent. Moreover, this allows us to further improve the efficiency of our algorithm by using carefully designed sampling strategy to optimize important pixels first. We will discuss the detailed algorithm in Section 3.

1.5 Contributions

We refer to the proposed attack method as black-box attacks using **z**eroth **o**rder **o**ptimization, or **ZOO** for short. Below we summarize our main contributions:

• We show that a coordinate descent based method using only the zeroth order oracle (without gradient information) can effectively attack black-box DNNs. Comparing to the substitute model based black-box attack [35], our method significantly increases the success rate for adversarial attacks, and attains comparable performance to the state-of-the-art white-box attack (C&W attack).

• In order to speed up the computational time and reduce number of queries for our black-box attacks to large-scale DNNs, we propose several techniques including attack-space dimension reduction, hierarchical attacks and importance sampling.

• In addition to datasets of small image size (MNIST and CIFAR10), we demonstrate the applicability of our black-box attack model to a large DNN - the Inception-v3 model trained on ImageNet. Our attack is capable of crafting a successful adversarial image within a reasonable time, whereas the substitute model based black-box attack in [35] only shows success in small networks trained on MNIST and is hardly scalable to the case of ImageNet.

2 RELATED WORK

The study of adversarial attacks roots in the need for understanding the robustness of state-of-the-art machine learning models [1, 2]. For instance, Biggio et el. proposed an effective attack to sabotaging the performance (test accuracy) of support vector machines (SVMs) by intelligently injecting adversarial examples to the training dataset [5]. Gradient based evasion attacks to SVMs and multi-layer perceptrons are discussed in [4]. Given the popularity and success of classification models trained by DNNs, in recent years there has been a surge in interest toward understanding the robustness of DNNs. A comprehensive overview of adversarial attacks and defenses for DNNs is given in Section 1.

Here we focus on related work on the black-box adversarial attack setting for DNNs. As illustrated in Figure 2, the black-box setting allows free query from a targeted DNN but prohibits any access to internal configurations (e.g., back propagation), which fits well to the scenario of publicly accessible machine learning services (e.g., mobile apps, image classification service providers, and computer vision packages). Under this black-box setting, the methodology of current attacks concentrates on training a substitute model and using it as a surrogate for adversarial attacks [8, 16, 17, 26, 34, 35]. In other words, a black-box attack is made possible by deploying a white-box attack to the substitute model. Therefore, the effectiveness of such black-box adversarial attacks heavily depends on the attack transferability from the substitute model to the target model. Different from the existing approaches, we propose a black-box attack via zeroth order optimization techniques. More importantly, the proposed attack spares the need for training substitute models by enabling a "pseudo back propagation" on the target model. Consequently, our attack can be viewed "as if it was" a white-box attack to the target model, and its advantage over current black-box methods can be explained by the fact that it avoids any potential loss in transferability from a substitute model. The performance comparison between the existing methods and our proposed black-box attack will be discussed in Section 4.

In principle, our black-box attack technique based on zeroth order optimization is a general framework that can be applied to any white-box attacks requiring back propagation on the targeted DNN. We note that all the effective adversarial attacks discussed in Section 1.1 have such a requirement, as back propagation on a targeted DNN provides invaluable information for an attacker. Analogously, one can view an attacker as an optimizer and an adversarial attack as an objective function to be optimized. Back propagation provides first-order evaluation (i.e., gradients) of the objective function for the optimizer for efficient optimization. For the purpose of demonstration, in this paper we proposed a black-box attack based on the formulation of the C&W attack, since the (white-box) C&W attack has significantly outperformed the other attacks discussed in Section 1.1 in terms of the quality (distortion)

of the crafted adversarial examples and attack transferability [8]. Experimental results in Section 4 show that our black-box version is as effective as the original C&W attack but at the cost of longer processing time for implementing pseudo back propagation. We also compare our black-box attack with the black-box attack via substitute models in [35], which trains a substitute model as an attack surrogate based on Jacobian saliency map [36]. Experimental results in Section 4 show that our attack significantly outperforms [35], which can be explained by the fact that our attack inherits the effectiveness of the state-of-the-art C&W attack, and also by the fact that zeroth order optimization allows direct adversarial attacks to a targeted DNN and hence our black-box attack does not suffer from any loss in attack transferability from a substitute model.

3 ZOO: A BLACK-BOX ATTACK WITHOUT TRAINING SUBSTITUTE MODELS

3.1 Notation for deep neural networks

As illustrated in Figure 2, since we consider the black-box attack setting where free query from a targeted DNN is allowed while accessing to internal states (e.g., performing back propagation) is prohibited, it suffices to use the notation $F(\mathbf{x})$ to denote a targeted DNN. Specifically, the DNN $F(\mathbf{x})$ takes an image $\mathbf{x} \in \mathbb{R}^p$ (a p-dimensional column vector) as an input and outputs a vector $F(\mathbf{x}) \in [0, 1]^K$ of confidence scores for each class, where K is the number of classes. The k-th entry $[F(\mathbf{x})]_k \in [0, 1]$ specifies the probability of classifying $\mathbf{x}$ as class k, and $\sum_{k=1}^{K} [F(\mathbf{x})]_k = 1$.

In principle, our proposed black-box attack via zeroth order optimization (ZOO) can be applied to non-DNN classifiers admitting the same input and output relationship. However, since DNNs achieved state-of-the-art classification accuracy in many image tasks, in this paper we focus on the capability of our black-box adversarial attack to DNNs.

3.2 Formulation of C&W attack

Our black-box attack is inspired by the formulation of the C&W attack [8], which is one of the strongest white-box adversarial attacks to DNNs at the time of our work. Given an image $\mathbf{x}_0$, let $\mathbf{x}$ denote the adversarial example of $\mathbf{x}_0$ with a targeted class label t toward misclassification. The C&W attack finds the adversarial example $\mathbf{x}$ by solving the following optimization problem:

$$\text{minimize}_{\mathbf{x}} \; \|\mathbf{x} - \mathbf{x}_0\|_2^2 + c \cdot f(\mathbf{x}, t) \qquad (1)$$
$$\text{subject to } \mathbf{x} \in [0, 1]^p,$$

where $\|\mathbf{v}\|_2 = \sqrt{\sum_{i=1}^{p} v_i^2}$ denotes the Euclidean norm (L_2 norm) of a vector $\mathbf{v} = [v_1, \ldots, v_p]^T$, and $c > 0$ is a regularization parameter.

The first term $\|\mathbf{x} - \mathbf{x}_0\|_2^2$ in (1) is the regularization used to enforce the similarity between the adversarial example $\mathbf{x}$ and the image $\mathbf{x}_0$ in terms of the Euclidean distance, since $\mathbf{x} - \mathbf{x}_0$ is the adversarial image perturbation of $\mathbf{x}$ relative to $\mathbf{x}_0$. The second term $c \cdot f(\mathbf{x}, t)$ in (1) is the loss function that reflects the level of unsuccessful adversarial attacks, and t is the target class. Carlini and Wagner compared several candidates for $f(\mathbf{x}, t)$ and suggested the following form for effective targeted attacks [8]:

$$f(\mathbf{x}, t) = \max\{\max_{i \neq t}[Z(\mathbf{x})]_i - [Z(\mathbf{x})]_t, -\kappa\}, \qquad (2)$$

where $Z(\mathbf{x}) \in \mathbb{R}^K$ is the logit layer representation (*logits*) in the DNN for $\mathbf{x}$ such that $[Z(\mathbf{x})]_k$ represents the predicted probability that $\mathbf{x}$ belongs to class k, and $\kappa \geq 0$ is a tuning parameter for attack transferability. Carlini and Wagner set $\kappa = 0$ for attacking a targeted DNN, and suggested large κ when performing transfer attacks. The rationale behind the use of the loss function in (2) can be explained by the softmax classification rule based on the logit layer representation; the output (confidence score) of a DNN $F(\mathbf{x})$ is determined by the softmax function:

$$[F(\mathbf{x})]_k = \frac{\exp([Z(\mathbf{x})]_k)}{\sum_{i=1}^{K} \exp([Z(\mathbf{x})]_i)}, \ \forall \ k \in \{1, \ldots, K\}. \tag{3}$$

Therefore, based on the softmax decision rule in (3), $\max_{i \neq t}[Z(\mathbf{x})]_i - [Z(\mathbf{x})]_t \leq 0$ implies that the adversarial example $\mathbf{x}$ attains the highest confidence score for class t and hence the targeted attack is successful. On the other hand, $\max_{i \neq t}[Z(\mathbf{x})]_i - [Z(\mathbf{x})]_t > 0$ implies that the targeted attack using $\mathbf{x}$ is unsuccessful. The role of κ ensures a constant gap between $[Z(\mathbf{x})]_t$ and $\max_{i \neq t}[Z(\mathbf{x})]_i$, which explains why setting large κ is effective in transfer attacks.

Finally, the box constraint $\mathbf{x} \in [0, 1]^p$ implies that the adversarial example has to be generated from the valid image space. In practice, every image can satisfy this box constraint by dividing each pixel value by the maximum attainable pixel value (e.g., 255). Carlini and Wagner remove the box constraint by replacing $\mathbf{x}$ with $\frac{1 + \tanh \mathbf{w}}{2}$, where $\mathbf{w} \in \mathbb{R}^p$. By using this change-of-variable, the optimization problem in (1) becomes an unconstrained minimization problem with $\mathbf{w}$ as an optimizer, and typical optimization tools for DNNs (i.e., back propagation) can be applied for solving the optimal $\mathbf{w}$ and obtain the corresponding adversarial example $\mathbf{x}$.

3.3 Proposed black-box attack via zeroth order stochastic coordinate descent

The attack formulation using (1) and (2) presumes a white-box attack because (i): the logit layer representation in (2) is an internal state information of a DNN; and (ii) back propagation on the targeted DNN is required for solving (1). We amend our attack to the black-box setting by proposing the following approaches: (i) modify the loss function $f(\mathbf{x}, t)$ in (1) such that it only depends on the output F of a DNN and the desired class label t; and (ii) compute an *approximate* gradient using a finite difference method instead of actual back propagation on the targeted DNN, and solve the optimization problem via zeroth order optimization. We elucidate these two approaches below.

• **Loss function $f(\mathbf{x}, t)$ based on F:** Inspired by (2), we propose a new hinge-like loss function based on the output F of a DNN, which is defined as

$$f(\mathbf{x}, t) = \max\{\max_{i \neq t} \log[F(\mathbf{x})]_i - \log[F(\mathbf{x})]_t, -\kappa\}, \tag{4}$$

where $\kappa \geq 0$ and $\log 0$ is defined as $-\infty$. We note that $\log(\cdot)$ is a monotonic function such that for any $x, y \geq 0$, $\log y \geq \log x$ if and only if $y \geq x$. This implies that $\max_{i \neq t} \log[F(\mathbf{x})]_i - \log[F(\mathbf{x})]_t \leq 0$ means $\mathbf{x}$ attains the highest confidence score for class t. We find that the log operator is essential to our black-box attack since very often a well-trained DNN yields a skewed probability distribution on its output $F(\mathbf{x})$ such that the confidence score of one class significantly dominates the confidence scores of the other classes. The use of the

log operator lessens the dominance effect while preserving the order of confidence scores due to monotonicity. Similar to (2), κ in (4) ensures a constant gap between $\max_{i \neq t} \log[F(\mathbf{x})]_i$ and $\log[F(\mathbf{x})]_t$.

For untargeted attacks, an adversarial attack is successful when $\mathbf{x}$ is classified as any class other than the original class label t_0. A similar loss function can be used (we drop the variable t for untargeted attacks):

$$f(\mathbf{x}) = \max\{\log[F(\mathbf{x})]_{t_0} - \max_{i \neq t_0} \log[F(\mathbf{x})]_i, -\kappa\}, \tag{5}$$

where t_0 is the original class label for $\mathbf{x}$, and $\max_{i \neq t_0} \log[F(\mathbf{x})]_i$ represents the most probable predicted class other than t_0.

• **Zeroth order optimization on the loss function:** We discuss our optimization techniques for any general function f used for attacks (the regularization term in (1) can be absorbed as a part of f). We use the symmetric difference quotient [23] to estimate the gradient $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_i}$ (defined as $\hat{g}_i$):

$$\hat{g}_i := \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_i} \approx \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x} - h\mathbf{e}_i)}{2h}, \tag{6}$$

where h is a small constant (we set $h = 0.0001$ in all our experiments) and $\mathbf{e}_i$ is a standard basis vector with only the i-th component as 1. The estimation error (not including the error introduced by limited numerical precision) is in the order of $O(h^2)$. Although numerical accuracy is a concern, accurately estimating the gradient is usually not necessary for successful adversarial attacks. One example is FGSM, which only requires the sign (rather than the exact value) of the gradient to find adversarial examples. Therefore, even if our zeroth order estimations may not be very accurate, they suffice to achieve very high success rates, as we will show in our experiments.

For any $\mathbf{x} \in \mathbb{R}^p$, we need to evaluate the objective function $2p$ times to estimate gradients of all p coordinates. Interestingly, with just one more objective function evaluation, we can also obtain the coordinate-wise Hessian estimate (defined as $\hat{h}_i$):

$$\hat{h}_i := \frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}_{ii}^2} \approx \frac{f(\mathbf{x} + h\mathbf{e}_i) - 2f(\mathbf{x}) + f(\mathbf{x} - h\mathbf{e}_i)}{h^2}. \tag{7}$$

Remarkably, since $f(\mathbf{x})$ only needs to be evaluated once for all p coordinates, we can obtain the Hessian estimates without additional function evaluations.

It is worth noting that stochastic gradient descent and batch gradient descent are two most commonly used algorithms for training DNNs, and the C&W attack [8] also used gradient descent to attack a DNN in the white-box setting. Unfortunately, in the black-box setting, the network structure is unknown and the gradient computation via back propagation is prohibited. To tackle this problem, a naive solution is applying (6) to estimate gradient, which requires $2p$ objective function evaluations. However, this naive solution is too expensive in practice. Even for an input image size of $64 \times 64 \times 3$, one full gradient descent step requires 24, 576 evaluations, and typically hundreds of iterations may be needed until convergence. To resolve this issue, we propose the following coordinate-wise update, which only requires 2 function evaluations for each step.

• **Stochastic coordinate descent:** Coordinate descent methods have been extensively studied in optimization literature [3]. At each iteration, one variable (coordinate) is chosen randomly and is

Algorithm 1 Stochastic Coordinate Descent

1: **while** not converged **do**
2:     Randomly pick a coordinate $i \in \{1, \ldots, p\}$
3:     Compute an update δ^* by approximately minimizing
$$\arg\min_{\delta} f(\mathbf{x} + \delta \mathbf{e}_i)$$
4:     Update $\mathbf{x}_i \leftarrow \mathbf{x}_i + \delta^*$
5: **end while**

updated by approximately minimizing the objective function along that coordinate (see Algorithm 1 for details). The most challenging part in Algorithm 1 is to compute the best coordinate update in step 3. After estimating the gradient and Hessian for $\mathbf{x}_i$, we can use any first or second order method to approximately find the best δ. In first-order methods, we found that ADAM [20]'s update rule significantly outperforms vanilla gradient descent update and other variants in our experiments, so we propose to use a zeroth-order coordinate ADAM, as described in Algorithm 2. We also use Newton's method with both estimated gradient and Hessian to update the chosen coordinate, as proposed in Algorithm 3. Note that when Hessian is negative (indicating the objective function is concave along direction $\mathbf{x}_i$), we simply update $\mathbf{x}_i$ by its gradient. We will show the comparison of these two methods in Section 4. Experimental results suggest coordinate-wise ADAM is faster than Newton's method.

Algorithm 2 ZOO-ADAM: Zeroth Order Stochastic Coordinate Descent with Coordinate-wise ADAM

Require: Step size η, ADAM states $M \in \mathbb{R}^p, v \in \mathbb{R}^p, T \in \mathbb{Z}^p$, ADAM hyper-parameters $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$
1: $M \leftarrow \mathbf{0}, v \leftarrow \mathbf{0}, T \leftarrow \mathbf{0}$
2: **while** not converged **do**
3:     Randomly pick a coordinate $i \in \{1, \cdots, p\}$
4:     Estimate $\hat{g}_i$ using (6)
5:     $T_i \leftarrow T_i + 1$
6:     $M_i \leftarrow \beta_1 M_i + (1 - \beta_1)\hat{g}_i, \quad v_i \leftarrow \beta_2 v_i + (1 - \beta_2)\hat{g}_i^2$
7:     $\hat{M}_i = M_i/(1 - \beta_1^{T_i}), \quad \hat{v}_i = v_i/(1 - \beta_2^{T_i})$
8:     $\delta^* = -\eta \frac{\hat{M}_i}{\sqrt{\hat{v}_i} + \epsilon}$
9:     Update $\mathbf{x}_i \leftarrow \mathbf{x}_i + \delta^*$
10: **end while**

Note that for algorithmic illustration we only update one coordinate for each iteration. In practice, to achieve the best efficiency of GPU, we usually evaluate the objective in batches, and thus a batch of $\hat{g}_i$ and $\hat{h}_i$ can be estimated. In our implementation we estimate $B = 128$ pixels' gradients and Hessians per iteration, and then update B coordinates in a single iteration.

3.4   Attack-space dimension reduction

We first define $\Delta \mathbf{x} = \mathbf{x} - \mathbf{x}_0$ and $\Delta \mathbf{x} \in \mathbb{R}^p$ to be the adversarial noise added to the original image $\mathbf{x}_0$. Our optimization procedure starts with $\Delta \mathbf{x} = 0$. For networks with a large input size p, optimizing over $\mathbb{R}^p$ (we call it *attack-space*) using zeroth order methods can be quite slow because we need to estimate a large number of gradients.

Algorithm 3 ZOO-Newton: Zeroth Order Stochastic Coordinate Descent with Coordinate-wise Newton's Method

Require: Step size η
1: **while** not converged **do**
2:     Randomly pick a coordinate $i \in \{1, \cdots, p\}$
3:     Estimate $\hat{g}_i$ and $\hat{h}_i$ using (6) and (7)
4:     **if** $\hat{h}_i \leq 0$ **then**
5:         $\delta^* \leftarrow -\eta \hat{g}_i$
6:     **else**
7:         $\delta^* \leftarrow -\eta \frac{\hat{g}_i}{\hat{h}_i}$
8:     **end if**
9:     Update $\mathbf{x}_i \leftarrow \mathbf{x}_i + \delta^*$
10: **end while**

Instead of directly optimizing $\Delta \mathbf{x} \in \mathbb{R}^p$, we introduce a dimension reduction transformation $D(\mathbf{y})$ where $\mathbf{y} \in \mathbb{R}^m, range(D) \in \mathbb{R}^p$, and $m < p$. The transformation can be linear or non-linear. Then, we use $D(\mathbf{y})$ to replace $\Delta \mathbf{x} = \mathbf{x} - \mathbf{x}_0$ in (1):

$$\text{minimize}_{\mathbf{y}} \; \|D(\mathbf{y})\|_2^2 + c \cdot f(\mathbf{x}_0 + D(\mathbf{y}), t) \quad (8)$$
$$\text{subject to} \; \mathbf{x}_0 + D(\mathbf{y}) \in [0, 1]^p.$$

The use of $D(\mathbf{y})$ effectively reduces the dimension of attack-space from p to m. Note that we do not alter the dimension of an input image $\mathbf{x}$ but only reduce the permissible dimension of the adversarial noise. A convenient transformation is to define D to be the upscaling operator that resizes $\mathbf{y}$ as a size-p image, such as the bilinear interpolation method[1]. For example, in the Inception-v3 network $\mathbf{y}$ can be a small adversarial noise image with dimension $m = 32 \times 32 \times 3$, while the original image dimension is $p = 299 \times 299 \times 3$. Other transformations like DCT (discrete cosine transformation) can also be used. We will show the effectiveness of this method in Section 4.

3.5   Hierarchical attack

When applying attack-space dimension reduction with a small m, although the attack-space is efficient to optimize using zeroth order methods, a valid attack might not be found due to the limited search space. Conversely, if a large m is used, a valid attack can be found in that space, but the optimization process may take a long time. Thus, for large images and difficult attacks, we propose to use a *hierarchical attack* scheme, where we use a series of transformations $D_1, D_2 \cdots$ with dimensions $m_1, m_2, \cdots$ to gradually increase m during the optimization process. In other words, at a specific iteration j (according to the dimension increasing schedule) we set $\mathbf{y}_j = D_i^{-1}(D_{i-1}(\mathbf{y}_{j-1}))$ to increase the dimension of $\mathbf{y}$ from m_{i-1} to m_i (D^{-1} denotes the inverse transformation of D).

For example, when using image scaling as the dimension reduction technique, D_1 upscales $\mathbf{y}$ from $m_1 = 32 \times 32 \times 3$ to $299 \times 299 \times 3$, and D_2 upscales $\mathbf{y}$ from $m_2 = 64 \times 64 \times 3$ to $299 \times 299 \times 3$. We start with $m_1 = 32 \times 32 \times 3$ variables to optimize with and use D_1 as the transformation, then after a certain number of iterations (when the decrease in the loss function is inapparent, indicating the need of a larger attack-space), we upscale $\mathbf{y}$ from $32 \times 32 \times 3$ to $64 \times 64 \times 3$, and use D_2 for the following iterations.

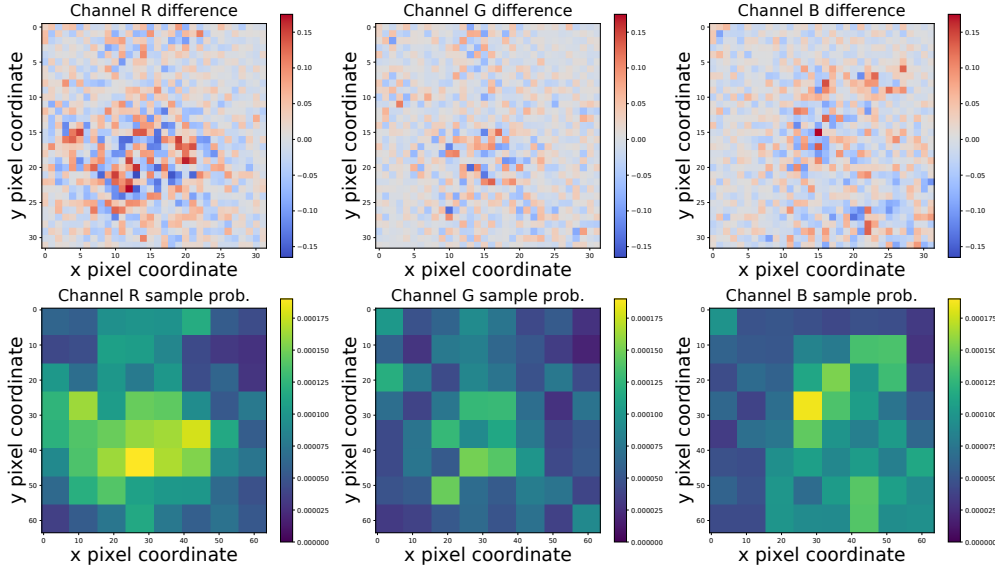[1]See the details at https://en.wikipedia.org/wiki/Bilinear_interpolation

Figure 3: Attacking the bagel image in Figure 1 (a) with importance sampling. Top: Pixel values in certain parts of the bagel image have significant changes in RGB channels, and the changes in the R channel is more prominent than other channels. Here the attack-space is $32 \times 32 \times 3$. Although our targeted attack in this attack-space fails, its adversarial noise provides important clues to pixel importance. We use the noise from this attack-space to sample important pixels after we increase the dimension of attack-space to a larger dimension. Bottom: Importance sampling probability distribution for $64 \times 64 \times 3$ attack-space. The importance is computed by taking the absolute value of pixel value changes, running a 4×4 max-pooling for each channel, up-sampling to the dimension of $64 \times 64 \times 3$, and normalizing all values.

3.6 Optimize the important pixels first

One benefit of using coordinate descent is that we can choose which coordinates to update. Since estimating gradient and Hessian for each pixel is expensive in the black-box setting, we propose to selectively update pixels by using *importance sampling*. For example, pixels in the corners or at the edges of an image are usually less important, whereas pixels near the main object can be crucial for a successful attack. Therefore, in the attack process we sample more pixels close to the main object indicated by the adversarial noise.

We propose to divide the image into 8×8 regions, and assign sampling probabilities according to how large the pixel values change in that region. We run a max pooling of the absolute pixel value changes in each region, up-sample to the desired dimension, and then normalize all values such that they sum up to 1. Every few iterations, we update these sampling probabilities according to the recent changes. In Figure 3, we show a practical example of pixel changes and how importance sampling probabilities are generated when attacking the bagel image in Figure 1 (a).

When the attack-space is small (for example, $32 \times 32 \times 3$), we do not use importance sampling to sufficiently search the attack-space. When we gradually increase the dimension of attack-space using hierarchical attack, incorporating importance sampling becomes crucial as the attack-space is increasingly larger. We will show the effectiveness of importance sampling in Section 4.

4 PERFORMANCE EVALUATION

4.1 Setup

We compare our attack (ZOO) with Carlini & Wagner's (C&W) white-box attack [8] and the substitute model based black-box attack [35]. We would like to show that our black-box attack can achieve similar success rate and distortion as the white-box C&W attack, and can significantly outperform the substitute model based black-box attack, while maintaining a reasonable attack time.

Our experimental setup is based on Carlini & Wagner's framework[2] with our ADAM and Newton based zeroth order optimizer included. For substitute model based attack, we use the reference implementation (with necessary modifications) in CleverHans[3] for comparison. For experiments on MNIST and CIFAR, we use a Intel Xeon E5-2690v4 CPU with a single NVIDIA K80 GPU; for experiments on ImageNet, we use a AMD Ryzen 1600 CPU with a single NVIDIA GTX 1080 Ti GPU. Our experimental code is publicly available[4]. For implementing zeroth order optimization, we use a batch size of $B = 128$; i.e., we evaluate 128 gradients and update 128 coordinates per iteration. In addition, we set $\kappa = 0$ unless specified.

4.2 MNIST and CIFAR10

DNN Model. For MNIST and CIFAR10, we use the same DNN model as in the C&W attack ([8], Table 1). For substitute model based attack, we use the same DNN model for *both* the target model

---

[2]https://github.com/carlini/nn_robust_attacks
[3]https://github.com/tensorflow/cleverhans/blob/master/tutorials/mnist_blackbox.py
[4]https://github.com/huanzhang12/ZOO-Attack

Table 1: MNIST and CIFAR10 attack comparison: ZOO attains comparable success rate and L_2 distortion as the white-box C& W attack, and significantly outperforms the black-box substitute model attacks using FGSM (L_∞ attack) and the C&W attack [35]. The numbers in parentheses in Avg. Time field is the total time for training the substitute model. For FGSM we do not compare its L_2 with other methods because it is an L_∞ attack.

	MNIST					
	Untargeted			Targeted		
	Success Rate	Avg. L_2	Avg. Time (per attack)	Success Rate	Avg. L_2	Avg. Time (per attack)
White-box (C&W)	100 %	1.48066	0.48 min	100 %	2.00661	0.53 min
Black-box (Substitute Model + FGSM)	40.6 %	-	0.002 sec (+ 6.16 min)	7.48 %	-	0.002 sec (+ 6.16 min)
Black-box (Substitute Model + C&W)	33.3 %	3.6111	0.76 min (+ 6.16 min)	26.74 %	5.272	0.80 min (+ 6.16 min)
Proposed black-box (ZOO-ADAM)	100 %	1.49550	1.38 min	98.9 %	1.987068	1.62 min
Proposed black-box (ZOO-Newton)	100 %	1.51502	2.75 min	98.9 %	2.057264	2.06 min
	CIFAR10					
	Untargeted			Targeted		
	Success Rate	Avg. L_2	Avg. Time (per attack)	Success Rate	Avg. L_2	Avg. Time (per attack)
White-box (C&W)	100 %	0.17980	0.20 min	100 %	0.37974	0.16 min
Black-box (Substitute Model + FGSM)	76.1 %	-	0.005 sec (+ 7.81 min)	11.48 %	-	0.005 sec (+ 7.81 min)
Black-box (Substitute Model + C&W)	25.3 %	2.9708	0.47 min (+ 7.81 min)	5.3 %	5.7439	0.49 min (+ 7.81 min)
Proposed Black-box (ZOO-ADAM)	100 %	0.19973	3.43 min	96.8 %	0.39879	3.95 min
Proposed Black-box (ZOO-Newton)	100 %	0.23554	4.41 min	97.0 %	0.54226	4.40 min

and the substitute model. If the architecture of a targeted DNN is unknown, black-box attacks based on substitute models will yield worse performance due to model mismatch.

Target images. For targeted attacks, we randomly select 100 images from MNIST and CIFAR10 test sets, and skip the original images misclassified by the target model. For each image, we apply targeted attacks to all 9 other classes, and thus there are 900 attacks in total. For untargeted attacks, we randomly select 200 images from MNIST and CIFAR10 test sets.

Parameter setting. For both ours and the C&W attack, we run a binary search up to 9 times to find the best c (starting from 0.01), and terminate the optimization process early if the loss does not decrease for 100 iterations. We use the same step size $\eta = 0.01$ and ADAM parameters $\beta_1 = 0.9, \beta_2 = 0.999$ for all methods. For the C&W attack, we run 1,000 iterations; for our attack, we run 3,000 iterations for MNIST and 1,000 iterations for CIFAR. Note that our algorithm updates far less variables because for each iteration we only update 128 pixels, whereas in the C&W attack all pixels are updated based on the full gradient in one iteration due to the white-box setting. Also, since the image size of MNIST and CIFAR10 is small, we do not reduce the dimension of attack-space or use hierarchical attack and importance sampling. For training the substitute model, we use 150 hold-out images from the test set and run 5 Jacobian augmentation epochs, and set the augmentation parameter $\lambda = 0.1$. We implement FGSM and the C&W attack on the substitute model for both targeted and untargeted transfer attacks to the black-box DNN. For FGSM, the perturbation parameter $\epsilon = 0.4$, as it is shown to be effective in [35]. For the C&W attack, we use the same settings as the white-box C&W, except for setting $\kappa = 20$ for attack transferability and using 2,000 iterations.

Other tricks. When attacking MNIST, we found that the change-of-variable via tanh can cause the estimated gradients to vanish due to limited numerical accuracy when pixel values are close to the boundary (0 or 1). Alternatively, we simply project the pixel values within the box constraints after each update for MNIST. But for

CIFAR10, we find that using change-of-variable converges faster, as most pixels are not close to the boundary.

Results. As shown in Table 1, our proposed attack (ZOO) achieves nearly 100% success rate. Furthermore, the L_2 distortions are also close to the C&W attack, indicating our black-box adversarial images have similar quality as the white-box approach (Figures 4 and 5). Notably, our success rate is significantly higher than the substitute model based attacks, especially for targeted attacks, while maintaining reasonable average attack time. When transferring attacks from the substitute models to the target DNN, FGSM achieves better success rates in some experiments because it uses a relatively large $\epsilon = 0.4$ and introduces much more noise than C&W attack. We also find that ADAM usually works better than the Newton's method in terms of computation time and L_2 distortion.

4.3 Inception network with ImageNet

Attacking a large black-box network like Inception-v3 [39] can be challenging due to large attack-space and expensive model evaluation. Black-box attacks via substitute models become impractical in this case, as a substitute model with a large enough capacity relative to Inception-V3 is needed, and a tremendous amount of costly Jacobian data augmentation is needed to train this model. On the other hand, transfer attacks may suffer from lower success rate comparing to white-box attacks, especially for targeted attacks. Here we apply the techniques proposed in Section 3.4, 3.5 and 3.6 to efficiently overcome the optimization difficulty toward effective and efficient black-box attacks.

• **Untargeted black-box attacks to Inception-v3.**

Target images. We use 150 images from the ImageNet test set for untargeted attacks. To justify the effectiveness of using attack-space dimension reduction, we exclude small images in the test set and ensure that all the original images are at least 299×299 in size. We also skip all images that are originally misclassified by Inception-v3.

Attack techniques and parameters. We use an attack-space of only $32 \times 32 \times 3$ (the original input space is $299 \times 299 \times 3$) and do not

(a)   (b) White-box C&W attack   (c) ZOO-ADAM black-box attack   (d) ZOO-Newton black-box attack
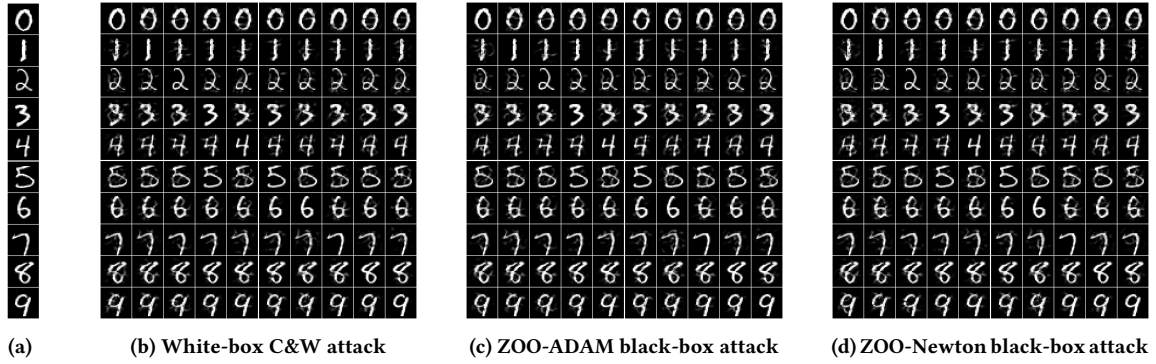
Figure 4: Visual comparison of successful adversarial examples in MNIST. Each row displays crafted adversarial examples from the sampled images in (a). Each column in (b) to (d) indexes the targeted class for attack (digits 0 to 9).

(a)   (b) White-box C&W attack   (c) ZOO-ADAM black-box attack   (d) ZOO-Newton black-box attack
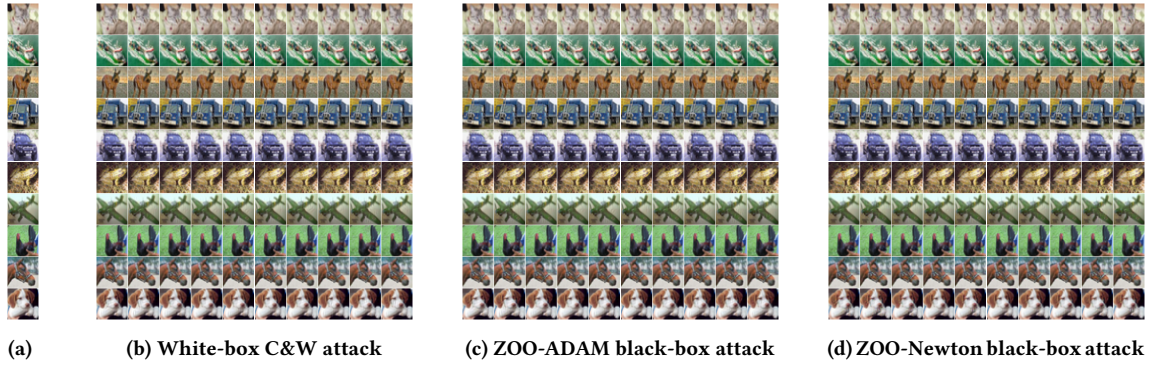
Figure 5: Visual comparison of successful adversarial examples in CIFAR10. Each row displays crafted adversarial examples from the sampled images in (a). Each column in (b) to (d) indexes the targeted class for attack.

Table 2: Untargeted ImageNet attacks comparison. Substitute model based attack cannot easily scale to ImageNet.

	Success Rate	Avg. L_2
White-box (C&W)	100 %	0.37310
Proposed black-box (ZOO-ADAM)	88.9 %	1.19916
Black-box (Substitute Model)	N.A.	N.A.

use hierarchical attack. We also set a hard limit of $1,500$ iterations for each attack, which takes about 20 minutes per attack in our setup. In fact, during $1,500$ iterations, only $1500 \times 128 = 192,000$ gradients are evaluated, which is even less than the total number of pixels ($299 \times 299 \times 3 = 268,203$) of the input image. We fix $c = 10$ in all Inception-v3 experiments, as it is too costly to do binary search in this case. For both C&W and our attacks, we use step size 0.002.
Results. We compare the success rate and average L_2 distortion between our ZOO attack and the C&W white-box attack in Table 2. Despite running only 1,500 iterations (within 20 minutes per image) and using a small attack-space ($32 \times 32 \times 3$), our black-box attack achieves about 90% success rate. The average L_2 distortion is about 3 times larger than the white-box attack, but our adversarial images are still visually indistinguishable (Figures 1). The success rate and distortion can be further improved if we run more iterations.

• **Targeted black-box attacks to Inception-v3.**

For Inception-v3, a targeted attack is much more difficult as there are 1000 classes, and a successful attack means one can manipulate the predicted probability of any specified class. However, we report that using our advanced attack techniques, $20,000$ iterations (each with $B = 128$ pixel updates) are sufficient for a *hard* targeted attack.
Target image. We select an image (Figure 1 (a)) for which our untargeted attack failed, i.e., we cannot even find an *untargeted* attack in the $32 \times 32 \times 3$ attack-space, indicating that this image is hard to attack. Inception-v3 classifies it as a "bagel" with 97.0% confidence, and other top-5 predictions include "guillotine", "pretzel", "Granny Smith" and "dough" with 1.15%, 0.07%, 0.06% and 0.01% confidence. We deliberately make the attack even harder by choosing the target class as "grand piano", with original confidence of only 0.0006%.
Attack techniques. We use attack-space dimension reduction as well as hierarchical attack. We start from an attack-space of $32 \times 32 \times 3$, and increase it to $64 \times 64 \times 3$ and $128 \times 128 \times 3$ at iteration 2,000 and 10,000, respectively. We run the zeroth order ADAM solver (Algorithm 2) with a total of 20,000 iterations, taking about 260 minutes in our setup. Also, when the attack space is greater than $32 \times 32 \times 3$, we incorporate importance sampling, and keep updating the sampling probability after each iteration.
Reset ADAM states. We report an additional trick to reduce the final distortion - reset the ADAM solver's states when a first valid
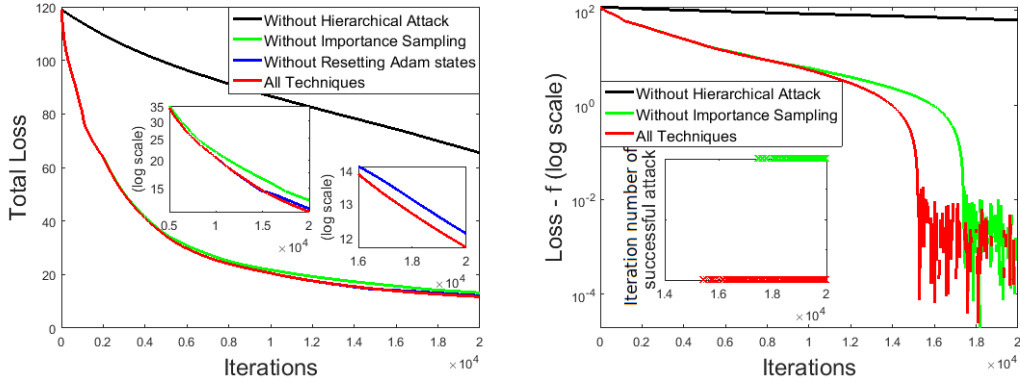
Figure 6: Left: total loss $\|\mathbf{x} - \mathbf{x}_0\|_2^2 + c \cdot f(\mathbf{x}, t)$ versus iterations. Right: $c \cdot f(\mathbf{x}, t)$ versus iterations (log y-scale). When $c \cdot f(\mathbf{x}, t)$ reaches 0, a valid attack is found. With all techniques applied, the first valid attack is found at iteration $15,227$. The optimizer then continues to minimize $\|\mathbf{x} - \mathbf{x}_0\|_2^2$ to reduce distortion. In the right figure we do not show the curve without resetting ADAM states because we reset ADAM states only when $c \cdot f(\mathbf{x}, t)$ reaches 0 for the first time.

Table 3: Comparison of different attack techniques. "First Valid" indicates the iteration number where the first successful attack was found during the optimization process.

Black-box (ZOO-ADAM)	Success?	First Valid	Final L_2	Final Loss
All techniques	Yes	15,227	3.425	11.735
No Hierarchical Attack	No	-	-	62.439
No importance sampling	Yes	17,403	3.63486	13.216
No ADAM state reset	Yes	15,227	3.47935	12.111

attack is found during the optimization process. The reason is as follows. The total loss consists of two parts: $l_1 := c \cdot f(\mathbf{x}, t)$ and $l_2 := \|\mathbf{x} - \mathbf{x}_0\|_2^2$. l_1 measures the difference between the original class probability P_{orig} and targeted class probability P_{target} as defined in (4). When $l_1 = 0$, $P_{\text{orig}} \leq P_{\text{target}}$, and a valid adversarial example is found. l_2 is the L_2 distortion. During the optimization process, we observe that before l_1 reaches 0, l_2 is likely to increase, i.e., adding more distortion and getting closer to the target class. After l_1 reaches 0 it cannot go below 0 because it is a hinge-like loss, and at this point the optimizer should try to reduce l_2 as much as possible while keeping P_{target} only slightly larger than P_{orig}. However, when we run coordinate-wise ADAM, we found that even after l_1 reaches 0, the optimizer still tries to reduce P_{orig} and to increase P_{target}, and l_2 will not be decreased efficiently. We believe the reason is that the historical gradient statistics stored in ADAM states are quite stale due to the large number of coordinates. Therefore, we simply reset the ADAM states after l_1 reaches 0 for the first time in order to make the solver focus on decreasing l_2 afterwards.

Results. Figure 6 shows how the loss decreases versus iterations, with all techniques discussed above applied in red; other curves show the optimization process without a certain technique but all others included. The black curve decreases very slowly, suggesting hierarchical attack is extremely important in accelerating our attack, otherwise the large attack-space makes zeroth order methods infeasible. Importance sampling also makes a difference especially after iteration 10,000 – when the attack-space is increased to $128 \times 128 \times 3$; it helps us to find the first valid attack over 2,000 iterations earlier,

thus leaving more time for reducing the distortion. The benefit of resetting ADAM states is clearly shown in Table 3, where the final distortion and loss increase noticeably if we do not reset the states.

The proposed ZOO attack succeeds in decreasing the probability of the original class by over 160x (from 97% to about 0.6%) while increasing the probability of the target class by over 1000x (from 0.0006% to over 0.6%, which is top-1) to achieve a successful attack. Furthermore, as shown in Figures 1, the crafted adversarial noise is almost negligible and indistinguishable by human eyes.

5 CONCLUSION AND FUTURE WORK

This paper proposed a new type of black-box attacks named ZOO to DNNs without training any substitute model as an attack surrogate. By exploiting zeroth order optimization for deploying pseudo back propagation on a targeted black-box DNN, experimental results show that our attack attains comparable performance to the state-of-the-art white-box attack (Carlini and Wagner's attack). In addition, our black-box attack significantly outperforms the substitute model based black-box attack in terms of attack success rate and distortion, as our method does not incur any performance loss in attack transferability. Furthermore, we proposed several acceleration techniques for applying our attack to large DNNs trained on ImageNet, whereas the substitute model based black-box attack is hardly scalable to a large DNN like Inception-v3.

Based on the analysis and findings from the experimental results on MNIST, CIFAR10 and ImageNet, we discuss some potential research directions of our black-box adversarial attacks as follows.
• Accelerated black-box attack: although our black-box attack spares the need for training substitution models and attains comparable performance to the white-box C&W attack, in the optimization process it requires more iterations than a white-box attack due to additional computation for approximating the gradient of a DNN and for performing pseudo back propagation. In addition to the computation acceleration tricks proposed in Section 3, a data-driven approach that take these tricks into consideration while crafting an adversarial example can make our black-box attack more efficient.

• Adversarial training using our black-box attack: adversarial training in DNNs is usually implemented in a white-box setting - generating adversarial examples from a DNN to stabilize training and making the retrained model more robust to adversarial attacks. Our black-box attack can serve as an independent indicator of the robustness of a DNN for adversarial training.

• Black-box attacks in different domains: in this paper we explicitly demonstrated our black-box attack to image classifiers trained by DNNs. A natural extension is adapting our attack to other data types (e.g., speeches, time series, graphs) and different machine learning models and neural network architectures (e.g., recurrent neural networks). In addition, how to incorporate side information of a dataset (e.g., expert knowledge) and existing adversarial examples (e.g., security leaks and exploits) into our black-box attack is worthy of further investigation.

ACKNOWLEDGMENT

REFERENCES

[1] Marco Barreno, Blaine Nelson, Anthony D Joseph, and JD Tygar. 2010. The security of machine learning. *Machine Learning* 81, 2 (2010), 121–148.
[2] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. 2006. Can machine learning be secure?. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*. ACM, 16–25.
[3] Dimitri P Bertsekas. *Nonlinear programming*.
[4] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. Evasion attacks against machine learning at test time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 387–402.
[5] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning Attacks Against Support Vector Machines. In *Proceedings of the International Coference on International Conference on Machine Learning*. 1467–1474.
[6] John Bradshaw, Alexander G de G Matthews, and Zoubin Ghahramani. 2017. Adversarial Examples, Uncertainty, and Transfer Testing Robustness in Gaussian Process Hybrid Deep Networks. *arXiv preprint arXiv:1707.02476* (2017).
[7] Nicholas Carlini and David Wagner. 2017. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. *arXiv preprint arXiv:1705.07263* (2017).
[8] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP)*. IEEE, 39–57.
[9] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. 2017. Robust Physical-World Attacks on Machine Learning Models. *arXiv preprint arXiv:1707.08945* (2017).
[10] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. 2017. Detecting Adversarial Samples from Artifacts. *arXiv preprint arXiv:1703.00410* (2017).
[11] Saeed Ghadimi and Guanghui Lan. 2013. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization* 23, 4 (2013), 2341–2368.
[12] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).
[13] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. 2017. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280* (2017).
[14] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. 2016. Adversarial perturbations against deep neural networks for malware classification. *arXiv preprint arXiv:1606.04435* (2016).
[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).
[16] Weiwei Hu and Ying Tan. 2017. Black-Box Attacks against RNN based Malware Detection Algorithms. *arXiv preprint arXiv:1705.08131* (2017).
[17] Weiwei Hu and Ying Tan. 2017. Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN. *arXiv preprint arXiv:1702.05983* (2017).
[18] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. 2016. Safety verification of deep neural networks. *arXiv preprint arXiv:1610.06940* (2016).
[19] Jonghoon Jin, Aysegul Dundar, and Eugenio Culurciello. 2015. Robust convolutional neural networks under adversarial noise. *arXiv preprint arXiv:1511.06306* (2015).
[20] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[21] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533* (2016).
[22] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236* (2016).
[23] Peter D Lax and Maria Shea Terrell. 2014. *Calculus with applications*. Springer.
[24] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.
[25] Xiangru Lian, Huan Zhang, Cho-Jui Hsieh, Yijun Huang, and Ji Liu. 2016. A comprehensive linear speedup analysis for asynchronous stochastic parallel optimization from zeroth-order to first-order. In *Advances in Neural Information Processing Systems*. 3054–3062.
[26] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2016. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770* (2016).
[27] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv preprint arXiv:1706.06083* (2017).
[28] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. 2017. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267* (2017).
[29] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2016. Universal adversarial perturbations. *arXiv preprint arXiv:1610.08401* (2016).
[30] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, Pascal Frossard, and Stefano Soatto. 2017. Analysis of universal adversarial perturbations. *arXiv preprint arXiv:1705.09554* (2017).
[31] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2574–2582.
[32] Yurii Nesterov et al. 2011. *Random gradient-free minimization of convex functions*. Technical Report. Université catholique de Louvain, Center for Operations Research and Econometrics (CORE).
[33] Nicolas Papernot and Patrick McDaniel. 2017. Extending Defensive Distillation. *arXiv preprint arXiv:1705.05264* (2017).
[34] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277* (2016).
[35] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the ACM on Asia Conference on Computer and Communications Security*. ACM, 506–519.
[36] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy (EuroS&P)*. 372–387.
[37] Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang. 2016. Crafting adversarial input sequences for recurrent neural networks. In *IEEE Military Communications Conference (MILCOM)*. 49–54.
[38] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy (SP)*. 582–597.
[39] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2818–2826.
[40] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).
[41] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. 2017. Ensemble Adversarial Training: Attacks and Defenses. *arXiv preprint arXiv:1705.07204* (2017).
[42] Weilin Xu, David Evans, and Yanjun Qi. 2017. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. *arXiv preprint arXiv:1704.01155* (2017).
[43] Weilin Xu, David Evans, and Yanjun Qi. 2017. Feature Squeezing Mitigates and Detects Carlini/Wagner Adversarial Examples. *arXiv preprint arXiv:1705.10686* (2017).
[44] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. 2017. Efficient Defenses Against Adversarial Attacks. *arXiv preprint arXiv:1707.06728*.
[45] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. 2016. Improving the robustness of deep neural networks via stability training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4480–4488.

Figure 7: Additional visual illustration of black-box untargeted attacks using ZOO-ADAM, sampled from ImageNet test set. The columns from left to right are original images with correct labels, additive adversarial noises from our attack (gray color means no modification), and crafted adversarial images with misclassified labels.

APPENDIX

Figure 7 displays some additional adversarial examples in ImageNet from our untargeted attack.