# CHAPTER 3

# Black-box adversarial attacks

In recent years, "machine learning as a service" has offered the world an effortless access to powerful machine learning tools for a wide variety of tasks. For example, commercially available services such as Google Cloud Vision API[1] and Clarifai.com[2] provide well-trained image classifiers to the public. One is able to upload and obtain the class prediction results for images at hand at a low price. However, the details of the model behind the service are unknown to an attacker/user, which means that the function/model $f$ can be viewed as a *black-box* model. A *query* is defined as an instance to upload a data input $\boldsymbol{x}$ and obtain the function/model output $f(\boldsymbol{x})$. In addition to attack success rate, the number of query counts to a targeted black-box model required to launch successful adversarial attacks (e.g., finding adversarial examples) is also important to evaluate the efficiency of black-box attacks.

## 3.1 Evasion attack taxonomy

Evasion attacks can be categorized based on attackers' knowledge of the target model. Fig. 3.1 illustrates the taxonomy of different attack types.

*White-box attack* introduced in Chapter 2 assumes complete knowledge about the target model, including model architecture and model parameters. Consequently, an attacker can exploit the autodifferentiation function offered by deep learning packages, such as backpropagation (input gradient) from the model output to the model input, to craft adversarial examples. This "attack" would be most practical for model developers to perform a series of in-house robustness testing, that is, a virtual adversary or a white-hat hacker. For real attackers, requiring white-box access to the target model is often not feasible.

*Black-box attack* assumes that an attacker can only observe the model prediction of a data input (i.e., a query) and does not know any other information. The target model can be viewed as a black-box function, and thus backpropagation for input gradient is infeasible without knowing the

[1] https://cloud.google.com/vision.
[2] https://clarifai.com.

*Adversarial Robustness for Machine Learning*
https://doi.org/10.1016/B978-0-12-824020-5.00012-0
Copyright © 2023 Elsevier Inc.
All rights reserved.

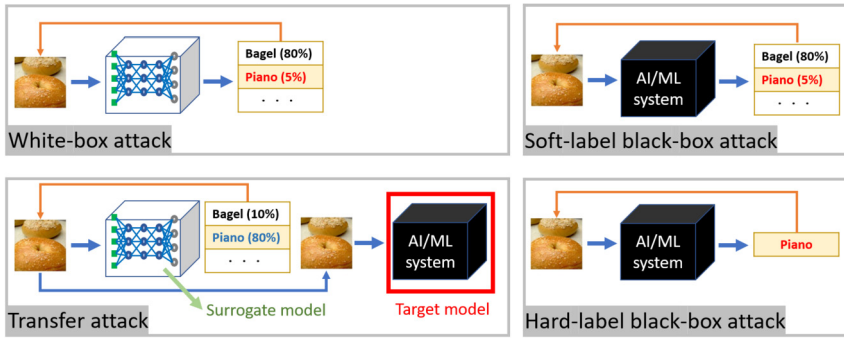


**Figure 3.1** Taxonomy and illustration of evasion attacks.

model details. In the *soft-label black-box attack* setting, an attacker can observe (parts of) class predictions and their associated confidence scores. In the *hard-label black-box attack* (decision-based) setting, an attacker can only observe the top-1 label prediction, which is the least information required to be returned to remain utility of the model. In addition to attack success rate, the query efficiency is also an important metric for performance evaluation of black-box attacks.

*Transfer attack* is a branch of black-box attack that uses adversarial examples generated from a white-box surrogate model to attack the target black-box model. The surrogate model can be either pretrained (Liu et al., 2017b) or distilled from a set of data samples with soft labels given by the target model as their training labels (Papernot et al., 2016, 2017).

Between the spectrum of white-box and black-box attacks, variations in the threat models and the attacker's capability will result in different *gray-box* attacks.

## 3.2 Soft-label black-box attack

Black-box attack algorithms often adopt either the penalty-based (e.g., Eq. (2.9)) or budget-based (e.g., Eq. (2.1)) formulation. Since the input gradient of the attacker's loss is unavailable to obtain in the black-box setting, one principal approach is performing gradient estimation using model queries and then using the estimated gradient to replace the true gradient in white-box attack algorithms, leading to the zeroth-order optimization (ZOO) based black-box attacks (Chen et al., 2017a).

Without loss of generality, it suffices to denote the target model as a classification function parameterized by $\boldsymbol{\theta}$, defined as $f_{\boldsymbol{\theta}} : [0, 1]^d \mapsto \mathbb{R}^K$, that takes a $d$-dimensional scaled data sample as its input and yields a vector of prediction scores of all $K$ classes, such as the prediction probabilities for each class. The term "soft labels" refers to the fact that an attacker can observe the prediction score of each class. We further consider the case of applying an entrywise monotonic transformation $M(f_{\boldsymbol{\theta}})$ to the output of $f_{\boldsymbol{\theta}}$ for black-box attacks, since a monotonic transformation preserves the ranking of the class predictions and can alleviate the problem of large score variation in $f_{\boldsymbol{\theta}}$ (e.g., the transformation of probability to log probability is monotonic).

Here we formulate soft-label black-box targeted attacks using penalty-based loss. The formulation can be easily adapted to untargeted attacks. Let $(\boldsymbol{x}_0, t_0)$ denote a natural image $\boldsymbol{x}_0$ and its ground-truth class label $t_0$, and let $(\boldsymbol{x}, t)$ denote the adversarial example of $\boldsymbol{x}_0$ and the target attack class label $t \neq t_0$. The problem of finding an adversarial example can be formulated as an optimization problem taking the generic form of

$$\min_{\boldsymbol{x} \in [0,1]^d} \text{Dist}(\mathbf{x}, \mathbf{x}_0) + \lambda \cdot \text{Loss}\big(\mathbf{x}, M\big(f_{\boldsymbol{\theta}}(\boldsymbol{x})\big), t\big), \tag{3.1}$$

where $\text{Dist}(\mathbf{x}, \mathbf{x}_0)$ measures the distortion between $\mathbf{x}$ and $\mathbf{x}_0$, $\text{Loss}(\cdot)$ is an attack objective reflecting the likelihood of predicting $t = \arg\max_{k \in \{1,\ldots,K\}} [M(f_{\boldsymbol{\theta}}(\boldsymbol{x}))]_k$, $\lambda \geq 0$ is a regularization coefficient, and the constraint $\mathbf{x} \in [0, 1]^d$ confines the adversarial image $\boldsymbol{x}$ to the valid scaled data space. Consider the case of additive input perturbation, $\text{Dist}(\boldsymbol{x}, \boldsymbol{x}_0)$ is often evaluated by the $\ell_p$ norm defined as $\text{Dist}(\boldsymbol{x}, \boldsymbol{x}_0) = \|\boldsymbol{x} - \boldsymbol{x}_0\|_p$, where $\boldsymbol{\delta} = \boldsymbol{x} - \boldsymbol{x}_0$ is the additive perturbation to $\boldsymbol{x}_0$. The attack objective $\text{Loss}(\cdot)$ can be the training loss (e.g., cross-entropy) for classification or the margin-based C&W loss (Carlini and Wagner, 2017b), as introduced in Chapter 2. The log operation is often adopted for the entrywise monotonic transformation function $M$.

In the black-box attack setting, the attack objective in (3.1) can be viewed as a black-box function $F$. Since only the function values of $F$ are observable, unlike white-box attacks, we cannot use gradient-based approaches for solving (3.1). A fundamental tool to address this issue is the use of *zeroth-order optimization* (ZOO), which follows gradient-based (first-order) optimization rules but uses function values at two closeby points to estimate the gradient instead of using the true gradient.

*Coordinatewise gradient estimation.* As a first attempt to enable gradient-free black-box attacks, Chen et al. (2017a) use the symmetric difference

quotient method (Lax and Terrell, 2014) to evaluate the gradient $\frac{\partial F(\boldsymbol{x})}{\partial \boldsymbol{x}_i}$ of the $i$th component by

$$\boldsymbol{g}_i = \frac{F(\boldsymbol{x} + h\boldsymbol{e}_i) - F(\boldsymbol{x} - h\boldsymbol{e}_i)}{2h} \approx \frac{\partial F(\boldsymbol{x})}{\partial \boldsymbol{x}_i} \tag{3.2}$$

using a small $h$, and here $\boldsymbol{e}_i$ denotes the $i$th elementary basis (having one at the $i$th entry and zero elsewhere). Theoretically this estimation will get the exact gradient value when $h \to 0$, however, in practice taking very small $h$ also results in some numerical error, so one usually choose some small-enough value such as $h = 0.001$.

Note that the nature of coordinatewise gradient estimation step in (3.2) must incur an enormous amount of model queries to obtain the full gradient estimate and is hence not query-efficient. To estimate the gradient of dimension $d$, it needs $2 \cdot d$ model queries to $F$. For example, the ImageNet dataset has $d = 299 \times 299 \times 3 \approx 270{,}000$ input dimensions, rendering coordinatewise zeroth-order optimization query-inefficient. Therefore, to use this coordinatewise gradient estimation, Chen et al. (2017a) uses a coordinate descent algorithm to solve the attack objective (3.1), where at each iteration the coordinate descent algorithm picks a coordinate $i$, computes gradient based on (3.2), and then update the coordinate by $x_i \leftarrow x_i - \alpha g_i$, where $\alpha$ is the step size.

*Random vector based gradient estimation.* The coordinatewise gradient estimation mentioned above can be viewed as computing the derectional derivative along the direction of $\boldsymbol{e}_i$. However, in standard attacks when the perturbation is bounded by small $\ell_2$ or $\ell_\infty$ norm, coordinate-wise perturbation is usually insufficient, so using coordinate gradient estimation will require a large amount of queries. Therefore, query-efficient black-box attacks such as (Tu et al., 2019) use the following gradient estimator which computes the directional derivative at a randomly sampled direction $\boldsymbol{u}$:

$$\boldsymbol{g} = b \cdot \frac{f(\boldsymbol{x} + \beta \boldsymbol{u}) - f(\boldsymbol{x})}{\beta} \cdot \boldsymbol{u}, \tag{3.3}$$

where $\beta > 0$ is a smoothing parameter, $\boldsymbol{u}$ is a unit-length vector that is uniformly drawn at random from a unit Euclidean sphere, and $b$ is a tunable scaling parameter that balances the bias and variance trade-off of the gradient estimation error.

To effectively control the error in gradient estimation, we consider a more general gradient estimator, in which the gradient estimate is averaged

over $q$ random directions $\{\mathbf{u}^{(j)}\}_{j=1}^{q}$, that is,

$$\overline{\boldsymbol{g}} = \frac{1}{q}\sum_{j=1}^{q} \boldsymbol{g}^{(j)}, \tag{3.4}$$

where $\boldsymbol{g}^{(j)}$ is a gradient estimate defined in (3.3) with $\mathbf{u} = \mathbf{u}^{(j)}$. The use of multiple random directions can reduce the variance of $\overline{\boldsymbol{g}}$ in (3.4) for convex loss functions (Duchi et al., 2015; Liu et al., 2018a).

Below we establish an error analysis of the averaged random gradient estimator in (3.4) for studying the influence of the parameters $b$ and $q$ on estimation error and query efficiency.

**Theorem 1.** *Let $F : \mathbb{R}^d \mapsto \mathbb{R}$ be a differentiable function with L-Lipschitz gradient $\nabla F$.*[3] *Then the mean squared estimation error of $\overline{\boldsymbol{g}}$ in (3.4) is upper bounded by*

$$\mathbb{E}\|\overline{\boldsymbol{g}} - \nabla F(\mathbf{x})\|_2^2 \leq 4\left(\frac{b^2}{d^2} + \frac{b^2}{dq} + \frac{(b-d)^2}{d^2}\right)\|\nabla F(\mathbf{x})\|_2^2 + \frac{2q+1}{q} b^2\beta^2 L^2. \tag{3.5}$$

*Proof.* See Section 3.7. □

Here we highlight the important implications based on Theorem 1: (i) The error analysis holds when $F$ is *nonconvex*; (ii) In neural network implementations, the true gradient $\nabla F$ can be viewed as the numerical gradient obtained via back-propagation; (iii) For any fixed $b$, selecting a small $\beta$ (e.g., $\beta = 1/d$) can effectively reduce the last error term in (3.5), and we therefore focus on optimizing the first error term; (iv) The first error term in (3.5) exhibits the influence of $b$ and $q$ on the estimation error and is independent of $\beta$. We further elaborate on (iv) as follows. Fixing $q$ and letting $\eta(b) = \frac{b^2}{d^2} + \frac{b^2}{dq} + \frac{(b-d)^2}{d^2}$ be the coefficient of the first error term in (3.5), the optimal $b$ that minimizes $\eta(b)$ is $b^* = \frac{dq}{2q+d}$. For query efficiency, we would like to keep $q$ small, which then implies $b^* \approx q$ and $\eta(b^*) \approx 1$ when the dimension $d$ is large. On the other hand, as $q \to \infty$, $b^* \approx d/2$ and $\eta(b^*) \approx 1/2$, which yields a smaller error upper bound but is query-inefficient. We also note that by setting $b = q$ the coefficient $\eta(b) = \frac{b^2}{d^2} + \frac{b^2}{dq} + \frac{(b-d)^2}{d^2} \approx 1$ and thus is independent of the dimension $d$ and parameter $q$.

[3] A function $W$ is $L$-Lipschitz if $\|W(\boldsymbol{w}_1) - W(\boldsymbol{w}_2)\|_2 \leq L\|\boldsymbol{w}_1 - \boldsymbol{w}_2\|_2$ for all $\boldsymbol{w}_1$, $\boldsymbol{w}_2$. For neural networks with ReLU activations, $L$ can be derived from the model weights (Szegedy et al., 2014).

*Pseudo-gradient descent*. With the estimated gradient $\boldsymbol{g}$, we can extend any gradient-based white-box attack to the soft-label black-box setting, by conducting the same gradient-based attack with gradient replaced by the gradient estimator $\boldsymbol{g}$. For example, we can solve (3.1) using a regular (projected) gradient descent/ascent algorithm following

$$\boldsymbol{x}_{t+1} \leftarrow \Pi\big(\boldsymbol{x}_t - \alpha \cdot \boldsymbol{g}(\boldsymbol{x}_t)\big), \tag{3.6}$$

where $t$ denotes the iteration index, $\Pi$ denotes the project operations (e.g., projection to $[0,1]^d$), and $\alpha$ denotes the step size. The choices in gradient estimators and the type of gradient-based optimization solvers (e.g., PGD, Adam or Mementum-SGD) will lead to different black-box attack algorithms.

## 3.3 Hard-label black-box attack

Hard-label black-box attacks assume the most information-restricted setting that only the top-1 class prediction (but not the scores) is observable to an attacker. In this setting, the naive attack objective in (3.1) will be ill-defined so one cannot directly extend white-box or soft-label black-box attacks to the hard-label setting. If we replace $f_\theta(x)$ by the hard-label prediction in (3.1), the objective function will be a discrete step function; we can observe small perturbation either doesn't affect the objective function value (when the prediction label is unchanged) or result in a discrete change to the objective function value (when the prediction label is changed). Therefore, zeroth order optimization attacks used in the soft-label black-box setting cannot be easily applied. Despite the above-mentioned difficulty, it has been shown that by carefully reformulating the attack objective function, zeroth-order optimization principles can still be applied by spending extra queries to explore local loss landscapes for gradient estimation (Cheng et al., 2019a, 2020c; Chen et al., 2020a).

For a given example $\boldsymbol{x}_0$, true label $t_0$, and the hard-label black-box function $f_{\boldsymbol{\theta}} : \mathbb{R}^d \to \{1, \ldots, K\}$, we define our objective function $h : \mathbb{R}^d \to \mathbb{R}$ depending on the type of attack:

$$\textit{Untargeted attack:}\ \ h(\boldsymbol{s}) = \min_{\lambda > 0} \lambda \quad \texttt{s.t.} \quad f_{\boldsymbol{\theta}}\left(\boldsymbol{x}_0 + \lambda \frac{\boldsymbol{s}}{\|\boldsymbol{s}\|}\right) \neq t_0. \tag{3.7}$$

$$\textit{Targeted attack (given target } t\textit{):}\quad h(\boldsymbol{s}) = \min_{\lambda>0} \lambda \quad \texttt{s.t.} \quad f_{\boldsymbol{\theta}}\left(\boldsymbol{x}_0 + \lambda \frac{\boldsymbol{s}}{\|\boldsymbol{s}\|}\right) = t. \tag{3.8}$$

In this formulation, $\boldsymbol{s}$ represents the direction of adversarial example, and $h(\boldsymbol{s})$ is the distance from $\boldsymbol{x}_0$ to the nearest adversarial example along the direction $\boldsymbol{s}$. The norm $\|s\|$ in the denominator makes the search vector $\frac{\boldsymbol{s}}{\|\boldsymbol{s}\|}$ having unit length. For untargeted attack, $h(\boldsymbol{s})$ also corresponds to the distance to the decision boundary along the direction $\boldsymbol{s}$.

The main idea of this reformulation proposed in (Cheng et al., 2019a) is that instead of directly optimizing the adversarial perturbation as in (3.1), we aim to find a direction $\boldsymbol{s}$ such that the worst-case perturbation will be along that direction. As $h(\boldsymbol{s})$ defined above measures the distance to the closest adversarial example along direction $\boldsymbol{s}$, finding the direction of adversarial example can be formulated as the following optimization problem:

$$\min_{\boldsymbol{s}} \; h(\boldsymbol{s}). \tag{3.9}$$

Finally, the adversarial example can be found by $\boldsymbol{x} = \boldsymbol{x}_0 + h(\boldsymbol{s}^*)\frac{\boldsymbol{s}^*}{\|\boldsymbol{s}^*\|}$, where $\boldsymbol{s}^*$ is the optimal solution of (3.9).

In the black-box attack setting, we are not able to obtain the gradient of $h$, but we can evaluate the function value of $h$ using the hard-label queries. For simplicity, we focus on untargeted attack here, but the same procedure can be applied to targeted attack as well.

Here we discuss how to compute $h(\boldsymbol{s})$ directly without additional information. For a given normalized $\boldsymbol{s}$, this can be computed by a coarse-grained search and then a binary search. Taking untargeted attack as an example, in coarse-grained search, the algorithm queries the points $\{\boldsymbol{x}_0 + \alpha\boldsymbol{s}, \boldsymbol{x}_0 + 2\alpha\boldsymbol{s}, \dots\}$ one by one until finding an adversarial example $f(\boldsymbol{x} + i\alpha\boldsymbol{s}) \neq t_0$. This means that the boundary lies between $[\boldsymbol{x}_0 + (i-1)\alpha\boldsymbol{s}, \boldsymbol{x}_0 + i\alpha\boldsymbol{s}]$. We can then enter the second phase to conduct a binary search to find the solution within this region. Note that there is an upper bound of the first stage if we choose $\boldsymbol{s}$ by the direction of $\boldsymbol{x} - \boldsymbol{x}_0$ with some $\boldsymbol{x}$ from another class. This procedure is used to find the initial $\boldsymbol{s}_0$ and corresponding $g(\boldsymbol{s}_0)$ in our optimization algorithm. The entire procedure for computing the $h$ value is presented in Algorithm 1. Note that this procedure is query-inefficient since it may take many queries in the coarse-grained search until finding the interval contains an adversarial example. However, the procedure is guaranteed to find the closest adversarial example when the interval is small enough.

**Algorithm 1** Local computation of $h(\boldsymbol{s})$.

**Input:** Hard-label model $f_{\boldsymbol{\theta}}$, original sample $x_0$, query direction $\boldsymbol{s}$, previous solution $v$, increase/decrease ratio $\alpha = 0.01$, stopping tolerance $\gamma$ (maximum tolerance of computed error)
$\boldsymbol{s} \leftarrow \boldsymbol{s}/||\boldsymbol{s}||$
**if** $f_{\boldsymbol{\theta}}(\boldsymbol{x}_0 + v\boldsymbol{s}) = t_0$ **then**
    $v_{left} \leftarrow v$, $v_{right} \leftarrow (1+\alpha)v$
    **while** $f_{\boldsymbol{\theta}}(\boldsymbol{x}_0 + v_{right}\boldsymbol{s}) = t_0$ **do**
        $v_{right} \leftarrow (1+\alpha)v_{right}$
    **end while**
**else**
    $v_{right} \leftarrow v$, $v_{left} \leftarrow (1-\alpha)v$
    **while** $f_{\boldsymbol{\theta}}(\boldsymbol{x}_0 + v_{left}\boldsymbol{s}) \neq t_0$ **do**
        $v_{left} \leftarrow (1-\alpha)v_{left}$
    **end while**
**end if**
## Binary Search within $[v_{left}, v_{right}]$
**while** $v_{right} - v_{left} > \gamma$ **do**
    $v_{mid} \leftarrow (v_{right} + v_{left})/2$
    **if** $f_{\boldsymbol{\theta}}(\boldsymbol{x}_0 + v_{mid}\boldsymbol{s}) = t_0$ **then**
        $v_{left} \leftarrow v_{mid}$
    **else**
        $v_{right} \leftarrow v_{mid}$
    **end if**
**end while**
**return** $v_{right}$

Although it requires many samples to exactly compute the $h$ value, when integrating it in the zeroth order optimization algorithm we can usually ignore the coarse-grained search and directly enter the binary search phase. This is because the computation of $h$ values is needed for finite-difference gradient estimator (such as (3.3)), which means we already know $h(\boldsymbol{x})$ and want to compute a closeby point $h(\boldsymbol{x}+\beta\boldsymbol{u})$. Therefore, the queried direction will only be a slight modification of the previous one. In this case, we first increase or decrease $v$ in the local region to find the interval that contains the nearby boundary (e.g., $f_{\boldsymbol{\theta}}(\boldsymbol{x}_0 + v\boldsymbol{s}) = t_0$ and $f_{\boldsymbol{\theta}}(\boldsymbol{x}_0 + v'\boldsymbol{s}) \neq t_0$), then conduct a binary search to find the final value of $h$.

**Algorithm 2** Hard-label black-box attack (OPT attack) in (Cheng et al., 2019a).

1: **Input:** Hard-label model $f_{\boldsymbol{\theta}}$, original sample $x_0$, initial $\boldsymbol{s}_0$, total iteration $T$, step size $\{\eta_t\}$
2: **for** $t = 0, 1, 2, \ldots, T$ **do**
3: Evaluate $h(\boldsymbol{s}_t)$ and $h(\boldsymbol{s}_t + \beta \boldsymbol{u})$ using Algorithm 1
4: Compute the estimated gradient $\overline{\boldsymbol{g}}$ using (3.3) and (3.4).
5: Update $\boldsymbol{s}_{t+1} = \boldsymbol{s}_t - \eta_t \overline{\boldsymbol{g}}$
6: **end for**
7: **return** $\boldsymbol{x}_0 + g(\boldsymbol{s}_T) \frac{\boldsymbol{s}_T}{\|\boldsymbol{s}_T\|}$

With the local computation of $h(\boldsymbol{s})$, we can apply zeroth-order gradient descent to solve the optimization problem (3.9) in the hard-label setting. The procedure is summarized in Algorithm 2. If $g(\boldsymbol{s})$ is Lipschitz-smooth, then the number of iterations needed achieve stationary points using Algorithm 2 is given by Cheng et al. (2019a).

In the follow-up work, Cheng et al. (2020c) further improve the query efficiency of Algorithm 2 by estimating the sign value of $h(\boldsymbol{s} + \alpha \boldsymbol{u}) - h(\boldsymbol{s})$ instead of its exact value. The main idea is that estimating the sign will only require 1 single query (see below), which already gives sufficient information for gradient-based optimizers to progress. In the untargeted attack case the sign can be computed by

$$\operatorname{sign}\big(h(\boldsymbol{s} + \alpha \boldsymbol{u}) - h(\boldsymbol{s})\big) = \begin{cases} +1, & f_{\boldsymbol{\theta}}(x_0 + g(\boldsymbol{s}) \frac{(\boldsymbol{s} + \alpha \boldsymbol{u})}{\|\boldsymbol{s} + \alpha \boldsymbol{u}\|}) = t_0, \\ -1 & \text{otherwise.} \end{cases} \tag{3.10}$$

Essentially, for a new direction $\boldsymbol{s} + \alpha \boldsymbol{u}$, we test whether a point at the original distance $h(\boldsymbol{s})$ from $x_0$ in this direction lies inside or outside the decision boundary, i.e., if the produced perturbation will result in a wrong prediction by the classifier. If the produced perturbation is outside the boundary, i.e., $f_{\boldsymbol{\theta}}(x_0 + h(\boldsymbol{s}) \frac{(\boldsymbol{s} + \alpha \boldsymbol{u})}{\|\boldsymbol{s} + \alpha \boldsymbol{u}\|}) \neq t_0$, then the new direction has a smaller distance to its decision boundary and thus gives a smaller value of $h$. It indicates that $\boldsymbol{u}$ is a descent direction to minimize $h$.

The resulting attack, which is called Sign-OPT attack, follows the same procedure as in Algorithm 2 by replacing the gradient estimation in step 4 with the following sign estimator:

$$\overline{\boldsymbol{g}} = \sum_{j=1}^{q} \operatorname{sign}\big(h\big(\boldsymbol{s} + \alpha \boldsymbol{u}^{(j)}\big) - h(\boldsymbol{s})\big) \cdot \boldsymbol{u}^{(j)}, \tag{3.11}$$

and since each sign can be computed using a single query, this gradient estimator requires siginficaintly less number of queries compared with the original gradient estimator. Intuitively, this new gradient estimator computes the gradient sign on each randomly sampled direction and then averages the results. As sign already gives sufficient information for the optimizer to progress, the algorithm enjoys similar convergence speed while being much more query efficient. The corresponding convergence analysis of zeroth order optimization using this sign estimator is also given by Cheng et al. (2020c).

## 3.4 Transfer attack

Generally speaking, transfer attack means generating adversarial examples from a source model to attack a target model. This typically corresponds to the "no-box" setting, where an attacker has no information about the target model and takes another pretrained model as a source to generate adversarial examples. Although the models can be vastly different, some adversarial examples can transfer from one model to another (Papernot et al., 2016; Liu et al., 2017b), and several attempts have been made, most empirically, to study how to generate more transferable adversarial examples. For instance, Dong et al. (2018) boosted the transferability by integrating the momentum term into the iterative process. Other techniques like data augmentations (Xie et al., 2019), exploiting gradients of skip-connection (Wu et al., 2020a), also contribute to stronger transferable attacks. A large-scale attack transferability study among 18 pretrained ImageNet models is conducted by Su et al. (2018). Instead of using pretrained surrogate models, it is also possible to construct a "meta-surrogate model" based on a set of pretrained surrogate models, where the meta-surrogate model can generate more transferable adversarial examples. This can be done with a meta-learning framework, as proposed by Qin et al. (2021).

In the soft-label black-box attack setting, Papernot et al. (2017) propose to train a substitute model using iterative model queries, perform white-box attacks on the substitute model, and leverage the transferability of adversarial examples to attack the target model. However, training a representative surrogate for a state-of-the-art classifier is challenging due to the complicated and nonlinear classification rules of the machine learning model (e.g., neural networks) and high dimensionality of the underlying dataset. The performance of black-box attacks can be severely degraded

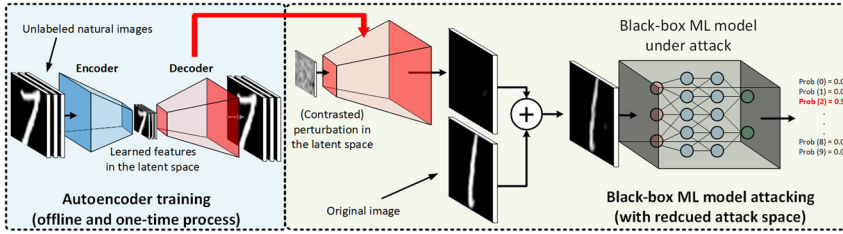


**Figure 3.2** Illustration of attack dimension reduction through a "decoder" in (Tu et al., 2019) for improving query efficiency in black-box attacks. The decoder has two modes: (i) An autoencoder (AE) trained on unlabeled natural images that are different from the attacked images and training data; (ii) a simple bilinear image resizer (BiLIN) applied channelwise to extrapolate low-dimensional feature to the original image dimension (width $\times$ height). In the latter mode, no additional training is required.

if the adversarial examples for the substitute model transfer poorly to the target model.

## 3.5 Attack dimension reduction

Different from the first-order convergence results, the convergence rate of zeroth-order gradient descent methods has an additional multiplicative dimension-dependent factor $d$. In the convex loss setting the rate is $O(\sqrt{d/T})$, sufficiently close to the optimal value, where $T$ is the number of iterations (Nesterov and Spokoiny, 2017; Liu et al., 2018a; Gao et al., 2014; Wang et al., 2018b). The same convergence rate has also been found in the nonconvex setting (Ghadimi and Lan, 2013). The dimension-dependent convergence factor $d$ suggests that vanilla black-box attacks using gradient estimations can be query inefficient when the (vectorized) image dimension $d$ is large, due to the curse of dimensionality in convergence.

To reduce the attack dimension and improve query efficiency in black-box attacks, Tu et al. (2019) propose to perform gradient estimation from a reduced dimension $d' < d$ to improve query efficiency. Specifically, as illustrated in Fig. 3.2, the additive perturbation to an image $\boldsymbol{x}_0$ is actually implemented through a "decoder" $D : \mathbb{R}^{d'} \mapsto \mathbb{R}^d$ such that $\boldsymbol{x} = \boldsymbol{x}_0 + D(\boldsymbol{\delta}')$, where $\boldsymbol{\delta}' \in \mathbb{R}^{d'}$. In other words, the adversarial perturbation $\boldsymbol{\delta} \in \mathbb{R}^d$ to $\boldsymbol{x}_0$ is in fact generated from a dimension-reduced space with an aim of improving query efficiency due to the reduced dimension-dependent factor in the convergence analysis. The **AutoZOOM** method proposed by Tu et al. (2019), which is short for **Auto**encoder-based **Z**eroth-**O**rder **O**ptimization **M**ethod, provides two modes for such a decoder $D$:

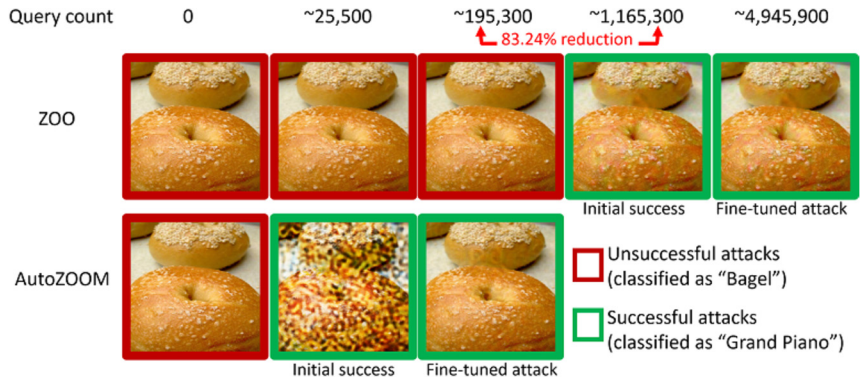


**Figure 3.3** AutoZOOM (Tu et al., 2019) significantly reduces the number of queries required to generate a successful adversarial Bagel image from the black-box Inception-v3 model.

- An autoencoder (AE) trained on unlabeled data that are different from the training data to learn reconstruction from a dimension-reduced representation. The encoder $E$ in an AE that compresses the data to a low-dimensional latent space, and the decoder $D$ reconstructs an example from its latent representation. The weights of an AE are learned to minimize the average $\ell_2$ reconstruction error. Note that training such an AE for black-box adversarial attacks is one-time and entirely offline (i.e., no model queries needed).
- A simple channelwise bilinear image resizer (BiLIN) that scales a small image to a large image via bilinear extrapolation.[4] Note that no additional training is required for BiLIN.

## 3.6 Empirical comparisons

There are two widely used criteria to evaluate the effectiveness of black-box attacks. Given a set of data samples to generate adversarial examples from, one criterion is to report the attack success rate and the average distortion given a model query budget. The other criterion is to report the attack success rate and the average query counts required to find an adversarial example with a given perturbation budget.

For soft-label black-box attacks, Fig. 3.3 displays a prediction-evasive adversarial example crafted via iterative model queries from a black-box image classifier (the Inception-v3 model Szegedy et al., 2016) trained

[4] See tf.image.resize_images, a TensorFlow example.

on ImageNet. The ZOO attack uses coordinatewise gradient estimation, whereas the AutoZOOM attack uses random vector-based gradient estimation. The results show that advanced gradient estimation and dimension reduction techniques can greatly improve the query efficiency while attaining visual quality in finding adversarial examples.

Next, we evaluate the following hard-label black-box attacks on three different standard datasets MNIST, CIFAR-10, and ImageNet-1K. For the black-box models, we use the convolutional neural networks provided by Carlini and Wagner (2017b) for both MNIST and CIFAR-10 and the pretrained Resnet-50 network (He et al., 2016) for ImageNet-1K.

We compare the performance of the following untargeted attacks:

- *Sign-OPT attack* (black box) (Cheng et al., 2020c)
- *OPT attack* (black box) (Cheng et al., 2019a)
- *Boundary attack* (black box) (Brendel et al., 2018): This method starts from a large adversarial perturbation and reduces it while staying adversarial via random walk on the decision boundary.
- *Guessing Smart Attack* (black box) (Brunner et al., 2019): This attack enhances boundary attack by biased sampling using some prior.
- *C&W attack* (white box) (Carlini and Wagner, 2017b): We use C&W $\ell_2$-norm attack as a baseline for the white-box attack performance.

For each attack, we randomly sample 100 examples from validation set and generate adversarial perturbations for them. We only consider examples that are correctly predicted by the target model. To compare different methods, we mainly use *average distortion* of successful adversarial attacks as the metric. We also report the *attack success rate (ASR)* for $B$ queries for a given $\ell_2$ perturbation threshold $\epsilon$, which is the percentage of the number of examples that have achieved an adversarial perturbation below $\epsilon$ with less than $B$ queries.

Table 3.1 summarizes the attack results. Given the same query budget, Sign-OPT consistently yields higher ASR and lower $\ell_2$ distortion (the latter is comparable to the C&W white-box attack), demonstrating its attack efficiency. Particularly for ImageNet dataset on ResNet-50 model, Sign-OPT attack reaches a median distortion below 3.0 in less than $30k$ queries, whereas other attacks need more than $200k$ queries for the same.

Fig. 3.4 shows some examples of adversarial examples generated by hard-label black-box attacks. The first two rows show comparison of Sign-OPT attack (Cheng et al., 2020c) and Opt attack (Cheng et al., 2019a), respectively, on an example from MNIST dataset. These figures show adversarial examples generated at almost same number of queries for both

**Table 3.1** $\ell_2$ Untargeted attack results. Comparison of average $\ell_2$ distortion achieved using a given query budget for different attacks. ASR stands for success rate.

| | MNIST | | | CIFAR10 | | | ImageNet (ResNet-50) | | |
|---|---|---|---|---|---|---|---|---|---|
| | **#Queries** | **Avg $\ell_2$** | **ASR ($\epsilon = 1.5$)** | **#Queries** | **Avg $\ell_2$** | **ASR ($\epsilon = 0.5$)** | **#Queries** | **Avg $\ell_2$** | **ASR ($\epsilon = 3.0$)** |
| Boundary attack | 4,000 | 4.24 | 1.0% | 4,000 | 3.12 | 2.3% | 4,000 | 209.63 | 0% |
| | 8,000 | 4.24 | 1.0% | 8,000 | 2.84 | 7.6% | 30,000 | 17.40 | 16.6% |
| | 14,000 | 2.13 | 16.3% | 12,000 | 0.78 | 29.2% | 160,000 | 4.62 | 41.6% |
| Guessing Smart | 4,000 | 1.74 | 41.0% | 4,000 | 0.29 | 75.0% | 4,000 | 16.69 | 12.0% |
| | 8,000 | 1.69 | 42.0% | 8,000 | 0.25 | 80.0% | 30,000 | 13.27 | 12.0% |
| | 14,000 | 1.68 | 43.0% | 12,000 | 0.24 | 80.0% | 160,000 | 12.88 | 12.0% |
| OPT attack | 4,000 | 3.65 | 3.0% | 4,000 | 0.77 | 37.0% | 4,000 | 83.85 | 2.0% |
| | 8,000 | 2.41 | 18.0% | 8,000 | 0.43 | 53.0% | 30,000 | 16.77 | 14.0% |
| | 14,000 | 1.76 | 36.0% | 12,000 | 0.33 | 61.0% | 160,000 | 4.27 | 34.0% |
| Sign-OPT attack | 4,000 | 1.54 | 46.0% | 4,000 | 0.26 | 73.0% | 4,000 | 23.19 | 8.0% |
| | 8,000 | 1.18 | 84.0% | 8,000 | 0.16 | 90.0% | 30,000 | 2.99 | 50.0% |
| | 14,000 | 1.09 | 94.0% | 12,000 | 0.13 | 95.0% | 160,000 | 1.21 | 90.0% |
| C&W (white-box) | – | 0.88 | 99.0% | – | 0.25 | 85.0% | – | 1.51 | 80.0% |

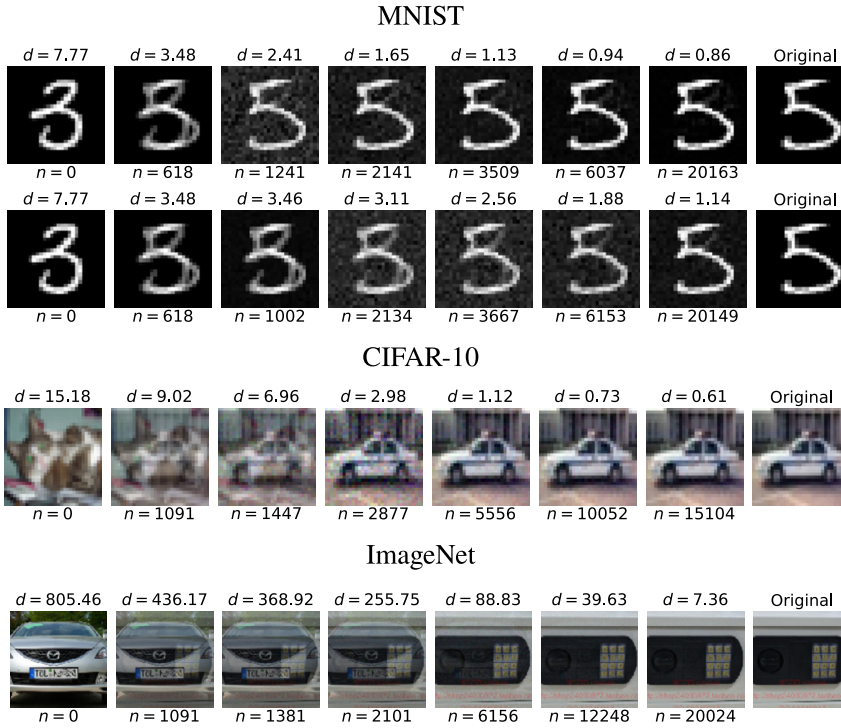


**Figure 3.4** Examples of hard-label black targeted attacks over query iterations. The first column is a randomly selected (unperturbed) image from the target class. Images in all other columns except for the last column (the original image) will give the same prediction as the first column. $\ell_2$ distortions and queries ($n$) used are shown above and below the images. First two rows: Example comparison of Sign-OPT (Cheng et al., 2020c) attack and OPT attack (Cheng et al., 2019a). Third and fourth rows: Examples of Sign-OPT attack on CIFAR-10 and ImageNet.

attacks. Sign-OPT method generates an $\ell_2$ adversarial perturbation of $0.94$ in $\sim 6k$ queries for this particular example, whereas Opt-based attack requires $\sim 35k$ for the same.

## 3.7 Proof of Theorem 1

Recall that the data dimension is $d$ and we assume $F$ to be differentiable and its gradient $\nabla F$ to be $L$-Lipschitz. Fixing $\beta$, we consider a smoothed version of $F$,

$$F_\beta(\mathbf{x}) = \mathbb{E}_{\mathbf{u}}\big[F(\mathbf{x}+\beta\mathbf{u})\big]. \tag{3.12}$$

Based on Gao et al. (2014, Lemma 4.1-a), we have the relation

$$\nabla F_\beta(\mathbf{x}) = \mathbb{E}_{\mathbf{u}}\left[\frac{d}{\beta}F(\mathbf{x}+\beta\mathbf{u})\mathbf{u}\right] = \frac{d}{b}\mathbb{E}_{\mathbf{u}}[\boldsymbol{g}], \tag{3.13}$$

which then yields

$$\mathbb{E}_{\mathbf{u}}[\mathbf{g}] = \frac{b}{d}\nabla F_\beta(\mathbf{x}), \tag{3.14}$$

where we recall that $\mathbf{g}$ is defined in (3.3). Moreover, based on Gao et al. (2014, Lemma 4.1-b), we have

$$\|\nabla F_\beta(\mathbf{x}) - \nabla F(\mathbf{x})\|_2 \leq \frac{\beta d L}{2}. \tag{3.15}$$

Substituting (3.14) into (3.15), we obtain

$$\|\mathbb{E}[\boldsymbol{g}] - \frac{b}{d}\nabla F(\mathbf{x})\|_2 \leq \frac{\beta b L}{2}.$$

This then implies that

$$\mathbb{E}[\boldsymbol{g}] = \frac{b}{d}\nabla F(\mathbf{x}) + \boldsymbol{\gamma}, \tag{3.16}$$

where $\|\boldsymbol{\gamma}\|_2 \leq \frac{b\beta L}{2}$.

Once again, by applying Gao et al. (2014, Lemma 4.1-b), we can easily obtain that

$$\mathbb{E}_{\mathbf{u}}\left[\|\boldsymbol{g}\|_2^2\right] \leq \frac{b^2L^2\beta^2}{2} + \frac{2b^2}{d}\|\nabla F(\mathbf{x})\|_2^2. \tag{3.17}$$

Now let us consider the averaged random gradient estimator in (3.4),

$$\overline{\mathbf{g}} = \frac{1}{q}\sum_{i=1}^{q}\mathbf{g}^{(i)} = \frac{b}{q}\sum_{i=1}^{q}\frac{F(\mathbf{x}+\beta\mathbf{u}^{(i)}) - F(\mathbf{x})}{\beta}\mathbf{u}^{(i)}.$$

Due to the properties of i.i.d. samples $\{\mathbf{u}^{(i)}\}$ and (3.16), we define

$$\boldsymbol{v} =: \mathbb{E}\left[\mathbf{g}^{(i)}\right] = \frac{b}{d}\nabla F(\mathbf{x}) + \boldsymbol{\gamma}. \tag{3.18}$$

Moreover, we have

$$\mathbb{E}\left[\|\overline{\mathbf{g}}\|_2^2\right] = \mathbb{E}\left[\left\|\frac{1}{q}\sum_{i=1}^{q}\left(\mathbf{g}^{(i)} - \boldsymbol{v}\right) + \boldsymbol{v}\right\|_2^2\right] \tag{3.19}$$

$$
\begin{aligned}
&= \|\boldsymbol{v}\|_2^2 + \mathbb{E}\left[\left\|\frac{1}{q}\sum_{i=1}^{q}\left(\mathbf{g}^{(i)} - \boldsymbol{v}\right)\right\|_2^2\right] \\
&= \|\boldsymbol{v}\|_2^2 + \frac{1}{q}\mathbb{E}\left[\left\|\mathbf{g}^{(1)} - \boldsymbol{v}\right\|_2^2\right] \qquad (3.20) \\
&= \|\boldsymbol{v}\|_2^2 + \frac{1}{q}\mathbb{E}\left[\|\mathbf{g}^{(1)}\|_2^2\right] - \frac{1}{q}\|\boldsymbol{v}\|_2^2, \qquad (3.21)
\end{aligned}
$$

where we have used the fact that $\mathbb{E}[\mathbf{g}^{(i)}] = \mathbb{E}[\mathbf{g}^{(1)}] = \boldsymbol{v}$ for all $i$. The definition of $\boldsymbol{v}$ in (3.18) yields

$$
\begin{aligned}
\|\boldsymbol{v}\|_2^2 &\leq 2\frac{b^2}{d^2}\|\nabla F(\mathbf{x})\|_2^2 + 2\|\boldsymbol{\gamma}\|_2^2 \\
&\leq 2\frac{b^2}{d^2}\|\nabla f(\mathbf{x})\|_2^2 + \frac{1}{2}b^2\beta^2L^2. \qquad (3.22)
\end{aligned}
$$

From (3.17) we also obtain that for all $i$,

$$
\mathbb{E}\left[\|\mathbf{g}^{(i)}\|_2^2\right] \leq \frac{b^2L^2\beta^2}{2} + \frac{2b^2}{d}\|\nabla F(\mathbf{x})\|_2^2. \qquad (3.23)
$$

Substituting (3.22) and (3.23) into (3.21), we obtain

$$
\begin{aligned}
\mathbb{E}\left[\|\overline{\mathbf{g}}\|_2^2\right] &\leq \|\boldsymbol{v}\|_2^2 + \frac{1}{q}\mathbb{E}\left[\left\|\mathbf{g}^{(1)}\right\|_2^2\right] \qquad (3.24) \\
&\leq 2\left(\frac{b^2}{d^2} + \frac{b^2}{dq}\right)\|\nabla F(\mathbf{x})\|_2^2 + \frac{q+1}{2q}b^2L^2\beta^2. \qquad (3.25)
\end{aligned}
$$

Finally, we bound the mean squared estimation error as

$$
\begin{aligned}
\mathbb{E}\left[\|\overline{\mathbf{g}} - \nabla f(\mathbf{x})\|_2^2\right] &\leq 2\mathbb{E}\left[\|\overline{\mathbf{g}} - \boldsymbol{v}\|_2^2\right] + 2\|\boldsymbol{v} - \nabla F(\mathbf{x})\|_2^2 \\
&\leq 2\mathbb{E}\left[\|\overline{\mathbf{g}}\|_2^2\right] + 2\|\frac{b}{d}\nabla F(\mathbf{x}) + \boldsymbol{\gamma} - \nabla F(\mathbf{x})\|_2^2 \\
&\leq 4\left(\frac{b^2}{d^2} + \frac{b^2}{dq} + \frac{(b-d)^2}{d^2}\right)\|\nabla F(\mathbf{x})\|_2^2 + \frac{2q+1}{q}b^2L^2\beta^2, \qquad (3.26)
\end{aligned}
$$

which completes the proof.

## 3.8 Extended reading

- Survey paper on zeroth-order optimization and applications (Liu et al., 2020a).

- Garcia et al. (2021) provide detailed analysis on hard-label attacks and dimension reduction.
- Chen et al. (2020a) proposed a query-efficient hard-label attack called Hopskipjump attack.
- Black-box attacks beyond pseudo-gradient descent: natural gradient descent (Zhao et al., 2020b), bandit algorithm (Ilyas et al., 2019), genetic algorithm (Alzantot et al., 2019), alternating direction method of multipliers (Zhao et al., 2019a), random walk around the decision boundary (Brendel et al., 2018), and ray searching (Chen and Gu, 2020).
- "No-box" attack that does not assume model queries nor accessing to the training data (Li et al., 2020b).