# Making Targeted Black-box Evasion Attacks Effective and Efficient

Mika Juuti, Buse Gul Atli, N. Asokan
{mika.juuti,buse.atli}@aalto.fi,asokan@acm.org
Aalto University, Finland

## ABSTRACT

We investigate how an adversary can optimally use its query budget for *targeted evasion attacks* against deep neural networks in a black-box setting. We formalize the problem setting and systematically evaluate what benefits the adversary can gain by using substitute models. We show that there is an exploration-exploitation tradeoff in that *query efficiency* comes at the cost of *effectiveness*. We present two new attack strategies for using substitute models and show that they are as effective as previous "query-only" techniques but require significantly fewer queries, by up to three orders of magnitude. We also show that an *agile adversary* capable of switching through different attack techniques can achieve pareto-optimal efficiency. We demonstrate our attack against Google Cloud Vision showing that the difficulty of targeted black-box attacks against real-world prediction APIs is significantly easier than previously thought (requiring ≈500 queries instead of ≈20,000 as in previous work).

## CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → **Neural networks**; *Object recognition*; Search methodologies;

## KEYWORDS

adversarial example, neural networks

**ACM Reference Format:**
Mika Juuti, Buse Gul Atli, N. Asokan. 2019. Making Targeted Black-box Evasion Attacks Effective and Efficient. In *12th ACM Workshop on Artificial Intelligence and Security (AISec '19), November 15, 2019, London, UK*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3338501.3357366

## 1 INTRODUCTION

The immense surge in the popularity of machine learning applications in recent years has been accompanied by concerns about their security and privacy. One such concern is *evasion*. Given a machine learning classifier (*victim*) and a particular input (*goal*), evasion is the process of finding an *adversarial example* [4, 40] that is sufficiently close to the goal, but will fool the victim classifier into outputting a different class label than that for the goal. When the adversary aims for a specific misclassified label, evasion is said to be *targeted*. For image classifiers, the difference between the adversarial example and goal images is imperceptible to humans.

Early techniques for finding adversarial examples against deep neural networks (DNNs) for image classification assumed a *white-box* setting, where the adversary knows the architecture and weights of the victim DNN [15, 40]. Since DNN cost functions are differentiable, these techniques calculated minimal changes (perturbations) to images which resulted in the DNN misclassifying the modified image. Later work addressed the black-box setting, where this perturbation cannot be directly calculated. Papernot et al [34] demonstrated that adversarial examples exhibit *transferability*: adversarial examples for one DNN (substitute model) are likely to be adversarial to another DNN (victim model) when they are trained on datasets with a similar distribution. Liu et al [29] empirically demonstrated that using an ensemble of substitute models instead of a single one results in better transferability. We call the state-of-the-art techniques in this category, such as [29] which rely exclusively on the use of ensembles as Ens. These are *efficient* in that the adversary needs to access the victim DNN only once (to test the adversarial example) but are not *effective* in targeted evasion because they may not always result in successful adversarial examples.

An alternative approach for targeted black-box evasion is where the adversary repeatedly queries the prediction API of a victim DNN, and estimates its gradient solely based on the responses to the queries. We call this class of techniques [5, 8, 22], Query-only: QO. These techniques are highly effective, reaching up to 100%. But they are *inefficient*. For example, even state-of-the-art methods [23] require up to 10,000 API queries for Google's Inception v3 [39] to reach a success rate of 95.6% assuming that the API returns probability scores for *all* labels. Real-life DNN APIs often work in a *partial information* (PI) setting [22], where the API only returns the *top-k scores* which further degrades the efficiency of Query-only techniques: the state-of-the-art to the best of our knowledge is [22] which reports requiring 350,000 queries reach a success rate of 93.6% on this network in a PI setting.

In this paper, we ask if we can design targeted black-box evasion techniques that are simultaneously efficient and effective. We argue that a realistic solution should (a) be designed for the partial-information setting since real-life APIs often use this setting[1]; and (b) use ensembles because of the widespread availability of *public, pre-trained* models. Our contributions are as follows:

- We show that Ensemble followed by Query-only (EQ) can outperform pure Query-only techniques (Sec. 4.6).
- We present PRISM and PRISM$_R$, two new targeted black-box evasion techniques that (a) starts from an input with the target label, and (b) repeatedly query the victim API (within a

[1] e.g. https://cloud.google.com/vision/, https://clarifai.com/demo


*AISec '19, November 15, 2019, London, United Kingdom*

ACM ISBN 978-1-4503-6833-9/19/11…$15.00
https://doi.org/10.1145/3338501.3357366

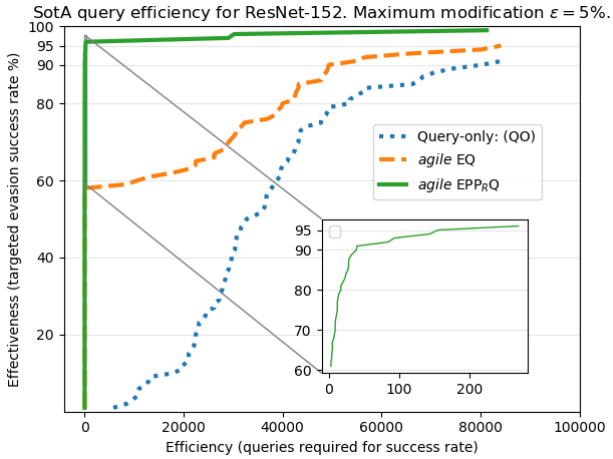



**Figure 1: Comparison of targeted black-box attacks against a partial information ResNet-152 using different strategies: Query-only [22] vs. EQ (using MIFGSM [9], followed by Query-only), vs. a fully agile adversary EPP$_R$Q switching through all methods (Sec.4.6).**

query budget) while concurrently using an ensemble to estimate victim's gradient in a PI setting. (Sec. 3.2). We show that the effectiveness of PRISM is comparable to Query-only on publicly available ImageNet models while typically requiring *fewer victim queries by three orders of magnitude* (Sec. 4.5).

- We systematically compare the different evasion approaches to show that an *agile adversary* can switch through these approaches in a particular order to achieve optimal efficiency, e.g. 13,000 queries on average to reach the effectiveness 94% on Inception v3 in PI setting (Sec. 4.6).
- We demonstrate the real-world applicability of PRISM by producing a targeted adversarial example against Google Cloud Vision, reducing the number of queries required from an estimate of 20,000 queries [22] to approximately 500 (Sec. 4.7).

Figure 1 shows the benefit of attacker agility by comparing different strategies: QO vs. EQ vs. a fully agile adversary EPP$_R$Q who switches through all methods (Ensemble, PRISM, PRISM$_R$ and Query-only). In Section 5, we discuss possible reasons why the PRISM approach is effective.

## 2 PRELIMINARIES

We first lay out definitions of frequently used concepts and functions in this work, with the focus on the image domain.

### 2.1 API definitions

*Classifier,* `clf`: a function that maps an arbitrary input *image* $\mathbf{x} \in [0,1]^{\mathrm{c}\times\mathrm{w}\times\mathrm{h}}$ to a vector of probabilities $\mathbf{p} \in [0,1]^{\mathrm{N}}$, denoting `clf`'s confidence in assigning $\mathbf{x}$ to any of the pre-defined classes $\{1,\ldots,\mathrm{N}\}$. The elements of $\mathbf{p}$ sum to 1. Here c refers to the number of color channels, w to the width and h to the height of input $\mathbf{x}$. A deep neural network dnn is a particular type of `clf` parameterized with $M$ sequential functions:

$$\mathsf{dnn}(\mathbf{x}) = f_M \circ f_{M-1} \circ \ldots f_2 \circ f_1 \circ \mathbf{x}, \tag{1}$$

where each function $f_i(\mathbf{z})$ can be expressed as $\sigma_i(\mathbf{w}_i^T\mathbf{z} + \mathbf{b}_i)$, where $\sigma_i$ is a (nonlinear) function, $\mathbf{w}_i$ is a weight matrix and $\mathbf{b}_i$ is a bias vector. In this work, we focus on dnn-based image classifiers. In a typical dnn, $\sigma_i$ is selected as a differentiable function. Therefore, it can calculate the gradient of classification error ($\nabla_x$dnn) in order to minimize this error easily in the training procedure [14].

*Preprocessor,* `pre`: A function that receives an input $\mathbf{x} \in \mathbb{R}^{\mathrm{c}\times\mathrm{h}\times\mathrm{w}}$ and produces an output $\mathbf{x'} \in \mathbb{R}^{\mathrm{c'}\times\mathrm{h'}\times\mathrm{w'}}$, and is primarily used for formatting, normalizing and resizing input $\mathbf{x}$ before it is classified by dnn, since dnn only processes fixed input sizes. `pre` can be used to break the differentiability property of dnn and act as a form of defense (shattered gradients, [3]).

*Postprocessor,* `post`: A function that receives input $\mathbf{p} \in \mathbb{R}^{\mathrm{N}}$ and produces an output $\mathbf{p'} \in \mathbb{R}^{\mathrm{N'}}$. It is used for formatting the output $\mathbf{p}$ of dnn, both for readability and limiting information from dnn. Common choices are:

- Identity function: $\mathtt{post}_I(\mathbf{x}) \leftarrow \mathtt{I}(\mathbf{x})\ \forall x$
- Label-only: $\mathtt{post}_{\mathcal{L}}(\mathbf{x}) \leftarrow \arg\max(\mathbf{x})$, i.e. the index $i \in \{1,\ldots,N\}$ with the largest value in $x$.
- Top-k: $\mathtt{post}_{\mathrm{k}}(\mathbf{x}) \leftarrow \mathbf{w}_{\mathrm{k}} \cdot \mathbf{x} \in \mathbb{R}^{\mathrm{N}}$, such that $\mathbf{w}_{\mathrm{k,i}} \leftarrow 1$ iff $i$ is among the k first values in $\mathtt{sorted}(\mathbf{x})$, where $\mathbf{x}$ is sorted in descending order. $\mathbf{w}_{\mathrm{k,i}} \leftarrow 0$ for others.

*API:.* A combination of pre, dnn and post that responds to arbitrary client, *CLI*, queries $\mathbf{x}$ with the following response:

$$_{\mathtt{pre}}\mathtt{API}_{\mathtt{post}}(\mathbf{x}) = \mathtt{post} \circ \mathsf{dnn} \circ \mathtt{pre} \circ \mathbf{x} \tag{2}$$

An ideal API always responds to *CLI*'s queries $\mathbf{x}$, as long as $\mathbf{x}$ is not malformatted. Prior work in black-box adversarial examples however typically do not use preprocessing on APIs $\mathtt{pre}(\mathbf{x}) = \mathtt{I}(\mathbf{x}) = \mathbf{x}$. In this paper, we only focus on preprocessing that resizes input correctly for dnn.

*White-box access: CLI* knows the precise definition of every intermediate function applied on any arbitrary input $\mathbf{x}$. Moreover, $\mathtt{pre}(\mathbf{x}) = \mathtt{post}(\mathbf{x}) = \mathtt{I}(\mathbf{x})$, i.e. *CLI* has access to all of dnn's outputs.

*Gray-box access: CLI* does not know the full definition of pre and post or the network parameters of dnn, but may know other information such as the architecture, hyper-parameters, training method and the training set of dnn [30].

*Black-box access: CLI* does not know the exact forms of any intermediate functions. Different authors define the minutiae of black-box API differently, we adapt these as follows (in order of decreasing privilege):

- *Maximum information* ($\mathtt{API}_I$): dnn is secret, while *CLI* has access to probabilities or logits from dnn for arbitrary input $\mathbf{x}$ [23].
- *Partial information* ($\mathtt{API}_k$): *CLI* has access to top-k output from dnn for arbitrary input $\mathbf{x}$ [22]. Generally, a realistic API has a long probability list $\mathbf{p} \in \mathbb{R}^{\mathrm{N}}$ with $N \geq 1000$ and returns a small subset of this list ($k \ll N$).
- *Label-only* ($\mathtt{API}_{\mathcal{L}}$): *CLI* has access only to labels from dnn for arbitrary input $\mathbf{x}$ [22].

## 2.2 Adversarial example definitions

*Adversarial example:* The adversary aims to produce an adversarial example $x_{\text{adv}}$ that is very similar to a goal image $x_{\text{goal}}$, but *evades* classification by API: $\texttt{API}(x_{\text{adv}}) \neq \texttt{API}(x_{\text{goal}})$ (non-targeted evasion). The similarity between $x_{\text{goal}}$ and $x_{\text{adv}}$ is often evaluated by an $L_p$-norm [37]: $\left\|x_{\text{goal}} - x_{\text{adv}}\right\|_{\text{p}}$. In this work, we set $p = \infty$ as is common. In targeted evasion, adversarial examples require that $y' \leftarrow \texttt{API}(x_{\text{adv}})$ for a pre-defined class $y' \neq \texttt{API}(x_{\text{goal}})$. Targeted evasion is described in Equation 3.

$$\begin{aligned} y' &\leftarrow \texttt{API}(x_{\text{adv}}) \\ \text{s.t. } &\left\|x_{\text{goal}} - x_{\text{adv}}\right\|_{\infty} \leq \epsilon \\ x_{\text{goal}}, x_{\text{adv}} &\in [0,1]^{\text{c}\times\text{w}\times\text{h}}, \end{aligned} \tag{3}$$

where $\epsilon$ is the allowed *perturbation* size.

*White-box attacker*, $\mathcal{A}_{\text{Wbox}}$, is a malicious client that has white-box access to API which it tries to evade. Since $\mathcal{A}_{\text{Wbox}}$ knows the precise definition of every intermediate function in dnn inside API, it is able to calculate the gradient of the classification error with respect to the input image: $\nabla_x \text{dnn}$. It uses this information to modify $x_{\text{adv}}$. Existing evasion methods such as single-step fast gradient sign method FGSM [16] and iterative version of it I-FGSM [27] find adversarial example $x_{\text{adv}}$ by maximizing the cross entropy loss function of dnn under $L_\infty$-norm.

*Black-box attacker*, $\mathcal{A}_{\text{Bbox}}$, is a malicious client that has black-box access to $\texttt{API}'$ that it tries to evade. Rotations and translation operations are often enough to create non-targeted evasion in black-box APIs [11]. However, in order to create targeted adversarial examples with a small perturbation, an approximation to the gradient information, $\hat{\text{G}}'_{\text{dnn}}$, of the target classifier $\text{dnn}'$ becomes necessary. There are two predominant ways of obtaining this information: (a) gradient approximation through *transferability* and (b) *finite-difference methods*.

*Transferability:* An adversarial example developed for evading one API (dnn) can be also adversarial to another $\texttt{API}'$ ($\text{dnn}'$), i.e.

$$\begin{aligned} y' &\leftarrow \texttt{API}'(x_{\text{adv}}) \\ &\text{s.t. Equation (3)} \end{aligned} \tag{4}$$

Recently, Adam et al. [1] found that the cosine similarity between the gradient $\nabla_x \text{dnn}'$ and the available gradient $\nabla_x \text{dnn}$ is a reliable estimator for transferability. Thus, implicitly the following approximation occurs during transferability:

$$\nabla_x \text{dnn} \approx \nabla_x \text{dnn}' \tag{5}$$

Liu et al. [29] states that transferability depends on the architectural similarity of $\text{dnn}'$ and dnn.

*Baseline transferability attack:* Adversary $\mathcal{A}_{\text{Bbox}}$ has a label-only black-box access to $\texttt{API}'$. It tries to evade $\texttt{API}'$ by creating adversarial examples using many available APIs and relies on the transferability property holds for the attacker's adversarial examples. Current state-of-the-art transferability attacks use ensembles of pre-trained DNNs as dnn [29]. A momentum-iterative version of FGSM (MIFGSM) [9], won both the targeted and non-targeted evasion competition at the NIPS workshop in 2017 and is since then considered to be the strongest $L_\infty$-bounded transferability attack. *We call this evasion method Ensemble, Ens.* We provide an illustration for Ens evasion for a toy example in Figure 2. The classifier in the figure is a multilayer perceptron (MLP) [32] with three classes.

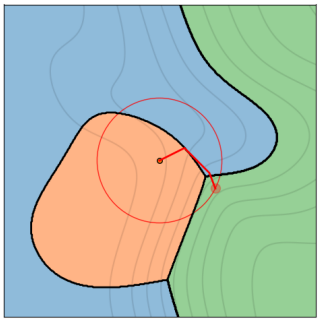


**Figure 2: Decision boundaries of an MLP classifier (3 classes). Targeted evasion with Ens, starting from *goal* at origin. Maximum modification $\epsilon$ indicated with a red square. Toy example: perturbations bounded with an $L_2$-norm.**

The effectiveness of transferability attacks is usually reported in terms of whether the victim model is fooled when it is queried with the adversarial example *once*. In Section 4.4, we explore whether allowing multiple queries brings any benefit to the adversary. Ens starts querying $\texttt{API}'$ once it has reached the maximum allowed modification $\epsilon$, as an effort to reduce the number of queries.

*Finite-difference methods,* FDM, which are also known as zero-order optimization methods, directly estimate gradients $\hat{\text{G}}_{\text{dnn}'}(x)$ for a target $\texttt{API}'$ by making repeated queries around $x$ [8, 22, 23, 42] and recording minute differences in the returned values. The baseline assumption is that $\texttt{API}'$ returns maximum information ($\texttt{API}'_I$). However, this is not a realistic assumption in practice. For example, Google Vision and Clarifai both return only top-k results from dnn.

Some papers [5, 22] report results under these more realistic APIs. The efficiency of FDMs under these API models degrades, e.g. [22] evaluates that the median number of queries grows to approximately 50,000 per example with $\texttt{API}'_{\text{k=1}}$, whereas they were found to be approximately 10,000 per example with $\texttt{API}'_I$. The number of queries further grows to 2.7 million with $\texttt{API}'_{\mathcal{L}}$

Evasion against $\texttt{API}'_{\text{k=1}}$ requires a change in the creation of targeted adversarial examples. Since $\texttt{API}'_{\text{k=1}}$ does not return feedback on any other class than the top-1 label, $x_{\text{adv}}$ must always remain as the top-1 label. Evasion needs to initialize the adversarial image with another image $x_{\text{start}}$ of the target class $y'$:

$$\begin{aligned} x^0_{\text{adv}} &\leftarrow x_{\text{start}} \\ \text{s.t. } &\texttt{API}'_{\text{k=1}}(x^0_{\text{adv}}) = y', \end{aligned} \tag{6}$$

We define j-th iteration of an adversarial image $x^j_{\text{adv}}$ as a series of $j$ modifications of $x^0_{\text{adv}}$ towards the original image $x_{\text{goal}}$. The distance between j-th adversarial image $x^j_{\text{adv}}$ and $x_{\text{goal}}$ gradually decreases when $j$ increases, so that the evasion process eventually ends with $x^i_{\text{adv}}$ ($i \geq j$) that is within a $\epsilon$-distance from a goal image $x_{\text{goal}}$:

$$y' \leftarrow \texttt{API}'_{k=1}(x^i_{adv})$$
$$\text{s.t. } \left\|x^i_{adv} - x_{goal}\right\|_\infty \leq \epsilon \quad (7)$$
$$1 \leq i \leq B,$$

where $B$ is maximum number of queries allowed by the $\texttt{API}'_{k=1}$. [22] reported success at attacking Google Cloud Vision (GCV, December 2017) using this strategy, successfully fooling the system with perturbation size $\epsilon = 25.5/255$. However, success came at a high cost: approximately some 170 gradient estimation steps, which we estimate is approximately 20,000 queries for one sample[2] they provide. *We call this evasion method Query-only, QO.* QO relies on the black-box optimization technique Natural Evolution Strategies (NES) by Wiestra et al. [44], which is used for gradient estimation. This is a common limitation in all Query-only methods, since they rely on querying $\texttt{API}'$ at all steps. The authors report a success rate of 93.6% on creating targeted adversarial examples for Inception v3 [39], using a budget of 1 million queries.

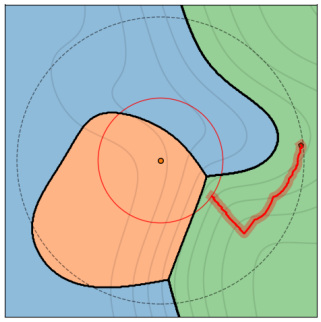


**Figure 3: Decision boundaries of an MLP classifier (3 classes). Targeted evasion with Query-only, starting from *start* and gradually approaching *goal* at origin. Maximum modification $\epsilon$ indicated with a red square. Toy example: perturbations bounded with an $L_2$-norm.**

## 3 PRISM

We further extend upon the previously mentioned methods, but evaluate a more realistic adversary. Our motivation is as follows. Due to the recent trend of striving for reproducibility in machine learning, tens or hundreds of pretrained models are available to the adversary. At the same time, model owners limit the APIs to only reveal partial information (Sec. 2.1). For that we propose the PRISM method: a novel way of attacking such APIs. We then define our evaluation criteria: *success* and *pareto-efficiency*. We begin with defining the adversary next.

### 3.1 Adversary model

*Goal and capabilities:* The goal of the adversary $\mathcal{A}_{\text{Bbox}}$ is to evade a black-box $\texttt{API}'$ hosting $\texttt{dnn}'$ which classifies ImageNet images. $\mathcal{A}_{\text{Bbox}}$ can query the $\texttt{API}'$ multiple times and continue attacking until a successful evasion attack is encountered or a reasonable query budget $B$ is exceeded. $\mathcal{A}_{\text{Bbox}}$ has access to several other publicly available pre-trained ImageNet DNNs [19–21, 38, 39] that it is free to combine in any way to reach its goal. $\mathcal{A}_{\text{Bbox}}$ is *agile*, meaning that given a set of evasion methods, $\mathcal{A}_{\text{Bbox}}$ will choose the method that is most likely to result in evasion with a minimum number of query. Given a setting $\texttt{s} = (x_{start}, x_{goal}, y', \texttt{API}')$, $\mathcal{A}_{\text{Bbox}}$ chooses an evasion method $m_i$ while considering the query budget $B$, and produces targeted adversarial example $x_{adv}$.

*Attack surface:* $\mathcal{A}_{\text{Bbox}}$ attacks a partial information $\texttt{API}'_{k=1}$: it has the format as defined in Equation 2 and returns the top-1 output from $\texttt{dnn}'$. $\mathcal{A}_{\text{Bbox}}$ does not know the native image size of $\texttt{dnn}'$ nor the resizing operator in $\texttt{pre}$. $\texttt{API}'$ is an ideal API: it always returns responses to queries.

### 3.2 PRISM attack technique

Next, we describe PRISM (*p*artial *i*nformation *s*ubstitute *m*odel), an approach for targeted evasion technique that combines strengths we identified in Ensemble and Query-only. We start by providing an illustration of PRISM in Figure 4. Conceptually, PRISM is similar to Query-only in Figure 3: it starts the evasion process with image $x_{start}$ of the target class $y'$ (Equation 6) and finishes when it finds a solution that is within an $\epsilon$-distance to *goal* image $x_{goal}$ (Equation 7). Initially, $x_{start}$ is at distance $\left\|x_{start} - x_{goal}\right\|_\infty = d \gg \epsilon$. The method consists of several iterations of increasing the classification likelihood of the j-th iteration $x^j_{adv}$ by stepping in the direction of the approximated gradient $\hat{\texttt{G}}_{\texttt{dnn}'}$ and then projecting it closer to $x_{goal}$. The procedure continues until the distance between $x^j_{adv}$ and $x$ decreases to $\epsilon$, and $x_{adv}$ is classified as the target class $y'$ by $\texttt{API}'_{k=1}$ or until a query budget $B$ has been exceeded. Although the process of finding $x_{adv}$ is similar to Query-only [22], the gradient estimator comes via substitute model ensembles and MIFGSM as is done in Ens [9]. We detail pseudocode for PRISM in Algorithm 1.

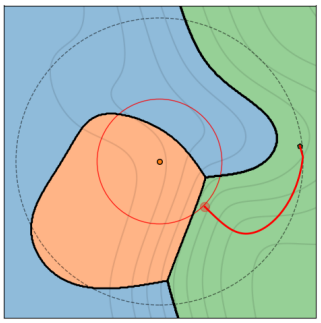


**Figure 4: Decision boundaries of an MLP classifier (3 classes). Targeted evasion with PRISM, starting from *start* and gradually approaching *goal* at origin. Maximum modification $\epsilon$ indicated with a red square. Toy example: perturbations bounded with an $L_2$-norm.**

The following steps highlight the differences between PRISM and Query-only algorithms:

[2]URL: https://www.labsix.org/media/2017/12/20/skier_adv.png. GCV has been retrained since then and the sample they provide does not evade GCV in March, 2019.

S1. We set $\hat{G}_{dnn'}(x) \leftarrow$ Ens$(x)$ in PRISM, which comes from the ensemble models and therefore does not consume queries. With default settings, Query-only [22] uses 100 queries to determine $\hat{G}_{dnn'}(x)$ via finite-difference methods.

S2. Query-only [22] uses line search [32] to find an appropriate step size with the purpose of reducing queries. PRISM does not do this, since no queries are used.

S3. Since PRISM relies on gradient estimates from substitute models, we can aggressively avoid using queries until $d$ has reached $\epsilon$. QO methods cannot do this, as they need to remain inside the top-k region in order to calculate gradients.

**Algorithm 1** PRISM attack technique

**Require:** Target $\texttt{API}_k(x')$, goal image $x_{goal}$, targeted class $y'$, starting image of target class $x_{start}$, gradient estimator $\hat{G}(x')$, goal $L_\infty$-distance $\epsilon \leftarrow 0.05$, patience $C \leftarrow 5$, counter $c \leftarrow 0$, update threshold $t_{adv} \leftarrow 20\%$, initial $L_\infty$-distance between $x_{start}$ and $x_{goal}$ $d \leftarrow 0.50$, $\delta_\epsilon \leftarrow 0.005$.

$x' \leftarrow \text{Clip}(x', x_{start} - d, x_{start} + d)$
**while** $d \geq \epsilon$ and $\texttt{API}_{\mathcal{L}}(x_{adv}) \neq y'$ **do**
S1. $g' \leftarrow \hat{G}(x')$ {gradient estimation}
S2. $x' \leftarrow x_{adv} + \eta \cdot sign(g')$.
$x' \leftarrow \text{Clip}(x', x_{goal} - d, x_{goal} + d)$.
**if** $d = \epsilon$ **then**
topk $\leftarrow \texttt{API}_k(x')$.
**else**
S3. topk $\leftarrow [y']$. {Pseudolabel until $d \leftarrow \epsilon$}
**end if**
**if** $y' \in$ topk **then**
$x_{adv} \leftarrow x'$
**if** topk$[y'] \geq t_{adv}$ **then**
$x_{backtrack} \leftarrow x_{adv}$ {update high-confidence $x'$}
**end if**
$d \leftarrow max(\epsilon, d - \delta_\epsilon)$, $c \leftarrow 0$
**else**
$c \leftarrow c + 1$
**if** $c > C$ **then**
$x_{adv} \leftarrow x_{backtrack}$
**end if**
**end if**
**end while**

It is important to note that PRISM succeeds only as long as Relation 5 holds. For this, we calculate $\hat{G}_{dnn'}$ using a large ensemble (approximately 10 models). We further define PRISM$_R$ as a variant of PRISM, with an emphasis on diversity. Instead of always using the same ensemble for gradient estimation like in PRISM, PRISM$_R$ subsamples a random number of ensembles and calculates the gradient with these ensembles using Ens method.

# 4 EVALUATION

## 4.1 Experimental setup

We first describe our experimental setup, target models and black-box evasion methods. We take the 100 examples from ImageNet as our initial images. These images were used in prior research [29][3], where they were chosen randomly from the ImageNet validation set, such that they were classified correctly by all models in their experiments. We use these images as $x_{goal}$. Our adversary model also specifies $x_{start}$. Since the dataset [29] does not consider the partial information setting, we choose $x_{goal}$, $x_{start}$ and target class $y'$ from this dataset according to Algorithm 2. We use this setup for all 100 experiments.

**Table 1: Shorthand notation for DNNs.**

| | Classifier name (acronym) | Input size |
|---|---|---|
| 1 | SqueezeNet 1.0 (SN1.0) [21] | 224 |
| 2 | SqueezeNet 1.1 (SN1.1) [21] | 224 |
| 3 | ResNet-18 (RN18) [19] | 224 |
| 4 | ResNet-34 (RN34) [19] | 224 |
| 5 | ResNet-50 (RN50) [19] | 224 |
| 6 | DenseNet-121 (DN121) [20] | 224 |
| 7 | DenseNet-169 (DN169) [20] | 224 |
| 8 | DenseNet-201 (DN201) [20] | 224 |
| 9 | VGG11 [38] | 224 |
| 10 | VGG16 [38] | 224 |
| 11 | ResNet-101 (RN101) [19] | 224 |
| 12 | ResNet-152 (RN152) [19] | 224 |
| 13 | Inception v3 (IncV3) [39] | 299 |

**Algorithm 2** Experiment Setup

**Require:** dataset $\mathcal{D}$ with 100 entries $e_i$, each entry in format $e_i = (x_i, c_i)$, where $x_i$ refers to an image, and $c_i$ is the assigned class
**for** $i \leftarrow 0$ to 100 **do**
$x_{goal} \leftarrow x_i$
$x_{start} \leftarrow x_{i+1 \bmod 100}$
$y' \leftarrow c_{i+1 \bmod 100}$
**end for**

We use the classifiers defined in Table 1 in our evaluation. All classifiers process input of the size $[0,1]^{3\times224\times224}$, apart from Inception v3, which processes input of size $[0,1]^{3\times299\times299}$. Further, all classifiers expect input to be normalized with RGB color-channel means $(0.485, 0.456, 0.406)$ and standard deviations $(0.229, 0.224, 0.225)$, whereas Inception v3 expects input to be normalized to range $[0,1]$ in all color channels. We re-normalize data to the correct range before processing it each classifier, and ensure that images are of correct size with bilinear interpolation.

We define our target APIs and substitute model ensembles in Table 2. For target APIs, we chose Inception v3, ResNet-101, ResNet-152 and VGG16 to experiment with different architecture choices having a low error rate in ImageNet examples[4]. The choice for ensemble components is also shown in Table 2. We choose to divide the target models and ensemble models in this way to study the effect of PRISM between similar architectures (ResNet models, VGG models) and different architectures (Inception as target model). We

[3] https://github.com/sunblaze-ucb/transferability-advdnn-pub/blob/master/data/image_label_target.csv
[4] https://pytorch.org/docs/stable/torchvision/models.html

**Table 2: Ensemble models and target API used in this work. Shorthand notation from Table 1.**

| Target | Ensemble components |
|---|---|
| IncV3 | Models 1–10 |
| RN101 | Models 1–10 |
| VGG16 | Models 1–9, 11 |
| RN152 | Models 1–11 |

**Table 3: Evasion methods. Methods 1–3 use substitute model ensembles from Table 2, while Method 4's formulation does not make use of substitute models. Method 1 starts adversarial example creation from the goal image $x_{\text{goal}}$, while Methods 2–4 have separate goal images and start images.**

| | Method | Grad. est. $\hat{\mathsf{G}}_{\text{dnn}'}$ | Start class of $x_{\text{start}}$ |
|---|---|---|---|
| 1 | Ens | MIFGSM [9], full ens. | $\texttt{API}'_{\mathcal{L}}(x_{\text{goal}})$ |
| 2 | PRISM | MIFGSM [9], full ens. | $y'$ |
| 3 | PRISM$_R$ | MIFGSM [9], subset ens. | $y'$ |
| 4 | QO | NES [44] | $y'$ |

also formulate target models as $\texttt{API}'_{\text{k=1}}$, i.e the attacker can only obtain top-1 output from the target model. The ensemble components were chosen with the principle of adding components of different architecture families (SqueezeNet, ResNet, VGG, DenseNet) until the graphics card memory (GeForce 1060, 6GB) was filled. While we could have an even created a stronger adversary by loading additional models in RAM, we found that this represented a reasonably strong and efficient adversary.

We investigate the methods shown in Table 3 in this work. Methods 1 (Ensemble) [9] and 4 (Query-only) [22] are state-of-the-art methods among black-box transferable evasion methods, and FDM attacks on partial information APIs, respectively. We investigate how well Ens can reach eventual success, by continuing to query API until a specified query budget is exceeded or success is encountered. Additionally, two variants are investigated. PRISM is a straightforward adaptation of Ens for the setting where $x_{\text{start}} \neq x_{\text{goal}}$. PRISM$_R$ is a variant of PRISM, using random subsets of the ensemble. We set the query limit to 100,000 queries for QO, and 1,000 for methods Ens, PRISM and PRISM$_R$ after a preliminary investigation. To ensure consistency, all methods are evaluated with PyTorch. We imported the code for QO from the authors' TensorFlow repository[5] with default parameters. Methods Ens, PRISM and PRISM$_R$ use MIFGSM with the suggested momentum parameter $\mu = 1$ and are evaluated with the same ensembles. We evaluate adversarial examples with maximum perturbation $\left\|x_{\text{adv}} - x_{\text{goal}}\right\|_\infty \leq \epsilon = 0.05$, and step size 0.005 (in Ens, PRISM and PRISM$_R$), meaning that at least 10 MIFGSM steps are taking before querying the API for the first time. Methods Ens, PRISM and PRISM$_R$ start querying target APIs only after reaching this perturbation. To make results comparable, we assume no knowledge of the resizing operator on each target API. Instead, black-box attacker $\mathcal{A}_{\text{Bbox}}$ produces input of the size $[0,1]^{3\times224\times224}$ in all methods.

[5] https://github.com/labsix/limited-blackbox-attacks

**Table 4: Effectiveness of baseline black-box evasion methods Ens and QO, and an *agile* adversary EQ. *Success rate* and *average* number of queries required for success.**

| | One query | Up to 100,000 queries | |
|---|---|---|---|
| | Ensemble | Query-only | EQ |
| IncV3 | 12% : 1 | 88% : 44158 | 89%: 40029 |
| RN101 | 47%: 1 | 89%: 32864 | 96%: 18874 |
| VGG16 | 47%: 1 | 94%: 28875 | 94%: 17433 |
| RN152 | 58%: 1 | 91%: 34689 | 95%: 14754 |

## 4.2 Evaluation criteria

We next define criteria that we use to compare evasion methods.

*Success:* A boolean value denoting whether a targeted adversarial example $x_{\text{adv}}$ created by method $m_i$, for $\texttt{API}'$ such that $y' \leftarrow \texttt{API}'(x_{\text{adv}})$, using at most $B$ queries. *Success rate* refers to how often success occurred in an experiment.

*Pareto-efficiency:* given certain evasion setting $\mathsf{s}$, a set of methods $\{m_1, \ldots, m_L\}$ and a criteria metric $q(x_{\text{start}}, x_{\text{goal}}, y', \texttt{API}', m_i) = q(\mathsf{s}, m_i)$, a method $m_i$ is said to be pareto-efficient for setting $\mathsf{s}$ if

$$q(\mathsf{s}, m_i) \leq q(\mathsf{s}, m_j)\ \forall i, j \in 1, \ldots, L, i \neq j, \tag{8}$$

i.e. $m_i$ produces the smallest criteria metric $q$ for setting $\mathsf{s}$. Pareto-efficiency is a *descriptive* property: given an experiment, we may calculate this statistic in hindsight.

*Dominance:* a method $m_{i*}$ is said to dominate other methods in a range $q \in [a, b]$ if

$$\begin{aligned} \mathrm{E}(q(\mathsf{s}, m_{i*})) <& \mathrm{E}(q(\mathsf{s}, m_j))\ \forall i*, j \in 1, \ldots, L, i* \neq j \\ q* &= \min_i q(\mathsf{s}, m_i)\ \forall i \in \{1, \ldots, L\} \\ q* &\in [a, b], \end{aligned} \tag{9}$$

where $\mathrm{E}(\cdot)$ is the expectation operator, i.e. we expect that $m_{i*}$ will produce the smallest performance metric, given that the pareto-efficient choice produces a metric in the range $[a, b]$. We consider the performance criteria $\mathsf{q}*$ in this paper. $\mathsf{q}*$ is the minimum number of queries to $\texttt{API}'$ it requires to produce an adversarial example that is classified as class $y'$ on $\texttt{API}'$ given setup $s$. Dominance is a *predictive* property: given previous tests, we may extrapolate future performance.

## 4.3 Baseline evaluations and basic agility

We first evaluate the baseline methods Ensemble and Query-only and how an agile adversary can simply increase efficiency and effectiveness. We show these results in Table 4. The success rate of Ensemble on the first try is shown in the leftmost column (up to 58% on RN152). The success rate on IncV3 is only 12%. We attribute this to the resize operator in IncV3, which resizes input from $(3 \times 224 \times 224)$ to $(3 \times 299 \times 299)$ (cf. Table 1). For example, Xie et al. [45] too found that resizing and cropping operations can act as a form of defense against adversarial examples.

On the other extreme, Query-only reaches approximately 90% success rate on all target APIs with up to 100,000 queries. QO takes between 28,000 and 44,000 queries in average to succeed.

**Table 5: Effectiveness of black-box evasion methods, *success rate* and *median* number of queries required for success.**

| | Up to 1000 queries | | | |
|---|---|---|---|---|
| | Ensemble | PRISM | PRISM$_R$ | Query-only |
| IncV3 | 26%: 2 | 69%: 11 | 75%: 14 | 0%: - |
| RN101 | 83%: 1 | 88%: 8 | 93%: 12 | 0%: - |
| VGG16 | 82%: 1 | 89%: 10 | 90%: 13 | 0%: - |
| RN152 | 84%: 1 | 95%: 8 | 96%: 11 | 0%: - |

We argue that the pareto-efficient choice for $\mathcal{A}_{\text{Bbox}}$ is to switch from one method to another when it becomes apparent that the first method will not succeed. Such an *agile adversary* can combine the previous methods: after the first query, is done through Ens, the remaining 99,999 queries can be done with Query-only. We call this simple agile method *EQ*. We see in Table 4 that the success rate for EQ is between 0–7 percentage points (pp) higher than with QO only. In these cases, Ens efficiently (1 query) finds an adversarial example that QO fails at. This occurs especially on RN101 and RN152. We suspect this occurs due to similarity of some of the ensemble components (Table 2). EQ can decrease the average queries between 42 – 58% on RN101, VGG16 and RN152, i.e. models where Ens produces satisfactory transferability.

## 4.4 PRISM effectiveness

Next we evaluate variants of PRISM and compare them to the baseline methods. We show success statistics for each of these four methods in Table 5, given our experimental setup and a query budget of 1000 queries. Columns are arranged in the order of methods presented in Table 3. Columns 1 – 3 represent query use with substitute models, which we advocate in this paper. If we continue querying for up to 1000 times Ens reaches a high success rate on RN101, VGG16 and RN152 (between 82% – 84%). However, Ens success with 1000 queries on IncV3 is significantly lower: only 26%. PRISM and PRISM$_R$ reach 69% and 75% success rates respectively on IncV3, while limited to the same query budget as Ens. By comparison, QO does not succed since the query budget is too low.

By comparing Tables 4 and 5, we see that the success rate of PRISM$_R$ is in fact similar as QO on RN101, VGG16 and RN152, while for most of the cases requiring *3 orders of magnitude fewer* queries. We also see that in most cases, methods that enable higher success rate do this at the expense of a higher number of median queries. This motivates us to study the comparative effectiveness of different methods. Knowledge of such patterns can help in developing more efficient agile attacks than the one presented in Section 4.3.

## 4.5 Pareto-efficiency of methods

We show the query efficiency of the four methods in Figures 5(a) to 5(c) on IncV3, RN101 and VGG16. The results are sorted so that the examples that required the least number of queries are ordered to the left side. We see that some of the examples are significantly harder than others (positive trend across colors). The most effective methods are connected by a grey dotted line, denoting the pareto-efficient choices. We also see a progression that the most efficient methods on the left side do not work efficiently on the right side. It is *harder* to find adversarial examples in some experiments than others: by this we mean that the minimum number of queries required to create adversarial examples are bigger than in others. This hints at an inherent exploration-exploitation trade-off: methods that are the most efficient find the universally easy solutions quickly (good exploitation), but tend to underperform on more difficult tasks (poor exploration).

## 4.6 Dominance and efficient strategies

Factoring out the trivial case of transferability, we may ask what is the optimal evasion method, given a certain "hardness" of the task. Using the data points in the previous figure, we can predict which method performs best given a certain region of required queries. Dominance (Equation 9) can be the treated as a multiclass classification problem: finding the most efficient evasion method, given a certain region of required queries. We solve the problem with multinomial logistic regression. Dominance regions are shown in Figures 5(d) to 5(f), and give hints when it is sensible to switch evasion algorithms, calculated separately for each target model.

**Table 6: Approximate dominance regions of evasion methods in Figures 5(a) to 5(c) on IncV3, RN101 and VGG16.**

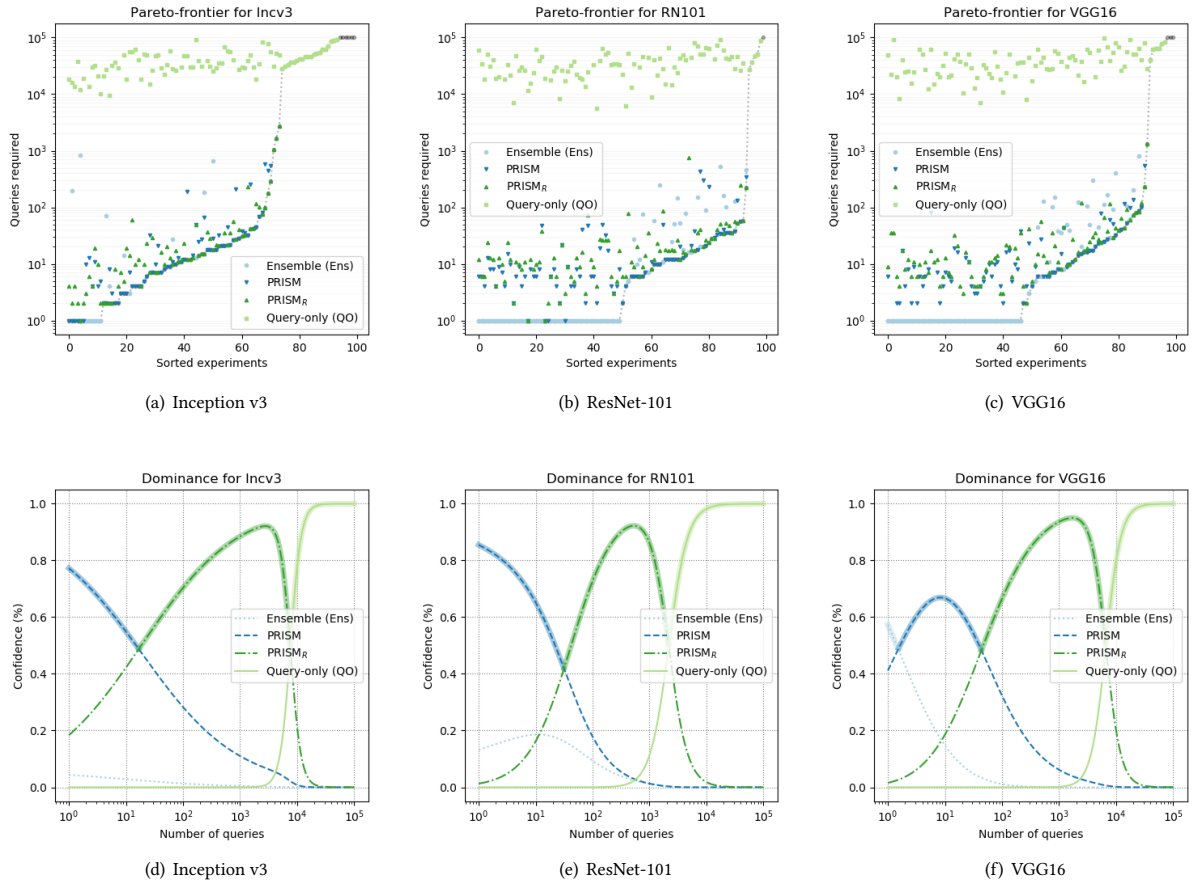
| Ensemble | PRISM | PRISM$_R$ | Query-only |
|---|---|---|---|
| 0–1 | 1–50 | 50–1,000 | 1,000–100,000 |

We identify approximate dominance regions in Table 6. The results indicate a optimal progression of methods to try out, in the order of 1 to 4. Using this we can calculate several instantiations of *efficient attacker strategies*, e.g. EPP$_R$Q tries Ens for the first query, PRISM during queries 2–50, PRISM$_R$ between 51–1000 and the rest with QO. Following this strategy, we calculate that the average number of queries required to create adversarial examples on each target model in Table 7. We compare this strategy to only using EPP$_R$ and the previously presented EQ and QO.

EPP$_R$Q is the most effective strategy. It reaches between *94% – 100% success rate* on the evaluated models, which is betwen 3 and 11 points higher than QO alone, while using between 2.27× and 24.43× less queries. EPP$_R$Q is also more efficient than EQ, while being significantly more effective. It is clear that PRISM is helpful towards increasing success rate and reducing queries.

**Table 7: Comparison on effectiveness and efficiency of EPP$_R$Q compared to EPQ, EQ and QO. Column 1 details the *success rate* and *average* number of queries for success. Columns 2 – 4 shows relative results of alternative attacker strategies: "comparative success rate in percentage points (pp): × (queries for success in EPP$_R$Q)". Row-wise best results are bolded.**

| | EPP$_R$Q | Cmp. EPQ | Cmp. EQ | Cmp. QO |
|---|---|---|---|---|
| IncV3 | **94**%:13477 | **±0 pp**: 1.12× | -5 pp: 1.97× | -6 pp: 2.27× |
| RN101 | **100**%:2882 | -1 pp: 1.56× | -5 pp: 6.55× | -11 pp:11.40× |
| VGG16 | **97**%: 3497 | **±0 pp**: **1.00**× | -3 pp: 4.98× | -3 pp: 8.26× |
| RN152 | **100**%:1419 | -1 pp: 1.36× | -5 pp:10.39× | -9 pp:24.43× |




(a) Inception v3 (b) ResNet-101 (c) VGG16

(d) Inception v3 (e) ResNet-101 (f) VGG16

**Figure 5: Subfigures 5(a) – 5(c) illustrate *pareto-efficiency* of Ens (light blue dot), and QO (light green square), PRISM (dark blue down-pointing triangle) and PRISM$_R$ (dark green up-pointing triangle), evaluated against number of queries required to generate an adversarial example against three models: IncV3, RN101 and VGG16. Subfigures 5(d) – 5(f) illustrate dominance regions. Dominance is calculated as per Equation 9 and illustrates the most effective methods given a certain query region. The thick lines illustrate when methods are optimal and the confidence of that optimality.**

**Table 8: Comparison on effectiveness and efficiency of EPP$_R$ compared to EQ and QO. Comparative success rate *success rate* and *average* number of queries for success. Row-wise best results are bolded.**

| | EPP$_R$ | Cmp. EQ | Cmp. QO |
|---|---|---|---|
| IncV3 | 74%: **107** | **+14 pp**: 387× | +13 pp: 427× |
| RN101 | 94%: **30** | **+1 pp**: 667× | -5pp: 1162× |
| VGG16 | 91%: **37** | **+3 pp**: 496× | **+3 pp**: 798× |
| RN152 | **98%**: **28** | -3 pp : 530× | -7 pp: 1246× |

We also compare EPP$_R$Q to EPQ, to confirm whether PRISM$_R$ has any impact on EPP$_R$Q. We see that the inclusion of PRISM$_R$ does not impact the success rate, but does impact the average number of queries on RN101 and RN152.

We additionally compare an attack strategy that only relies on gradient estimates from substitute model ensembles: EPP$_R$. We compare this strategy to EQ and QO in Table 8. In one case out of four, EPP$_R$ is most effective, and beats all strategies in efficiency: it uses 2.6–2.8 orders of magnitude fewer queries than the basic agile attack EQ, and *2.6–3.1 orders of magnitude fewer queries than QO.*

With these results we wish to highlight that agile attackers can present a realistic threat to prediction APIs. We next discuss the threat that PRISM poses to real-life prediction APIs.

## 4.7 Applicability to real-life APIs

As a proof-of-concept, we tested PRISM against Google Cloud Vision (GCV) API[6]. GCV does not exactly fit our adversary model (Sect. 3.1), as it is trained with non-public data and uses different

[6]Real-time attack demo: https://drive.google.com/open?id=1CWXlWd_rcDz6t0-zkG3jbsjY9UOLXaN8

labels than the substitute models have. This means that the approximation in Relation 5 does not hold well. As we saw in Section 4.5, some setups are harder than others. To provide a comparison with previous techniques, we took the same goal image as in Ilyas et al. [22], with the task of changing the image classification of an image with two skiers to "Canidae" (dog-like mammal), and set $\epsilon = 0.10$ (as in [22]). For the the attack on GCV, we thus relaxed the optimization step S3 (Algorithm 1), and queried all intermediate crafted samples (thus requiring at a minimum 40 queries to go from $d = 0.50$ to $d = \epsilon = 0.10$ with step size 0.01). Nevertheless, we still succeeded. Figure 6 shows the adversarial image that fools GCV in May 2019. The attack required in average 500 queries.

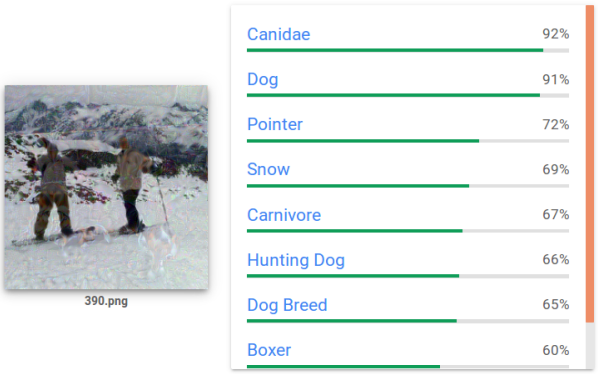



**Figure 6: Adversarial example by PRISM$_R$ against GCV (May 14th, 2019, $\epsilon = 10\%$). Found using 390 queries.**

Google Cloud Vision is under development. The original attack example in Ilyas et al. [22] does not fool GCV anymore, and we found that we were no longer able to create an adversarial example with their method using 100,000 queries in June 2019. We similarly found a large difference in the efficiency of our attack between January 2019 and June 2019. In January 2019, we found that approximately 400 queries were enough to fool GCV with modifications of size $\epsilon = 5\%$, whereas approximately the same amount of queries were required for $\epsilon = 10\%$ in May 2019. For example with 500 queries, the monetary cost of producing adversarial examples can be quite cheap: with current pricing ≈ $0.60 per example.

Comparing performance in-the-wild to laboratory conditions lets us determine both the current threat level of a technique, as well as future potential. Adversarial attacks on online prediction APIs like GCV are important to understand, since these may act as components in access control systems; for example in content filtering/moderation, or biometric-based premise access control.

## 5 DISCUSSION

Next, we study the effect of the size of the ensemble and attack effectiveness. In order to understand the relationship between the number of ensemble components and the effectiveness of substitute model attacks like PRISM and Ens, we calculated the success rate and the median number of queries required for success for different ensemble sizes in Table 9. We increased the number of ensemble components step-by-step from one to ten. We added models from Table 1 to the ensemble in the order of their ImageNet validation set accuracies[7]. As usual, we evaluate targeted adversarial examples with $\epsilon = 5\%$.

We make the following observations from Table 9. For a fixed $\epsilon$, we see that adding more components to the ensemble both *increases the success rate* while *decreasing* the median *number of queries*, for all attacks. Adding components is helpful, even when the component itself might have fairly low accuracy (SN1.1 & SN1.0). Table 9 also shows that the effectiveness of the attack increases rapidly when a component with a similar architecture is added to the ensemble (bold font). For example, when attacking VGG16, adding VGG11 to the ensemble increases the success rate from 54% to 85%. We see that PRISM performs better than Ens when several ensemble components are used. We provide intuition as to why PRISM performs better compared to Ens in Appendix A.

## 6 RELATED WORK

Our paper explored a limited knowledge *substitute learner* adversary model [31]. Other adversary models have also been considered:

*Limited knowledge* surrogate data*:* Tramer et al [41], Papernot et al [35], and others [25, 36], develop model extraction attacks against DNNs by training a substitute model using synthetic data. Labels are obtained by querying the target model. The substitute model is later used to form transferable adversarial examples.

*Model confidentiality:* There have been several attempts [13, 26, 28] to make the DNNs APIs *oblivious*, such that the API can process inputs correctly without learning anything about the client's input, the client simultaneously does not learning anything about the model behind the API.

*Other black-box attacks using substitute models:* Wang et al [43] explore transfer learning, where a student model for a specific application initialized by a publicly available pretrained teacher model (e.g., Inception v3, VGG16 etc). They show that one can compute adversarial perturbations that can mimic hidden layer representations copied from the teacher in order to fool the student model. They also used ensembles in the case of student model knowledge and an unknown teacher model. They show that their attack performance degrades if several layers of the student models are fine-tuned. Ji et al [24] maliciously train pre-trained models in order to implement model-reuse attacks against ML systems without knowing the developer's dataset or fine-tuning strategies. Hashemi et al [18] query the target model with images that come from a similar distribution as the training images of the target model, augment the dataset with random noise and use this augmented dataset to train a substitute model. They craft adversarial examples against the substitute model using Carlini & Wagner [7] method and perform transferability attacks. However, for training substitute models, they require logits from the target model in order to mimic decision boundaries and this attack can fail in case of more limited information. For example, Guo et al [17] implemented non-differentiable image transformation techniques as a preprocessor in order to defend against black-box and gray-box model evasion attacks. This type of gradient obfuscation techniques are effective when the adversary does not have the knowledge about the preprocessing method.

[7] https://pytorch.org/docs/stable/torchvision/models.html

**Table 9: Ablation study on the size of ensembles for PRISM and Ensemble. The ensemble size is gradually increased from only one (left) to ten (right). Components with highest top-1 accuracy evaluated first. Success rate and median number of queries for success shown.**

| Number of components. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Target model | IncV3 | | | | | | | | | |
| added model to ens. | DN201 | RN101 | RN50 | DN169 | DN121 | RN34 | RN18 | VGG11 | SN1.1 | SN1.0 |
| Ensemble (1 query) | 2% : 1 | 4%: 1 | 5%: 1 | 6%: 1 | 6%: 1 | 8%: 1 | 10%: 1 | 12%: 1 | 12%: 1 | 12%: 1 |
| Ensemble (up to 1000 queries) | 4% : 9 | 6%: 1 | 7%: 1 | 10%: 1 | 13%: 2 | 14%: 1 | 18%: 1 | 24%: 1 | 23%: 1 | 26%: 2 |
| PRISM (up to 1000 queries) | 2%: 89 | 6%: 60 | 12%: 28 | 16%: 27 | 26%: 14 | 41%: 15 | 54%: 12 | 60% 10 | 62%: 9 | 69%: 11 |
| Target model | RN101 | | | | | | | | | |
| added model to ens. | DN201 | **RN50** | DN169 | DN121 | **RN34** | VGG16 | **RN18** | VGG11 | SN1.1 | SN1.0 |
| Ensemble (1 query) | 4%: 1 | 11%: 1 | 14%: 1 | 21%: 1 | 34%: 1 | 34%: 1 | 39%: 1 | 45%: 1 | 44%: 1 | 47%: 1 |
| Ensemble (up to 1000 queries) | 7%: 1 | **29%**: 8 | 35%: 3 | 43%: 2 | 59%: 1 | 62%: 1 | 74%: 1 | 76%: 1 | 78%: 1 | 83%: 1 |
| PRISM (up to 1000 queries) | 7%: 84 | **34%**: 29 | 49%: 26 | 56%: 16 | **69%**: 14 | 69%: 12 | **83%**: 8 | 85%: 8 | 87%: 9 | 88%: 8 |
| Target model | VGG16 | | | | | | | | | |
| added model to ens. | DN201 | RN101 | RN50 | DN169 | DN121 | RN34 | RN18 | **VGG11** | SN1.1 | SN1.0 |
| Ensemble (1 query) | 1%: 1 | 3%: 1 | 3%: 1 | 6%: 1 | 13%: 1 | 13%: 1 | 21%: 1 | **41%**: 1 | 42%: 1 | 47%: 1 |
| Ensemble (up to 1000 queries) | 6%: 11 | 9%: 87 | 13%: 17 | 20%: 10 | 26%: 1 | 34%: 5 | 44%: 2 | **75%**: 1 | 80%: 1 | 82%: 1 |
| PRISM (up to 1000 queries) | 3%: 248 | 7%: 77 | 15%: 52 | 27%: 34 | 32%: 22 | 38%: 21 | 54%: 17 | **85%**: 12 | 86%: 11 | 86%: 10 |

*Finite-difference method attacks:* Similarly to us, Du et al [10] also consider the partial information attack, and separate between start and goal images. However, they initialize the starting image with a gray color and adopt NES for gradient estimation. They attacked a cloud API (Clarifai food detection) by choosing a valid label from top-k classes and minimizing the probability of so-called non-object or background predictions. Although their gray-image attack requires fewer queries than typical finite-difference methods as in [8, 22], the adversarial examples are unrecognizable by humans, which is different from our case. Brendel et al [5] introduce a decision-based attack which initializes the starting sample that is already adversarial and walks along the boundary between the adversarial and non-adversarial region as well as decreasing the distance towards the target image. They only used top label for initializing the starting image and finding the direction along the boundary, which is similar to our evaluations, but their attack requires more than an order of magnitude more iterations than the attacks evaluated in this work. Brunner et al. [6] suggest that start images can be 'initialized' by 'copy-pasting' content from other images, before query-only methods are used. Our initial tests suggested that such initialization can be done with PRISM. We leave a rigorous evaluation for future work. Other publications have used gradient-free optimization techniques, such as genetic algorithms [2] or greedy local search [33] over the image space in order to craft adversarial image in a black-box setting.

## 7 CONCLUSIONS

We presented targeted evasion attacks using substitute model ensembles for black-box APIs. We showed that such attacks can achieve very high effectiveness and efficiency: reaching similar effectiveness as state-of-the-art finite-difference attacks on partial-information APIs, while requiring up to 3 orders of magnitude fewer queries. We showed that the attack relies on the appropriateness of an implicit gradient estimation, and that this gradient approximation benefits from large substitute model ensembles. Query use with ensembles seems like an interesting direction to explore for future research. We argue that query-using substitute-model attacks form a pervasive threat against present-day cloud APIs due to the availability of substitute models and relatively cheap pricing.

## ACKNOWLEDGMENTS

This work was supported in part by the Intel (ICRI-CARS). We thank Samuel Marchal and Sebastian Szyller for interesting discussions, and Aalto Science-IT project for computational resources.

## REFERENCES

[1] George Adam, Petr Smirnov, Benjamin Haibe-Kains, and Anna Goldenberg. 2019. Reducing Adversarial Example Transferability Using Gradient Regularization. *arXiv preprint arXiv:1904.07980* (2019).
[2] Moustafa Alzantot, Yash Sharma, Supriyo Chakraborty, and Mani Srivastava. 2018. Genattack: Practical black-box attacks with gradient-free optimization. *arXiv preprint arXiv:1805.11090* (2018).
[3] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *International Conference on Machine Learning*. 274–283.
[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] Wieland Brendel, Jonas Rauber, and Matthias Bethge. 2018. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *International Conference on Learning Representations*.
[6] Thomas Brunner, Frederik Diehl, and Alois Knoll. [n. d.]. Copy and Paste: A Simple But Effective Initialization Method for Black-Box Adversarial Attacks. *arXiv:1906.06086* ([n. d.]).
[7] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 39–57.
[8] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. 2017. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 15–26.
[9] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. 2018. Boosting adversarial attacks with momentum. In *Proceedings of*

*the IEEE Conference on Computer Vision and Pattern Recognition*. 9185–9193.
[10] Yali Du, Meng Fang, Jinfeng Yi, Jun Cheng, and Dacheng Tao. 2018. Towards Query Efficient Black-box Attacks: An Input-free Perspective. In *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*. ACM, 13–24.
[11] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. 2018. A Rotation and a Translation Suffice: Fooling CNNs with Simple Transformations. In *ICLR 2018*.
[12] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, Pascal Frossard, and Stefano Soatto. 2017. Classification regions of deep neural networks. *arXiv preprint arXiv:1705.09552* (2017).
[13] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. 2016. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*. 201–210.
[14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.
[15] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *ICLR 2015*.
[16] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*.
[17] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. 2018. Countering Adversarial Images using Input Transformations. In *ICLR 2018*.
[18] Mohammad Hashemi, Greg Cusack, and Eric Keller. 2018. Stochastic Substitute Training: A Gray-box Approach to Craft Adversarial Examples Against Gradient Obfuscation Defenses. In *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*. ACM, 25–36.
[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
[20] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
[21] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. *arXiv:1602.07360* (2016).
[22] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. 2018. Black-box Adversarial Attacks with Limited Queries and Information. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*. https://arxiv.org/abs/1804.08598
[23] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. 2018. Prior convictions: Black-box adversarial attacks with bandits and priors. *arXiv preprint arXiv:1807.07978* (2018).
[24] Yujie Ji, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. 2018. Model-reuse attacks on deep learning systems. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 349–363.
[25] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N Asokan. 2019. PRADA: Protecting against DNN model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE.
[26] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. 2018. {GAZELLE}: A Low Latency Framework for Secure Neural Network Inference. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*. 1651–1669.
[27] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533* (2016).
[28] Jian Liu, Mika Juuti, Yao Lu, and N Asokan. 2017. Oblivious neural network predictions via minionn transformations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 619–631.
[29] 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).
[30] Dongyu Meng and Hao Chen. 2017. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 135–147.
[31] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. 2017. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of ACM AISec*.
[32] K Murphy. 2012. Machine learning: a probabilistic approach. *Massachusetts Institute of Technology* (2012).
[33] Nina Narodytska and Shiva Kasiviswanathan. 2017. Simple black-box adversarial attacks on deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 1310–1318.
[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 2017 ACM on Asia Conference on Computer and Communications Security*. ACM, 506–519.
[36] Li Pengcheng, Jinfeng Yi, and Lijun Zhang. 2018. Query-Efficient Black-Box Attack by Active Learning. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1200–1205.
[37] Mahmood Sharif, Lujo Bauer, and Michael K Reiter. 2018. On the suitability of lp-norms for creating and preventing adversarial examples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 1605–1613.
[38] K. Simonyan and A. Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* abs/1409.1556 (2014).
[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, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*. 601–618.
[42] Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. 2018. AutoZOOM: Autoencoder-based Zeroth Order Optimization Method for Attacking Black-box Neural Networks. *arXiv preprint arXiv:1805.11770* (2018).
[43] Bolun Wang, Yuanshun Yao, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2018. With great training comes great vulnerability: Practical attacks against transfer learning. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*. 1281–1297.
[44] Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. 2008. Natural evolution strategies. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. IEEE, 3381–3387.
[45] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. 2018. Mitigating Adversarial Effects Through Randomization. In *ICLR 2018*.

# A APPENDIX

Figure 7 shows an evasion example created by PRISM in January, 2019.

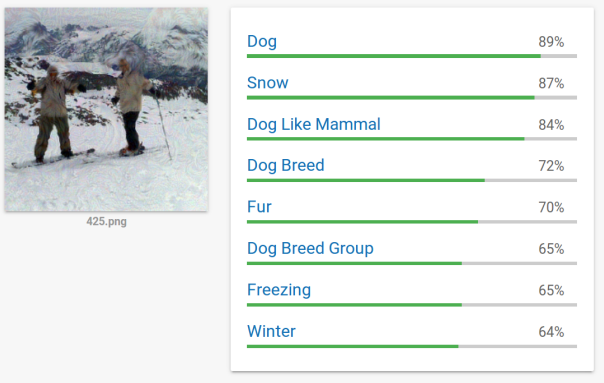
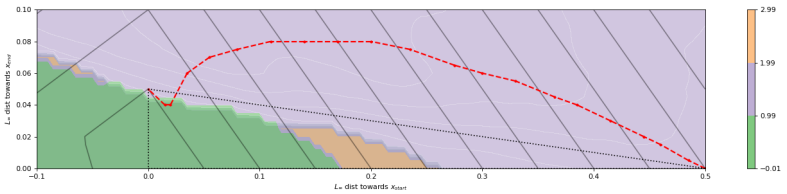



**Figure 7: Adversarial example by PRISM against GCV (January 7th, 2019, $\epsilon = 5\%$). Found using 425 queries.**

We show an example of how PRISM evades black-box APIs next. Figure 8 shows a resulting adversarial example, given a goal image and a start image. Figure 9 shows the process of finding the adversarial example with PRISM. The path that PRISM induces is shown in Figure 9(a). Although it is a black-box attack, it essentially follows a hill-climbing route due to the similarity of the gradients of the substitute model and target model. The results in Figure 9(a) suggest that a path between $x_{\text{start}}$ and $x_{\text{adv}}$ may be found without breaking the classification region of DNNs. The success behind PRISM implies that DNNs have connected but complex classification

(a) $x_{\text{goal}}$ (b) $x_{\text{adv}}$ found by PRISM (c) $x_{\text{start}}$

**Figure 8: An example of an adversarial example $x_{\text{adv}}$ found by PRISM, given goal image $x_{\text{goal}}$ and start image $x_{\text{start}}$. Created against against IncV3 with perturbation $\epsilon = 5\%$.**



(a) (Closed) linear span of ($x_{\text{start}}$, $x_{\text{adv}}$) found by PRISM. Confined to range ((-0.1, 0.5), (0. 0.1)). PRISM finds an $x_{\text{adv}}$ where a linear interpolation maintains the same classification region as $x_{\text{start}}$.

**Figure 9: Example of linear spans of $(x_{\text{start}}, x_{\text{adv}})$, starting with the same input, comparing two methods. The sample creation starts in the lower right corner (0.5,0) and progresses towards (0,0). The process ends when the coordinate (0, 0.05) is reached and the sample is still in the same classification region as in $x_{\text{start}}$. Evaluated with RN101. Green marks the original class (sea gull), purple the target class (water buffalo) and orange other classes. Contour regions inside the purple region mark logit values of RN101. Red dashed lines mark the path from $x_{\text{start}}$ to $x_{\text{adv}}$, projected down to the closest point in the linear span of $(x_{\text{start}}, x_{\text{adv}})$, in terms of $L_1$-distance. Diagonal lines marks successive 5% absolute increments in $L_\infty$-distance: from $x_{\text{start}}$. Classification regions sampled with resolution 121x21.**

(a) Ens (b) QO (c) PRISM

**Figure 10: Perturbations of adversarial examples. Created against against IncV3 with perturbation $\epsilon = 5\%$.**

regions. These results reinforce empirical results by Fawzi et al [12] who claim that classification regions of DNNs form connected regions, rather than isolated pockets.

We compare perturbations created by Ens, QO and PRISM in Figure 10, evaluated with IncV3. The goal image and start image are the same as in Figure 8(a) and 8(c). QO perturbations resemble random noise, whereas perturbations created via Ens and PRISM contain regular grid-resembling structures. The perturbation found by PRISM additionally contains localized perturbations influenced from $x_{\text{start}}$ (Figure 8(c)), as suggested by Figure 9(a). Note the gradient at the gull wing, greenish tint in background and retained buffalo head.