A Geometry-Inspired Decision-Based Attack

Yujia Liu* yjcaihon@mail.ustc.edu.cn
Seyed-Mohsen Moosavi-Dezfooli† seyed.moosavi@epfl.ch
Pascal Frossard† pascal.frossard@epfl.ch

Abstract

Deep neural networks have recently achieved tremendous success in image classification. Recent studies have however shown that they are easily misled into incorrect classification decisions by adversarial examples. Adversaries can even craft attacks by querying the model in black-box settings, where no information about the model is released except its final decision. Such decision-based attacks usually require lots of queries, while real-world image recognition systems might actually restrict the number of queries. In this paper, we propose qFool, a novel decision-based attack algorithm that can generate adversarial examples using a small number of queries. The qFool method can drastically reduce the number of queries compared to previous decision-based attacks while reaching the same quality of adversarial examples. We also enhance our method by constraining adversarial perturbations in low-frequency subspace, which can make qFool even more computationally efficient. Altogether, we manage to fool commercial image recognition systems with a small number of queries, which demonstrates the actual effectiveness of our new algorithm in practice.

1. Introduction

Deep neural networks have led to major breakthroughs in recent years and have been developing as powerful tools in various applications, including computer vision [15, 14], speech [19, 11], health-care [13, 17], etc. Despite their huge success, it has been shown that these networks are vulnerable to deliberate attacks. An adversary can generate adversarial examples [26] that look very similar to the original images, yet that can mislead a deep neural network to give an incorrect output.

Many existing attacks focus on white-box settings [8, 4], where adversaries have full access to all information of the model. They can attack models by computing gradients of the loss function. White-box attacks are interesting to derive a better understanding of deep neural networks, but they are unrealistic in a more practical setting, where adversaries have no knowledge about model parameters. In this case, called the black-box setting, some recent works focus on the score-based attacks [21, 5], where they make a large number of queries to the target model and exploit the output probabilities to generate adversarial examples. Most existing black-box attacks do however not consider that queries are usually costly in terms of both time and money, especially for commercial models. And, in fact, some commercial models only provide users with the final decision (top-1 label), and not even any output probabilities. This scenario corresponds to decision-based settings, where attacks [3] are actually the most challenging for adversaries.

In this paper, we propose a novel method of decision-based attack called qFool that computes adversarial examples with only a few queries. We consider both non-targeted and targeted attacks, advantageously exploiting the fact that the decision boundary is locally flat around adversarial examples. In non-targeted attacks, we estimate the gradient direction of the decision boundary only by relying on the top-1 label of each query result. We then search for an adversarial example directly from the original image in the estimated direction. In targeted attacks, we make iterative gradient estimations on multiple points along the boundary and seek for adversarial examples with the help of a target image. Both our non-targeted and targeted attacks can mislead networks with only few queries. Moreover, our experiments show that searching perturbations in a low-dimensional subspace is even more efficient in query-limited settings. We thus propose to extend qFool to subspaces in order to further reduce the number of queries. Finally, we use qFool to fool a famous commercial image recognition service with only a small number of queries to the model.

Our main contributions are the following:

- We propose a novel method, qFool, for generating adversarial examples when only having access to the top-1 label decision of a trained deep network model.
- We demonstrate that qFool needs fewer queries than previous decision-based attacks in both non-targeted and targeted settings.
- We enhance the proposed qFool with properly chosen

*University of Science and Technology of China
†École Polytechnique Fédérale de Lausanne, Switzerland

subspace constraints, thus further reducing the number of queries while maintaining a high fooling rate.

- We apply qFool to Google Cloud Vision, an example of commercially deployed black-box machine learning system. We demonstrate that such commercial classifiers can be fooled with a small number of queries.

2. Related Work

Szegedy et al. [26] were the first to show the vulnerability of deep networks to adversarial examples. Depending on how much adversaries know about networks, adversarial attacks are divided into white-box and black-box attacks.

In white-box setting, adversaries have all the knowledge about the network itself. It is a more favorable situation for adversaries to attack.

Gradient-based attacks. Most white-box attacks need the gradient of loss function of the network. Goodfellow et al. [8] proposed a fast gradient sign method (FGSM), which explores the gradient direction. The Jacobian saliency map attack (JSMA) by Papernot et al. [25] uses the saliency map to compute the adversarial perturbation. DeepFool by Moosavi-Dezfooli et al. [20] generates an adversarial example by exploring the nearest decision boundary and crossing it to deceive the network. C&W attack by Carlini and Wanger [4] solves a more efficient optimization problem under three distance measurements with the Adam optimizer. Houdini by Cisse [6] is an approach for generating adversarial examples specifically tailored for task losses, and it can be applied to a range of applications. Baluja and Fischer [1] trained Adversarial Transformation Networks to generate adversarial examples with a very large diversity against a target network or set of networks.

All these white-box attacks need to compute gradients of models, but sometimes attackers might have no access to the model or it may contain non-differentiable operations. So black-box attacks have gained more attention recently.

In the black-box attacks, adversaries do not have knowledge of the network. They only have access to the predictions by querying it. Black-box attacks can roughly be divided into three families: transfer-based, score-based and decision-based attacks.

Transfer-based attacks. Szegedy et al. [26] found that adversarial examples can transfer between models, thus enabling black-box attacks on deployed models. Transfer-based attacks aim to train a surrogate model by exploiting predictions from an underlying target model. Papernot et al. [22] showed that adversarial examples crafted based on surrogate models can mislead the target model. Liu et al. [18] developed an ensemble transfer-based attack and showed its high success against Clarifai.com, a commercial image classification service.

Score-based attacks. Adversaries only rely on the predicted scores of a model to generate adversarial examples in score-based attacks. Narodytska et al. [21] proposed the local-search attack that measures a models sensitivity to individual pixels. Chen et al. [5] proposed Zeroth Order Optimization (ZOO), that approximates gradient information by the symmetric difference quotient to generate adversarial examples. Hayes et al. [9] used the predicted scores to train an attacker model for black-box attacks.

Decision-based attacks. Decision-based attacks rely only on the class decision (top-1 label) of the model. A simple method is the additive Gaussian noise attack [23], which probes the robustness of a model to i.i.d. normal noises. Adversaries can also use uniform noise, salt and pepper noise, contrast reduction or Gaussian blur. But perturbations computed by these simple methods are usually very perceptible. Brendel et al. [3] proposed the Boundary attack, that can generate much smaller adversarial perturbations effectively. It starts from a large adversarial perturbation and iteratively reduces the norm of the perturbation by making the perturbed data point walk along the decision boundary while ensuring that it stays in the adversarial region.

Most black-box attacks usually require a large number of queries to the network model, while this is an important constraint in practice. Li et al. [16] introduced an active learning strategy to significantly reduce the number of queries for transfer-based attacks. For score-based attacks, Ilyas et al. [12] applied natural evolution strategies and only used two to three orders of magnitude fewer queries than previous methods. Bhagoji et al. [2] proposed random feature grouping and principal component analysis, which can achieve both high query efficiency and high success rate. Finally in decision-based attacks, since adversaries get less information from each query, the required number of queries is definitely large. Reducing the number of queries in decision-based is definitely challenging and in line with practical settings. In this paper, we propose a novel decision-based attack, which greatly improves the query efficiency even with commercial black-box systems.

3. Non-targeted decision-based attacks

We describe now our new decision-based attack algorithm. We consider a trained model with parameters θ that can be represented as $f_\theta : \boldsymbol{x} \rightarrow y$, where $\boldsymbol{x} \in \mathbb{R}^d$ is an input normalized image and y is the final decision of the model, *i.e.*, the top-1 classification label.

We first consider the non-targeted attack problem, where an adversary computes an adversarial perturbation $\boldsymbol{v}$ to change the estimated label of an image $\boldsymbol{x}_0$, *i.e.*, $f_\theta(\boldsymbol{x}_0 + \boldsymbol{v}) \neq f_\theta(\boldsymbol{x}_0)$. Furthermore, there shall be no perceptible difference between the perturbed and original image, *i.e.*, the Euclidean norm $||\boldsymbol{v}||_2 \leq \tau$ for some small τ. The adver-

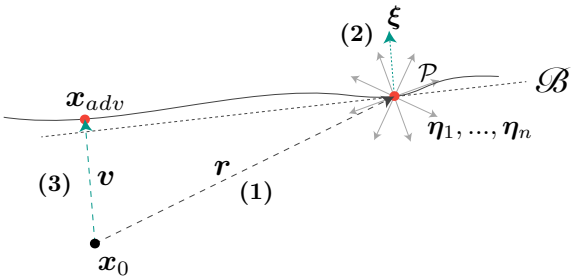


Figure 1: A simple illustration of three steps to compute an adversarial example by qFool. (1) Compute starting point $\mathcal{P}$ with a random perturbation $\boldsymbol{r}$. (2) Estimate gradient direction $\boldsymbol{\xi}$ at $\mathcal{P}$ by n perturbation vectors $\boldsymbol{\eta}_1, ..., \boldsymbol{\eta}_n$. (3) Search an adversarial example $\boldsymbol{x}_{adv}$ with a perturbation $\boldsymbol{v}$ in the estimated gradient direction $\boldsymbol{\xi}$.

sarial example is constructed by making queries to the unknown deep network. Theoretically, the more queries, the more information that one can get about the target model, and the smaller the adversarial perturbation. We aim to use as few queries as possible such that the norm of the perturbation is smaller than a certain threshold τ.

3.1. Query-efficient design

The basic idea of our decision-based attack algorithm is the following. Fawzi et al. [7] have shown that the decision boundary has a quite small curvature in the vicinity of adversarial examples. This indicates that some geometry properties of the decision boundary can be approximated in a similar way at different neighbour points around adversarial examples. We therefore exploit this observation to compute the adversarial perturbation $\boldsymbol{v}$. The direction of the minimal adversarial perturbation $\boldsymbol{v}$ for the input sample $\boldsymbol{x}_0$ is theoretically the gradient direction of the decision boundary at $\boldsymbol{x}_{adv}$. As we do not have full access to the classifier's structure in the black-box scenario, it is impossible to compute the direction directly. But, since the boundary is quite flat, the gradient of the classifier at point $\boldsymbol{x}_{adv}$ is almost the same as the gradient at other neighboring points on the boundary. Hence, the direction of $\boldsymbol{v}$ can be approximated well by $\boldsymbol{\xi}$, the gradient estimated at neighbour point $\mathcal{P}$. Then we can seek for an adversarial $\boldsymbol{x}_{adv}$ from $\boldsymbol{x}_0$ along the approximated direction $\boldsymbol{\xi}$. The basic logic is illustrated in Fig. 1. More details about each step are given below.

(1) Initial point. First, we find a small random noise $\boldsymbol{r}$ to perturb the original image $\boldsymbol{x}_0$ and identify a starting point $\mathcal{P}$ on the boundary. Formally,

$$\mathcal{P} := \boldsymbol{x}_0 + \min_{\boldsymbol{r}} ||\boldsymbol{r}||_2 \text{ s.t. } f_\theta(\mathcal{P}) \neq f_\theta(\boldsymbol{x}_0), \boldsymbol{r} \sim \mathcal{N}(0, \sigma) \tag{1}$$

To do that, we add some random Gaussian noises $\boldsymbol{r}_i \sim \mathcal{N}(0, \sigma_i)(i = 1, 2, ..., k)$ to the original image and make queries to the system, until the image $\mathcal{P} = \boldsymbol{x}_0 + \boldsymbol{r}_j$ is misclassified. In order to make the starting point closer to the decision boundary, we then do a binary search, in the direction of $\boldsymbol{r}_j$ to find a small Gaussian noise $\boldsymbol{r}$ that makes the perturbed image located on the boundary.

(2) Gradient estimation. The gradient of boundary $\nabla f(\mathcal{P})$ is estimated using only the top-1 label of the classifier. We randomly generate n small vectors $\boldsymbol{\eta}_1, ..., \boldsymbol{\eta}_n$ with the same norm to perturb $\mathcal{P}$ and query the classifier to get the predicted label $f(\mathcal{P} + \boldsymbol{\eta}_i)$. We define:

$$z_i = \begin{cases} -1 & f(\mathcal{P} + \boldsymbol{\eta}_i) = f(\boldsymbol{x}_0) \\ +1 & f(\mathcal{P} + \boldsymbol{\eta}_i) \neq f(\boldsymbol{x}_0) \end{cases}, \ i = 1, 2, ..., n \tag{2}$$

Theoretically, the vectors $(\boldsymbol{\eta}_1, ..., \boldsymbol{\eta}_n)$ are likely to be symmetrically distributed on both sides of the decision boundary. For large enough n, $\frac{1}{n}\sum_{i=1}^{n} z_i \boldsymbol{\eta}_i$ converges to the normal of the decision boundary as the components of $\boldsymbol{\eta}_i$'s along the decision boundary will cancel out each other. Hence, the direction of the gradient $\nabla f(\mathcal{P})$ can be estimated as

$$\boldsymbol{\xi} = \frac{\sum_{i=1}^{n} z_i \boldsymbol{\eta}_i}{||\sum_{i=1}^{n} z_i \boldsymbol{\eta}_i||_2}. \tag{3}$$

(3) Directional search. Due to the flatness of the boundary, the direction of the gradient at point $\boldsymbol{x}_{adv}$ can be approximated by the one at point $\mathcal{P}$, *i.e.*, $\nabla f(\mathcal{P}) \approx \boldsymbol{\xi}$, which can be computed by Eq. (3). We can thus find the adversarial example $\boldsymbol{x}_{adv}$ by perturbing the original image $\boldsymbol{x}_0$ in the direction of $\boldsymbol{\xi}$ until we hit the decision boundary. It only costs a few queries to the classifier by using a binary search algorithm.

The qFool design described above is highly parallelizable, which can bring important speed-ups. In the step of gradient estimation, where the vast majority of queries are consumed, we can conduct parallel queries, as they are independent of each other. On the contrary, in all previous decision-based attacks, the algorithms are usually executed in an iterative way, and the results are highly dependent on the ones of the previous iterations. Hence, our algorithm can not only reduce the number of queries, but also save considerable computation time due to parallelization.

3.2. qFool algorithm

The distance between the starting point $\mathcal{P}$ and original point $\boldsymbol{x}_0$ directly impacts the quality of adversarial perturbations computed in Section 3.1. If we find a better starting point $\mathcal{P}$ that is closer to the boundary, the final perturbation is generally smaller and more likely to be imperceptible.

Therefore, we design the qFool attack as an iterative algorithm where the starting point $\mathcal{P}$ is iteratively updated. The total number of queries n is divided into s iterations, *i.e.*, $n = n^{(0)} + n^{(1)} + ... + n^{(s-1)}$. s and $n^{(i)}$ $(i =$

Algorithm 1: Non-targeted qFool

Input: original image $\boldsymbol{x}_0$, initial estimation number n_u, a threshold ϵ.

$i = 0$;
Search starting point $\mathcal{P}_0$;
while $\sum_{j=0}^{i} n^{(j)} < n$ **do**
  $n^{(i)} = 0$;
  $\boldsymbol{x}_{adv}^{(i)} = \boldsymbol{x}'_{adv} = \mathcal{P}_i$;
  while $\frac{\left\|\boldsymbol{x}_{adv}^{(i)} - \mathcal{P}_i\right\|}{n^{(i)}+n_u} \leq \epsilon \cdot \frac{\left\|\boldsymbol{x}'_{adv} - \mathcal{P}_i\right\|}{n^{(i)}}$ ***and*** $\sum_{j=0}^{i} n^{(j)} < n$ **do**
    Make n_u new queries ;
    $\boldsymbol{x}'_{adv} = \boldsymbol{x}_{adv}^{(i)}$;
    $n^{(i)} = n^{(i)} + n_u$;
    Estimate gradient $\boldsymbol{\xi}_i$ at $\mathcal{P}_i$ with $n^{(i)}$ queries in full/sub space;
    Binary search for $\boldsymbol{x}_{adv}^{(i)} = \boldsymbol{x}_0 + \beta \cdot \boldsymbol{\xi}_i$;
  end
  $\mathcal{P}_{i+1} = \boldsymbol{x}_{adv}^{(i)}$, $i = i + 1$;
end
return $\boldsymbol{x}_{adv}^{(i-1)}$

$0, ..., s-1$) can be found adaptively. At i-th iteration, we query the model $n^{(i)}$ times. These queries permit to estimate the gradient direction $\boldsymbol{\xi}_i$ at starting point $\mathcal{P}_i$, and to search for the adversarial example $\boldsymbol{x}_{adv}^{(i)}$ in the direction of $\boldsymbol{\xi}_i$. Then we update the starting point $\mathcal{P}_{i+1}$ to $\boldsymbol{x}_{adv}^{(i)}$ in the next iteration.

The allocation of the number of queries $n = n^{(0)} + n^{(1)} + ... + n^{(s-1)}$ in the gradient estimation plays an important role in the efficiency of the algorithm. Here we find $n^{(i)}$ $(i = 0, 1, ...s-1)$ in a heuristic and greedy way. In one iteration, we first initialize the number of queries with a small n_u and compute an adversarial perturbation following the three steps described in Section 3.1. Then we gradually increase the number of queries in this iteration to get a smaller perturbation. This process will stop when the reduction rate of the perturbation is less than a preset threshold or the number of used queries reaches n. With this method, we can find an appropriate number of queries for each iteration $n^{(i)}(i = 0, 1, ..., s-1)$.

The norm of noise vectors in the step of estimating gradient direction is another important parameter. It should be small enough so that the boundary between the adversarial and non-adversarial regions can be considered as approximately linear. If so, we expect to get an equal split of $+1$ and -1 for z_i in Eq. (2). We dynamically adjust the norm $\omega^{(k)}$ in the k-th iteration according to the norm and the percentage of $z_i = +1$ in previous iterations. We formalize this adjustment in Eq. (4).

$$\begin{aligned} \omega^{(k+1)} &= \omega^{(k)} \cdot (1 + \phi^{(k)} \cdot \rho^{(k)}) \\ \phi^{(k)} &= -\text{sign}(\rho^{(k)}) \cdot \phi^{(k-1)} \end{aligned} \tag{4}$$

where $\omega^{(0)} = \omega_0$, $\phi^{(0)} = -1$. $\phi^{(k)}$ controls the increase or decrease of $\omega^{(k)}$, and $\rho^{(k)}$ is the difference between 0.5 and the proportion of z_i that are equal to $+1$. In this way, we can always ensure that the noise vectors are evenly distributed on both sides of the boundary.

3.3. qFool in subspace

In order to further reduce the number of queries, we now propose to concentrate on perturbations that belong to a properly chosen subspace, in order to simplify the search for adversarial perturbations. Indeed, it has been shown that adversarial perturbations that cause data misclassification can be found within specific subspaces [7]. Hence, in the step of gradient estimation, instead of using n noise vectors $\boldsymbol{\eta}_1, ..., \boldsymbol{\eta}_n$ of dimension d, we randomly choose n noise vectors in a pre-defined subspace of dimension $m \ll d$. The other steps of the algorithm remain the same. The non-targeted qFool algorithm is described in Alg. 1.

Generating adversarial perturbations in subspaces rather than full data space clearly reduces the complexity of querying the model. The more symmetrically the sampled noise vectors are distributed, the better the components in the other directions than the gradient direction will cancel out, so the more accurate the estimated gradient direction. Given a limited number of sampled noise vectors, these vectors are distributed more sparsely in the full space than in a lower dimensional subspace. Thus, the probability that these vectors are symmetrically distributed on both sides of gradient direction in a subspace is larger than the one in the full space. This leads to a smaller adversarial perturbation in a subspace. Alternatively, given a threshold of the norm of perturbation, the subspace method requires fewer queries.

4. Targeted attacks

We now extend our method to the targeted attack problem. In this case, an adversary tries to compute an adversarial perturbation $\boldsymbol{v}$ to change the label towards a specific target class t, *i.e.*, $f_\theta(\boldsymbol{x}_0 + \boldsymbol{v}) = t$. The norm of $\boldsymbol{v}$ should be small enough. The only information available from the classification system is the top-1 label and the number of queries is assumed to be limited.

Similarly to the non-targeted attack, we need a starting point in the target adversarial class, close to the decision boundary. But it is almost impossible to change the label of the original image to a specific target label by adding random noise. We therefore utilize an arbitrary image $\boldsymbol{x}_t$ belonging to the target class t to find a proper starting point $\mathcal{P}$.

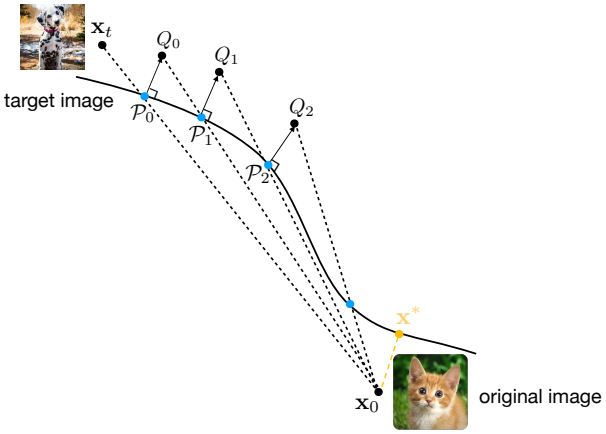


Figure 2: An illustration of qFool in targeted attacks.

As the distance between $\boldsymbol{x}_0$ and $\boldsymbol{x}_t$ can be quite large, the decision boundary between original and target adversarial region cannot be treated as flat. We cannot find a good estimation around $\mathcal{P}$ as we did in the non-targeted attack case. In the targeted attack, we therefore use a new way to compute and update the starting point $\mathcal{P}_i$ in each iteration, as shown in Fig. 2. We first perform a linear interpolation in the direction of $(\boldsymbol{x}_t - \boldsymbol{x}_0)$ to find a starting point $\mathcal{P}_0$,

$$\mathcal{P}_0 := \min_{\alpha}(\boldsymbol{x}_0 + \alpha \cdot \frac{\boldsymbol{x}_t - \boldsymbol{x}_0}{||\boldsymbol{x}_t - \boldsymbol{x}_0||_2}) \quad \text{s.t. } f_\theta(\mathcal{P}_0) = t \quad (5)$$

The gradient direction $\boldsymbol{\xi}_0$ at $\mathcal{P}_0$ can be estimated with the same method as for non-targeted attacks. We then get a point Q_0, which is slightly away from the boundary, by adding a small perturbation at $\mathcal{P}_0$ in this direction, *i.e.*, $Q_0 = \mathcal{P}_0 + \delta \cdot \boldsymbol{\xi}_0$. In the direction of $(Q_0 - \boldsymbol{x}_0)$, $\mathcal{P}_1$ can be searched with the method in Eq. (5). Motivated by $\mathcal{Q}_i$ $(i = 0, 1, ..., m)$, another point $\mathcal{P}_{i+1}$ near the boundary can be found quickly. Points $\mathcal{P}_i$ can walk along the boundary until the computed adversarial perturbation converges, *i.e.*, $(\mathcal{P}_m - \boldsymbol{x}_0) \parallel \nabla f(\mathcal{P}_m)$, and finally $\mathcal{P}_m$ is the adversarial example we get. Alg. 2 describes the algorithm in details. We note that the targeted qFool attack algorithm can also be executed in either the full data space or limited to a predefined subspace.

Algorithm 2: Targeted qFool

Input: original image $\boldsymbol{x}_0$, target image $\boldsymbol{x}_t$, initial estimation number n_u, a threshold ϵ.

$i = 0$;
Compute direction $\boldsymbol{\nu}_0 = \frac{\boldsymbol{x}_t - \boldsymbol{x}_0}{||\boldsymbol{x}_t - \boldsymbol{x}_0||_2}$;
Search starting point $\mathcal{P}_0$ in direction $\boldsymbol{\nu}_i$ by Eq. (5) ;
while $\sum_{j=0}^{i} n^{(j)} < n$ **do**
  $n^{(i)} = 0$;
  $\boldsymbol{x}_{adv}^{(i)} = \boldsymbol{x}'_{adv} = \mathcal{P}_i$;
  while $\frac{\left\|\boldsymbol{x}_{adv}^{(i)} - \mathcal{P}_i\right\|}{n^{(i)} + n_u} \leq \epsilon \cdot \frac{\left\|\boldsymbol{x}'_{adv} - \mathcal{P}_i\right\|}{n^{(i)}}$ ***and*** $\sum_{j=0}^{i} n^{(j)} < n$ **do**
    Make n_u new queries ;
    $\boldsymbol{x}'_{adv} = \boldsymbol{x}_{adv}^{(i)}$;
    $n^{(i)} = n^{(i)} + n_u$;
    Estimate gradient $\boldsymbol{\xi}_i$ at $\mathcal{P}_i$ with $n^{(i)}$ queries in full/sub space;
    $Q_i = \mathcal{P}_i + \delta \cdot \boldsymbol{\xi}_i$;
    $\boldsymbol{\nu}_i = \frac{Q_i - \boldsymbol{x}_0}{||Q_i - \boldsymbol{x}_0||_2}$;
    Search $\boldsymbol{x}_{adv}^{(i)}$ in the direction $\boldsymbol{\nu}_i$ by Eq. (5) ;
  end
  $\mathcal{P}_{i+1} = \boldsymbol{x}_{adv}^{(i)}$, $i = i + 1$;
end
return $\boldsymbol{x}_{adv}^{(i-1)}$

5. Experimental Results

We evaluate qFool using 250 images randomly chosen from the validation set of ImageNet on the networks VGG-19 [24], ResNet-50 [10] and Inception-v3 [25]. We measure the median of the average norm of all adversarial perturbations after a specific number of queries, defined by

$$\mathcal{M}_{\mathcal{X}}(n) = \underset{\boldsymbol{x}_i \in \mathcal{X}}{\text{median}} \left(\frac{1}{m} \left\| \boldsymbol{v}(\boldsymbol{x}_i, n) \right\|_2^2 \right), \quad (6)$$

where $\boldsymbol{v}(\boldsymbol{x}_i, n) \in \mathbb{R}^m$ is an adversarial perturbation generated for sample $\boldsymbol{x}_i$ using n queries to the model. The median is taken over the images in dataset $\mathcal{X}$.

5.1. Non-targeted attacks

For the non-targeted attack, Fig. 3 shows samples of adversarial perturbations on different models. We compare qFool with Boundary attack [3] on ImageNet. The results can be seen in Fig. 4. When the distances between adversarial examples generated by the two methods and original images are similar, qFool always spends fewer queries. qFool can converge much faster: if the number of queries is limited to a small value (*e.g.*, 10000), our method can achieve much better performance.

In addition, compared with qFool in full space, the subspace version can reduce the number of queries even more. Specifically, in this work, we use a 2-dimensional Discrete Cosine Transform (DCT) basis to define the low-dimensional subspace. Let $\mathcal{S} = \{\boldsymbol{\psi}_{i,j}\}_{i,j=0,...,\sqrt{m}-1}$ be the basis in the subspace of dimension m. When estimating gradients in the subspace, we use n noise vectors $\boldsymbol{\eta}_i^* = \mathcal{S}\boldsymbol{\gamma}_i$, where $\boldsymbol{\gamma}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_m)$, instead of vectors $\boldsymbol{\eta}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$.

In our experiments, the dimension d of full space is 224×224 or 299×299, we use 250 images in the ImageNet training set to find the best subspace. From Fig. 5, the best dimensions m of subspace range from 70×70 to 90×90.

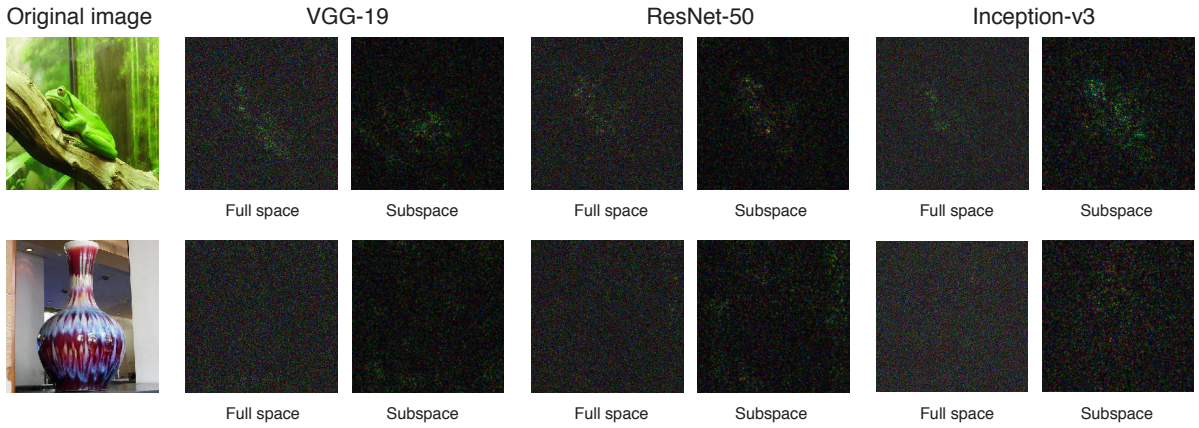
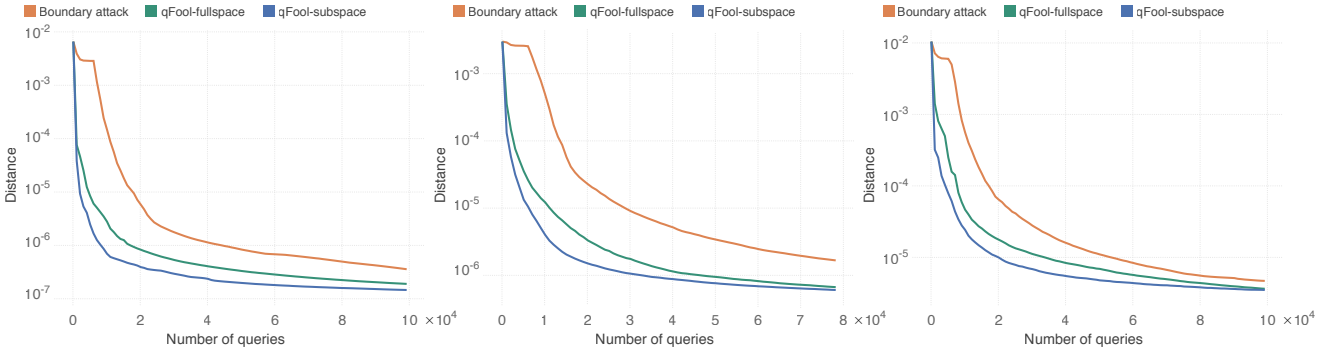


Figure 3: Original images and adversarial perturbations by qFool in the full space and subspace on different models for a "frog" and a "vase" (20,000 queries). The resulting perturbed images are classified as "snake" (first row) and "goblet" (second row). In addition, qFool in the subspace can get smaller perturbation than full space for the same number of queries.



(a) VGG-19 (b) ResNet-50 (c) Inception-v3

Figure 4: Distance (the median of MSE in Eq. (6)) between adversarial example and original image over numbers of queries in non-targeted qFool and Boundary attack [3] against different models.

We choose $\sqrt{m} = 75$, *i.e.*, we use a low frequency subspace $\mathcal{S}$ of dimension $m = 75 \times 75$. The method of using perturbation vectors in the subspace to estimate the gradient of boundary can achieve higher efficiency, as shown in Fig. 4.

We also compare qFool with several white-box methods on 50 images when applying no more than 10,000 queries, shown in Table. 1. FGSM and DeepFool are gradient-based attacks, that unsurprisingly need fewer predictions (queries). C&W, a gradient-based attack as well, achieves the smallest perturbation with more queries. As decision-based attacks, qFool can get much smaller perturbation than Boundary attack. Besides, qFool has an obvious advantage in the number of iterations, that can bring significant speed-ups because of its high parallelization.

In addition, we observe in our experiments that, for 250 different images, the number of iterations is no more than 4, and the allocation of n queries in consecutive iterations computed by our algorithm conforms to a similar rule: in the first few iterations, the numbers of queries are small, while the last iteration needs a much larger number of queries, shown in Fig. 6. It indicates that the basic logic of qFool algorithm is reasonable and effective. Allocating some small numbers of queries to the first several iterations can make the starting point closer to the theoretical adversarial example, which leads to a more locally flat boundary at the starting point, and thus a more accurate approximation of adversarial perturbation direction.

5.2. Targeted attacks

In targeted attacks, we use 250 samples in ImageNet. For each of them, we draw a target label randomly and pick one target image belonging to the target label randomly. We keep this original-target image pair consistent in all experiments of targeted attacks. The results are shown in Fig. 8. We can see that, when the number of queries is not so large, our algorithm is more efficient than the Boundary attack method [3]. Although the two curves cross at $100,000$ queries, qFool converges much faster. In the first several

Table 1: Comparison with different non-targeted attacks on the total number of backward gradient computation (#gradients), the total number of forward predictions (#predictions) and the total number of iterations (#iterations). (FGSM* is a modified version of FGSM that tries to find the smallest perturbation in the direction of the sign of the gradient.)

	#gradients	#predictions	#iterations	MSE		
				VGG-19	ResNet-50	Inception-v3
FGSM*	1	∼69	∼69	1.28e-6	6.37e-7	1.20e-5
DeepFool	∼20	∼20	∼ 2	2.76e-7	1.71e-7	6.12e-7
C&W	10000	10000	10000	2.22e-7	2.26e-7	3.25e-7
Boundary attack	0	∼10000	∼500	4.61e-4	1.15e-3	1.79e-3
qFool-fullspace	0	∼ 10000	∼3	1.16e-5	1.03e-4	2.48e-4
qFool-subspace	0	∼ 10000	∼3	6.15e-6	4.14e-5	1.80e-4

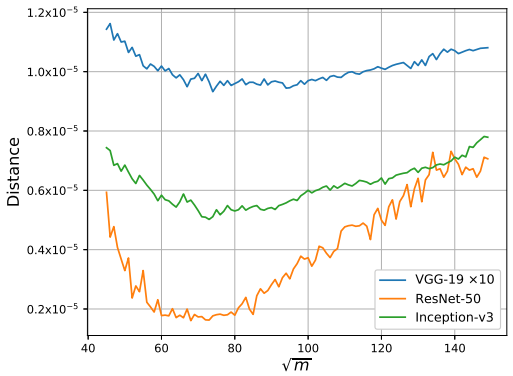


Figure 5: ℓ_2-distance between adversarial examples and original images in ImageNet training set over different subspace dimensions in non-targeted qFool attacks against three classifier models. (Results for VGG-19 are multiplied by 10 to make the dynamic range roughly the same.)

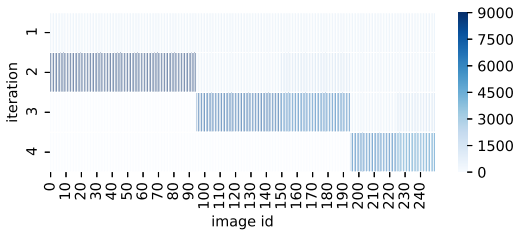


Figure 6: Distribution of 10,000 queries in different iterations for 250 images. We have grouped the images according to the number of used iterations. More than 30% images only need 2 iterations, and most queries appear in their second iteration. Other images need three or four iterations, and the last iteration similarly contain the most queries.

thousand queries, qFool can find a much smaller adversarial perturbation than Boundary attack. Therefore, in settings where the number of queries is limited, our algorithm is more efficient. Some samples are shown in Fig. 9.

Different target labels or target images obviously result

(a) (b) (c) (d)

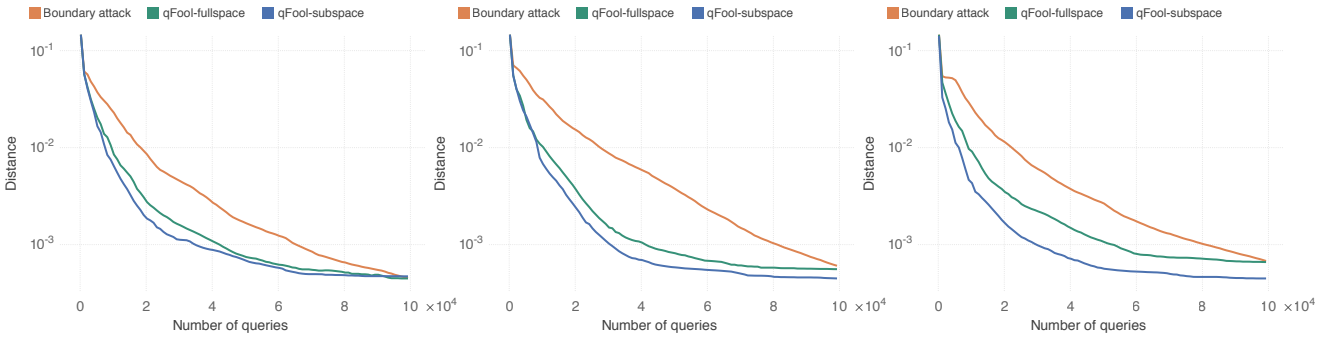
Figure 7: Attacking Google Cloud Vision. (a) original: 'dog like mammal' (b) adversarial: 'hair' (**1500** queries) (c) original: 'flag' (d) adversarial: 'sky' (**500** queries)

in different numbers of queries, as shown in Table 2. We list results at around 100,000 queries. It shows that MSE of adversarial examples is not positively correlated with the distance between the original and target image. Two images that are close in the spatial domain may not be similar in the feature domain.

5.3. Attacks on commercial classifiers

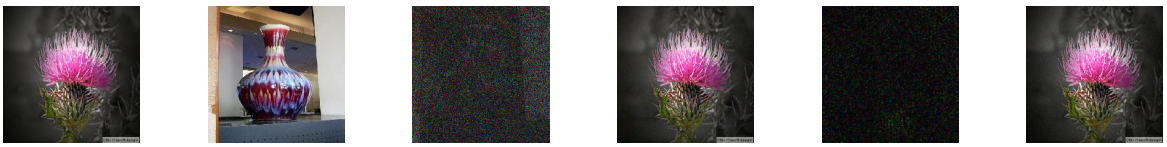
In order to demonstrate the effectiveness of qFool, we apply our non-targeted attack algorithm to one sample commercial system, namely Google Cloud Vision Image Recognition service, where users can get some tags associated with the input image and the corresponding confidence scores after querying the model. But we only use the final decision (top-1 tag) in order to attack this classifier.

Adversaries would ideally want to use the smallest number of queries to get an adversarial example with a relatively good visual quality. Since qFool performs particularly well when the number of queries is not very large, it is suitable for attacking commercial classifiers. Fig. 7 demonstrates some adversarial examples generated by only $500 \sim 1500$ queries. In the first 'dog' image, the MSE is 7.53e-5, which is totally imperceptible by human eyes. We only use 1500 queries to the model. Its new top-1 label is 'hair', and it indicates that the generated perturbation might mislead the model to ignore the whole content of this image yet focus on some local information. In the second 'flag' image, we use only 500 queries and the MSE achieves 2.66e-6. The perturbation makes the model ignore the existence of the object flag, and classify the image as 'sky'.



(a) VGG-19 (b) ResNet-50 (c) Inception-v3

Figure 8: Distance between adversarial example and original image versus numbers of queries in targeted qFool and Boundary attack [3] in three different network models.



(a) (b) (c) (d) (e) (f)

Figure 9: qFool in targeted attacks on VGG-19. (a) original:'cardoon' (b) target:'vase' (c) adversarial perturbation in full space (d) adversarial example in full space (e) adversarial perturbation in subspace (f) adversarial example in subspace. We use 50,000 queries in this experiment. The MSE is 4.40e-4 in full space and 1.12e-4 in subspace.

original class	target class								
"school bus"	"French loaf"			"golden retriever"			"sunflower"		
									
original distance	0.159	0.180	0.182	0.122	0.192	0.211	0.126	0.155	0.200
#query	99993	99127	99443	99514	99998	99135	99973	99938	99928
adversarial distance	5.27e-4	7.90e-4	5.79e-4	3.09e-4	7.07e-4	3.08e-4	7.06e-4	8.91e-4	3.60e-4

Table 2: Targeted qFool attacks on different target labels and target images. Our goal is to fool the "school bus" to be classified as three different classes: "French loaf", "golden retriever" and "sunflower". For each class, we select three different images randomly. Original distance is the ℓ_2 distance between the original image and target image. Adversarial distance is the ℓ_2 distance between the original image and the generated adversarial example after approximately 100,000 queries by targeted qFool attack.

6. Conclusion

In this work, we propose a novel query-efficient decision-based attack algorithm, namely qFool, for settings where the attacker only has access to the final decision (top-1 label) of a model. Our algorithm is based on the observation that the curvature of the boundary is small around adversarial examples. We use a simple way to estimate the gradient direction of the boundary, and construct imperceptible adversarial examples using the estimated direction. With this method, the required number of queries in both our non-targeted and targeted attacks can be reduced drastically compared to previous decision-based attacks. We also enhance qFool by constraining the gradient estimation to a pre-defined subspace, which further reduces the number of queries. We finally apply qFool on Google Cloud Vision, demonstrating the effectiveness of our attack on a sample real-world black-box classifier.

Acknowledgements

This work has been supported by a Google Faculty Research Award, and the Hasler Foundation, Switzerland, in the framework of the ROBERT project.

References

[1] Shumeet Baluja and Ian Fischer. Adversarial transformation networks: Learning to generate adversarial examples. *arXiv preprint arXiv:1703.09387*, 2017.

[2] Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. Practical black-box attacks on deep neural networks using efficient query mechanisms. In *Proceedings of the European Conference on Computer Vision*, 2018.

[3] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

[4] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP)*, 2017.

[5] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. 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*, 2017.

[6] Moustapha Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured prediction models. *arXiv preprint arXiv:1707.05373*, 2017.

[7] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: from adversarial to random noise. In *Proceedings of Advances in Neural Information Processing Systems*, 2016.

[8] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proceedings of International Conference on Learning Representations*, 2015.

[9] Jamie Hayes and George Danezis. Machine learning as an adversarial service: Learning black-box adversarial examples. *arXiv preprint arXiv:1708.05207*, 2017.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[11] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 2012.

[12] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Query-efficient black-box adversarial examples. *arXiv preprint arXiv:1712.07113*, 2017.

[13] Nikolaus Kriegeskorte. Deep neural networks: a new framework for modeling biological vision and brain information processing. *Annual review of vision science*, 2015.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of Advances in Neural Information Processing Systems*, 2012.

[15] Yann LeCun, Koray Kavukcuoglu, Clément Farabet, et al. Convolutional networks and applications in vision. In *ISCAS*, 2010.

[16] Pengcheng Li, Jinfeng Yi, and Lijun Zhang. Query-efficient black-box attack by active learning. *arXiv preprint arXiv:1809.04913*, 2018.

[17] Znaonui Liang, Gang Zhang, Jimmy Xiangji Huang, and Qmming Vivian Hu. Deep learning for healthcare decision making with emrs. In *Bioinformatics and Biomedicine*, 2014.

[18] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

[19] Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černockỳ. Strategies for training large scale neural network language models. In *Automatic Speech Recognition and Understanding*, 2011.

[20] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[21] Nina Narodytska and Shiva Prasad Kasiviswanathan. Simple black-box adversarial perturbations for deep networks. *arXiv preprint arXiv:1612.06299*, 2016.

[22] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of Asia Conference on Computer and Communications Security*, 2017.

[23] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox v0. 8.0: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017.

[24] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[25] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[26] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.