Blackbox Attacks via Surrogate Ensemble Search

Zikui Cai, Chengyu Song, Srikanth Krishnamurthy,
Amit Roy-Chowdhury, M. Salman Asif*
University of California Riverside

Abstract

Blackbox adversarial attacks can be categorized into transfer- and query-based attacks. Transfer methods do not require any feedback from the victim model, but provide lower success rates compared to query-based methods. Query attacks often require a large number of queries for success. To achieve the best of both approaches, recent efforts have tried to combine them, but still require hundreds of queries to achieve high success rates (especially for targeted attacks). In this paper, we propose a novel method for Blackbox Attacks via Surrogate Ensemble Search (BASES) that can generate highly successful blackbox attacks using an extremely small number of queries. We first define a perturbation machine that generates a perturbed image by minimizing a weighted loss function over a fixed set of surrogate models. To generate an attack for a given victim model, we search over the weights in the loss function using queries generated by the perturbation machine. Since the dimension of the search space is small (same as the number of surrogate models), the search requires a small number of queries. We demonstrate that our proposed method achieves better success rate with at least $30\times$ fewer queries compared to state-of-the-art methods on different image classifiers trained with ImageNet (including VGG-19, DenseNet-121, and ResNext-50). In particular, our method requires as few as 3 queries per image (on average) to achieve more than a 90% success rate for targeted attacks and 1–2 queries per image for over a 99% success rate for untargeted attacks. Our method is also effective on Google Cloud Vision API and achieved a 91% untargeted attack success rate with 2.9 queries per image. We also show that the perturbations generated by our proposed method are highly transferable and can be adopted for hard-label blackbox attacks. Furthermore, we argue that BASES can be used to create attacks for a variety of tasks and show its effectiveness for attacks on object detection models. Our code is available at https://github.com/CSIPlab/BASES.

1 Introduction

Deep neural network (DNN) models are known to be vulnerable to adversarial attacks [1–4]. Many methods have been proposed in recent years to generate adversarial attacks [2, 5–11] (or to defend against such attacks [6, 12–20]). Attack methods for blackbox models can be divided into two broad categories: transfer- and query-based methods. Transfer-based methods generate attacks for some (whitebox) surrogate models via backpropagation and test if they fool the victim models [3, 4]. They are usually agnostic to victim models as they do not require or readily use any feedback; and they often provide lower success rates compared to query-based methods. On the other hand, query-based attacks achieve high success rate but at the expense of querying the victim model several times to find perturbation directions that reduce the victim model loss [21–25]. One possible way to achieve a high success rate while keeping the number of queries small, is to combine the transfer and query attacks. While there has been impressive recent work along this direction [26, 10, 27, 11], the

*Corresponding authors: Zikui Cai (zcai032@ucr.edu) and M. Salman Asif (sasif@ucr.edu)

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

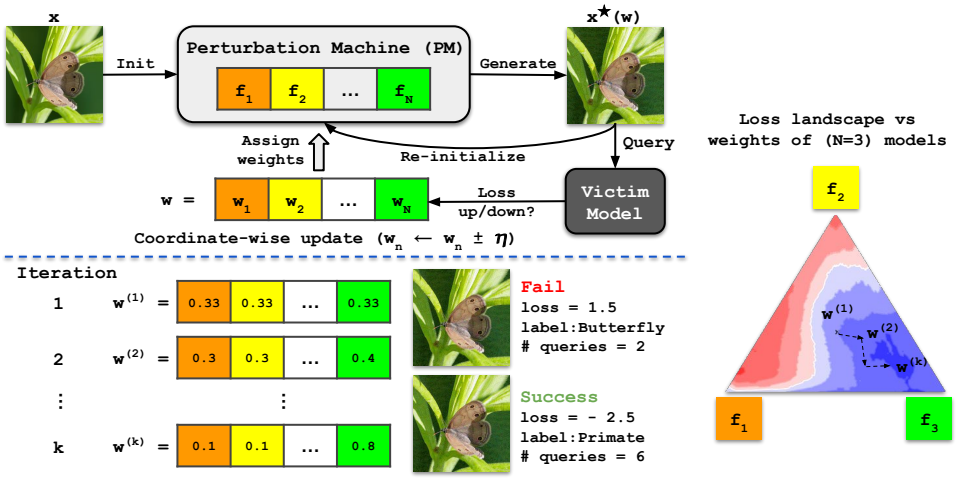


Figure 1: BASES for score-based attack. (**Top-left**) We define a perturbation machine (PM) using a fixed set of N surrogate models, each of which is assigned a weight value as $\mathbf{w} = [w_1, \ldots, w_N]$. The PM generates a perturbed image $x^\star(\mathbf{w})$ for a given input image x by minimizing the perturbation loss that is defined as a function of $\mathbf{w}$. To fool a victim model, we update one coordinate in $\mathbf{w}$ at a time while querying the victim model using $x^\star(\mathbf{w})$ generated by the PM. We can view this approach as a bi-level optimization or search procedure; the PM generates a perturbed image with the given weights $x^\star(\mathbf{w})$ in the inner level, while we update $\mathbf{w}$ in the outer level. (**Bottom-left**) We visualize weights and perturbed images for a few iterations. We stop as soon as the attack is successful (e.g. original label - 'Butterfly' is changed to target label - 'Primate' for targeted attack). (**Right**) Victim loss values for different weights along the Barycentric coordinates on the triangle. We start with equal weights (at the centroid) and traverse the space of $\mathbf{w}$ to reduce loss (concentrate on model f_3). Red color indicates large loss values (unsuccessful attack), and blue indicates low loss (successful attack).

state-of-the-art methods [27, 11] still require hundreds of or more queries to be successful at targeted attacks. Such attacks are infeasible for limited-access settings where a user cannot query a model that many times[28].

Given this premise, we design a new method for blackbox attacks via surrogate ensemble search (BASES), combining transfer and query ideas, to fool a given victim model with higher success rates and fewer queries compared to state-of-the-art methods. For example, our evaluation shows that BASES (on average) only requires 3 queries per image to achieve over a 90% success rate for targeted attacks, which is at least $30\times$ fewer queries compared to state-of-the-art methods [10, 11]. BASES consists of two key steps that can be viewed as bilevel optimization steps. 1) A perturbation machine generates a query for the victim model based on weights assigned to the surrogate models. 2) The victim model's feedback is used to change weights of the perturbation machine to refine the query. Figure 1 depicts these steps.

We first define a perturbation machine (PM) that generates a single perturbation to fool all the (whitebox) models in the surrogate ensemble. We use a surrogate ensemble for two reasons: 1) It is known to provide better transfer attacks [29, 7]. The assumption is that if an adversarial image can fool multiple surrogate models, then it is very likely to fool a victim model as well. For the same reason, an ensemble with different and diverse surrogate models provides better attack transfer. 2) Our main interest is in searching for perturbations that can fool the given victim model. A single surrogate model provides a fixed perturbation; hence, it does not offer flexibility to search over perturbations. To facilitate search over perturbations, we define the adversarial loss for the PM as a function of weights assigned to each model in the ensemble. By changing the weights of the loss function, we can generate different perturbations and steer in a direction that fools the victim model. It is worth noting that perturbations generated by a surrogate ensemble with an arbitrary set of weights often fools all the surrogate models, but they do not guarantee success on unseen victim models; therefore, searching over the weights space for surrogate models is necessary.

Since the number of models in the surrogate ensemble is small, the search space is low dimensional and requires extremely small number of queries compared to other query-based approaches. In our method, we further simplify the search process by updating one weight element at a time, which is equivalent to coordinate descent, which has been shown to be effective in query-based attacks

[21, 23]. Since it searches in orthogonal directions instead of estimating the full gradients, it is query efficient. This strategy requires 2 queries per coordinate update but offers success rates as good as that given by performing a full gradient update step (as shown in Section 4). Reducing the dimension of the search space while maintaining high success rate for query-based attacks is an active area of research [23, 10, 27, 11], and our proposed method pushes the boundary in this area.

We perform extensive experiments for (score-based) blackbox attacks using a variety of surrogate and blackbox victim models for both targeted and untargeted attacks. We select PyTorch Torchvision [30] as our model zoo, which contains 56 image classification models trained on ImageNet [31] that span a wide range of architectures. We demonstrate superior performance by a large margin over state-of-the-art approaches, especially for targeted attacks. Furthermore, we tested the perturbations generated by our method for attacks on hard-label classifiers. Our results show that the perturbations generated by our method are highly transferable.

The main contributions of this paper are as follows.

- We propose a novel, yet simple method, BASES, for effective and query-efficient blackbox attacks. The method adjusts weights of the surrogate ensemble by querying the victim model and achieves high fooling rate targeted attack with a very small number of queries.
- We perform extensive experiments to demonstrate that BASES outperforms state-of-the-art methods [26, 10, 27, 11] by a large margin; over 90% targeted success rate with less than 3 queries, which is at least $30\times$ fewer than other method.
- We also demonstrate the effectiveness under a real-world blackbox setting by attacking Google Cloud Vision API and achieve 91% untargeted fooling rate with 2.9 queries ($3\times$ less than [10]).
- The perturbations from BASES are highly transferable and can also be used for hard-label attacks. In this challenging setting, we can achieve over 90% fooling rate for targeted and almost perfect fooling rate for untargeted attacks on a variety of models using less than 3 and 2 queries, respectively.
- We show that BASES can be used for different tasks by creating attacks for object detectors that significantly improve the fooling rate over transfer attacks.

2 Related work

Ensemble-based transfer attacks. Transferable adversarial examples that can fool one model can also fool a different model [3, 4, 29, 32]. Transfer-based untargeted attacks are considered 'easy' since the adversarial examples can disrupt feature extractors into unrelated directions (e.g., in MIM [7], the fooling rate for some models can be as high as 87.9%). In contrast, transfer-based targeted attacks often suffer from low fooling rates (e.g., MIM shows a transfer rate of about 20% at best). To improve the transfer rate, several methods use ensemble based approach. To combine the information from different surrogate models, [29] fuses probability scores, and [7] proposes combining logits. While these methods have been effective, the most natural and generic approach is to combine losses, which can be used for tasks beyond classification [33, 34]. MGAA [35] iteratively selects a set of surrogate models from an ensemble, to perform meta train and meta test steps to shrink the gap between whitebox and blackbox gradient directions. Simulator-Attack [36] trains a substitute model as a "Simulator" for the victim model, using several surrogate models, however training such simulators is computationally expensive and thus difficult to scale to large datasets. Previous ensemble approaches typically assign equal weights for each surrogate model. In contrast, we update weights for different surrogate models based on the victim model feedback.

Query-based attacks. Unlike transfer-based attacks, query-based attacks do not make assumptions that surrogate models share similarity with the victim model. They can usually achieve high fooling rates even for targeted attacks (but at the expense of queries) [21, 25, 22]. The query complexity is proportional to the dimension of the search space. Queries over the entire image space can be extremely expensive [21], requiring millions of queries for targeted attack [22]. To reduce the query complexity, a number of approaches have attempted to reduce the search space dimension or leverage transferable priors or surrogate models to generate queries. SimBA-DCT [23] searches over the low DCT frequencies. P-RGF [26] utilizes surrogate gradients as a transfer-based prior, and draws random vectors from a low-dimensional subspace for gradient estimation. TREMBA [10] trains a perturbation generator and traverses over the low-dimensional latent space. ODS [27] optimizes in the logit space to diversify perturbations for the output space. GFCS [11] searches along the direction of surrogate

gradients, and falls back to ODS if surrogate gradients fail. Some other methods [37, 38, 36] also reuse the query feedback to update surrogate models or 'blackbox simulator', but such a fine-tuning process provides very slight improvements. We summarize the typical search space and average number of queries for some state-of-the-art methods in Table 1. In our approach, we further shrink the search dimension to as low as the number of models in the ensemble. Since our search space is dense with adversarial perturbations, we show that a moderate-size ensemble with 20 models can generate successful targeted attacks for a variety of victim models while requiring only 3 queries (on average), which is at least 30 time fewer than that of existing methods.

3 Method

3.1 Preliminaries

We use additive perturbation [1, 2, 6] to generate a perturbed image as $x^\star = x + \delta$, where δ denotes the perturbation vector of same size as input image x. To ensure that the perturbation is imperceptible to humans, we usually constrain its ℓ_p norm to be less than a threshold, i.e., $\|\delta\|_p \leq \varepsilon$, where p is usually chosen from $\{2, \infty\}$. Such adversarial attacks for a victim model f can be generated by minimizing the so-called adversarial loss function $\mathcal{L}$ over δ such that the output $f(x+\delta)$ is as close to the desired (adversarial) output as possible. Specifically, the attack generator function maps the input image x to an adversarial image $x^\star$ such that the output $f(x^\star)$ is either far/different from the original output y for untargeted attacks, or close/identical to the desired output $y^\star$ for targeted attacks.

Let us consider a multi-class classifier $f(x) : x \mapsto z$, where $z = [z_1, \ldots, z_C]$ represents a logit vector at the last layer. The logit vector can be converted to a probability vector $p = \text{softmax}(z)$. We refer to such a classifier as a "score-based" or "soft-label" classifier. In contrast, a "hard-label" classifier provides a single label index out of a total of C classes. We can derive the hard label from the soft labels as $y = \mathbf{arg\,max}_c\, f(x)_c$. For untargeted attacks, the objective is to find $x^\star$ such that $\mathbf{arg\,max}_c\, f(x^\star)_c \neq y$. For targeted attacks, the objective is to find $x^\star$ such that $\mathbf{arg\,max}_c\, f(x^\star)_c = y^\star$, where $y^\star$ is the target label.

Many efforts on adversarial attacks use iterative variants of the fast signed gradient method (FGSM) [2] because of their simplicity and effectiveness. Notable examples include I-FGSM [5], PGD [6], and MIM [7]. We use PGD attack in our PM, which iteratively optimizes perturbations as

$$\delta^{t+1} = \Pi_\varepsilon \left(\delta^t - \lambda\, \mathbf{sign}(\nabla_\delta \mathcal{L}(x + \delta^t, y^\star)) \right), \tag{1}$$

where $\mathcal{L}$ is the loss function and Π_ε denotes a projection operator. There are many loss functions suitable for crafting adversarial attacks. We mainly employ the following margin loss, which has been shown to be effective in C&W attacks [39]:

$$\mathcal{L}(f(x), y^\star) = \max\left(\max_{j \neq y^\star} f(x)_j - f(x)_{y^\star}, -\kappa \right), \tag{2}$$

where κ is the margin parameter that adjusts the extent to which the example is 'adversarial.' A larger κ corresponds to a lower optimization loss. One advantage of C&W loss function is that its sign directly indicates whether the attack is successful or not ($+ve$ value indicates failure, $-ve$ value indicates success). Cross-entropy loss is also a popular loss function to consider, which has similar performance as margin loss (comparison results provided in supplementary material).

3.2 Perturbation machine with surrogate ensemble

Controlled query generation with PM. We define a perturbation machine (PM) to generate queries for the victim model as shown in Figure 1. The PM accepts an image and generates a perturbation to fool all the surrogate models. Furthermore, we seek some control over the perturbations generated by the PM to steer them in a direction that fools the victim model. To achieve these goals, we construct the PM such that it minimizes a weighted adversarial loss function over the surrogate ensemble.

Adversarial loss functions for ensembles. Suppose our PM consists of N surrogate models given as $\mathcal{F} = \{f_1, \ldots, f_N\}$, each of which is assigned a weight in $\mathbf{w} = [w_1, \ldots, w_N]$ such that $\sum_{i=1}^N w_i = 1$. For any given image x and $\mathbf{w}$, we seek to find a perturbed image $x^\star(\mathbf{w})$ that fools the surrogate ensemble. Below we discuss three possible weighted ensemble loss functions-based optimization

problems for targeted attack. Loss functions for untargeted attack can be derived similarly.

$$\textbf{weighted probabilities} \qquad x^\star(\mathbf{w}) = \arg\min_x -\log\left(\mathbf{1}_{y^\star} \cdot \sum_{i=1}^{N} w_i \,\mathrm{softmax}(f_i(x))\right), \tag{3}$$

$$\textbf{weighted logits} \qquad x^\star(\mathbf{w}) = \arg\min_x \ \mathcal{L}(\sum_{i=1}^{N} w_i f_i(x), y^\star), \tag{4}$$

$$\textbf{weighted loss} \qquad x^\star(\mathbf{w}) = \arg\min_x \ \sum_{i=1}^{N} w_i \mathcal{L}(f_i(x), y^\star). \tag{5}$$

$y^\star$ denotes the target label and $\mathbf{1}_{y^\star}$ denotes its one-hot encoding. $\mathcal{L}$ represents some adversarial loss function (e.g., C&W loss). The first problem in (3) is the minimization of the softmax cross-entropy loss defined on the weighted combination of probability vectors from all ensemble models [29]. The second problem in (4) uses adversarial loss on the weighted combination of logits from the models [7], as the argument for the optimization. The third problem in (5) optimizes a weighted combination of adversarial losses over all models. The weighted loss formulation is the simplest and most generic ensemble approach that works not only for the classification task with logit or probability vectors, but also other tasks (e.g., object detection, segmentation) as long as the model losses can be aggregated [34]. Here, we focus on the weighted loss formulation, since it shows superior performance compared to weighted probabilities and logits formulations in our experiments (results in supplementary material).

Algorithm 1 presents a pseudocode for the PM module for a fixed set of weights. The PM accepts an image x and weights $\mathbf{w}$ along with the surrogate ensemble and returns the perturbed image $x^\star = x + \delta$ after a fixed number of signed gradient descent steps for the ensemble loss.

Algorithm 1 Perturbation Machine: $\delta, x^\star(\mathbf{w}) = \mathbf{PM}(x, \mathbf{w}, \delta_{\text{init}})$

Input:
Input x and the target class $y^\star$ (for untargeted attack $y^\star \neq y$);
Surrogate ensemble $\mathcal{F} = \{f_1, f_2, ..., f_N\}$;
Ensemble weights $\mathbf{w} = \{w_1, w_2, ..., w_N\}$;
Initial perturbation δ_{init}; Step size λ; Perturbation norm (ℓ_2/ℓ_∞) and bound ε
Output: Adversarial perturbation $\delta, x^\star(\mathbf{w})$

```
δ = δ_init
for t = 1 to T do
    Calculate L_ens = Σ_{i=1}^{N} w_i L_i(x + δ, y*)          ▷ Ensemble loss
    Update δ ← δ − λ · sign(∇_δ L_ens)                        ▷ Gradient of ensemble via backpropagation
    Project δ ← Π_ε(δ)                                        ▷ Project to the feasible set of ℓ∞ or ℓ2 ball
end for
x*(w) ← x + δ
return δ, x*(w)
```

3.3 Surrogate ensemble search as bilevel optimization

Let us assume that we are given a blackbox victim model, $f_{\mathbf{v}}$, that we seek to fool using a perturbed image generated by the PM (as illustrated in Figure 1). Suppose the adversarial loss for the victim model is defined as $\mathcal{L}_{\mathbf{v}}$. To generate a perturbed image that fools the victim model, we want to solve the following optimization problem:

$$\mathbf{w} = \arg\min_{\mathbf{w}} \ \mathcal{L}_{\mathbf{v}}(f_{\mathbf{v}}(x^\star(\mathbf{w})), y^\star). \tag{6}$$

The problem in (6) is bilevel optimization that seeks to update the weight vector $\mathbf{w}$ for the PM so that the generated $x^\star(\mathbf{w})$ fools the victim model. The PM in Algorithm 1 can be viewed as a function that solves the inner optimization problem in our bilevel optimization. The outer optimization problem searches over $\mathbf{w}$ to steer the PM towards a perturbation that fools the victim model.

BASES: blackbox attacks via surrogate ensemble search. Our objective is to maximize the attack success rate and minimize the number of queries on the victim model; hence, we adopt a simple yet

effective iterative procedure to update the weights $\mathbf{w}$ and generate a sequence of queries. Pseudocode for our approach is shown in Algorithm 2. We initialize all entries in $\mathbf{w}$ to $1/N$ and generate the initial perturbed image $x^\star(\mathbf{w})$ for input x. We stop if the attack succeeds for the victim model; otherwise, we update $\mathbf{w}$ and generate a new set of perturbed images. We follow [21] and update $\mathbf{w}$ in a coordinate-wise manner, where at every outer iteration, we select nth index and generate two instances of $\mathbf{w}$ as $\mathbf{w}^+, \mathbf{w}^-$ by updating w_n as $w_n + \eta, w_n - \eta$, where η is a step size. We normalize the weight vectors so that the entries are non-negative and add up to 1. We generate perturbations $x^\star(\mathbf{w}^+), x^\star(\mathbf{w}^-)$ using the PM and query the victim model. We compute the victim loss (or score) for $\{\mathbf{w}^+, \mathbf{w}^-\}$ and select the weights, the perturbation vector, and the perturbed images corresponding to the smallest victim loss. We stop if the attack is successful with any query.

Algorithm 2 BASES: Blackbox Attack via Surrogate Ensemble Search

Input:
Input x and the target class $y^\star$ (for untargeted attack $y^\star \neq y$);
Victim model $f_{\mathbf{v}}$; Maximum number of queries Q; Learning rate η;
Perturbation machine with surrogate ensemble
Output: Adversarial perturbation $\delta, x^\star$

Initialize $\delta = 0$; $q = 0$; $\mathbf{w} = \{1/N, 1/N, ..., 1/N\}$
Generate perturbation via PM: $\delta, x^\star(\mathbf{w}) = \mathbf{PM}(x, \mathbf{w}, \delta)$ ▷ *first query with equal weights*
Query victim model: $z = f_{\mathbf{v}}(x + \delta)$
Update query count: $q \leftarrow q + 1$
if $\arg\max_c z_c = y^\star$ **then**
  break ▷ *stop if attack is successful*
end if
while $q < Q$ **do**
  Update surrogate ensemble weights as follows. ▷ *outer level updates weights*
  Pick a surrogate index n ▷ *cyclic or random order*
  Compute $\mathbf{w}^+, \mathbf{w}^-$ by updating w_n as $w_n + \eta, w_n - \eta$, respectively
  Generate perturbation $x^\star(\mathbf{w}^+), x^\star(\mathbf{w}^-)$ via PM ▷ *inner level generates query*
  Query victim model: $f_{\mathbf{v}}(x^\star(\mathbf{w}^+)), f_{\mathbf{v}}(x^\star(\mathbf{w}^-))$ ▷ *2 queries per coordinate*
  Calculate victim model loss for $\{\mathbf{w}^+, \mathbf{w}^-\}$ as $\mathcal{L}_{\mathbf{v}}(\mathbf{w}^+), \mathcal{L}_{\mathbf{v}}(\mathbf{w}^-)$
  Select $\mathbf{w}, \delta, x^\star(\mathbf{w})$ for the weight vector with the smallest loss
  Increment q after every query, and stop if the attack is successful for any query
end while
return δ

4 Experiments

4.1 Experiment setup

In this section, we present experiments on attacking the image classification task. Additional experiments on attacking object detection task can be found in the supplementary material.

Surrogate and victim models. We present experiments for blackbox attacks mainly using pretrained image classification models from Pytorch Torchvision [30], which is a comprehensive and actively updated package for computer vision tasks. At the time of writing this paper, Torchvision offers 56 classification models trained on ImageNet dataset [31]. These models have different architectures and include the family of VGG [40], ResNet [41], SqueezeNet [42], DenseNet[43], ResNeXt [44], MobileNet [45, 46], EfficientNet [47], RegNet [48], Vision Transformer [49], and ConvNeXt [50]. We choose different models as the victim blackbox models for our experiments, as shown in Figures 2, 3, and 4. To construct an effective surrogate ensemble for the PM, we sample 20 models from different families: {`VGG-16-BN`, `ResNet-18`, `SqueezeNet-1.1`, `GoogleNet`, `MNASNet-1.0`, `DenseNet-161`, `EfficientNet-B0`, `RegNet-y-400`, `ResNeXt-101`, `Convnext-Small`, `ResNet-50`, `VGG-13`, `DenseNet-201`, `Inception-v3`, `ShuffleNet-1.0`, `MobileNet-v3-Small`, `Wide-ResNet-50`, `EfficientNet-B4`, `RegNet-x-400`, `VIT-B-16`}. We vary our ensemble size $N \in \{4, 10, 20\}$ by picking the first N model from the set. In most of the experiments, our method uses $N = 20$ models in the **PM**, unless otherwise specified. We also tested a different set of

models pretrained on TinyImageNet dataset, the details of which are included in the supplementary material. To validate the effectiveness of our methods in a practical blackbox setting, we also tested Google Cloud Vision API.

Comparison with other methods. We compare our method with some of the state-of-the-art methods for score-based blackbox attacks. TREMBA [10] is a powerful attack method that searches for perturbations by changing the latent code of a generator trained using a set of surrogate models. GFCS [11] is a recently proposed surrogate-based attack method that probes the victim model using the surrogate gradient directions. We use their original code repositories [51, 52]. For completeness, we also compare with two earlier methods, ODS [27] and P-RGF [26], that leverage transferable priors, even though they have been shown to be less effective than GFCS and TREMBA. Additional details about comparison with TREMBA and GFCS are provided in the supplementary material.

Dataset. We mainly use 1000 ImageNet-like images from the NeurIPS-17 challenge [53, 54], which provides the ground truth label and a target label for each image. We also provide evaluation results on TinyImageNet in the supplementary material.

Query budget. In this paper, we move towards a limited-access setting, since for many real-life applications, legitimate users will not be able to run many queries [28]. In contrast with TREMBA and GFCS, which set the maximum query count to $10,000$ and $50,000$, respectively, we set the maximum count to be 500 and only run our method for 50 queries in the worst case. (TREMBA also uses only 500 queries for Google Cloud Vision API to cut down the cost.)

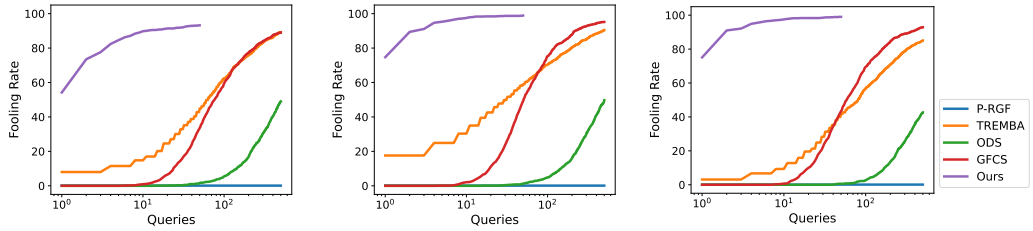
Perturbation budget. We evaluated our method under both ℓ_∞ and ℓ_2 norm bound, with commonly used perturbation budgets of $\ell_\infty \leq 16$ and $\ell_2 \leq 255\sqrt{0.001D} = 3128$ on a 0–255 pixel intensity scale, where D denotes the number of pixels in the image. For attacking Google Cloud Vision API, we reduce the norm bound to $\ell_\infty \leq 12$ to align with the setting in TREMBA. Results for ℓ_2 norm bound are provided in the supplementary material.

Targeted vs untargeted attacks. All the methods achieve near perfect fooling rates for untargeted attacks in our experiments. This is because untargeted attack on image classifiers is not challenging [11], especially when the number of classes is large. Thus, we primarily report experimental results on targeted attacks in the main text. We use the target labels provided in the dataset [53], and we provide analysis on using different target label selection methods such as 'easiest' and 'hardest' according to the confidence scores. Detailed analyses and results for untargeted attacks are in the supplementary material.

4.2 Score-based attacks

Targeted attacks. Figure 2 presents a performance comparison of five methods for targeted attacks on three blackbox victim models. Our proposed method provides the highest fooling rate with the least number of queries. P-RGF is found to be ineffective (almost 0% success) for targeted attacks under low query budgets. TREMBA and GFCS are similar in performance; TREMBA shows better performance when query count is small, but GFCS matches TREMBA after nearly 100 queries. Nevertheless, our method clearly outperforms these two powerful methods by a large margin at any level of query count. We summarize the search space dimension $\mathcal{D}$ and query counts vs fooling rate of different methods under a limited (and realistic) query budget for both the targeted and untargeted attacks in Table 1. Our method is the most effective in terms of fooling rate vs number of queries (and has the smallest search dimension). *Additional results and details about fair comparison and fine tuning of TREMBA and GFCS are provided in the supplementary material.*

Surrogate ensemble size (N)**.** To evaluate the effect of surrogate ensemble size on the performance of our method, we performed targeted blackbox attacks experiment on three different victim models using three different sizes of surrogate ensemble: $N \in \{4, 10, 20\}$. The results are presented in Figure 3 in terms of fooling success rate vs number of queries. As we increase the ensemble size, the fooling rate also increases. With $N = 20$, the targeted attack fooling rate is almost perfect within 50 queries. Specifically, for `VGG-19` with $N = 20$, we improve from 54% success rate at the first query (with equal ensemble weights) to 96% success rate at the end of 50 queries; this equals 78% improvement. `DenseNet-121` and `ResNext-50` can achieve 100% fooling rate with $N = 20$. With `DenseNet-121`, using 10 surrogate models, we can achieve a fooling rate of 98%. While using 4 models is challenging with respect to all victim models, we can see a rapid and significant improvement in fooling rates when the number of queries increases.



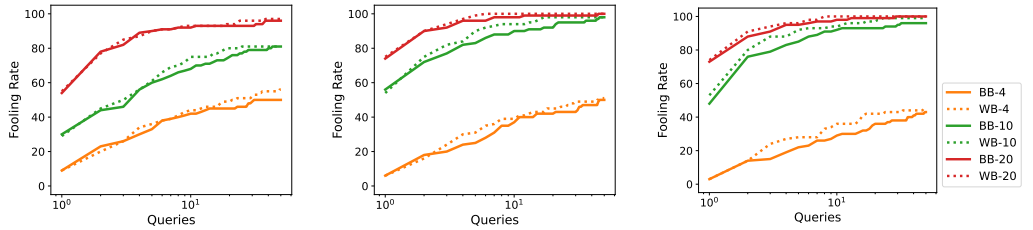
(a) VGG-19 **(b)** DenseNet-121 **(c)** ResNext-50

Figure 2: Comparison of 5 attack methods on three victim models under perturbation budget $l_\infty \leq 16$ for targeted attack. Our method achieves high success rate (over 90%) with few queries (average of 3 per image).

Table 1: Number of queries vs fooling rate of different methods and the search space dimension $\mathcal{D}$.

Method	$\mathcal{D}$	Number of queries ($\mathbf{mean} \pm \mathbf{std}$) per image and fooling rate					
		VGG-19		DenseNet-121		ResNext-50	
		Targeted	Untargeted	Targeted	Untargeted	Targeted	Untargeted
P-RGF [26]	7,500	-	156 ± 113 93.5%	-	164 ± 112 92.9%	-	166 ± 116 92.5%
TREMBA [10]	1,568	92 ± 107 89.2%	2.4 ± 14 99.7%	70 ± 104 90.5%	5.9 ± 28 99.5%	100 ± 109 85.1%	7.5 ± 38 98.9%
ODS [27]	1,000	261 ± 125 49.0%	38 ± 48 99.9%	266 ± 123 49.7%	52 ± 64 99.0%	270 ± 116 42.7%	54 ± 65 98.4%
GFCS [11]	1,000	101 ± 95 89.1%	14 ± 21 100.0%	76 ± 75 95.2%	16 ± 36 99.9%	86 ± 87 92.9%	15 ± 18 99.7%
Ours	**20**	$\mathbf{3.0 \pm 5.4}$ **95.9%**	$\mathbf{1.2 \pm 2.4}$ **99.8%**	$\mathbf{1.8 \pm 2.7}$ **99.4%**	$\mathbf{1.2 \pm 1.8}$ **99.9%**	$\mathbf{1.8 \pm 2.6}$ **99.7%**	$\mathbf{1.2 \pm 0.9}$ **100.0%**

Comparison of whitebox (gradient) vs blackbox (queries). To check the effectiveness of our coordinate descent approach for updating $\mathbf{w}$, we compare its performance with the alternative approach of calculating the exact gradient of victim loss under the whitebox setting. The results are presented in Figure 3 as dotted lines. We observe that our blackbox query approach provides similar results as the whitebox version, which implies the coordinate-wise update of $\mathbf{w}$ is as good as a complete gradient update.

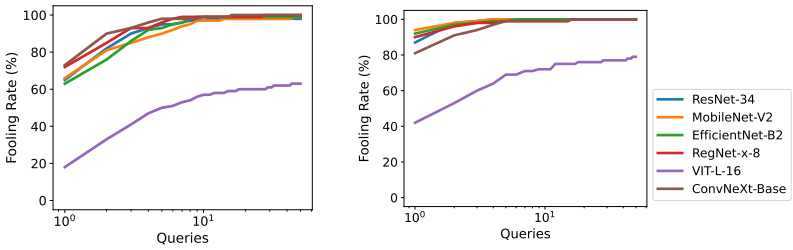


(a) VGG-19 **(b)** DenseNet-121 **(c)** ResNext-50

Figure 3: Comparison of targeted attack fooling rate with different number of ensemble models $N \in \{4, 10, 20\}$ in PM. Every experiment is performed with whitebox gradient (denoted as 'WB' with dotted lines) and blackbox score-based coordinate descent (denoted as 'BB' with solid lines). Experiment was run on 100 images.

4.3 Hard-label attacks

The queries generated by our PM are highly transferable and can be used to craft successful attacks for hard-label classifiers. To generate a sequence of queries for hard-label classifiers, we pick a 'surrogate victim' model and generate queries by updating $\mathbf{w}$ in the same manner as the score-based attacks for Q iterations (without termination). We store the queries generated at every iterations in a query set $\{\delta^1, \ldots, \delta^Q\}$. We test the victim hard-label blackbox model using $x + \delta$ by selecting δ from the set in a sequential order until either the attack succeeds or the queries finish.



(a) Targeted **(b)** Untargeted

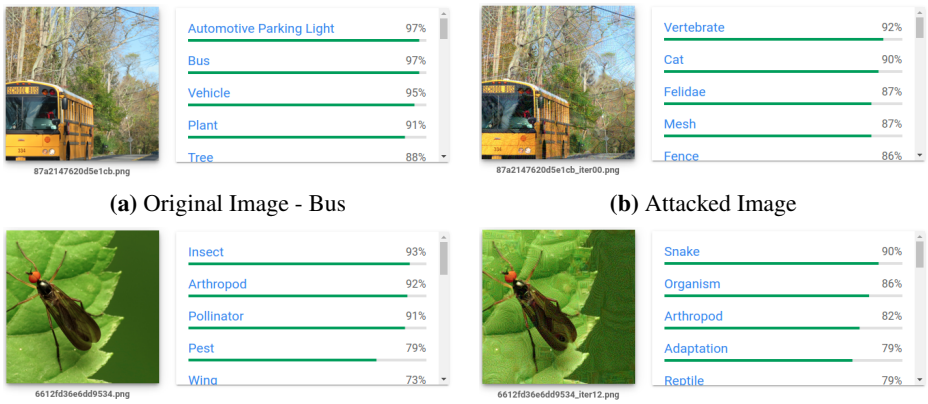
Figure 4: Performance of blackbox attack on 6 hard-label classifiers. Our method generates a sequence of queries for targeted attack using `VGG-19` as a victim model while the PM has $N = 20$ models in the surrogate ensemble. Experiment performed on 100 images.

In our experiments, we observed that this approach can achieve a high targeted attack fooling rate on a variety of models. We present the results of our experiment in Figure 4, where we report attack success rate vs query count for 6 models: {`MobileNet-V2, ResNet-34, ConvNeXt-Base, EfficientNet-B2, RegNet-x-8, VIT-L-16`}. We used `VGG-19` as the 'surrogate victim' model to generate the queries using the PM with 20 surrogate models. Using the saved surrogate perturbations, we can fool all models almost 100%, except for `VIT-L-16` [49] that is a vision transformer and architecturally very different from the majority of surrogate ensemble models (thus difficult to attack). Nevertheless, the fooling rate increases from $18\% \rightarrow 63\%$, which is a 250% improvement.

4.4 Attack on commercial Google Cloud Vision API

We demonstrate the effectiveness of our approach under a practical blackbox setting by attacking the Google Cloud Vision (GCV) label detection API. GCV detects and extracts information about entities in an image, across a very broad group of categories containing general objects, locations, activities, animal species, and products. Thus, the label set is very different from that of ImageNet, and largely unknown to us. We have no knowledge about the detection models in this API either. We randomly select 100 images from the aforementioned ImageNet dataset that are correctly classified by GCV, and perform untargeted attacks against GCV using 20 surrogate models with perturbation budget of $\ell_\infty \leq 12$ to align with the setting in TREMBA [10].

For each input image, GCV returns a list of labels, which are usually the top 10 labels ranked by probability. Under the success metric of changing the top 1 label to any other label, same as in [10], our attack can achieve a fooling rate of 91% with only 2.9 queries per image on average, which is much lower than 8 queries TREMBA reported for similar experiment. We present some successful examples in Figure 5. We present additional results in the supplementary material that show our attacks from classification can transfer to object detection models.



(a) Original Image - Bus **(b)** Attacked Image

(c) Original Image - Fly **(d)** Attacked Image

Figure 5: Visualization of some successful attacks on Google Cloud Vision.

5 Conclusion and discussion

We propose a novel and simple approach, BASES, to effectively perform blackbox attacks in a query-efficient manner, by searching over the weight space of ensemble models. Our extensive experiments demonstrate that a wide range of models are vulnerable to our attacks at the fooling rate of over 90% with as few as 3 queries for targeted attacks. The attacks generated by our method are highly transferable and can also be used to attack hard-label classifiers. Attacks on Google Cloud Vision API further demonstrates that our attacks are generalizable beyond the surrogate and victim models in our experiments.

Limitations. 1) Our method needs a diverse ensemble for attacks to be successful. Even though the search space is low-dimensional, the generated queries should span a large space so that they can fool any given victim model. This is not a major limitation for image classification task as a large number of models are available, but it can be a limitation for other tasks. 2) Our method relies on the PM to generate a perturbation query for every given set of weights. The perturbation generation over surrogate ensemble is computationally expensive, especially as the ensemble size becomes large. In our experiments, one query generation with $\{4, 10, 20\}$ surrogate models requires nearly $\{2.4s, 9.6s, 18s\}$ per image on Nvidia GeForce RTX 2080 TI. Since our method requires a small number of queries, the overall computation time of our method remains small.

Societal impacts. We propose an effective and query efficient approach for blackbox attacks. Such adversarial attacks can potentially be used for malicious purposes. Our work can help further explain the vulnerabilities of DNN models and reduce technological surprise. We also hope this work will motivate the community to develop more robust and reliable models, since DNNs are widely used in real-life or even safety-critical applications.

Acknowledgments. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under agreement number HR00112090096. Approved for public release; distribution is unlimited.

References

[1] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

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

[3] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[4] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[5] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[6] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[7] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting Adversarial Attacks with Momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

[8] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L Yuille. Improving transferability of adversarial examples with input diversity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2730–2739, 2019.

[9] Jiadong Lin, Chuanbiao Song, Kun He, Liwei Wang, and John E Hopcroft. Nesterov accelerated gradient and scale invariance for adversarial attacks. *arXiv preprint arXiv:1908.06281*, 2019.

[10] Zhichao Huang and Tong Zhang. Black-box adversarial attack with transferable model-based embedding. In *International Conference on Learning Representations*, 2019.

[11] Nicholas A. Lord, Romain Mueller, and Luca Bertinetto. Attacking deep networks with surrogate-based adversarial black-box methods is easy. In *International Conference on Learning Representations*, 2022.

[12] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations*, 2018.

[13] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *Annual Network and Distributed System Security Symposium*, 2017.

[14] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. In *International Conference on Learning Representations*, 2017.

[15] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 135–147, 2017.

[16] Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. Towards robust neural networks via random self-ensemble. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 369–385, 2018.

[17] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018.

[18] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 501–509, 2019.

[19] Kui Ren, Tianhang Zheng, Zhan Qin, and Xue Liu. Adversarial attacks and defenses in deep learning. *Engineering*, 6(3):346–360, 2020.

[20] Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. Recent advances in adversarial training for adversarial robustness. In *International Joint Conference on Artificial Intelligence*, 2021.

[21] 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*, pages 15–26, 2017.

[22] Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 742–749, 2019.

[23] Chuan Guo, Jacob Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Weinberger. Simple black-box adversarial attacks. In *International Conference on Machine Learning*, pages 2484–2493. PMLR, 2019.

[24] Huichen Li, Xiaojun Xu, Xiaolu Zhang, Shuang Yang, and Bo Li. Qeba: Query-efficient boundary-based blackbox attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1221–1230, 2020.

[25] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. *International Conference on Machine Learning*, 2018.

[26] Shuyu Cheng, Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Improving black-box adversarial attacks with a transfer-based prior. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[27] Yusuke Tashiro, Yang Song, and Stefano Ermon. Diversity can be transferred: Output diversification for white-and black-box attacks. In *Advances in Neural Information Processing Systems*, volume 33, pages 4536–4548, 2020.

[28] Ian Goodfellow. A research agenda: Dynamic models to defend against correlated attacks. *arXiv preprint arXiv:1903.06293*, 2019.

[29] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *International Conference on Learning Representations*, 2017.

[30] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[31] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. IEEE, 2009.

[32] Maosen Li, Cheng Deng, Tengjiao Li, Junchi Yan, Xinbo Gao, and Heng Huang. Towards transferable targeted attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 641–649, 2020.

[33] Zhaohui Che, Ali Borji, Guangtao Zhai, Suiyi Ling, Jing Li, and Patrick Le Callet. A new ensemble adversarial attack powered by long-term gradient memories. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3405–3413, 2020.

[34] Zuxuan Wu, Ser-Nam Lim, Larry S Davis, and Tom Goldstein. Making an invisibility cloak: Real world adversarial attacks on object detectors. In *European Conference on Computer Vision*, pages 1–17. Springer, 2020.

[35] Zheng Yuan, Jie Zhang, Yunpei Jia, Chuanqi Tan, Tao Xue, and Shiguang Shan. Meta gradient adversarial attack. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7748–7757, 2021.

[36] Chen Ma, Li Chen, and Jun-Hai Yong. Simulating unknown target models for query-efficient black-box attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11835–11844, 2021.

[37] Fnu Suya, Jianfeng Chi, David Evans, and Yuan Tian. Hybrid batch attacks: Finding black-box adversarial examples with limited queries. *USENIX Security Symposium*, 2019.

[38] Jiancheng Yang, Yangzhou Jiang, Xiaoyang Huang, Bingbing Ni, and Chenglong Zhao. Learning black-box attackers with transferable priors and query feedback. *Advances in Neural Information Processing Systems*, 33:12288–12299, 2020.

[39] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

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

[41] 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*, pages 770–778, 2016.

[42] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[43] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[44] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[45] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[46] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019.

[47] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[48] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020.

[49] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[50] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2022.

[51] Nicholas A. Lord, Romain Mueller, and Luca Bertinetto. Gfcs. `https://github.com/fiveai/GFCS`, 2022. [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License].

[52] Zhichao Huang and Tong Zhang. Tremba. `https://github.com/TransEmbedBA/TREMBA`, 2019. [No license provided].

[53] Google Brain. Neurips 2017: Targeted adversarial attack. `https://www.kaggle.com/competitions/nips-2017-targeted-adversarial-attack/data`, 2017. [On Kaggle].

[54] Alexey Kurakin, Ian Goodfellow, Samy Bengio, Yinpeng Dong, Fangzhou Liao, Ming Liang, Tianyu Pang, Jun Zhu, Xiaolin Hu, Cihang Xie, et al. Adversarial attacks and defences competition. In *The NIPS'17 Competition: Building Intelligent Systems*, pages 195–231. Springer, 2018.

[55] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pages 484–501. Springer, 2020.

[56] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[57] Jiawei Du, Hu Zhang, Joey Tianyi Zhou, Yi Yang, and Jiashi Feng. Query-efficient meta attack to deep neural networks. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=Skxd6gSYDS`.

[58] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. *arXiv preprint arXiv:1807.07978*, 2018.

[59] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[60] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Mmdetection. `https://github.com/open-mmlab/mmdetection`, 2019. [Apache License 2.0].

[61] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[62] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[63] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[64] Xiaosong Zhang, Fang Wan, Chang Liu, Rongrong Ji, and Qixiang Ye. FreeAnchor: Learning to match anchors for visual object detection. In *Neural Information Processing Systems*, 2019.

[65] Ze Yang, Shaohui Liu, Han Hu, Liwei Wang, and Stephen Lin. Reppoints: Point set representation for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9657–9666, 2019.

[66] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. In *arXiv preprint arXiv:1904.07850*, 2019.

[67] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[68] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=gZ9hCDWe6ke`.

[69] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] *Code is available at https://github.com/CSIPlab/BASES.*
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] . *We provide these details in the experiment section and supplementary material.*
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] . *Our method provides (almost) identical results for each run.*
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] . *We provide these details in the experiment section and supplementary material.*
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Blackbox Attacks via Surrogate Ensemble Search
Supplementary Material

Summary

In this supplementary document, we provide additional discussion on the selection of hyper parameters, results for classification and object detection tasks, and influence of ensemble weights on the loss landscape of the victim model. Below is a summary of main sections of this document.

A. As mentioned in Section 3 of the main text, we provide experimental results justifying our selection of hyper-parameters, such as the choice of loss function for individual surrogate models, ensemble loss function, step size of PGD attack in PM, and the selection of models in the surrogate ensemble. We select the hyper-parameters that achieve the best performance for our experiments. We also discuss the effects of using different target labels.

B. As promised in Section 4 of the main text, we provide more details about the comparison with TREMBA and GFCS. We present comparisons with the Simulator Attack [36] and a hybrid attack using query-based square attack [37, 55]. We also provide additional comparisons with state-of-the-art methods for untargeted attacks and attacks under ℓ_2 norm constraints.

C. We present experiments and results for vanishing attacks on object detectors. Our results indicate that the proposed approach is also effective for tasks beyond classification.

D. We present some examples of adversarial images generated in our experiments.

E. We analyze the effect of ensemble weights on the loss landscape of different victim models.

F. We present average top-1 classification accuracy of all the models on clean images.

A Analysis of hyper-parameters

A.1 Hyper-parameters for inner optimization (PM)

Hyper-parameters can greatly impact the attack performance but can get overlooked sometimes. In our experiments, we analyzed how different hyper-parameters influence the performance of our algorithm. The experiment setup is similar to Figure 2a (in main text) using only the first 100 images to speed up experiments (because we observed similar trends using all 1000 images).

Loss function. Two popular candidates of loss functions (C&W, Cross Entropy) show similar performance as shown in Figure 6a. We choose C&W for the sake of convenience in determining success from its sign.

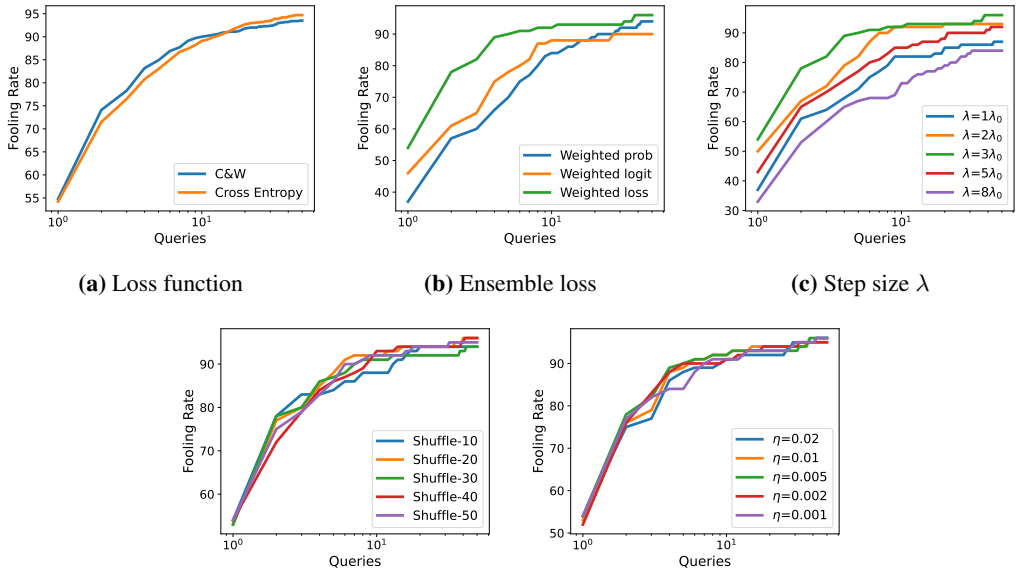
Ensemble loss function. Some previous papers (e.g., MIM [7]) claimed that ensemble with weighted logits (equation (4) in main text) outperforms ensemble with weighted probabilities and weighted combination of loss (equations (3) and (5) in main text). In our experiments, shown in Figure 6b, we observe that weighted combination of surrogate loss functions provide similar or even higher fooling rate compared to weighted probabilities or logits.

PGD step size λ. Since we are running the PGD-based attack in PM, the step size λ can influence the attack success rate. For perturbation budget $\varepsilon = 16$ and $T = 10$ iterations, I-FGSM will use a step size of $\lambda_0 = \varepsilon/T = 1.6$ to prevent the perturbation from exceeding the ℓ_∞ norm bound. Since PGD projects the perturbation back to its feasible set at each iteration, we can increase the step size λ such that more pixels saturate, which leads to a higher attack success rate. In Figure 6c, we report results using λ as a multiple of λ_0 using multiplying factors $\{2, 3, 5, 8\}$. We choose the best step size for our experiments, which is $\lambda = 3\lambda_0$.

Selection of surrogate models. As specified in the main text, our method needs a diverse ensemble for attacks to be successful. Following the setting in TREMBA [10], we start with four surrogate models, `{VGG-16-BN, ResNet-18, SqueezeNet-1.1, GoogleNet}`. To improve the diversity of our ensemble, we insert more models from different families. We expand it to ten by adding `{MNASNet-1.0, DenseNet-161, EfficientNet-B0, RegNet-y-400, ResNeXt-101, Convnext-Small}`. Finally, we add `{VGG-13, ResNet-50, DenseNet-201, Inception-v3, ShuffleNet-1.0, MobileNet-v3-Small, Wide-ResNet-50, EfficientNet-B4, RegNet-x-400, VIT-B-16}` to create an ensemble with 20 models. We observe that by using 20 models in the ensemble, our method can already achieve an almost perfect targeted attack fooling rate.

A.2 Hyper-parameters for outer optimization

Order of surrogate models. Since we are using a coordinate descent approach, the order of coordinates (i.e., surrogate models in our case) may influence the performance of our method. We performed different experiments



(a) Loss function **(b)** Ensemble loss **(c)** Step size λ

(d) Shuffle model orders **(e)** Learning rate η

Figure 6: Analysis on the effect of some hyper-parameters. (a) Loss function for individual surrogate models. (b) Ensemble loss function using three types of weighted combinations. (c) Step size λ of inner optimization in PM. (d) Order of surrogate models in PM. (e) Learning rate η in updating ensemble weights $\mathbf{w}$ for outer optimization. All the experiment are performed for targeted attacks on victim model `VGG-19`, with ensemble size $N = 20$, and evaluated on 100 images.

by shuffling the order of the models using different random seeds (10, 20, 30, 40, 50). The results in Figure 6d suggest that our method provides identical results for different sequences of surrogate models.

Learning rate η. Learning rate is often an important hyper-parameter that can influence performance, and we selected our learning rate to be $1/10th$ of the average ensemble weight with 20 models (i.e., $\eta = 0.005$). We compare different learning rates in the range of $\{0.02, 0.01, 0.005, 0.002, 0.001\}$ while ensuring that all individual surrogate weights remain non-negative. The results in Figure 6e suggest that our approach is robust to variations in the learning rates.

A.3 Selection of target labels

Different selections of target labels may result in different levels of difficulty in attacks. Here we evaluate different proposals for selecting the target labels, including the 'easiest' label (the label with the second highest original confidence score), the 'hardest' label (the label with the lowest original confidence score), and a random label. We performed an experiment to test the difficulty of three types of target labels and report our results below. Corresponding to Table 1 in our paper, we use `DenseNet-121` as the victim model. Here we randomly select 100 images for evaluation. We see that the original confidence scores of the NeurIPS17 target classes are already close to 0 (which means they are already challenging cases), for which our method requires an average of 1.64 queries to achieve a 100% fooling rate. The second most likely label has an average confidence of 0.08 (whereas top 1 is 0.8), and our method achieves a 100% fooling rate with an average of 1.02 queries. For the 'hardest' setting of the least confident class label, our method shows a slight drop in fooling success rate and achieves 97% success using 2.20 queries on average.

Target classes	Avg. confidence	Fooling rate	Query counts			
			mean	**min**	**max**	**median**
'Easiest'	0.08	100%	1.02	1	2	1
NeurIPS17	8.92×10^{-6}	100%	1.64	1	13	1
'Hardest'	1.74×10^{-8}	97%	2.20	1	15	1

B Experiments on classification

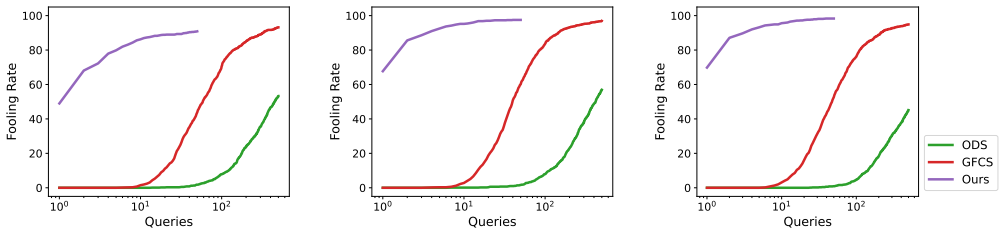
Comparison with TREMBA. TREMBA [10] requires one trained generator for each target class; thus, it is not feasible to test it for any arbitrary target label selected from 1000 classes in ImageNet. For a fair comparison, we

attack each image using one of the 6 target labels available in trained TREMBA model $\{0, 20, 40, 60, 80, 100\}$ and average the query counts. Furthermore, TREMBA generator was trained using an ensemble of 4 surrogate models; while it is possible to train the generator with more surrogate models, training one generate per target label is expensive and non-trivial in terms of hyper-parameter tuning. Therefore, in our experiments, we used the trained generator from the paper. It is worth pointing out that our method with 4 surrogate models (as shown in Figure 3) is still better than TREMBA in the low query count regime. TREMBA can provide better success rate at the expense of increased queries.

Why is our method better than TREMBA? TREMBA generates patterns by optimizing over the latent code of a trained generator, which contributes to the high success rate. TREMBA generator has a large enough range that it can generate adversarial perturbations that fool a victim model. Our experimental results suggests that the space of perturbations generated by our PM (via weighted surrogate ensemble) is better (in terms of diversity and low dimensionality) than TREMBA's generator. That is the reason why we see a steep slope for the first few queries in our success vs query curve.

Comparison with GFCS. To perform our experiments, we used the same set of $N = 20$ surrogate models for GFCS [11] that are used in our PM. GFCS used ℓ_2 norm constraint and did not compare with TREMBA. While our method can generate perturbations with ℓ_2 and ℓ_∞ constraints, TREMBA generates perturbations with ℓ_∞ constraint. To perform a fair comparison, we modified GFCS code to have ℓ_∞ constraint and tuned the hyper-parameters to achieve the best performance. The step-size is the key parameter that we choose as 0.005 after searching over a grid of $\{0.2, 0.02, 0.01, 0.005, 0.001, 0.0005\}$. As shown in 7, the performance reported in Figure 2 for ℓ_∞ attacks is on par with the performance achieved with original settings of ℓ_2 norm constraint.

Why is our method better than GFCS? Our method is more query efficient because we leverage all surrogate models for each query, whereas GFCS only uses one surrogate model per query. We can see that our method has the steepest slope in Figure 7 and the highest success at the starting point.



(a) VGG-19 **(b)** DenseNet-121 **(c)** ResNext-50

Figure 7: Adversarial attacks generated with ℓ_2 constraint (equivalent to Figure 2 in main text that uses ℓ_∞ constraints). Comparison of our method with GFCS / ODS on three victim models under perturbation budget $\ell_2 \leq 3128$ for targeted attacks.

Note about P-RGF. The original implementation of P-RGF is in Tensorflow, but to unify the platform, we use the Pytorch implementation provided by GFCS [51].

Comparison with Simulator Attack. We use the same setting as in simulator attack [36] that tests 3 victim blackbox models {`DenseNet-121`, `ResNeXt-101 (32×4d)`, `ResNeXt-101 (64×4d)`} and uses 16 surrogate models {`VGG-11/13/16/19`, `VGG-11/13/16/19 (BN)`, `ResNet-18/34/50/101/152`, `DenseNet-161/169/201`}. All of these models are trained on TinyImageNet [56] dataset and we obtain the pretrained weights from [36]. We randomly select 1000 tinyImageNet images and use incremental target label selection for targeted attacks. Target label $y_{adv} = (y + 1) \mod C$, where y is the original label and total number of classes is $C = 200$. Perturbation budget for targeted attack is $\ell_2 \leq 4.6 \times 255 = 1173$, and for untargeted attack $\ell_\infty \leq 8$. As shown in Table 2, we achieve perfect fooling rates with less than two queries on average for both targeted and untargeted attacks. Specifically, for `ResNeXt-101 (32×4d)`, we achieve 100% targeted fooling rate with an average query count of 2.0, (min = 1, max = 26, median = 1). In contrast, simulator attack achieves 84.9% fooling rate using 2558 queries, which is $1279\times$ more expensive than ours. For untargeted attack, the trend is similar that our method is 811–$1445\times$ more query efficient than simulator attack.

Comparison with combining transfer and query-based attacks. Hybrid attack in [37] is one of the earliest works that combines transfer and query-based attacks. It uses surrogate models to generate the initial query, which is later updated using feedback from the blackbox victim model via pure query-based methods. To verify that our proposed method is advantageous, we use the perturbations generated by our ensemble models with equal weights as the initial query, and for every failed query we deploy a powerful pure query-based method square attack [55]. Following the same setting as in our Table 1 and Figure 2, we perform targeted attack on `DenseNet-121` with a perturbation budget of $\ell_\infty \leq 16$. The transfer rate of initial perturbed images is 75.5%. We attack the remaining 24.5% failed perturbed images using square attack by allowing a maximum query count of 500 (same setting as other baseline methods). On this subset of images, we observed a 33.9% fooling rate

Table 2: Number of queries vs fooling rate of different methods on TinyImageNet dataset.

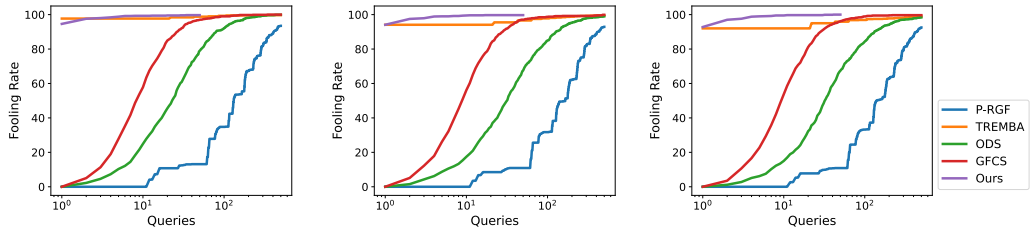
Method	Number of queries ($\mathbf{mean}/\mathbf{median}$) per image and fooling rate					
	DenseNet-121		ResNeXt-101 (32×4d)		ResNeXt-101 (64×4d)	
	Targeted	Untargeted	Targeted	Untargeted	Targeted	Untargeted
NES [25]	4625 / 4337 88.5%	1306 / 510 74.3%	4959 / 4703 88.0%	2104 / 765 45.3%	4758 / 4440 88.2%	2078 / 816 45.5%
Meta [57]	5420 / 5506 24.2%	3789 / 3202 71.1%	5440 / 5249 21.0%	4101 / 3712 33.8%	5661 / 5250 18.2%	4012 / 3649 36.0%
Bandits [58]	2724 / 1860 85.1%	964 / 520 99.2%	3550 / 2700 72.2%	1737 / 954 94.1%	3542 / 2854 72.4%	1662 / 1014 95.3%
Simulator [36]	1959 / 1399 89.8%	811 / 431 99.4%	2558 / 1966 84.9%	1380 / 850 96.8%	2488 / 1982 83.9%	1445 / 878 97.9%
Ours	**1.5 / 1** **100.0%**	**1.0 / 1** **100.0%**	**2.0 / 1** **100.0%**	**1.0 / 1** **100.0%**	**2.0 / 1** **100.0%**	**1.0 / 1** **100.0%**

Table 3: Number of queries vs fooling rate for hybrid methods that combine transfer and query-based attacks.

Models	Fooling rate and number of queries ($\mathbf{mean} \pm \mathbf{std}$) per image	
	Combine [37] and [55]	**Ours**
VGG-19	64.7% ; 34.1 ± 99.2	**95.9% ; 3.0 ± 5.4**
DenseNet-121	83.8% ; 24.5 ± 81.4	**99.4% ; 1.8 ± 2.7**
ResNext-50	84.3% ; 24.7 ± 81.4	**99.7% ; 1.8 ± 2.6**

with query count ($\mathbf{mean} \pm \mathbf{std}$): 238.2 ± 127.4. Overall, including the images that can initially transfer, the combination of [37] and [55] achieves a fooling rate of 83.8%, with query count ($\mathbf{mean} \pm \mathbf{std}$): 24.5 ± 81.4. In comparison, our method achieves a 99.4% fooling rate with a query count of 1.8 ± 2.7. Similar trends appear for other victim models, as shown in Table 3. Our main takeaway is that even though the surrogate ensemble provides highly transferable perturbation or perturbations that can be used as initialization for query-based optimization methods. The query-based methods lose their advantage by querying over a high dimensional image space. Our method searches over the weights of the ensemble loss, which is very low dimension and provides query efficiency.

Untargeted attacks. Un-targeted attacks are 'easy' [11] in image classification, especially when the number of classes is large (e.g., in ImageNet that has 1000 categories). We show that our method can readily achieve a fooling rate over 99% with only 1–2 queries (on average), as depicted in Figure 8 below and Table 1 in the main text. The initial perturbations from the PM (with all ensemble weights set to $1/N$) can already achieve a fooling rate of over 94%, close to that of TREMBA. Other methods require tens or hundreds of queries to achieve near-perfect success rate.



(a) VGG-19 **(b)** DenseNet-121 **(c)** ResNext-50

Figure 8: Untargeted attacks (version of Figure 2 in the main text). Comparison of 5 attack methods on three victim models under perturbation budget $l_\infty \leq 16$ for untargeted attack. All methods can achieve near perfect success rate within 500 queries.

C Experiments on object detection

To demonstrate the generalizability of BASES beyond classification tasks, we also performed experiments for vanishing attacks on object detectors. The results indicate that our proposed method can be easily adopted for other tasks.

C.1 Experiment setup

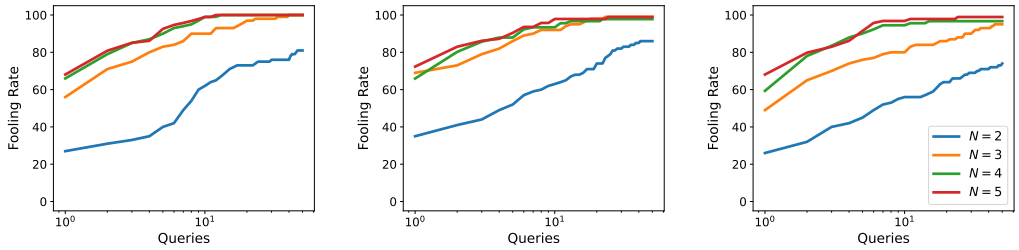
Surrogate and victim models. We evaluate BASES using object detectors from MMDetection [59, 60], which provides a diverse set of models form over fifty model families, including `Faster R-CNN [61]`, `YOLOv3 [62]`, `RetinaNet [63]`, `FreeAnchor [64]`, `RepPoints [65]`, `CenterNet [66]`, `DETR [67]`, and `Deformable DETR [68]`. We choose different models {`RetinaNet`, `RepPoints`, `Deformable DETR`} as victim blackbox models, as shown in Figure 9. For surrogate models in the PM, we select some popular models {`Faster R-CNN`, `YOLOv3`, `FreeAnchor`, `DETR`, `CenterNet`} and vary our ensemble size $N \in \{2, 3, 4, 5\}$ by choosing the first N models from the set.

Dataset, attacks, query, and perturbation budgets. All models are trained on COCO 2017 train dataset [69]. We randomly sample 100 images of stop sign from COCO 2014 validation dataset to perform blackbox vanishing attacks. The attack is considered successful if the victim model fails to detect the stop sign in the adversarial image. The constraints on the query budget $Q \leq 50$ and perturbation budget $\ell_\infty \leq 16$ are the same as the classification setting.

Loss functions and ensemble loss. For individual surrogate models, we use the original loss function used for their training. We defined the ensemble loss as a weighted combination of loss over all the surrogate models. The confidence score of stop sign detected by the victim model is used as a feedback from the victim model.

C.2 Attacks on object detection

The results of attacking object detectors are shown in Figure 9 and Table 4. We observe that our attack method is effective and query efficient in attacking object detectors. In particular, for RetinaNet, a simple transfer attack (first iteration) has 27% fooling rate with $N = 2$ surrogate models. The fooling rate improve from $27\% \rightarrow 81\%$ with a small number of queries, which is a 300% improvement. Our attack gets stronger as the number of surrogate models increases. When $N = 5$, we can get almost perfect ($\geq 99\%$) fooling rate for all victim models with less than 3 queries on average.



(a) RetinaNet **(b)** RepPoints **(c)** Deformable DETR

Figure 9: Fooling rates for vanishing attacks on three victim object detectors using different number ($N \in \{2, 3, 4, 5\}$) of surrogate models in PM.

Table 4: Number of queries per image and fooling rate of attacks on three victim models using different number N of surrogate models in PM.

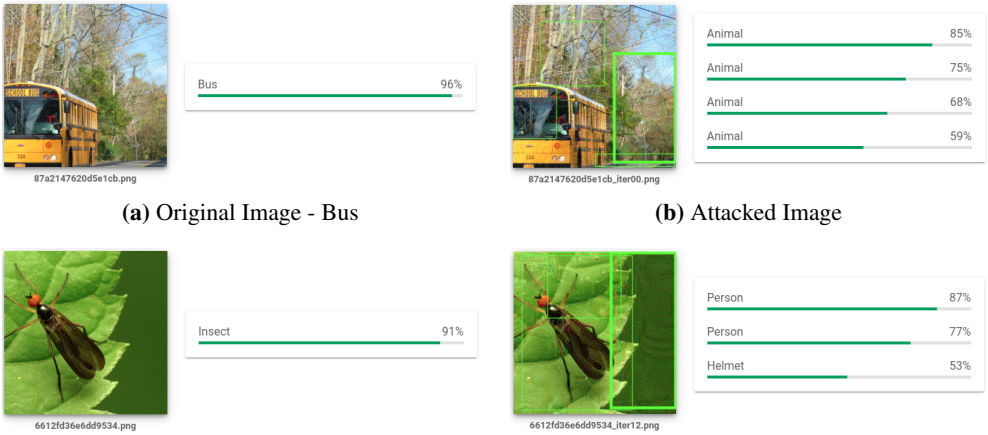
N	Fooling rate and number of queries (**mean** $\pm$ **std**) per image		
	RetinaNet	RepPoints	Deformable DETR
2	81% ; 8.5 $\pm$ 11	86% ; 8.0 $\pm$ 9.9	74% ; 8.5 $\pm$ 11
3	100% ; 3.9 $\pm$ 6.5	99% ; 2.8 $\pm$ 4.1	95% ; 5.4 $\pm$ 9.3
4	100% ; 2.2 $\pm$ 2.4	98% ; 2.2 $\pm$ 3.1	97% ; 2.1 $\pm$ 2.2
5	100% ; 2.0 $\pm$ 2.1	99% ; 2.1 $\pm$ 3.0	99% ; 2.1 $\pm$ 2.9

C.3 Attacks on Google Cloud Vision API

We also observe that the attacks generated by our method can also fool object detection models, as shown in Figure 10.

D Visualization of adversarial examples

Classifiers. We present some examples of adversarial images generated by different methods for targeted attack on VGG-19 classifier in Figure 11. We observe that even with the same perturbation budget, $\ell_\infty \leq 16$,

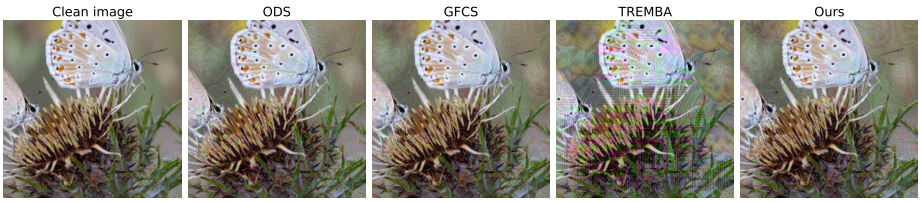


(a) Original Image - Bus **(b)** Attacked Image

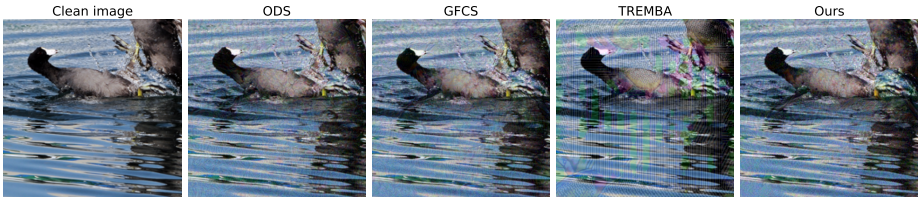
(c) Original Image - Fly **(d)** Attacked Image

Figure 10: Attacks generated by our PM can fool object detection models. Visualization of some successful attacks on Google Cloud Vision object detection API. (Compare to Figure 5 in main text.)

perturbation from our method is less visible than TREMBA, and is comparable with the ones from ODS and GFCS. TREMBA perturbs all images to 'Tench' and has a very structured semantic pattern that becomes visible. ODS, GFCS, and our method perturb 'Butterfly' to 'Dog', 'Coot' to 'Jacamar', and 'Parrot' to 'Fountain'.



(a) Original: Butterfly; Target: Dog for ODS, GFCS, and Ours; Target: 'Tench' for TREMBA.



(b) Original: Coot; Target: Jacamar for ODS, GFCS, and Ours; Target: 'Tench' for TREMBA.

(c) Original: Parrot; Target: Fountain for ODS, GFCS, and Ours; Target: 'Tench' for TREMBA.

Figure 11: Visualization of adversarial images generated by different methods for targeted attack. (Corresponds to experiments in Figure 2 in main text.)

Detectors. We visualize some example images of attacking different object detectors in Figure 12. Our method effectively vanishes stop sign in the scene.

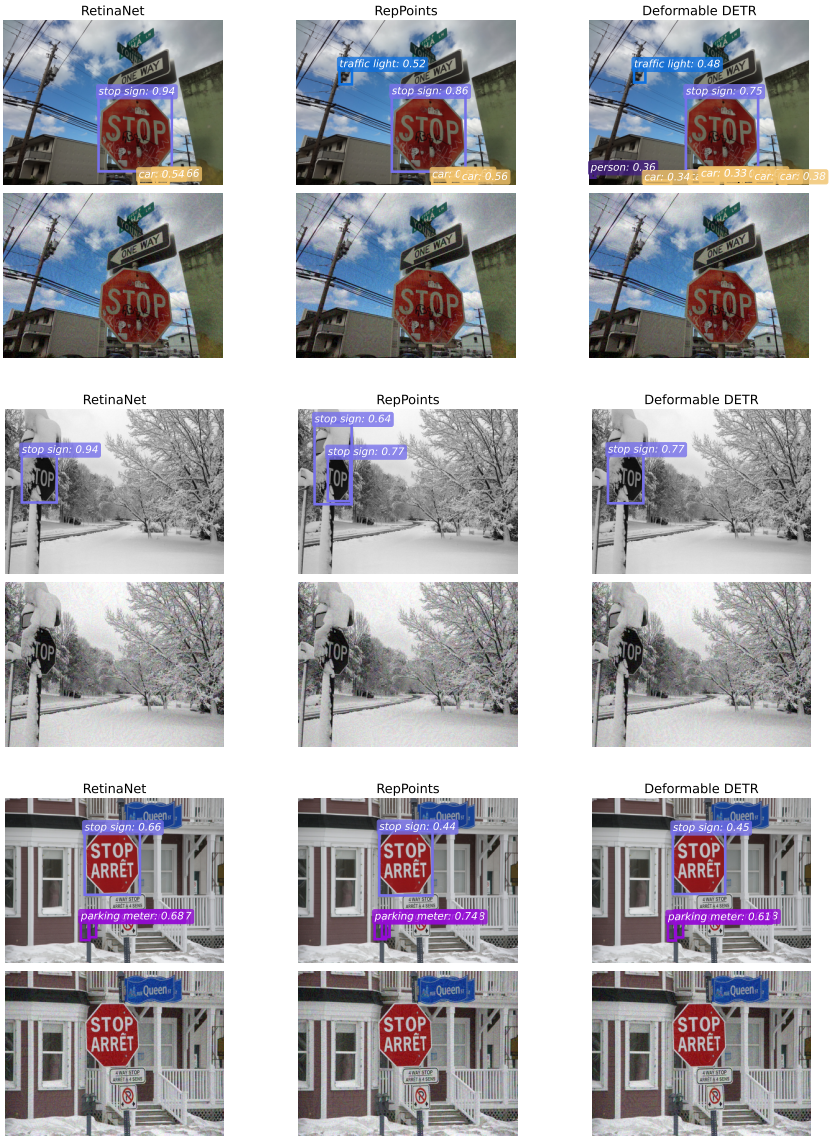
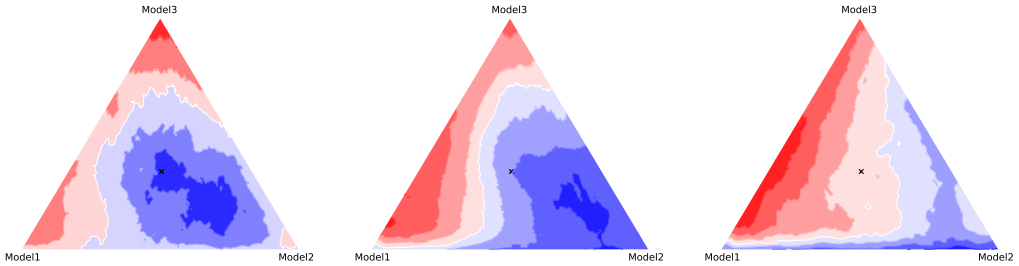


Figure 12: Visualization of adversarial images generated by different methods for vanishing attacks on 'stop sign'. Top row is detection on clean images and bottom row is detection on adversarial images. (Corresponds to results in Figure 9 with $N = 5$.)

E Loss landscape vs ensemble weights

Why does ensemble weights-based query update work? We visualize the loss landscape of some victim models with respect to ensemble weights of three surrogate models in the PM. The plots in Figure 13 illustrate the loss, where the vertices of each triangle represent the surrogates models in the PM used for attacking a victim model on one image (as shown in sub-caption). The location of each point inside the triangle corresponds to the weight vector $\mathbf{w}$ (in terms of Barycentric coordinates). For instance, the centroid (marked by $\times$) has the barycentric coordinate $\mathbf{w} = [1/3, 1/3, 1/3]$, which implies the losses for all the surrogate models in the ensemble are weighted equally. More weight is given to a model if the weight vector moves closer to the vertex of that model. The color of each point inside the triangle represents the victim loss value for the corresponding $\mathbf{w}$. The attack is more successful when the loss value is low (indicated by blue color) and less successful when the loss value is high (indicated by red color). We created this figure using VGG-16, ResNet-18, and SqueezeNet as Model 1,2, and 3, respectively. The main takeaway is that, in many cases, an arbitrary weight vector does not provide successful perturbation for a given victim model; therefore, we need to adjust the weights to generate successful attacks.



(a) VGG-19, Image A **(b)** ResNet-34, Image A **(c)** ResNet-34, Image B

Figure 13: Illustration of the effect of weights of ensemble models on the attack loss for a victim model. Red color indicates large loss values (unsuccessful attack), and blue indicates small loss (successful attack).

F Classification performance on clean images

To ensure that all the models provide reasonably correct classification results on clean images, we calculate the classification accuracy of all ImageNet models on the 1000 test images. Our calculation shows that they have a Top-1 accuracy of ($\mathbf{mean} \pm \mathbf{std}$): $89.1\% \pm 6.5\%$. Among all the models tested, `Convnext-Smal` achieves the highest accuracy at 96.8%, and `SqueezeNet-1.1` gets the lowest at 68.8%.

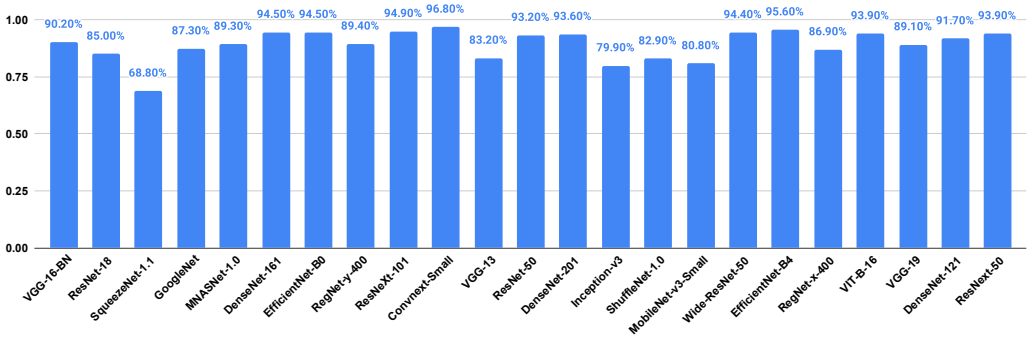


Figure 14: Top 1 classification accuracies of different ImageNet models used in our experiments.