# BAYESOPT ADVERSARIAL ATTACK

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Black-box adversarial attacks require a large number of attempts before finding successful adversarial examples that are visually indistinguishable from the original input. Current approaches relying on substitute model training, gradient estimation or genetic algorithms often require an excessive number of queries. Therefore, they are not suitable for real-world systems where the maximum query number is limited due to cost. We propose a query-efficient black-box attack which uses Bayesian optimisation in combination with Bayesian model selection to optimise over the adversarial perturbation and the optimal degree of search space dimension reduction. We demonstrate empirically that our method can achieve comparable success rates with 2-5 times fewer queries compared to previous state-of-the-art black-box attacks.

## 1 INTRODUCTION

Deep learning algorithms are widely deployed in many real-world systems and are increasingly being used for tasks, ranging from identity verification (Liu et al., 2018), to financial services (Heaton et al., 2017) to autonomous driving (Bojarski et al., 2016). However, even the most accurate deep learning models can be easily deceived by perturbations which are visually imperceptible to the human eye (Szegedy et al., 2013; Carlini et al., 2016). The growing costs and risks associated with the potential model failures has led to the importance of studying adversarial attacks, both in assessing their robustness and their ability to detect such attacks. In this paper we focus on highly practical adversarial attacks that fulfill the following two criteria. First, the attack is designed for a black-box setting because in real-world examples, the attacker would normally have no knowledge of the target deep learning model and can only interact with the model by querying it. Second, query efficiency is highly prioritised because in practical cases where the damage caused by the attack is high, the query budget available to the attacker will be highly limited due to the risk of being detected by the defence system or other high inherent costs (monetary or computational) of model evaluations.

Despite of the large array of adversarial attacks proposed in the literature, many of them are white-box approaches that assume full access to the target model architecture and the ability of performing back-propagation to get gradient information (Moosavi-Dezfooli et al., 2016; Kurakin et al., 2016; Gu & Rigazio, 2014; Goodfellow et al., 2014; Chen et al., 2018; Carlini & Wagner, 2017). On the other hand, for black-box attacks, there are various techniques that have been used which do not require access to the model architecture. One class of methods trains a white-box substitute model and attacks the target model with adversarial examples that successfully fool the substitute (Papernot et al., 2017). However, this type of method requires the availability of the original training data or large query data to train the substitute network and the performance is often limited by the mismatch between the substitute and the target models (Su et al., 2018). The second class of black-box attacks, which show better empirical performance than the substitute model approaches, numerically estimate the gradient of the target model by repeatedly querying it (Chen et al., 2017; Ilyas et al., 2018; Tu et al., 2018) and attack with the estimated gradient. Although various techniques are employed to increase the query efficiency for the gradient estimation, they need an excessively large query budget to achieve a successful attack (Alzantot et al., 2018). Another line of work removes the need for gradient estimation and uses decision-based techniques (Brendel et al., 2017) or genetic algorithms (Alzantot et al., 2018) to generate adversarial examples. One popular technique that is adopted by many black-box attacks (Chen et al., 2017; Alzantot et al., 2018; Tu et al., 2018) to significantly improve the query efficiency is to search for adversarial perturbations in a low-dimensional latent space (search dimensionality reduction). However, learning the effective dimensionality of the latent

search space can be challenging by itself, and has not been investigated by the prior works to the best of our knowledge.

In light of the above limitations, we propose a query efficient black-box attack that iteratively optimises over both the adversarial perturbation and the effective dimensionality of the latent search space. Our main contributions are summarised as follows:

- We introduce a novel gradient-free black-box attack method, BayesOpt attack, which uses Bayesian optimisation with Gaussian process surrogate models to find the effective adversarial example and is capable of dealing with high-dimensional image inputs.
- We proposes a Bayesian technique which learns the optimal degree of search dimensionality reduction by harnessing our statistical surrogate and information from query data. This technique can be incorporated naturally into our attack procedure, leading to efficient optimisation over both adversarial perturbation and the latent search space dimension.
- We empirically demonstrate that under the $L_\infty$ constraint, our proposed attack method can achieve comparable success rate with about *2 to 5 times* fewer model queries in comparison to current state-of-the-art query-efficient black box attack methods.

## 2 Related Work on Black-Box Attacks

Most of the existing adversarial attacks (Moosavi-Dezfooli et al., 2016; Kurakin et al., 2016; Gu & Rigazio, 2014; Goodfellow et al., 2014; Chen et al., 2018; Carlini & Wagner, 2017) focus on white-box settings, where the attacker can get full access to the target model and has complete knowledge of the architecture, weights and gradients to generate successful adversarial examples. However, real-world systems are usually attacked in a black-box setting, where one has no knowledge about the model and can only observe the input-output correspondences by querying the model. Here we give a brief overview of various existing black-box attacks.

**Substitute model** One class of black-box attacks uses data acquired from querying the target black-box model to train a substitute model (Papernot et al., 2017), which mimics the classification behaviour of the target model. The adversary can then employ any white-box method to attack the fully observable substitute model and apply the successful adversarial example to the target model. Such approaches rely on the transferability assumption that adversarial examples which are effective to the substitute model are also very likely to conceive the target model given their similar classification performance on the same data (Szegedy et al., 2013). Moreover, to provide training data for the substitute model, the adversary either requires information on the target model's training set, which is highly unrealistic in real-world applications, or needs to build a synthetic training set by querying the target model (Papernot et al., 2017), which implies a large number of model evaluations and becomes hardly feasible for large models or complex datasets (Brendel et al., 2017).

**Gradient estimation** An alternative to training a substitute model is to estimate the gradient via finite differences and use the estimated gradient information to produce attacks. However, the naive coordinate-wise gradient estimation requires excessive queries to the target model (2 queries per coordinate per descent/attack step) and thus is not feasible for attacking models with high dimensional inputs (e.g. classifiers on ImageNet). Chen et al. (2017) overcome this limitation by using stochastic coordinate gradient descent and selecting the batch of coordinates via importance sampling, introducing the state-of-the-art zeroth-order attack, ZOO, which achieves comparable attack success rate and perturbation costs as many white-box attacks. Although ZOO makes it computationally tractable to perform black-box attack on high-dimensional image data, it still requires millions of queries to generate a successful adversarial example, making it impracticable for attacking real-world systems where model query can be expensive and the budget limited. Improving on ZOO, AutoZOOM (Tu et al., 2018) significantly enhances the query efficiency by using random vectors to estimate the full gradient and adjusting the estimation parameter adaptively to trade off query efficiency vs. input perturbation cost. More importantly, AutoZOOM shows the benefits of employing dimension reduction techniques in accelerating attacks (i.e. searching for adversarial perturbation in a low-dimensional latent space and decoding it back to the high-dimensional input space). Parallel work by Ilyas et al. (2018) estimate the gradient through a modified version of natural evolution strategy which can be viewed as a finite-difference method over a random Gaussian basis. The estimated gradient is then used with projected gradient descent, a white-box attack method, to generate adversarial examples.

**Gradient-free optimisation** As discussed, gradient-estimation approaches in general need an excessive number of queries to achieve successful attack. Moreover, their dependence on the gradient information makes them less robust to defences which manipulate the gradients (Athalye et al., 2018; Brendel et al., 2017; Guo et al., 2017). Thus, truly gradient-free methods are more likely to bypass such defences. One recent example, which has demonstrated state-of-the-art query efficiency, is GenAttack (Alzantot et al., 2018). GenAttack uses genetic algorithms to iteratively evolve a population of candidate adversarial examples. Besides having an annealing scheme for mutation rate and range, GenAttack also adopts dimensional reduction techniques, similar to AutoZOOM, to improve the query efficiency. In parallel to GenAttack, Brendel et al. (2017) introduce a decision-based attack, Boundary Attack, which only requires access to the final model decision. Boundary Attack starts from a huge adversarial perturbation and then iteratively reduces the perturbation through a random walk along the decision boundary. However, Boundary Attack takes about 72 times more queries than GenAttack to fool an undefended ImageNet model (Alzantot et al., 2018). Another recent approach introduced in Moon et al. (2019) reduces the search space to a discrete domain and subsequently uses combinatorial optimisation to find successful attacks, mostly focusing on the less challenging setting of untargeted attacks. Finally, to the best of our knowledge, the only prior work that uses Bayesian optimisation is a workshop paper by (Suya et al., 2017). However, their work also only deals with the untargetted attack setting and only demonstrates the effectiveness of their method in comparison to random search on a low-dimensional ($d = 57$) email classification task. In contrast, we concentrate on image data, and tackle the more challenging (and realistic) black-box setup we describe next.

## 3 PRELIMINARIES

### 3.1 PROBLEM DEFINITION

We focus on the black-box attack setting, where the adversary has no knowledge about the network architecture, weights, gradient or training data of the target model $f$, and can only query the target model with an input $\boldsymbol{x}$ to observe its prediction scores on all $C$ classes (i.e. $f : \mathbb{R}^d \longrightarrow [0, 1]^C$) (Tu et al., 2018; Alzantot et al., 2018).

Moreover, we aim to perform *targeted* attacks, which is more challenging than untargeted attacks, subject to a constraint on the maximum change to any of the coordinates (i.e., a $L_\infty$ constraint) (Warde-Farley & Goodfellow, 2016; Alzantot et al., 2018). Specifically, *targeted* attacks refer to the case where given a valid input $\boldsymbol{x}_{origin}$ of class $t_{origin}$ (i.e. $\arg\max_{i\in\{1,\dots,C\}} f(\boldsymbol{x}_{origin})_i = t_{origin}$) and a target $t \neq t_{origin}$, we aim to find an adversarial input $\boldsymbol{x}$, which is close to $\boldsymbol{x}_{origin}$ according to the $L_\infty-$norm, such that $\arg\max_{i\in\{1,\dots,C\}} f(\boldsymbol{x})_i = t$. *Untargeted* adversarial attacks refer to the case that instead of classifying $\boldsymbol{x}_{origin}$ as $t_{origin}$, we try to find an input $x$ so that $\arg\max_{i\in\{1,\dots,C\}} f(\boldsymbol{x})_i \neq t_{origin}$.

In our approach, we follow the convention to optimise over the perturbation $\boldsymbol{\delta}$ instead of the adversarial example $\boldsymbol{x}$ directly (Chen et al., 2017; Alzantot et al., 2018; Tu et al., 2018). Therefore, our problem can be formulated as:

$$\underset{i\in\{1,\dots,C\}}{\arg\max}\ f(\boldsymbol{x}_{origin} + \boldsymbol{\delta})_i = t \quad \text{s.t.} \quad \|\boldsymbol{\delta}\|_\infty \leq \delta_{max} \tag{1}$$

### 3.2 BAYESIAN OPTIMISATION

Bayesian optimisation (BayesOpt) is a query-efficient approach to tackle global optimisation problems (Brochu et al., 2010). It is particularly useful when the objective function is a black-box and is very costly to evaluate. There are 2 key components in BO: a statistical surrogate, such as a Gaussian process (GP) or a Bayesian neural network (BNN) which models the unknown objective, and an acquisition function $\alpha(\cdot)$ which is maximised to recommend the next query location by trading off exploitation and exploration. The standard algorithm for BayesOpt is shown in Algorithm 3 in the Appendix.

# 4 BAYESOPT ATTACK

## 4.1 BAYESOPT ATTACK OBJECTIVE

It has been shown that reducing the dimensionality of the search space in Equation (1) increases query efficiency significantly (Chen et al., 2017; Tu et al., 2018; Alzantot et al., 2018). Due to our focus on the small query regime where our surrogate model needs to be trained with a very small number of observation data, we adopt the previously suggested dimensionality reduction technique, bilinear resizing, to reduce the challenging problem of optimising over high-dimensional input space of $\boldsymbol{x} \in \mathbb{R}^d$ to one over a relatively low-dimensional input space, setting $\boldsymbol{x} = \boldsymbol{x}_{origin} + g(\boldsymbol{\delta})$ where $\boldsymbol{\delta} \in \mathbb{R}^{d^r}$ with $d^r < d$ and $g : \mathbb{R}^{d^r} \longrightarrow \mathbb{R}^d$ being the bilinear resizing decoder[1].

Furthermore, we follow the approach of smoothing the discontinuous objective function in Equation (1), which has been found to be beneficial in previous work (Chen et al., 2017; Tu et al., 2018; Alzantot et al., 2018). Together with the dimensionality reduction, this leads to the following black-box objective problem for our Bayesian optimisation:

$$\boldsymbol{\delta}^* = \arg\max_{\boldsymbol{\delta}} y(\boldsymbol{\delta}) \quad \text{s.t.} \quad \boldsymbol{\delta} \in [-\delta_{max}, \delta_{max}]^{d^r} \tag{2}$$

$$\text{where} \quad y(\boldsymbol{\delta}) = \left[ \log f(\boldsymbol{x}_{origin} + g(\boldsymbol{\delta}))_t - \log \sum_{j \neq t}^{C} f(\boldsymbol{x}_{origin} + g(\boldsymbol{\delta}))_j \right]$$

## 4.2 GP-BASED BAYESOPT ATTACK

We first use BayesOpt with a standard GP as the surrogate to solve for the black-box attack objective in the reduced input dimension in Equation (2). The GP encodes our prior belief on the objective $y$ :

$$y \sim \mathcal{GP}(\mu(\boldsymbol{\delta}), k(\boldsymbol{\delta}, \boldsymbol{\delta}')) \tag{3}$$

which is specified by a mean function $\mu$ and a kernel/covariance function $k$ (we use the Matern-5/2 kernel in our work). In our work, we normalise the objective function value and thus use a zero-mean prior $\mu(\cdot) = 0$. The predictive posterior distribution for $y_t$ at a test point $\delta_t$ conditioned on the observation data $\mathcal{D}_{t-1} = \{\boldsymbol{\delta}_i, y_i\}_{i=1}^{t-1} = \{\boldsymbol{\delta}_{1:t-1}, \boldsymbol{y}_{1:t-1}\}$ then has the form:

$$p(y_t|\boldsymbol{\delta}_t, \mathcal{D}_{t-1}) = \mathcal{GP}(y; \mu_y(\cdot), k_y(\cdot, \cdot)) \tag{4}$$

where

$$\mu_y(\boldsymbol{\delta}_t) = \boldsymbol{k}(\boldsymbol{\delta}_t, \boldsymbol{\delta}_{1:t-1}) \boldsymbol{K}_{1:t-1}^{-1} \boldsymbol{y}_{1:t-1}, \tag{5}$$

$$k_y(\boldsymbol{\delta}_t, \boldsymbol{\delta}_t') = k(\boldsymbol{\delta}_t, \boldsymbol{\delta}_t') - \boldsymbol{k}(\boldsymbol{\delta_t}, \boldsymbol{\delta}_{1:t-1}) \boldsymbol{K}_{1:t-1}^{-1} \boldsymbol{k}(\boldsymbol{\delta}_{1:t-1}, \boldsymbol{\delta}_t'). \tag{6}$$

where $\mathbf{K}_{1:t-1} = \mathbf{K}(\boldsymbol{\delta}_{1:t-1}, \boldsymbol{\delta}_{1:t-1})$ is the $t-1 \times t-1$ matrix with pairwise covariances $k(\boldsymbol{\delta}_l, \boldsymbol{\delta}_m')$, $l, m \leq t-1$ as entries. The optimal GP hyper-parameters $\boldsymbol{\theta}^*$ such as the length scales and variance of the kernel function $k$ can be learnt by maximising the marginal likelihood $p(\mathcal{D}_{t-1}|\boldsymbol{\theta})$ which has analytic form as presented in Appendix C. For a detailed introduction on GPs, please refer to (Rasmussen, 2003).

Based on the predictive posterior distribution, we construct the acquisition function $\alpha(\boldsymbol{\delta}|\mathcal{D}_{t-1})$ to help select the next query point $\boldsymbol{\delta}_t$. The acquisition function can be considered as a utility measure which balances the exploration and exploitation by giving higher utility to input regions where the functional value is high (high $\mu_y$) and where the model is very uncertain (high $k_y$). The approach of using BayesOpt with a GP surrogate to attack the target model is described in Algorithm 1.

[1]Note that our method is not dependent on a particular choice of decoder, i.e. it could also be used together with an autoencoder- or PCA-based dimensionality reduction technique.

**Algorithm 1** BayesOpt Attack

1: **Input:** A black-box function $y$, observation data $\mathcal{D}_0$, iteration budget $T$, a decoder $g(\cdot)$
2: **Output:** The best recommended adversarial example $x^* = \boldsymbol{x}_{origin} + g(\boldsymbol{\delta}^*)$
3: **for** $t = 1, \ldots, T$ **do**
4: Select $\boldsymbol{\delta}_t = \arg\max \alpha_t(\boldsymbol{\delta}|\mathcal{D}_{t-1})$
5: $y_t = y(\boldsymbol{\delta}_t)$ and $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup (\boldsymbol{\delta}_t, y_t)$
6: Update the surrogate model with $\mathcal{D}_t$
7: **end for**

### 4.3 Additive GP-based BayesOpt Attack

Although techniques such as bilinear resizing are able to reduce the input dimension from the original image size (e.g. $d = 3072$ for CIFAR10 image) to a significantly lower dimension (e.g. $d' = 192$), making the problem amenable to GP-based BO, the reduced search space for the adversarial attack is still considered very high dimensional for GP-based BayesOpt (which is usually applied on problems with $d' \leq 20$).

There are two challenges in using BayesOpt for high dimensional problems. The first is the curse of dimensionality in modelling the objective function. When the unknown function is high-dimensional, estimating it with non-parametric regression becomes very difficult because it is impossible to densely fill the input space with finite number of sample points, even if the sample size is very large (Györfi et al., 2006). The second challenge is the computational difficulty in optimising the acquisition function. The computation cost needed for optimising the acquisition function to within a desired accuracy grows exponentially with dimension (Kandasamy et al., 2015).

We adopt the additive-GP model (Duvenaud et al., 2011; Kandasamy et al., 2015) to deal with the above-mentioned challenges associated with searching for adversarial perturbation in the high dimensional space. The key assumption we make is that the objective can be decomposed into a sum of low-dimensional composite functions:

$$y(\delta) = \sum_{j=1}^{M} y^{(j)}(\delta^{(A_j)}) \tag{7}$$

where $\boldsymbol{\delta}^{(A_j)}$ denotes disjoint low-dimensional subspaces, $\boldsymbol{\delta} = \cup_{j=1}^{M} \boldsymbol{\delta}^{(A_j)}$ and $\boldsymbol{\delta}^{(A_j)} \cap \boldsymbol{\delta}^{(A_i)} = \emptyset$ for all $j \neq i$. If we impose a GP prior for each $y^{(j)}$, the prior for the overall objective $y$ is also a GP: $y \sim \mathcal{GP}(\mu(\boldsymbol{\delta}), k(\boldsymbol{\delta}, \boldsymbol{\delta}'))$ where $\mu(\boldsymbol{\delta}) = \sum_{j=1}^{M} \mu^{(j)}\left(\boldsymbol{\delta}^{(A_j)}\right)$ and $k(\boldsymbol{\delta}, \boldsymbol{\delta}') = \sum_{j=1}^{M} k^{(j)}\left(\boldsymbol{\delta}^{(A_j)}, \boldsymbol{\delta}'^{(A_j)}\right)$. The predictive posterior distribution for each subspace $p(y_{1:t-1}^{(j)}|\boldsymbol{\delta}_t^{(A_j)}, \mathcal{D}_t)$ is:

$$\begin{aligned}
\mu_y^{(j)}(\boldsymbol{\delta}_t^{(A_j)}) &= \boldsymbol{k}^{(j)}(\boldsymbol{\delta}_t^{(A_j)}, \boldsymbol{\delta}_{1:t-1}^{(A_j)}) \boldsymbol{K}_{1:t-1}^{-1} \mathbf{y}_{1:t-1}, \\
k_y^{(j)}(\boldsymbol{\delta}_t^{(A_j)}, \boldsymbol{\delta}_t'^{(A_j)}) &= k^{(j)}(\boldsymbol{\delta}_t^{(A_j)}, \boldsymbol{\delta}_t'^{(A_j)}) - \boldsymbol{k}^{(j)}(\boldsymbol{\delta}_t^{(A_j)}, \boldsymbol{\delta}_{1:t-1}^{(A_j)}) \mathbf{K}_{1:t-1}^{-1} \boldsymbol{k}^{(j)}(\boldsymbol{\delta}_{1:t-1}^{(A_j)}, \boldsymbol{\delta}_t'^{(A_j)}).
\end{aligned}$$

In our case, the exact decomposition (i.e. which input dimension belongs to which low-dimensional subspace) is unknown but we can learn it together with other GP hyperparameters by maximising marginal likelihood (Kandasamy et al., 2015). Note that in this case, the acquisition function is formulated based on $p(y_t^{(j)}|\boldsymbol{\delta}_t^{(A_j)}, \mathcal{D}_t)$ for each subspace and is also optimised in the low-dimensional subspace, thus leading to much more efficient optimisation task. The optimal perturbations in all the subspaces $\{\boldsymbol{\delta}^{(A_j)*}\}_{j=1}^{M}$ are then combined to give the next query point $\boldsymbol{\delta}_t$.

### 4.4 Learning The Optimal $d^r$

Generating the successful adversarial example $\boldsymbol{x} \in \mathbb{R}^d$ by searching perturbation in a reduced dimension $\boldsymbol{\delta} \in \mathbb{R}^{d^r}$ has become a popular practice that leads to significant improvement in query efficiency (Chen et al., 2017; Tu et al., 2018; Alzantot et al., 2018). However, what the optimal $d^r$ is and how to decide it efficiently have not been investigated in previous work (Chen et al., 2017; Tu et al., 2018; Alzantot et al., 2018). As we shown empirically in Section 5.1, setting $d^r$ arbitrarily can lead to suboptimal attack performance in terms of query efficiency, attack success rate as well

**Algorithm 2** Bayesian selection of $d^r$

1: **Input:** A decoder $g(\cdot)$, observation data $\mathcal{D}_{t-1}^d = \{g(\boldsymbol{\delta}_i), y_i\}_{i=1}^{t-1}$ where $g(\boldsymbol{\delta}_i) \in \mathbb{R}^d$ and a set of possible $d^r : \{d_j^r\}_{j=1}^N$
2: **Output:** The optimal reduced dimension $d^{r*}$ and the corresponding GP model
3: **for** $j = 1, \ldots, N$ **do**
4: $\quad \mathcal{D}_{t-1}^{d_j^r} = \{g^{-1}(g(\boldsymbol{\delta}_i)), y_i\}_{i=1}^{t-1}$ where $g^{-1}(g(\boldsymbol{\delta}_i)) \in \mathbb{R}^{d_j^r}$
5: $\quad$ Fit a GP model to $\mathcal{D}_{t-1}^{d_j^r}$ and computes its maximum marginal likelihood $p(\mathcal{D}_{t-1}^d|\boldsymbol{\theta}^*, d_j^r)$
6: **end for**
7: Select $d^{r*} = \arg\max_{d_j^r \in \{d_j^r\}_{j=1}^N} p(\mathcal{D}_{t-1}^d|\boldsymbol{\theta}^*, d_j^r)$ and its correspond GP model

as perturbation costs while finding a good $d^r$ by trial and error or progressive increasing is computationally expensive and highly inefficient because the optimal $d^r$ varies with different attack input $\boldsymbol{x}_{origin}$. An effective way of learning $d^r$ is thus very important for adversarial attacks. In this section, we propose a rigorous method, which is neatly compatible with our attack technique, to learn the optimal $d^r$ from the query information.

The optimal $d^r$ should be the one that both takes into consideration our prior knowledge on the discrete $d^r$ choices and at the same time best explain the observed query data. Given that our BayesOpt attack uses a statistical surrogate (i.e. GP in our case) to model the unknown relation between the attack objective score $y$ and the adversarial perturbation $\boldsymbol{\delta}$, this naturally translates to the criterion of maximising the posterior for $d^r$:

$$p(d_i^r|\mathcal{D}_{t-1}) = \frac{p(\mathcal{D}_{t-1}|d_i^r)p(d_i^r)}{p(\mathcal{D}_{t-1})} \tag{8}$$

where $p(\mathcal{D}_{t-1}|d_j^r)$ is the marginal likelihood (evidence) of adopting a specific $d_j^r$, $p(d_j^r)$ is our prior over the possible $d^r$ values and $p(\mathcal{D}_{t-1}) = \sum_i p(\mathcal{D}_{t-1}|d_j^r)p(d_j^r)$ is a normalising constant. If we match different $d_j^r$ to different model choice, the problem of choosing $d^r$ then becomes the classic *Bayesian model selection* task (Rasmussen, 2003). In most cases, our prior assumption is that we do not prefer one $d^r$ over another (i.e. flat prior $p(d_j^r) = p(d_i^r)$ for $j \neq i$) and thus we can select $d^r$ by comparing their evidence(MacKay & Mac Kay, 2003):

$$\frac{p(d_j^r|\mathcal{D}_{t-1})}{p(d_j^r|\mathcal{D}_{t-1})} = \frac{p(\mathcal{D}_{t-1}|d_j^r)p(d_j^r)}{p(\mathcal{D}_{t-1}|d_j^r)p(d_j^r)} = \frac{p(\mathcal{D}_{t-1}|d_j^r)}{p(\mathcal{D}_{t-1}|d_j^r)}. \tag{9}$$

The exact computation of the evidence term requires marginalisation over model hyper-parameters, which is intractable. We approximate the integral with point estimates (i.e. marginal likelihood of our GP model): $p(\mathcal{D}_{t-1}|d_j^r) = \int p(\mathcal{D}_{t-1}|\boldsymbol{\theta}, d_j^r)p(\boldsymbol{\theta}|d_j^r)d\boldsymbol{\theta} \approx p(\mathcal{D}_{t-1}|\boldsymbol{\theta}^*, d_j^r)$ where $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} p(\mathcal{D}_{t-1}|\boldsymbol{\theta}, d_j^r)$. For query efficiency, we project the same perturbation query data in the original image dimension $\mathcal{D}_{t-1}^d = \{g(\boldsymbol{\delta}_i), y_i\}_{i=1}^{t-1}$ to different latent spaces to get corresponding low-dimensional training data sets for separate GP models. The GP model that corresponds to $d_j^r$ is trained on $\mathcal{D}_{t-1}^{d_j^r} = \{g^{-1}(g(\boldsymbol{\delta}_i)), y_i\}_{i=1}^{t-1}$ where $g^{-1}(g(\boldsymbol{\delta}_i)) \in \mathbb{R}^{d_j^r}$. Then we select the optimal $d^r$ by comparing the marginal likelihood $p(\mathcal{D}_{t-1}^{d_j^r}|\boldsymbol{\theta}^*, d_j^r)$ of each GP surrogate. The overall procedure for $d^r$ selection is described in Algorithm 2. We would like to highlight that the use of the statistical surrogate in our BayesOpt attack approach enables us to naturally use the Bayesian model selection technique to learn the optimal $d^r$ that automatically enjoy the trade-off between data-fit quality and model complexity. And we also show empirically in Section 5.3 that by automating and incorporating the learning of $d^r$ into our BayesOpt attack, we can gain higher success rate and query efficiency. Other adversarial attacks methods can also use our proposed approach to decide $d^r$ but it would require the additional efforts of constructing statistical models to provide $p(\mathcal{D}_{t-1}|d_j^r)$.

## 5 EXPERIMENTS

We empirically compare the performance of our BayesOpt attacks against the state-of-the-art black-box methods such as ZOO (Chen et al., 2017), AutoZOOM(Tu et al., 2018) and GenAttack (Alzantot

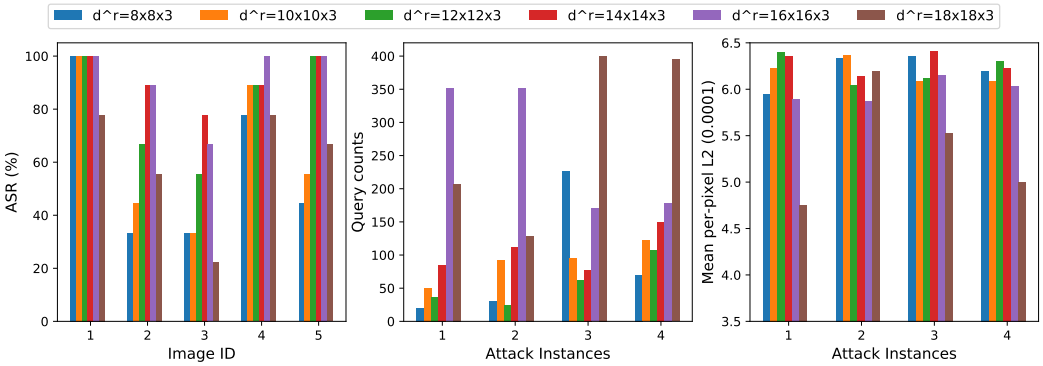



Figure 1: Performance of BayesOpt attack with GP surrogate in different reduced dimension $d^r$. Each color represents a particular $d^r$. The **left** subplot shows the attack success rates(ASR) for attacking all the other 9 classes on each of the 5 original images. The $d^r$ that leads to the highest ASR varies with the original images. The **middle** subplot shows the number of queries needed to attack the same attack instances (i.e. the same original image and the same attack target) by using different $d^r$. For the same 4 attack instances, the **right** subplot show the effect of the choice of $d^r$ on the $L_2$ distances between the resultant adversarial examples and the original image.

et al., 2018). We denote the BayesOpt attack with normal GP surrogate as GP-BO, its variant that learns the $d^r$ automatically is GP-BO-auto-$d^r$ as well as the attack with additive GP surrogate as ADDGP-BO. The target models that we attack follow the same architectures as that used in AutoZOOM and GenAttack; These are image classifiers for MNIST (a CNN with 99.5% test accuracy) and CIFAR10 (a CNN with 80% test accuracy). Following the experiment design in (Tu et al., 2018), we randomly select 3 correctly classified images for each class from CIFAR10 test data which sums up to 27 CIFAR10 images, and randomly select 7 correctly classified images from MNIST test data. We then perform targeted attacks on these images. Each selected image is attacked 9 times, targeting at all but its true class and this gives a total of 243 attack instances for CIFAR10 and 63 attack instances for MNIST. We set $\delta_{max} = 0.3$ for attacking MNIST and $\delta_{max} = 0.05$ for CIFAR10, which are used in (Alzantot et al., 2018). We use the recommended parameter setting and their open sourced implementations for performing all competing attack methods (Chen et al., 2017; Tu et al., 2018; Alzantot et al., 2018).

### 5.1 Effect of $d_r$

We first empirically investigate the effect of the reduced dimensionality $d^r$ of the latent space in which we search for the adversarial perturbation. We experiment with the GP-based BayesOpt attacks for the CIFAR10 classifier using reduced dimension of $d_r = \{6\times6\times3, 8\times8\times3, 10\times10\times3, 12\times12\times3, 14\times14\times3, 16\times16\times3, 18\times18\times3\}$ and perform targeted attacks on 5 target images with each image being attacked 9 times, leading to 45 attack instances. We first investigate the attack success rate (ASR) achieved at different $d^r$ for all the 5 images. The results on ASR out of the 9 targeted attacks for each of the 5 images, which are indicated by Image ID 1 to 5, are shown in the left subplot of Figure 1. It's evident that the $d^r$ which leads to highest attack success rate varies for different original images $\boldsymbol{x}_{origin}$.

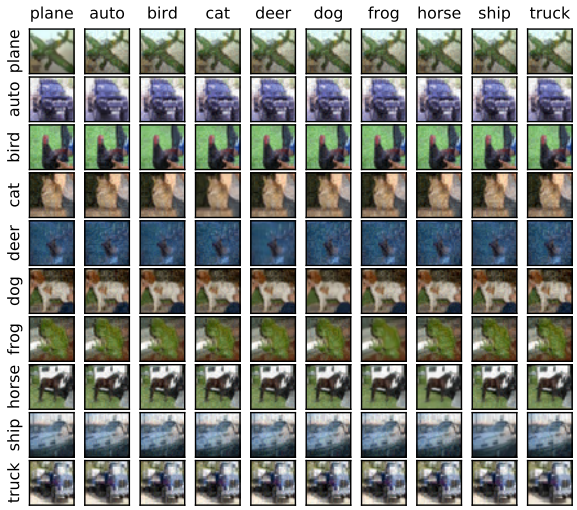



Figure 2: CIFAR10 adversarial examples generated by our BayesOpt attack. True labels and target labels correspond to the rows and columns.

Table 1: Summary of attack results on MNIST by using $d^r = 14 \times 14 \times 3$. $Q$ denotes the query count. *ASR* denotes attack success rate. The standard errors are in parentheses.

| *Attack method* | $ASR$ | *Q (Max, Median, Mean)* | Average $L_2$ perturbation (per pixel) |
|---|---|---|---|
| ADDGP-BO | 98% | $584, 34, 67(\pm 13)$ | $6.50 \times 10^{-3}(\pm 8.39 \times 10^{-5})$ |
| GP-BO-auto-$d^r$ | 98% | $833, 67, 134(\pm 14)$ | $6.48 \times 10^{-3}(\pm 1.68 \times 10^{-4})$ |
| GP-BO | 92% | $899, 53, 119(\pm 28)$ | $7.01 \times 10^{-3}(\pm 1.13 \times 10^{-4})$ |
| GenAttack | 97% | $976, 184, 239(\pm 22)$ | $5.56 \times 10^{-3}(\pm 9.25 \times 10^{-5})$ |
| AutoZOOM | 97% | $892, 186, 247(\pm 27)$ | $5.10 \times 10^{-3}(\pm 1.92 \times 10^{-4})$ |
| ZOO | 3% | $0, 0, 0(\pm 0)$ | $2.83 \times 10^{-3}(\pm 2.37 \times 10^{-6})$ |

Table 2: Summary of attack results on CIFAR10 with $d^r = 14 \times 14 \times 3$. $Q$ denotes the query count. *ASR* denotes attack success rate.The standard errors are in parentheses.

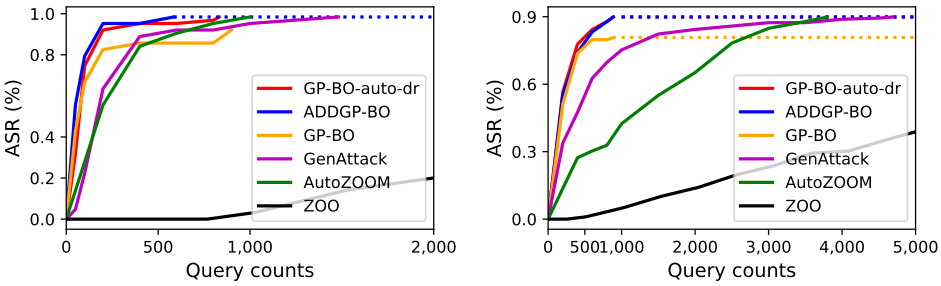
| *Attack method* | *ASR* | *Q (Max, Median, Mean)* | Average $L_2$ perturbation (per pixel) |
|---|---|---|---|
| ADDGP-BO | 90% | $885, 141, 213(\pm 15)$ | $5.79 \times 10^{-4}(\pm 5.16 \times 10^{-6})$ |
| GP-BO-auto-$d^r$ | 90% | $890, 153, 211(\pm 14)$ | $5.90 \times 10^{-4}(\pm 8.10 \times 10^{-6})$ |
| GP-BO | 81% | $899, 142, 192(\pm 11)$ | $5.74 \times 10^{-4}(\pm 6.96 \times 10^{-6})$ |
| GenAttack | 75% | $971, 251, 321(\pm 21)$ | $7.35 \times 10^{-4}(\pm 6.92 \times 10^{-6})$ |
| AutoZOOM | 42% | $376, 95, 133(\pm 12)$ | $9.82 \times 10^{-4}(\pm 1.95 \times 10^{-5})$ |
| ZOO | 5% | $768, 256, 384(\pm 65)$ | $1.83 \times 10^{-4}(\pm 1.86 \times 10^{-5})$ |

We then examines how the $d^r$ affects the query efficiency and attack quality (average $L_2$ distance of the adversarial image from the original image) for the attack instances (e.g. make the classifier to mis-classify a airplane image as a cat) that GP-based BayesOpt can attack successfully at all $d^r$. We present the results on 4 attack instances of a airplane image in the middle and right subplots of Figure 1. We can see that even for the same original image and attack instances, varying $d^r$ can impact query efficiency and $L_2$ norm of the successful adversarial perturbation significantly. For example, $d^r = 8 \times 8 \times 3$ is most query efficient for attack instance 1 and 4 but is outperformed by other dimensions in attack instance 2 and 3. Therefore, the importance of $d^r$ and the difficulty of finding the optimal $d^r$ for a specific target/image motivates us to derive our method for learning it automatically from the data. As shown in the following sections, our $d^r$ learner does lead to more superior attack performance.

### 5.2 Performance under a fixed query budget

In this experiment, we limit the total query number to be 1000, which is slightly above the median query counts needed for GenAttack to make a successful attack on MNIST and CIFAR10 (Alzantot et al., 2018). We adopt the reduced search dimension recommended by AutoZOOM, $d^r = 14 \times 14 \times 3$ for all methods. For BayesOpt attack, each iteration requires 1 query to the objective function so we limit its iteration to 1000 and early terminate the BayesOpt attack algorithm when successful adversarial example is found. AutoZOOM comprises 2 stages in their attack algorithm. The first exploration stage aims to find the successful adversarial example. Once a successful attack is found, it switches to the fine-tuning stage to reduce the perturbation cost (e.g. $L_2$ norm) of the successful attack. We report its performance on the attack success rate and $L_2$ norm of adversarial perturbations after a budget of 1000 queries, which allows it to fine-tune the successful adversarial perturbations found. Moreover, AutoZOOM uses $L_2$ norm to measure the perturbation costs but our method and GenAttack limit the search space via $L_\infty$ norm. We observe that the successful attacks found by AutoZOOM incur much higher perturbation costs in terms of $L_\infty$ norm than GenAttack and our method. Therefore, we only consider the final adversarial examples whose $L_\infty$ distances from the original images lie within $[-\delta_{max}, \delta_{max}]$ as the successful attacks [2]. The results on MNIST in Table 1 show that all our BayesOpt attack can achieve a comparable success rate but at a much

[2]This still gives an advantage to ZOO and AutoZOOM because it allows them to inject perturbation whose magnitude is larger than $\boldsymbol{\delta}_{max}$ at certain pixels but our BayesOpt methods and GenAttack limit perturbation at all the pixels to be less than $\boldsymbol{\delta}_{max}$.




(a) Attacks on MNIST

(b) Attacks on CIFAR10

Figure 3: Query efficiency of BayesOpt Attacks. Within a budget of 1000 queries, the best BayesOpt Attacks (i.e. GP-BO-auto-$d^r$ and ADDGP-BO) can achieve a attack success rate of 98% on MNIST(a) and 90% on CIFAR10 (b). To achieve the same success rates, GenAttack(in purple) takes 1481 for MNIST and 4691 for CIFAR10 and AutoZOOM(in green) takes 1000 for MNIST and 3880 for CIFAR10.

lower query count (71% less for simple GP-BO and 81% less for ADDGP-BO ) in comparison to GenAttack (Alzantot et al., 2018) and AutoZOOM (Tu et al., 2018). As for results on attacking on CIFAR10, Table 2 shows that all our BayesOpt attack can achieve significantly higher success rate but again at a remarkably lower query counts than the existing black-box approaches. For example, our ADDGP-BO can achieve 15% higher success rate while using 43.8% less queries in terms of the median as compared to GenAttack. In addition, our approaches also lead to better quality adversarial examples (Figure 2) which are closer to the original image than the benchmark methods as reflected by the lower average $L_2$ perturbation (21.9% less). More importantly, this set of experiments also demonstrate the effectiveness of our Bayesian method for learning $d^r$ as GP-BO-auto-$d^r$ leads to 9% increase in attack success rate compared to GP-BO while maintaining the competitiveness in query efficiency and $L_2$ distance.

### 5.3 Query efficiency comparison

We finish by comparing the query efficiency, measuring the change in the attack success rate over query counts for all the methods. We limit the query budget of our BayesOpt attacks to 1000 but let the competing methods to continue running until they achieve the same best attack success rate as our best BayesOpt attacks or exceeds a much higher limit (2000 queries for MNIST and 5000 queries for CIFAR10). As shown in Figure 3, BayesOpt attacks converge much faster to the high attack success rates than the other methods. Specifically, ADDGP-BO takes only 584 queries to achieve the success rate of 98% for MNIST, which is 50% of the query count by AutoZOOM(1000) and 30% of that by GenAttack (1481). As for CIFAR10, both ADDGP-BO and GP-BO-auto-$d^r$ takes around 890 queries to achieve a success rate of 90% which is 23% of the query count by AutoZOOM and 19% of that by GenAttack. One point to note is that AutoZOOM appears to be slightly more query efficient than GenAttack in this set of experiments. However, we need to bear in mind that although we limit the mean $L_{\text{inf}}$ norm of the adversarial perturbation found by AutoZOOM to $\delta_{max}$, AutoZOOM still have the advantage of exploring beyond the $\delta_{max}$ for many dimensions.

## 6 Conclusion

We introduce a new black-box adversarial attack which leverages Bayesian optimisation to find successful adversarial perturbations with high query efficiency. We also improve our attack by adopting an additive surrogate structure to ease the optimisation challenge over the typically high-dimensional task. Moreover, we take full advantage of our statistical surrogate model and the available query data to learn the optimal degree of dimension reduction for the search space via Bayesian model selection. In comparison to several existing black-box attack methods, our BayesOpt attacks can achieve high success rates with 2-5 times fewer queries while still producing adversarial examples that are closer to the original image (in terms of average $L_2$ distance). We believe our BayesOpt attacks can be a competitive alternative for accessing the model robustness, especially in real-world applications where the available query budget is highly limited and the model evaluation is expensive or risky.

## REFERENCES


Moustafa Alzantot, Yash Sharma, Supriyo Chakraborty, and Mani Srivastava. Genattack: Practical black-box attacks with gradient-free optimization. *arXiv preprint arXiv:1805.11090*, 2018.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

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.

Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57. IEEE, 2017.

Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden voice commands. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pp. 513–530, 2016.

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*, pp. 15–26. ACM, 2017.

Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Ead: elastic-net attacks to deep neural networks via adversarial examples. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

David K Duvenaud, Hannes Nickisch, and Carl E Rasmussen. Additive gaussian processes. In *Advances in neural information processing systems*, pp. 226–234, 2011.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017.

László Györfi, Michael Kohler, Adam Krzyzak, and Harro Walk. *A distribution-free theory of nonparametric regression*. Springer Science & Business Media, 2006.

JB Heaton, NG Polson, and Jan Hendrik Witte. Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*, 33(1):3–12, 2017.

Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. *arXiv preprint arXiv:1804.08598*, 2018.

Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional Bayesian optimisation and bandits via additive models. In *International Conference on Machine Learning*, pp. 295–304, 2015.

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

Yi Liu, Jie Ling, Zhusong Liu, Jian Shen, and Chongzhi Gao. Finger vein secure biometric template generation based on deep learning. *Soft Computing*, 22(7):2257–2265, 2018.

David JC MacKay and David JC Mac Kay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Seungyong Moon, Gaon An, and Hyun Oh Song. Parsimonious black-box adversarial attacks via efficient combinatorial optimization. *arXiv preprint arXiv:1905.06635*, 2019.

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*, pp. 2574–2582, 2016.

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*, pp. 506–519. ACM, 2017.

C E Rasmussen and C K I Williams. *Gaussian processes for machine learning*. 2006.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pp. 63–71. Springer, 2003.

Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. Is robustness the cost of accuracy?–a comprehensive study on the robustness of 18 deep image classification models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 631–648, 2018.

Fnu Suya, Yuan Tian, David Evans, and Paolo Papotti. Query-limited black-box attacks to classifiers. *NIPS Workshop*, 2017.

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.

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. *arXiv preprint arXiv:1805.11770*, 2018.

David Warde-Farley and Ian Goodfellow. 11 adversarial perturbations of deep neural networks. *Perturbations, Optimization, and Statistics*, 311, 2016.

## A BAYESIAN OPTIMISATION

Here we present a generic algorithm for Bayesian optimisation.

**Algorithm 3** BayesOpt Algorithm

1: **Input:** A black-box function $y$, observation data $\mathcal{D}_0$, iteration budget $T$
2: **Output:** The best recommendation $\delta^*$
3: **for** $t = 1, \ldots, T$ **do**
4: Select $\delta_t = \arg\max \alpha_t(\delta|\mathcal{D}_{t-1})$
5: $y_t = y(\delta_t)$ and $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup (\delta_t, y_t)$
6: Update the surrogate model with $\mathcal{D}_t$
7: **end for**

## B ADDITIONAL EXPERIMENTS ON MNIST AND CIFAR10

Table 3: Summary of attack results on MNIST. $Q$ denotes the query count. *ASR* denotes attack success rate. The standard errors are in parentheses.

| *Attack method* | $d_r$ | $ASR$ | *Q (Max, Median, Mean)* | Average $L_2$ perturbation (per pixel) |
|---|---|---|---|---|
| GP-BO | $14 \times 14 \times 3$ | 92% | $899, 53, 119(\pm 28)$ | $7.01 \times 10^{-3}(\pm 1.13 \times 10^{-4})$ |
| ADDGP-BO | $14 \times 14 \times 3$ | 98% | $584, 34, 67(\pm 13)$ | $6.50 \times 10^{-3}(\pm 8.39 \times 10^{-5})$ |
| AutoZOOM | $14 \times 14 \times 3$ | 97% | $892, 186, 247(\pm 27)$ | $5.10 \times 10^{-3}(\pm 1.92 \times 10^{-4})$ |
| GenAttack | $14 \times 14 \times 3$ | 97% | $976, 184, 239(\pm 22)$ | $5.56 \times 10^{-3}(\pm 9.25 \times 10^{-5})$ |
| GP-BO | $16 \times 16 \times 3$ | 97% | $400, 50, 80(\pm 12)$ | $7.08 \times 10^{-3}(\pm 1.09 \times 10^{-4})$ |
| ADDGP-BO | $16 \times 16 \times 3$ | 100% | $540, 42, 72(\pm 12)$ | $6.49 \times 10^{-3}(\pm 9.41 \times 10^{-5})$ |
| AutoZOOM | $16 \times 16 \times 3$ | 98% | $812, 251, 295(\pm 28)$ | $5.98 \times 10^{-3}(\pm 2.03 \times 10^{-4})$ |
| GenAttack | $16 \times 16 \times 3$ | 97% | $928, 202, 237(\pm 19)$ | $5.61 \times 10^{-3}(\pm 8.58 \times 10^{-5})$ |
| GP-BO | $28 \times 28 \times 3$ | 98% | $430, 201, 225(\pm 11)$ | $8.71 \times 10^{-3}(\pm 1.94 \times 10^{-4})$ |
| ADDGP-BO | $28 \times 28 \times 3$ | 100% | $666, 57, 100(\pm 15)$ | $9.71 \times 10^{-3}(\pm 1.12 \times 10^{-4})$ |
| AutoZOOM | $28 \times 28 \times 3$ | 98% | $860, 185, 244(\pm 27)$ | $1.01 \times 10^{-2}(\pm 1.63 \times 10^{-4})$ |
| GenAttack | $28 \times 28 \times 3$ | 83% | $976, 365, 399(\pm 33)$ | $6.26 \times 10^{-3}(\pm 6.69 \times 10^{-5})$ |

Table 4: Summary of attack results on CIFAR10. $Q$ denotes the query count. *ASR* denotes attack success rate. The standard errors are in parentheses.

| *Attack method* | $d_r$ | *ASR* | *Q (Max, Median, Mean)* | Average $L_2$ perturbation (per pixel) |
|---|---|---|---|---|
| GP-BO | $8 \times 8 \times 3$ | 52% | $314, 48, 70(\pm 13)$ | $5.78 \times 10^{-4}(\pm 1.44 \times 10^{-5})$ |
| ADDGP-BO | $8 \times 8 \times 3$ | 75% | $899, 140, 234(\pm 33)$ | $5.55 \times 10^{-4}(\pm 8.46 \times 10^{-6})$ |
| AutoZOOM | $8 \times 8 \times 3$ | 56% | $382, 126, 139(\pm 17)$ | $9.58 \times 10^{-4}(\pm 2.66 \times 10^{-5})$ |
| GenAttack | $8 \times 8 \times 3$ | 67% | $671, 236, 286(\pm 32)$ | $7.40 \times 10^{-4}(\pm 1.30 \times 10^{-5})$ |
| GP-BO | $14 \times 14 \times 3$ | 81% | $899, 142, 192(\pm 11)$ | $5.74 \times 10^{-4}(\pm 6.96 \times 10^{-6})$ |
| ADDGP-BO | $14 \times 14 \times 3$ | 90% | $885, 141, 213(\pm 15)$ | $5.79 \times 10^{-4}(\pm 5.16 \times 10^{-6})$ |
| AutoZOOM | $14 \times 14 \times 3$ | 42% | $376, 95, 133(\pm 12)$ | $9.82 \times 10^{-4}(\pm 1.95 \times 10^{-5})$ |
| GenAttack | $14 \times 14 \times 3$ | 75% | $971, 251, 321(\pm 21)$ | $7.35 \times 10^{-4}(\pm 6.92 \times 10^{-6})$ |
| GP-BO | $16 \times 16 \times 3$ | 83% | $785, 280, 292(\pm 22)$ | $5.34 \times 10^{-4}(\pm 1.43 \times 10^{-5})$ |
| ADDGP-BO | $16 \times 16 \times 3$ | 87% | $878, 168, 229(\pm 29)$ | $5.76 \times 10^{-4}(\pm 8.93 \times 10^{-6})$ |
| AutoZOOM | $16 \times 16 \times 3$ | 3% | $298, 170, 170(\pm 90)$ | $7.50 \times 10^{-4}(\pm 1.35 \times 10^{-5})$ |
| GenAttack | $16 \times 16 \times 3$ | 79% | $986, 281, 339(\pm 36)$ | $7.36 \times 10^{-4}(\pm 1.55 \times 10^{-5})$ |

# C GAUSSIAN PROCESSES

Gaussian processes (GPs) are popular models for inference in regression problems, especially when a quantification of the uncertainty around predictions is of relevance and little prior information is available to the inference problem. These qualities, together with their analytical tractability, make them the most commonly used surrogate model in Bayesian optimisation. A comprehensive overview on GPs can be found in Rasmussen & Williams (2006). Below we highlight the concepts of marginal likelihood and how to use it for hyperparameter optimisation and model selection based on the notation introduced in Section 4.2.

**Hyperparameter tuning.** The key to finding the right hyperparameters $\boldsymbol{\theta}^*$ in light of $\mathcal{D}_{t-1}$ in a principled way is given by the marginal likelihood in Equation (10).

$$p(\mathcal{D}_{t-1}|\boldsymbol{\theta}) = (2\pi)^{-\frac{t-1}{2}} |\mathbf{K}_{1:t-1}(\boldsymbol{\theta})|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\boldsymbol{y}_{1:t-1}^T \mathbf{K}_{1:t-1}^{-1}(\boldsymbol{\theta})\boldsymbol{y}_{1:t-1}\right), \tag{10}$$

where we have introduced a slightly augmented notation of $\mathbf{K}_{1:t-1} \equiv \mathbf{K}_{1:t-1}(\boldsymbol{\theta})$ to highlight the dependence on the kernel hyperparamaters $\boldsymbol{\theta}$. In a truly Bayesian approach, one could shy away from fixing one set of hyperparameters and use Equation (10) to derive a posterior distribution of $\boldsymbol{\theta}$ based on one's prior beliefs about $\boldsymbol{\theta}$ expressed through some distribution $p_0(\boldsymbol{\theta})$. However, as such an approach generally requires sampling using methods like Markov chain Monte Carlo (MCMC) due to intractability of integrals, in practice it is often easier and computationally cheaper to replace the fully Bayesian approach by a maximum likelihood approach and simply maximise the likelihood $p(\mathcal{D}_{t-1}|\boldsymbol{\theta})$ w.r.t. $\boldsymbol{\theta}$. We follow this approach and perform a mix of grid- and gradient based search to maximise the logarithm of the r.h.s. of Equation (10). After evaluating a grid of $5'000$ points, we start gradient ascent on each of the $5$ most promising points using the following equation for the gradient (Rasmussen & Williams, 2006) to find the hyperparameters $\boldsymbol{\theta}^*$ which maximise the marginal likelihood:

$$\frac{\partial}{\partial\theta_j} \log p(\mathcal{D}_{t-1}|\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{y}_{1:t-1}^T \mathbf{K}_{1:t-1}^{-1} \frac{\partial \mathbf{K}_{1:t-1}}{\partial\theta_j} \mathbf{K}_{1:t-1}^{-1} \boldsymbol{y}_{1:t-1} - \frac{1}{2} tr\left(\mathbf{K}_{1:t-1}^{-1} \frac{\partial \mathbf{K}_{1:t-1}}{\partial\theta_j}\right) \tag{11}$$

$$= \frac{1}{2} tr\left((\mathbf{K}_{1:t-1}^{-1}\boldsymbol{y}_{1:t-1}\boldsymbol{y}_{1:t-1}^T \mathbf{K}_{1:t-1}^{-1} - \mathbf{K}_{1:t-1}^{-1}) \frac{\partial \mathbf{K}_{1:t-1}}{\partial\theta_j}\right) \tag{12}$$

**Model selection.** Once the optimal hyperparameters $\boldsymbol{\theta}^*$ are found as described above, they can be plugged back into Equation (10) to choose amongst different models. In Section 4.4 this is described for the case of choosing between different values of the reduced dimensionality $d^r$.