Transferable Adversarial Examples with Bayesian Approach

Mingyuan Fan
East China Normal University
Shanghai, China
fmy2660966@gmail.com

Wenmeng Zhou
Alibaba Group
Hangzhou, China
wenmeng.zwm@alibaba-inc.com

Cen Chen*
East China Normal University
Shanghai, China
cenchen@dase.ecnu.edu.cn

Yinggui Wang
Ant Group
Beijing, China
wyinggui@gmail.com

Abstract
The vulnerability of deep neural networks (DNNs) to black-box adversarial attacks is one of the most heated topics in trustworthy AI. In such attacks, the attackers operate without any insider knowledge of the model, making the cross-model transferability of adversarial examples critical. Despite the potential for adversarial examples to be effective across various models, it has been observed that adversarial examples that are specifically crafted for a specific model often exhibit poor transferability. In this paper, we explore the transferability of adversarial examples via the lens of Bayesian approach. Specifically, we leverage Bayesian approach to probe the transferability and then study what constitutes a transferability-promoting prior. Following this, we design two concrete transferability-promoting priors, along with an adaptive dynamic weighting strategy for instances sampled from these priors. Employing these techniques, we present BayAtk. Extensive experiments illustrate the significant effectiveness of BayAtk in crafting more transferable adversarial examples against both undefended and defended black-box models compared to existing state-of-the-art attacks.

CCS Concepts
- Do Not Use This Code → Generate the Correct Terms for Your Paper; Generate the Correct Terms for Your Paper; Generate the Correct Terms for Your Paper.

Keywords
Adversarial examples, Deep neural networks, Security, Transferability, Black-box attack

ACM Reference Format:
Mingyuan Fan, Cen Chen, Wenmeng Zhou, and Yinggui Wang. 2018. Transferable Adversarial Examples with Bayesian Approach. In Proceedings of Make sure to enter the correct conference title from your rights confirmation

*Corresponding author.

1 Introduction
Deep neural networks (DNNs) have achieved remarkable success in a broad spectrum of domains [1, 2, 26]. Despite their impressive performance, DNNs are highly vulnerable to adversarial attacks [8, 25, 35]. Specifically, these attacks manipulate DNNs into making incorrect predictions by employing adversarial examples, generated by imposing carefully-crafted slight perturbations on natural samples. This inherent vulnerability of DNNs presents a significant challenge for deploying DNNs in security-critical scenarios [24, 35]. Consequently, the identification of DNN vulnerabilities is a top priority, and developing sophisticated adversarial attacks emerges as a cornerstone of research in the AI security area.

There have been substantial advances in the development of adversarial attacks. These attacks operate under a white-box assumption [12, 15, 22], where attackers can access the target model's details, including specific architectures and parameters. However, the white-box assumption does not align with typical real-world situations where the internal specifics of the target model remain opaque to attackers, i.e., a black-box scenario. Hence, launching adversarial attacks in the black-box scenario is more challenging and practical.

There are two primary types of black-box attacks, namely query-based attacks [25] and transfer-based attacks [28, 30]. Query-based attacks entail an avalanche of queries to the target model for estimating the necessary information for crafting adversarial examples (input gradients). However, this resource-intensive query budget incurs not only considerable expense but also risks detection by model administrators, thereby constraining their practical applicability. In contrast, another approach involves employing a proxy model to generate adversarial examples, known as transfer-based attacks. The underlying intuition is that if the proxy model shares certain similarities with the target model, the adversarial examples generated by the proxy model may also transfer successfully to the target model.

The effectiveness of transfer-based attacks largely relies on the transferability of adversarial examples. Unfortunately, empirical examinations [34, 36] indicate that such examples often present limited transferability, sparking considerable research interest in enhancing transferability. Intuitively, the concept of transferability can be likened to the generalization ability of models. Generalization refers to a model’s capability to accurately classify unseen data, whereas transferability pertains to the adversarial examples’ effectiveness in...
deceiving unknown target models. Drawing from this analogy, improving transferability can be viewed through the lens of enhancing generalization. One well-established method to boost generalization is the Bayesian approach, which incorporates prior knowledge into model parameters to alleviate overfitting issues [3, 32]. However, the field lacks investigations into studying transferability via the lens of the Bayesian approach, leaving a research gap.

To bridge this gap, this paper represents the first attempt to leverage the Bayesian approach to study transferability. Our investigation aims to answer three fundamental questions: first, how to integrate the transferability of adversarial examples within the Bayesian framework; second, the essence of the Bayesian approach lies in prior selection, yet it remains unclear what constitutes an appropriate prior for the transferability of adversarial examples; and third, after identifying such priors, how to develop suitable priors. Our responses to these questions lay the foundation for future research seeking to merge the Bayesian approach with the transferability of adversarial examples:

- To tackle the first question, we first revisit the conventional approach of generating adversarial examples on the proxy model. This re-examination elucidates that the vanilla transfer-based attack can be interpreted as a Maximum Likelihood Estimation (MLE) problem, under the assumption of a constant prior distribution. While MLE can craft optimal adversarial examples for a specific proxy model, it does not account for the performance of such adversarial examples against different models. Thus, we embrace the Bayesian approach by substituting the MLE’s prior with more suitable priors.
- Regarding the second question, we base our definition of a suitable prior on the decision-making process of DNNs. Specifically, DNNs work by extracting and analyzing features contained in images, and poor transferability is often due to the excessive concentration on corrupting proxy-model-specific features. In response, we define transferability-promoting prior as one that enables the corruption of a broader range of features.
- To answer the third question, we develop pixel-level removal prior and region-based soft removal prior. We demonstrate that both priors qualify as transferability-promoting prior. Furthermore, recognizing that the amount of feature information may vary in instances drawn from the priors, we introduce an adaptive dynamic weighting strategy to mitigate.

Building on these advances, we introduce our attack method termed as BayAtk, which generates adversarial examples by solving an optimization problem associated with the proposed priors and the adaptive dynamic weighting strategy. The extensive experiments in the benchmark dataset ImageNet show the superior transferability of adversarial examples generated by BayAtk over state-of-the-art transfer-based attack methods. Moreover, the adversarial examples crafted by BayAtk also present considerable effectiveness in misleading the physical-world large multi-modal model Claude3. To summarize, our contributions are threefold:

- We are the first to leverage the Bayesian approach to investigate the transferability of adversarial examples. Our answers to the first and second questions not only bridge a significant research gap but also pave the way for a novel line that intertwines the Bayesian approach with the transferability of adversarial examples.
- We develop pixel-level removal prior and region-based soft removal prior and demonstrate their effectiveness as transferability-promoting prior. Moreover, we design an adaptive dynamic weighting strategy to mitigate the issue of varying feature information amongst different instances sampled from priors.
- We conduct extensive experiments on the large-scale dataset ImageNet and Claude3 to examine the performance of BayAtk. The results show that BayAtk generates more transferable adversarial examples compared to state-of-the-art attacks.

2 Background & Related Work

2.1 Adversarial Attack

Adversarial attacks impose human-imperceptible adversarial perturbation into natural samples to fool the target model and can be roughly categorized into two types: white-box attacks and black-box attacks. White-box attacks assume that the attackers have full knowledge of the architecture and parameters of the target model; in contrast, attackers are not allowed to access the internal information about the target model in the black-box scenario.

White-box Attacks. White-box attacks [12, 15, 22] typically infuse inputs with the sign of the input gradient to generate adversarial examples. One of the most prominent techniques among white-box attacks is the Fast Gradient Sign Method (FGSM) [12], which crafts adversarial examples through a single-step adjustment aligned with the gradient of the input’s loss function. Let x denote the input image and y its ground-truth label. Given the target model T and loss function L(·), FGSM crafts an adversarial example x_{adv} for x as follows:

\[ x_{adv} = x + \alpha \cdot \text{sign}(\nabla_x L(T(x), y)), \]

where \( \alpha \) is the step size (typically set to \( \alpha = \frac{1}{255} \)) and \text{sign}(\cdot) is element-wise sign operation. FGSM just needs a single forward and backward propagation and thus is computationally efficient. Building on FGSM, basic iterative method (BIM) [15] refines the adversarial example generation process by applying the FGSM technique iteratively with small step sizes. BIM is formulated as follows:

\[ x_{t+1} = \Pi_{x, \epsilon} \{ x_t + \alpha \cdot \text{sign}(\nabla_x L(T(x_t), y)) \}, \]

\[ t = 0, \ldots, T - 1, \quad x_0 = x, \quad x_{adv} = x_T, \]

where \( x_t \) is the adversarial example obtained after the \( t \)-th iteration, \( \epsilon \) is the perturbation magnitude, and \( \Pi_{x, \epsilon} \) projects inputs back into a permissible range centered around \( x \) with radius \( \epsilon \). Projected gradient descent (PGD) [22] is like an extension of BIM but with a random start, i.e., \( x_0 \sim \mathcal{U}(x - \epsilon, x + \epsilon) \) where \( \mathcal{U}(a, b) \) denotes a uniform distribution between \( a \) and \( b \). The randomness helps in avoiding poor initial points that could lead to less effective adversarial examples. Although these attacks are designed in the white-box scenario, they commonly serve as a backend component for transfer-based attacks.

Black-box Attacks. In the black-box scenario, attackers employ two primary strategies: query-based attacks [25] and transfer-based attacks [6, 7, 21, 30]. We here mainly review transfer-based attacks, as this paper focuses on exploring the transferability of adversarial examples. Transfer-based attacks commonly train a proxy model \( M \) and then produce adversarial examples with the proxy model using
white-box attacks like BIM against the target model. However, adversarial examples produced purely through white-box attacks often fail to effectively mislead unknown target models [6, 28]. As such, several methods have been proposed to enable more transferable adversarial examples.

From the optimization perspective, momentum attack (MI) [6] incorporates momentum trick into BIM to stabilize the update direction and escape local optima. Follow-up works have improved MI by leveraging more advanced optimization methods [19, 29, 37].

From the model perspective, ensemble attack [20] improves transferability by attacking multiple models at once. Gao et al. [11] targeted the feature space of the proxy model to create more transferable adversarial examples. Other research [9, 16] explored which model architectures can enable better transferability.

From the input perspective, diverse input (DI) [34] resizes and pads input images before gradient calculation. Scale invariant method (SIM) [18] leverages the gradient information from images at different scales, and Admix [30] introduces a minor fraction of images from alternative categories into the original input image. Spectrum simulation attack (SSA) [21] aims to augment the proxy model in frequency space, which can be approximated by applying a spectrum transformation in inputs. PAM [36] incorporates multiple augmentation pathways to diversify input images. BSR [28] randomly shuffles and rotates the different regions of inputs. TPA [10] penalizes gradients of generated adversarial examples with random noises.

### 2.2 Adversarial Defense

The surge in the sophistication of adversarial attacks has driven progress in the development of countermeasures to these threats. The most straightforward and effective way is adversarial training [27, 35], in which adversarial examples are integrated into the training phase to instruct the model on the correct identification and management of these inputs. Despite its effectiveness, adversarial training is accompanied by significant training expenses due to the costly nature of generating adversarial examples.

As an alternative, input transformation-based defenses [4, 13, 17, 23, 33] present a cost-effective option, which tries to remove adversarial perturbations before feeding them to DNNs. Basic methods in this category include JPEG compression, total variance minimization, and image quilting. More sophisticated methods commonly utilize DNNs. High-level Representation Guided Denoiser (HGD) [17] employs a U-Net while Neural Representation Purifier (NRP) [23] learns to purify adversarial examples in a self-supervised manner.

### 3 The Proposed Attack: BayAtk

In this section, we develop BayAtk, which is outlined in Algorithm 1 and Figure 1.

#### 3.1 Motivation

The generation of adversarial examples can be formulated as an optimization task that shares similarities with the training process of DNNs. The essence of training a DNN lies in identifying a suitable set of parameters that enables the model’s high performance on both seen and unseen data. The primary goal of transfer-based attacks is to produce adversarial examples that are transferable across a diverse range of black-box models. Therefore, the transferability of adversarial examples is naturally akin to the generalization ability of DNNs. Recognizing this relationship, it is straightforward to explore model-generalization-promoting techniques to nourish the transferability of adversarial examples.

To this end, we investigate typical model-generalization-promoting techniques [1, 31] and summarize them as follows:

- **More data.** Collecting more data is always a simple yet effective strategy to improve the model’s generalization. In the context of adversarial examples, this strategy amounts to training more models with diverse architectures as the proxy models, i.e., ensemble attack [20]. However, despite the simplicity of this strategy, model training is inherently resource-intensive, not to mention the computational costs involved in training multiple models, especially on large-scale datasets like ImageNet.
- **Robust optimization algorithms.** Robust optimization algorithms, like momentum optimization, enjoy the capacity to prevent convergence into suboptimal local minima. These algorithms have been extensively examined in existing literature [6, 18].
- **Model regularizations.** This strategy involves restricting the model’s expression ability by imposing specific constraints or penalties on its parameters, such as $L_1$ or $L_2$ regularization. These regularizations encourage the model to concentrate on identifying the most discriminative features, thereby mitigating the risk of overfitting. Furthermore, these regularizations embody Bayesian approach, wherein each model parameter is assigned a specific prior distribution, reflecting prior knowledge. For example, the adoption of a Gaussian or Laplacian prior leads to the derivation of $L_1$ or $L_2$ regularizations, respectively.

Drawing upon the above analysis, Bayesian approach can considerably mitigate the overfitting problem and hence enjoys the potential to promote the cross-model transferability of adversarial examples. To our best knowledge, there is an absence of research specifically focused on harnessing Bayesian approach to analyze and boost the transferability of adversarial examples, leaving a considerable research gap. To bridge this gap, we seek to leverage Bayesian approach to study the transferability of adversarial examples.

#### 3.2 Transferable Adversarial Examples with Bayesian Approach

The adversarial example of $x$ can be written as $x_{\text{adv}} = x + \delta$, where $\delta$ is the adversarial perturbation for $x$. We formulate the following optimization task for generating adversarial perturbation $\delta$:

$$\delta^* = \arg \min_{\delta} p(x + \delta, y|M), \ s.t.., ||\delta||_\infty \leq \epsilon,$$

(3)

where $p(x + \delta, y|M)$ represents the joint probability of $x + \delta$ and $y$ given the proxy model $M$, and $\epsilon$ is a constant to control the maximum magnitude of $\delta$, i.e., attack strength. Intuitively, given $x$ and $y$, Equation 3 aims to identify the most effective $\delta$ that minimizes the association between $x + \delta$ and $y$. The above optimization task
Algorithm 1 BayAtk

Input: \( M \): the proxy model; \( x, y \): the natural sample and its ground-truth label; \( \varepsilon \): perturbation budget; \( \delta \): adversarial perturbation; \( T \): the number of iterations; \( \alpha \): step size; \( Q \): the number of instances to sample per iteration.

1: Adversarial perturbation \( \delta \) is randomly initialized within the range between \(-\varepsilon \) and \( \varepsilon \).
2: for each iteration \( i = 1 \) to \( T \) do
3:    Sample \( Q \) instances, denoted as \((x + \delta) \odot m_1, \ldots, (x + \delta) \odot m_Q\) from \( p(x + \delta) \). Usage pixel-level removal prior or region-based soft removal prior.
4:    Compute the optimization objective \( \mathcal{L} = \frac{1}{Q} \sum_{i=1}^{Q} M((x + \delta) \odot m_i)[y] \cdot M((x + \delta) \odot m_i)[y] \). The first \( M((x + \delta) \odot m_i)[y] \) serves as weights, i.e., our adaptive dynamic weighting strategy.
5:    Calculate the gradient \( g = \nabla_{\delta} \mathcal{L} \).
6:    Update adversarial perturbation \( \delta = \delta - \alpha \text{sign}(g) \).
7:    Clip adversarial perturbation \( \delta = \max(\min(\delta, \varepsilon), -\varepsilon) \).

4:     max and min are element-wise operations.
8: end for
9: Return: the crafted adversarial example \( x + \delta \).

can be rewritten as follows:

\[
\delta^* = \arg \min_{\delta} p(x + \delta, y|M) = \arg \min_{\delta} \frac{p(x + \delta, y, M)}{p(x + \delta, M)}
= \arg \min_{\delta} \frac{p(x + \delta, y, M)}{p(x + \delta, M)} \frac{p(x + \delta, M)}{p(M)}
= \arg \min_{\delta} \frac{p(y|x + \delta, M)}{p(x + \delta|M)} \frac{p(x + \delta, M)}{p(M)}
\]

s.t., \(|\delta|\leq \varepsilon \),

where \( p(y|x + \delta, M) \) represents the probability that the proxy model \( M \) classifies \( x + \delta \) as belonging to class \( y \), and \( p(x + \delta|M) \) represents the appearing likelihood of \( x + \delta \) under model \( M \). In common transfer-based attacks \([6, 15]\), \( p(x + \delta|M) \) is often set to 1, hence focusing primarily on the minimization of \( p(y|x + \delta, M) \), i.e., the probability of the model classifying \( x + \delta \) as belonging to class \( y \).

Unfortunately, this often makes resulting adversarial perturbation \( \delta^* \), while potentially optimal for the proxy model \( M \), ineffective when applied to the target model. This reduction in effectiveness is generally attributed to the overfitting of \( \delta^* \) to the proxy model \( M \). By contrast, Bayesian approach advocates for assigning a specific distribution to \( p(x + \delta|M) \) based on prior knowledge, rather than treating \( p(x + \delta|M) \) as a constant. Yet, to our best knowledge, there remains a research gap in understanding what constitutes an appropriate prior for transferable adversarial examples. We first address this gap, followed by designing our attack.

What constitutes a suitable prior? It is widely shared that DNNs classify images by extracting and analyzing features within those images. Accordingly, the effectiveness of adversarial examples is grounded in the capability of their associated adversarial noises to
corrupt these features. For clarity, we categorize these features as either cross-model features or model-specific features. Cross-model features are those that are universally learned by various DNNs, whereas model-specific features are exclusive to each model. This distinction suggests that the limited transferability of adversarial examples can be primarily attributed to their overemphasis on compromising model-specific features while neglecting the cross-model features. In other words, model-specific features offer a shortcut to fool the proxy model and focusing exclusively on optimizing the probability \( p(y|x + \delta, \mathcal{M}) \) may excessively exploit this shortcut. The consequence is the under-emphasis on attacking cross-model features, resulting in adversarial examples that exhibit less transferability.

Let us delve deeper into this analysis. We can describe the probability of the proxy model \( \mathcal{M} \) identifying \( x \) as \( y \) in the following manner:

\[
 p(y|x + \delta, \mathcal{M}) = \mathcal{M}(x)[y] = \sum_{h_i \in \mathcal{H}} f_t(h_i^T x), \quad \mathcal{H} = \{ h_i \}, \tag{5}
\]

where \( \mathcal{H} \) represents the collection of features \( h_i \) learned by the proxy model during its training process. The term \( h_i^T x \) quantifies the relevance of each feature \( h_i \) to the input image \( x \). To accommodate the inherent non-linearity of DNNs, we incorporate a nonlinear transformation denoted by \( f_t(\cdot) \). Moreover, \( f_t \) is monotonic, as a stronger presence of feature \( h_i \) in \( x \) typically amplifies \( h_i \)'s influence on the model’s decision-making process. If \( h_i \) positively correlates with the similarity in the gradient directions \( \nabla_x M(x)[y] \) and \( \nabla_x G(x)[y] \), the target model’s predictions.

In light of the above insight, we define transferability-promoting prior as follows:

**Definition 3.1. (Transferability-promoting Prior)** A prior is considered transferability-enhancing if it directs adversarial examples to solely disrupt cross-model features.

However, due to the black-box nature of the target model and the complexity of DNNs, identifying the cross-model features is indeed intractable, which somewhat undermines the applicability of Definition 3.1. Therefore, we propose an inclusive version of Definition 3.1:

**Definition 3.2. (Transferability-promoting Prior, Relaxed Version)** A prior is considered transferability-enhancing if it encourages adversarial examples to corrupt a broader range of features in a more balanced manner.

In Definition 3.2, the focus shifts from solely targeting cross-model features to disrupting a wider range of features in a balanced manner. Intuitively, this also can mitigate the excessive attention on corrupting model-specific features during the generation of adversarial examples, thereby promoting the transferability of the crafted adversarial examples. Actually, the intuition of disrupting a broader range of features has been explored in previous literature [14] and our contribution lies in formally validating this intuition and providing an explanation within a Bayesian framework. In the remainder of this paper, unless explicitly stated otherwise, we adopt Definition 3.2 by default.
We here present a transferability-promoting prior which we term \textit{ward}. Applying the pixel-level removal prior to Equation 4 intuitively as follows:

1. Adversarial perturbation \( \delta \) is randomly initialized within the range between \(-\epsilon \) and \( \epsilon \).
2. for each iteration \( i = 1 \) to \( T \) do
   3. Sample \( Q \) instances, denoted as \((x+\delta)\odot m_1, \ldots, (x+\delta)\odot m_Q\) from \( p(x+\delta) \).
   4. Compute the optimization objective \( \mathcal{L} = \frac{1}{Q} \sum_{i=1}^{Q} M((x+\delta)\odot m_i)[-y_i] \cdot M((x+\delta)\odot m_i)[-y_i] \).
   5. Calculate the gradient \( \nabla_{\delta} \mathcal{L} \).
   6. Update adversarial perturbation \( \delta = \delta - \alpha \text{sign}(g) \).
   7. Clip adversarial perturbation \( \delta = \max(\min(\delta, \epsilon), -\epsilon) \).
8. end for
9. Return: the crafted adversarial example \( x + \delta \).

### 3.3 Pixel-level Removal Prior

We here present a transferability-promoting prior which we term the pixel-level removal prior. Simply put, the pixel-level removal prior randomly nullifies a pixel value from \( x \). Let \( x \) be of dimension \( K \). Formally, the pixel-level removal prior is expressed as \( p(x) = x \otimes m_l \), where \( m_l \) is a binary vector of the same shape as \( x \). In \( m_l \), all elements are set to 0, except for \( l \)-th element, which is set to 0. The index \( l \) is sampled from a categorical distribution over the set \( \{1, 2, \ldots, K\} \), with each element having an equal probability of \( \frac{1}{K} \) of being selected.

The intuition behind the pixel-level removal prior is straightforward. Applying the pixel-level removal prior to Equation 4 intuitively makes the generated adversarial examples that are not only threatening to the proxy model but also less sensitive to the removal of a single-pixel value as well as the features associated with the pixel. In this way, pixel-level removal prior could potentially address the issue of adversarial examples being overly reliant on exploiting specific features. To substantiate this rationale, we engage in a formal examination (Theorem 3.3).

**THEOREM 3.3.** Pixel-level removal prior is a transferability-promoting prior.

We first restate the minimization of Equation 4 with \( p(x+\delta|M) = 1 \) as follows:

\[
\delta^* = \arg \min_{\delta} p(y|x+\delta,M) = \arg \min_{\delta} \sum_{h_i \in \mathcal{H}} f_i(h_i^T(x+\delta))
\]

\[
s.t., ||\delta||_\infty \leq \epsilon.
\]

Let us now proceed to the proof of Theorem 3.3. When leveraging the pixel-level removal prior as \( p(x) \), the optimization of \( p(y|x,M)p(x) \) translates to:

\[
p(y|x+\delta,M)p(x+\delta) = \mathbb{E}_{p(x+\delta)} \{ p(y|x+\delta,M) \} \\
= \frac{1}{K} \sum_{l=1}^{K} \sum_{h_i \in \mathcal{H}} f_i(h_i^T((x+\delta)\odot m_l)) \\
= \frac{1}{K} \sum_{l=1}^{K} \sum_{h_i \in \mathcal{H}} f_i(h_i^T(x+\delta) - h_i[l] \cdot (x+\delta)[l]) \\
= \frac{1}{K} \sum_{l=1}^{K} \sum_{h_i \in \mathcal{H}} f_i(h_i^T(x+\delta)) \\
= \frac{1}{K} \sum_{h_i \in \mathcal{H}} f_i(h_i^T(x+\delta)) \\
= \frac{1}{K} \sum_{h_i \in \mathcal{H}} f_i(h_i^T(x+\delta)) \sum_{l=1}^{K} (h_i[l] \cdot (x+\delta)[l]) \\
= \frac{1}{K} \sum_{h_i \in \mathcal{H}} f_i(h_i^T(x+\delta)) - \frac{1}{K} f_i(h_i^T(x+\delta)) h_i^T(x+\delta)
\]

\[
s.t., ||\delta||_\infty \leq \epsilon,
\]

where the operator \((x)[l]\) signifies the extraction of the \( l \)-th element from the vector \( x \). The summand regarding \( l \) averages the effect of removing each pixel across all positions indexed by \( l \). The third line to the fourth line in Equation 11 hires Taylor expansion.

The last row of Equation 11 illustrates that the focal point of optimization becomes more uniformly allocated among features, in contrast with Equation 10. More precisely, in Equation 10, if \( x+\delta \) overly concentrates on compromising feature \( h_i \), the gradient magnitude associated with \( h_i \) should be considerable, so as to shift optimization attention towards \( h_i \). When coming to Equation 11, \( f_i(h_i^T(x+\delta)) \) is coupled with a countering force term \( \frac{1}{K} f_i(h_i^T(x+\delta)) h_i^T(x+\delta)^2 \). The magnitude of the countering term is proportionally related to the gradient magnitude associated with \( h_i \), thus ensuring that features that are dominantly corrupted in Equation 10 bear greater regulation in Equation 11. Therefore, the pixel-level deletion prior is qualified as a corrective mechanism, encouraging a more balanced corruption across features, i.e., Theorem 3.3.

### 3.4 Region-based Soft Removal Prior

In Equation 11, we observe that the effectiveness of the countering force term scales linearly with \( K \), the total number of pixels of \( x \). The inherently high number of pixels in typical images may lead to a reduction in the intended countering effect. To mitigate this problem, we introduce the region-based soft removal prior. The core idea of the region-based soft removal prior is straightforward: as shown in Figure 1, it groups multiple contiguous pixels into a region, analogous to treating a group of pixels as a single entity much like the single pixel in the pixel-level removal prior. Doing

\[\text{Based on mathematical principles, as long as } f_i \text{ is monotonic, it can be derived that the latter term always act as a counteracting force to the preceding term, regardless of the sign of } f_i \text{ or } h_i^T(x+\delta).\]
Table 1: The attack success rates (%) on six normally-trained models. The adversarial examples are crafted in Inc-v3, Inc-v4, IncRes-v2, and Res-152, respectively. The best attack results are highlighted in bold. "-" denotes the white-box scenario.

| Proxy   | Attack   | Inc-v3 | Inc-v4 | IncRes-v2 | Res-50 | Res-101 | Res-152 |
|---------|----------|--------|--------|-----------|--------|---------|---------|
| MI      | 50.3     | 46.6   | 45.0   | 40.7      | 39.6   |         |         |
| DI      | 48.1     | 37.4   | 37.7   | 32.0      | 30.2   |         |         |
| PI      | 55.4     | 49.0   | 49.1   | 44.1      | 43.4   |         |         |
| SSA     | 64.3     | 58.2   | 54.9   | 51.4      | 49.1   |         |         |
| PAM     | 74.8     | 73.0   | 67.6   | 68.9      | 62.6   |         |         |
| BSR     | 94.3     | 92.9   | 91.5   | 85.5      | 88.7   |         |         |
| BayAtk  | 97.0     | 95.3   | 95.6   | 89.7      | 92.4   |         |         |
| MI      | 61.6     | 45.7   | 46.7   | 42.4      | 41.0   |         |         |
| DI      | 53.0     | 35.8   | 31.9   | 28.5      | 30.4   |         |         |
| PI      | 59.5     | 45.2   | 49.7   | 41.6      | 43.5   |         |         |
| SSA     | 68.5     | 55.8   | 55.5   | 47.6      | 47.2   |         |         |
| PAM     | 86.5     | 80.0   | 78.4   | 75.4      | 72.8   |         |         |
| BSR     | 93.8     | 91.6   | 97.3   | 87.2      | 87.4   |         |         |
| BayAtk  | 96.3     | 95.0   | 98.6   | 90.4      | 91.9   |         |         |
| MI      | 59.6     | 52.4   | -      | 47.3      | 46.2   | 45.6    |         |
| DI      | 54.7     | 48.3   | -      | 37.9      | 36.5   | 34.0    |         |
| PI      | 61.4     | 57.6   | -      | 49.4      | 46.8   | 46.1    |         |
| SSA     | 75.7     | 67.3   | -      | 58.7      | 58.1   | 54.9    |         |
| PAM     | 88.0     | 84.7   | -      | 75.9      | 80.1   | 70.5    |         |
| BSR     | 91.1     | 91.2   | -      | 90.1      | 88.6   | 86.0    |         |
| BayAtk  | 92.8     | 95.2   | -      | 93.6      | 91.1   | 91.5    |         |
| MI      | 54.1     | 48.5   | 45.2   | 83.7      | 85.5   | -       |         |
| DI      | 56.4     | 50.7   | 46.5   | 81.9      | 85.0   | -       |         |
| PI      | 62.7     | 54.7   | 47.1   | 81.4      | 83.0   | -       |         |
| SSA     | 65.9     | 61.7   | 56.8   | 91.7      | 94.0   | -       |         |
| PAM     | 76.2     | 73.4   | 73.8   | 89.6      | 98.5   | -       |         |
| BSR     | 90.9     | 88.8   | 95.0   | 96.5      | 97.5   | -       |         |
| BayAtk  | 93.4     | 91.8   | 95.8   | 98.8      | 98.4   | -       |         |

To achieve this, it is necessary to incorporate a weighting mechanism during the optimization process that discriminatively adjusts according to the amount of feature information present in different sampled instances. Intuitively, information-rich instances ought to be more distinguishable by the model, typically reflected through higher classification confidence. Motivated by this understanding, we adjust the weighting of the sampled instances in the optimization process in direct proportion to the model’s confidence in accurately classifying them as $y$. In this way, instances contributing valuable features are prioritized, thereby potentially leading to more transferable adversarial examples. Formally, the final loss function of BayAtk is defined as follows:

$$\delta^\ast = \arg\min_\delta p(y|x + \delta, M)p(y|x + \delta, M)\ p(x + \delta|M)$$

$$= M(\langle x + \delta \rangle \odot m_1)[y] \cdot M(\langle x + \delta \rangle \odot m_1)[y],$$

where $\delta$ is the weight considering the feature space which is not involved in the gradient computation during backpropagation and $i$ is random integer between 1 and $d$.

3.6 Targeted Version of BayAtk

Algorithm 1 provides a description of the non-targeted version of BayAtk, whose objective is to cause misclassification without directing the model towards a specific erroneous prediction. In contrast, targeted attacks take this a step further. Their goal is not just to make the model misclassify adversarial examples but to force the model to predict a specific, predetermined incorrect class. Targeted attack is more challenging to execute because they require fine-tuning the adversarial input in such a way that the model consistently arrives at the chosen, incorrect label. To make BayAtk more versatile, we here develop the targeted version of BayAtk.

To extend BayAtk to targeted attack scenario, we need to make some adjustments to Equation 12. The goal of targeted attack is to ensure that the model classifies $x + \delta$ into the desired target label. In other words, we aim to maximize the probability that the model assigns the target label to $x + \delta$. Intuitively, this is equivalent to minimizing the sum of the predicted probabilities for all labels other than the target label. By reducing the likelihood that the adversarial example is classified into any label except the target, we can increase the probability of it being assigned to the target label. Let target label be denoted as $y_t$. The loss function for the target version of BayAtk can thus be formulated as:

$$\delta^\ast = \arg\min_\delta \ p(-y_t|x + \delta, M)p(-y_t|x + \delta, M)\ p(x + \delta|M)$$

$$= M(\langle x + \delta \rangle \odot m_1)[-y_t] \cdot M(\langle x + \delta \rangle \odot m_1)[-y_t],$$

where $\delta$ is the weight considering the feature space which is not involved in the gradient computation during backpropagation and $i$ is random integer between 1 and $d$.

4 Experiment

4.1 Setup

Unless otherwise specified, the settings detailed below adhere to the standard benchmark configuration [21, 28, 36].

3.5 Adaptive Dynamic Weighting Strategy

The instances drawn from $p(x + \delta)$ hold varying amounts of feature information. For example, a complete removal of a region leads to a more substantial loss of feature information as compared to a soft removal. Attacking a less-information instance would yield less effective adversarial examples. Consequently, the information-rich instances should be endowed with more significant attention during the optimization process.

this can significantly decrease the value of $K$, thereby strengthening the influence of the counteracting force term.

Moreover, the region-based soft removal prior does not make the outright elimination of a region; instead, it employs an element-wise scaling operation on all elements within a region. This soft scaling serves as a generalized extension of the removal operation, thus it holds the potential to confer additional advantages (See Section 4.5 for validation).

In practice, we partition $x$ into a grid of non-overlapping regions, i.e., chessboard shape. Then, through random selection, we choose one region to undergo the element-wise scaling operation. To formalize, the partitioned $x$ along with $m$ are denoted as $x = [B_1, \ldots, B_d]$ and $m = [m_1, \ldots, m_d]$, respectively. Each $m_i$ starts with a value of 1, and one randomly picked $m_i$ is filled with a random scalar between 0 and 1.

Notice that the above process amounts to sampling from a joint probability density function: one sample from $[B_1, \ldots, B_d]$ and another from a uniform distribution over $[0, 1]$. These sampling procedures are independent, ensuring that the region-based soft removal prior adheres to probabilistic principles.
Models. We employ six normally-trained models to serve as our proxy and target models, including Inception-v3 (Inc-v3), Inception-v4 (Inc-v4), Inception-Resnet-v2 (IncRes-v2), Resnet-v2-50 (Res-50), Resnet-v2-101 (Res-101), and Resnet-v2-152 (Res-152). With the emergence of vision models based on transformer architecture, we include four transformer-based visual models, ViT, PiT, Visformer, and Swin, to better evaluate the effectiveness of BayAtk. Additionally, to thoroughly evaluate the effectiveness of BayAtk, we also include nine defense models, namely Inc-v3\textsubscript{ens}, Inc-v3\textsubscript{ens4}, IncRes-v2\textsubscript{ens}, HGD [17], R&P [33], NIPS-v3\textsuperscript{1}, JPEG [13], RS [4], and NRP [23]. These defenses cover the previously mentioned two mainstream adversarial defense directions, with the first three using adversarial training and the latter six utilizing input transformation techniques. Normally-trained models serve as the proxy model throughout the experiment, while normal and defense models are the target models in Section 4.2 and Section 4.3, respectively.

\textsuperscript{1}https://github.com/anlthms/nips-2017/tree/master/mmd

Table 3: The targeted attack success rates (%) with the default attack setting.

| Proxy  | Attack | Inc-v3 | Inc-v4 | IncRes-v2 | Res-50 | Res-101 | Res-152 |
|--------|--------|--------|--------|-----------|--------|---------|---------|
| Inc-v3 | PAM    | -      | 2.2    | -         | 1.0    | 1.2     | 1.5     |
|        | BSR    | -      | 3.0    | 2.1       | 2.6    | 2.3     | 3.3     |
|        | BayAtk | -      | -      | 4.6       | 4.1    | 4.5     | 4.8     |
| Inc-v4 | PAM    | -      | 2.3    | 0.9       | 1.2    | 1.2     | 3.4     |
|        | BSR    | -      | 3.1    | 2.3       | 2.7    | 2.1     | 3.1     |
|        | BayAtk | -      | -      | 4.9       | -      | 4.4     | 4.6     |
|        | -      | -      | -      | 4.5       | 4.5    | -       | -       |

Table 4: The targeted attack success rates (%). The adversarial examples are crafted in an ensemble of white-box proxy models with iteration number of 500 and logit loss function.

| Attack   | Inc-v3 | Inc-v4 | IncRes-v2 | ViT-B | PiT-B |
|----------|--------|--------|-----------|-------|-------|
| PAM + Logit | 17.2   | 13.4   | 10.7      | 5.4   | 6.3   |
| BSR + Logit | 22.2   | 18.2   | 12.9      | 6.6   | 7.6   |
| BayAtk + Logit | 29.8   | 27.1   | 22.0      | 14.6  | 15.6  |

4.2 Attack Performance on Normal Models

Convolution-based models. We first evaluate the performance of BayAtk on normally-trained models. To do this, we generate adversarial examples using four proxy models and subsequently evaluate their effectiveness across a variety of target models. Table 1 reports the performance of different attacks over diverse proxy-target model pairs. Overall, regardless of the specific proxy and target model pairing, BayAtk consistently boosts the transferability of adversarial examples across nearly all proxy models by a clear margin. This consistency suggests that the effectiveness of BayAtk is not dependent on particular proxy models being used. More impressively, the average ASR of BayAtk across all tested models stands at approximately 93%, underscoring BayAtk’s effectiveness in generating transferable adversarial examples. This performance is particularly striking when we consider the varying degrees of success across different attack types, such as MI, DI, PI, SSA, PAM, and BSR. Diving deeper into the individual results, when employing Inception-v3 as the proxy model, the ASR of BayAtk surpasses that of the state-of-the-art attack BSR by approximately 3% – 4%. These attack results demonstrate BayAtk’s superior ability in generating transferable adversarial examples.

Transformer-based models. Table 2 presents the ASRs achieved by different attack methods on four vision transformer models. The
Table 5: The attack success rates (%) on three models trained in CIFAR-10. We use Res-50 trained in CIFAR-10 as the proxy model.

| Attack | Inc-v3 | Inc-v4 | Res-101 |
|--------|--------|--------|---------|
| PAM    | 82.89  | 79.15  | 85.68   |
| BSR    | 85.64  | 82.30  | 88.26   |
| BayAtk | 89.32  | 86.32  | 92.82   |

Table 6: The attack success rates (%) on three adversarially-trained models. The adversarial examples are crafted in Inc-v3, Inc-v4, IncRes-v2, and Res-152, respectively.

| Proxy     | Attack | Inc-v3 | Inc-v4 | IncRes-v2 | Res-152 |
|-----------|--------|--------|--------|-----------|---------|
| Inc-v3    | Admix  | 35.4   | 34.0   | 25.9      |         |
|           | SSA    | 38.3   | 37.9   | 27.1      |         |
|           | PAM    | 39.0   | 38.8   | 28.0      |         |
|           | BayAtk | 58.6   | 52.2   | 32.5      |         |
| Inc-v4    | Admix  | 45.2   | 44.3   | 30.1      |         |
|           | SSA    | 47.7   | 45.9   | 31.7      |         |
|           | PAM    | 55.4   | 50.5   | 33.2      |         |
|           | BSR    | 51.0   | 46.6   | 29.1      |         |
|           | BayAtk | 60.5   | 56.8   | 37.8      |         |
| IncRes-v2 | Admix | 52.6   | 49.9   | 49.5      |         |
|           | SSA    | 55.3   | 52.0   | 50.2      |         |
|           | PAM    | 66.0   | 58.3   | 51.0      |         |
|           | BSR    | 71.4   | 63.1   | 51.0      |         |
|           | BayAtk | 75.4   | 66.7   | 52.6      |         |
| Res-152   | Admix  | 44.2   | 34.7   | 27.7      |         |
|           | SSA    | 45.4   | 37.0   | 31.7      |         |
|           | PAM    | 51.2   | 46.3   | 35.2      |         |
|           | BSR    | 78.7   | 74.7   | 51.6      |         |
|           | BayAtk | 82.3   | 77.8   | 55.1      |         |

Table 7: The attack success rates (%) on other defenses. The adversarial examples are crafted in Inc-v3, Inc-v4, IncRes-v2, and Res-152, respectively.

| Proxy      | Attack | HGD | R&P | NIPS-r3 | JPEG | RS | NR |
|------------|--------|-----|-----|--------|------|----|----|
| Inc-v3     | Admix  | 30.2| 28.0| 30.3   | 40.2 | 33.2| 21.9|
|            | SSA    | 37.1| 33.3| 36.7   | 41.1 | 33.8| 26.7|
|            | PAM    | 38.7| 34.5| 38.3   | 43.0 | 35.0| 27.5|
|            | BSR    | 49.2| 43.3| 48.6   | 53.0 | 44.7| 35.5|
|            | BayAtk | 53.6| 49.9| 53.1   | 58.4 | 51.4| 40.8|
| Inc-v4     | Admix  | 53.2| 47.8| 51.0   | 59.4 | 48.2| 34.4|
|            | SSA    | 54.8| 51.8| 55.2   | 60.1 | 52.4| 39.0|
|            | PAM    | 60.4| 57.6| 61.7   | 67.6 | 58.5| 43.4|
|            | BSR    | 66.4| 62.3| 66.6   | 73.3 | 63.1| 46.9|
|            | BayAtk | 71.4| 66.3| 73.3   | 77.0 | 66.4| 52.5|
| IncRes-v2  | Admix  | 86.7| 84.6| 85.2   | 89.8 | 77.6| 64.5|
|            | PAM    | 91.4| 88.9| 91.4   | 96.0 | 82.0| 67.5|
|            | BSR    | 93.5| 93.7| 95.9   | 97.2 | 85.8| 70.8|
|            | BayAtk | 71.4| 66.3| 70.2   | 75.7 | 65.0| 57.0|
| Res-152    | Admix  | 62.4| 56.9| 60.5   | 66.4 | 56.6| 47.8|
|            | PAM    | 68.3| 61.9| 67.8   | 71.8 | 61.4| 51.9|
|            | BSR    | 71.8| 66.3| 70.2   | 75.7 | 65.0| 57.0|
|            | BayAtk | 71.8| 66.3| 70.2   | 75.7 | 65.0| 57.0|

Table 8: The oddness scores of BIM and BayAtk. The results are averaged over 100 samples.

| Proxy      | Attack | Res-50 | Res-152 | Inc-v3 |
|------------|--------|--------|---------|--------|
| BIM        | 0.615  | 0.584  | 0.560   |
| BayAtk     | 0.282  | 0.287  | 0.322   |

attack results in Table 2 demonstrate the effectiveness of BayAtk across different vision transformer models. In particular, when using ResNet-152 as the proxy model, BayAtk leads with ASRs of 61.7% for ViT-B, 65.7% for PiT-B, 78.8% for Visformer-S, and 78.3% for Swin. In contrast, BSR, which performs relatively well, still lags behind BayAtk by substantial margins, particularly in the case of ViT-B and Visformer-S, where the differences are most pronounced.

**Targeted attack results.** We also evaluate the targeted attack performance of BayAtk in comparison to two-state-of-the-art transfer-based attack methods, PAM and BSR. The evaluation setup follows [38], utilizing the same target setting for consistency. Table 3 presents targeted ASRs of three attack methods. The results indicate that BayAtk consistently surpasses PAM and BSR across various proxy-target model pairs. For instance, in Inc-v3, BayAtk achieves the highest ASR of 4.6% for Inc-v4 and 4.5% for both Res-50 and Res-152, surpassing the results of PAM and BSR. Similarly, in the Inc-v4, BayAtk records an ASR of 4.9%, which is higher than the other methods. In Res-152, the superiority of BayAtk is even more pronounced, with ASRs reaching 15.7% compared to 8.0% for PAM and 12.4% for BSR against Inception-v3. This trend highlights the effectiveness of BayAtk in executing targeted attacks.

Moreover, we see that the targeted ASRs reported in Table 3 are somewhat low. This is primarily because targeted attacks require more attack iterations. To this end, following [38], we modify the loss function to logit and increase the number of attack iterations to 500. We then employ an ensemble of white-box proxy models, including ResNet-50, DenseNet121, and VGG16, for re-evaluation. Table 4 reports the targeted ASRs for various attack methods. The results indicate a significant improvement in the targeted ASRs across different methods, with BayAtk generating more transferable adversarial examples.

The Evaluation on CIFAR-10. We further evaluate the effectiveness of BayAtk on CIFAR-10 dataset. We train Res-50, Res-101, Inc-v3, and Inc-v4 in the training set of CIFAR-10. ResNet-50 is then used as the proxy model to generate 10000 adversarial examples for the test set of CIFAR-10. The ASRs of three attack methods are summarized in Table 5. We find that BayAtk consistently outperforms both PAM and BSR on CIFAR-10 by a significant margin, demonstrating the generalizability of BayAtk across different datasets.
### Table 9: Ablation study on different sub-components of BayAtk. We report ASRs (%).

| Strategy                          | Inception-v4 | Resnet-50 | Resnet-101 |
|-----------------------------------|--------------|-----------|------------|
| BIM                               | 21.7         | 15.2      | 13.4       |
| + Pixel-level Removal             | 54.5         | 44.8      | 42.2       |
| + Region-based Hard Removal       | 90.7         | 88.2      | 83.5       |
| + Region-based Soft Removal       | 95.8         | 94.4      | 88.2       |
| + Region-based Soft Removal & Adaptive Dynamic Weight Strategy | 97.0 | 95.6 | 89.7 |

### Table 10: The ASRs (%) of BayAtk with varying chessboard layouts.

| Chessboard Layout | Inception-v4 | Resnet-50 | Resnet-101 |
|-------------------|--------------|-----------|------------|
| 7x7               | 95.8         | 94.2      | 87.7       |
| 14x14             | 97.0         | 95.6      | 89.7       |
| 28x28             | 96.5         | 94.8      | 88.5       |
| 56x56             | 94.6         | 92.9      | 86.3       |

### 4.3 Attack Performance on Defense Models

Although many attacks are capable of misleading normally trained models, these attacks might not be as effective against models with defense mechanisms in place. To further substantiate the superiority of BayAtk, we examine the effectiveness of BayAtk against defense models. Table 6 and Table 7 report the results of different attacks in two mainstream defense paradigms, respectively.

Adversarial training, deemed a gold standard in defense, iteratively exposes models to adversarial examples during the training phase, allowing them to be resilient to similar attacks in the inference stage. We assess the attack effectiveness of BayAtk against three top-tier adversarially-trained models. Looking at the results, we see a consistent improvement by BayAtk over baselines. Specifically, when employing Inc-v3 as the proxy model, BayAtk yields ASRs of 55.6%, 52.2%, and 32.5% against the three adversarially-trained models, respectively. These results represent a substantial improvement over the best-performing baseline, BSR, which records ASRs of 51.0%, 46.6%, and 29.1%. BayAtk exceeds BSR by margins of 4.6%, 5.6%, and 3.4%. Furthermore, when evaluating the performance of BayAtk with Inception-v4, IncRes-v2, and Resnet-152 as the proxy models, we observe similar trends. In short, BayAtk is effective for enhancing the cross-model transferability of crafted adversarial examples, in spite of normal or adversarially-trained models.

Input transformation-based defenses, which differ fundamentally from adversarial training, tackle adversarial attacks by applying preprocessing techniques to model inputs with the aim of diminishing the impact of adversarial examples by transforming them back into their benign counterparts. Overall, as outlined in Table 6, BayAtk maintains its superiority when confronted with six state-of-the-art input transformation-based defenses. While these defenses can strip certain adversarial features, BayAtk appears to have a higher capacity to generate adversarial examples that retain their effectiveness after transformation, as evidenced by significantly higher ASRs. For example, BayAtk’s ASRs are significantly higher (58.4% to 77.0%) compared to BSR (53.0% to 73.3%) against JPEG compression defense when leveraging Inception-v3 and Inception-v4 as the proxy models. When we shift our focus to IncRes-v2 as the proxy model, BayAtk achieves the highest ASRs of 93.5%, 93.7%, 95.9%, 97.2%, 85.8%, and 70.8%, significantly surpassing BSR’s performance, which records ASRs of 91.4%, 88.9%, 91.4%, 96.0%, 82.0%, and 67.5%. The improvements of 2.1% to 4.3% across these defenses further illustrate the exceptional capability of BayAtk in generating resilient adversarial examples.

### 4.4 Oddness Evaluation

We here validate that BayAtk can corrupt a wider range of features in a more balanced manner. We employ BIM and BayAtk to generate adversarial examples and then compute the differences between these examples and the original images. The differences are partitioned into $14 \times 14$ regions, and we calculate the sum of values for each region. The standard deviation of these sums across different regions
is defined as the oddness score. Intuitively, a lower oddness score suggests that features contained in input samples are disrupted in a more uniform manner. Table 8 reports the oddness scores achieved by different methods, averaged over 100 samples. As observed, BayAtk attains a lower oddness score, demonstrating its superior capability to disrupt features in images more uniformly.

4.5 Sub-component Analysis and Sensitivity Analysis

In this subsection, we investigate the contributions of different sub-components of BayAtk to its performance, alongside examining the influence of chessboard size on its attack effectiveness.

Sub-component analysis. Our exploration initiates with BIM, which acts as the foundation attack, i.e., the vanilla transfer-based attack, upon which BayAtk and other transfer-based attacks are built. Table 9 reports the attack performance of BIM combined with various sub-components. The effectiveness of BIM alone is observed to be relatively low, with ASRs spanning from 13.4% to 21.7%. This initial performance sets the baseline from which the impact of different sub-components can be quantitatively measured.

Upon integrating pixel-level removal prior to BIM, a significant surge in attack performance is noted, with ASRs jumping to 54.5%, 44.8%, and 42.2%. This substantial increase validates the effectiveness of pixel-level removal prior as a transferability-promoting prior. The next evolutionary step replaces the pixel-level removal prior with the region-based hard removal prior. The performance leap in this step is even more pronounced, propelling ASRs to impressive numbers of 90.7%, 88.2%, and 83.5%. As discussed in Section 3.4, we attribute this significant improvement to its focus on coherent regions rather than isolated pixels, which is key to decreasing the value of $K$ and enhancing the influence of the counteracting force term. Switching from hard to soft removal yields a further increment in ASRs to 95.8%, 94.4%, and 88.2%, highlighting the benefits of a gentler, perhaps more probabilistic method to region removal. We speculate that the superior performance of soft over hard removal may stem from its ability to more intricately scale the feature information within images instead of outright elimination.

The impact of chessboard layout. Table 10 reports the performance of BayAtk with varying chessboard sizes and a $14 \times 14$ chessboard achieves the highest attack effectiveness. Intuitively, chessboard size that is either excessively large or small detracts from the effectiveness of BayAtk. On the one hand, enlarging chessboard layout, for instance, from $14 \times 14$ to $56 \times 56$, results in each grid representing a smaller subset of pixels. This change suggests a movement away from effectively leveraging the region-based prior towards an emphasis on the pixel-level prior, which, in turn, leads to a degradation in performance of BayAtk. On the other hand, a smaller chessboard implies that an overly large part of $x$ is modified, leading to the elimination of significant portions of image features. Hence, it appears that the $14 \times 14$ chessboard size likely offers an optimal balance.

4.6 Towards Efficient BayAtk

We focus here on the practical costs associated with BayAtk. Generally, when conducting large-scale attacks in real-world scenarios, it is crucial to consider the runtime costs of attack methods. The primary factors influencing the runtime of BayAtk include the size of the proxy model and the number of attack iterations. We demonstrate that using smaller models and fewer attack iterations allows effectively reducing the overall overhead of BayAtk while maintaining competitive attack effectiveness.

Figure 4 illustrates the ASRs and time costs of BayAtk when using Res-18 as the proxy model across different attack iterations. Here, we adjust the step size to ensure that the product of the step size and the number of attack iterations equals $\frac{16}{255}$. Besides, we employ a batch size of 64 and use Inc-v4 as the target model. We utilize a GTX 4090 GPU and report the average values over 10 trials. We observe that when the number of attack iterations reaches 5, the ASRs stabilize around 85%, with a runtime of approximately 4.9 seconds. Therefore, in practice, we can set the number of attack iterations to 5 to reduce the attack costs.

Additionally, when using Res-152 with 10 iterations, the time cost is about 98.8 seconds. As shown in Table 1, although replacing Res-18 with Res-152 and using an attack iteration of 5 results in a decrease in the ASR by about 7 percentage points, the attack cost is reduced to one-fortieth of the original. Furthermore, by employing mixed precision, we find that the runtime can be further reduced from 4.9 seconds to approximately 2.5 seconds, resulting in a cost that is one-eighth of the original. We believe that generating adversarial examples for 64 samples in just 2.5 seconds is highly efficient in practice.

5 Real-world Evaluation

To conduct a more dependable and thorough evaluation of BayAtk, we extend our evaluation to include attacks against Google MLaaS Classification System\(^4\) on a multi-modal large-scale model Claude\(^5\) developed by AWS. The Google MLaaS Classification System is recognized as one of the most advanced AI platforms available. Claude ranks among the top-tier multi-modal models, according to

\(^4\)https://cloud.google.com/vision/docs/drag-and-drop
\(^5\)https://www.anthropic.com/claud
Atk yields 25, 26, and 25 entities in the Top-5 predictions. These We record the Top-5 predictions for the generated adversarial exam-

Vision System recognizes the background colors of three images, this within given images.

The evaluation protocol is to determine whether the entities in the samples are included in the Top-5 predictions. In contrast, BayAtk can indeed effectively identify and exploit model-cross features to highlight that the training dataset, model architecture, and task and craft adversarial examples using BSR and BayAtk against Google Classification System are 43.33% and 74.67%, respectively, results indicate that the average ASRs for BSR and BayAtk the samples are included in the Top-5 predictions. In contrast, BayAtk demonstrates the superior performance of Google Classification System are 43.33% and 74.67%, respectively, indicating ASRs of 36%, 32%, and 40%, respectively. For BayAtk, the volunteers find that 25, 29, and 18 of the samples, respectively, are semantically consistent, resulting in ASRs of 75%, 71%, and 82% for BayAtk. These ASRs further underscore the vulnerabilities of DNNs.

To provide a tangible understanding, Figure 5 presents three illustrative examples. Figure 5 is divided into two columns: the left column displays the adversarial examples, while the right column showcases Claude3’s corresponding responses. Due to space constraints, only the initial summary paragraph of Claude3’s responses is presented. Specifically, the top image in Figure 5 shows an image of a butterfly, but Claude3 fails to recognize it. More impressively, in the bottom images in Figure 5, Claude3 misidentifies an insect and a hot-air balloon as an electronic circuit board and a hand, respectively. These misclassifications highlight the significant challenges DNNs face when confronted with adversarial examples, emphasizing the need for improved robustness in AI systems.

Figure 5: The responses of Claude3 to the adversarial examples generated by BayAtk are presented here. We prompted Claude3 with "Describe this image.".

5.2 Claude3

We feed adversarial examples produced by BSR and BayAtk into Claude3, prompting it with "Describe this image". Following this, we collect Claude3’s responses and ask the same volunteers to assess the consistency between Claude3’s responses and the semantic content of the generated adversarial examples, responding with either a yes or a no. For BSR, the volunteers identify that 64, 68, and 60 of the samples, respectively, are semantically consistent, resulting in ASRs of 36%, 32%, and 40%, respectively. For BayAtk, the volunteers find that 25, 29, and 18 of the samples, respectively, are semantically consistent, indicating ASRs of 75%, 71%, and 82% for BayAtk. These ASRs further underscore the vulnerabilities of DNNs.

To provide a tangible understanding, Figure 5 presents three illustrative examples. Figure 5 is divided into two columns: the left column displays the adversarial examples, while the right column showcases Claude3’s corresponding responses. Due to space constraints, only the initial summary paragraph of Claude3’s responses is presented. Specifically, the top image in Figure 5 shows an image of a butterfly, but Claude3 fails to recognize it. More impressively, in the bottom images in Figure 5, Claude3 misidentifies an insect and a hot-air balloon as an electronic circuit board and a hand, respectively. These misclassifications highlight the significant challenges DNNs face when confronted with adversarial examples, emphasizing the need for improved robustness in AI systems.

6 Conclusion

In this paper, we studied the transferability of adversarial examples through the lens of Bayesian approach. We answered how to combine the transferability of adversarial examples with Bayesian approach and identified what constitutes a transferability-promoting prior, so as to gain deeper insights into the underlying principles that govern the transferability. Building upon this foundation, we designed two specific transferability-promoting priors, namely, pixel-level removal prior and region-based soft removal prior, accompanied by an adaptive dynamic weighting strategy. We conducted extensive experiments to examine the performance of the proposed attack. We also tested BayAtk on a large-scale model deployed in a real-world scenario to further validate its effectiveness. Looking forward, we hope that this study can alarm relevant parties about the potential vulnerabilities of DNNs and spur further development of sophisticated adversarial defense mechanisms.

7 Acknowledgments

This work was supported by the National Natural Science Foundation of China under grant number 62202170 and Alibaba Group through the Alibaba Innovation Research Program.

6https://www.anthropic.com/news/claudie-3-family

7During the assessment, the volunteers primarily focus on whether the entities in the images matched the given Top-5 predictions, rather than the background colors. As a result, all three samples are marked as successful attacks.
Transferable Adversarial Examples with Bayesian Approach

References

[1] Ahmed Ali Mohammed Al-Saffar, Hai Tao, and Mohammed Ahmed Talab. 2017. Review of deep convolution neural network in image classification. In 2017 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET). IEEE, 26–31.

[2] Zeynep Batmaz, Ali Yurekli, Alper Bilge, and Cihan Kalez. 2019. A review on deep learning for recommender systems: challenges and remedies. Artificial Intelligence Review 52, 1 (2019), 1–37.

[3] William M Bolstad and James M Curran. 2016. Introduction to Bayesian statistics. John Wiley & Sons.

[4] Jeremy Cohen, Elian Rosenfeld, and Zico Kolter. 2019. Certified adversarial robustness via randomized smoothing. In International conference on machine learning. PMLR, 1310–1320.

[5] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. 2019. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In 28th USENIX security symposium (USENIX security 19). 321–338.

[6] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. 2018. Boosting adversarial attacks with momentum. In Proceedings of the IEEE conference on computer vision and pattern recognition. 9185–9193.

[7] Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. 2019. Evading defenses to transferable adversarial examples by translation-invariant attacks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 4312–4321.

[8] Mingyu Fan, Cen Chen, Chengyu Wang, and Jun Huang. 2023. On the trustworthiness of AdaBoost. In The Thirty-eighth Annual Conference on Neural Information Processing Systems.

[9] Mingyu Fan, Cen Chen, Wenzhong Guo, Zuobin Ying, and Ximeng Liu. 2023. Enhance transferability of adversarial examples with model architecture. In ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 1–5.

[10] Mingyu Fan, Xiaodan Li, Cen Chen, Wenhong Zhou, and Yaliang Li. (n. d.). Transferability Bound Theory: Exploring Relationship between Adversarial Transferability and Flattness. In The Thirty-eighth Annual Conference on Neural Information Processing Systems.

[11] Lijian Gao, Yaya Cheng, Qilong Zhang, Xing Xu, and Jingkuan Song. 2021. Feature space targeted attacks by statistic alignment. arXiv preprint arXiv:2105.11645 (2021).

[12] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014).

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

[14] Yang Hu, Ning Wang, Yimin Chen, Wenjing Lou, and Y Thomas Hou. 2022. Transferability of adversarial examples in machine learning-based malware detection. In 2022 IEEE Conference on Communications and Network Security (CNS). IEEE, 28–36.

[15] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. 2016. Adversarial Examples in The Physical World. arXiv preprint arXiv:1607.02533 (2016).

[16] Yingwei Li, Song Bai, Yuin Zhou, Chiang Xie, Zhishuai Zhang, and Alan Yuille. 2020. Learning transferable adversarial examples via ghost networks. In Proceedings of the AAAI conference on artificial intelligence, Vol. 34. 11458–11465.

[17] Fangzhou Liao, Ming Liang, Yinpeng Dong, Xiaolin Hu, and Jun Zhu. 2018. Defense against adversarial attacks using high-level representation guided denoiser. In Proceedings of the IEEE conference on computer vision and pattern recognition. 1778–1787.

[18] Jiadong Lin, Chaobao Song, Kun He, Liwei Wang, and John E Hopcroft. 2019. Nesterov accelerated gradient and scale invariance for adversarial attacks. arXiv preprint arXiv:1908.05281 (2019).

[19] Jiadong Lin, Chaobao Song, Kun He, Liwei Wang, and John E Hopcroft. 2020. Nesterov Accelerated Gradient and Scale Invariance for Adversarial Attacks. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net. https://openreview.net/forum?id=SJHkwBYDH

[20] Yanpei Liu, Xinyu Chen, Chang Liu, and Dawn Song. 2016. Delving into transferable adversarial examples and black-box attacks. arXiv preprint arXiv:1611.02770 (2016).

[21] Yuyang Long, Qilong Zhang, Boheong Zeng, Lianli Gao, Xionglong Liu, Jian Zhang, and Jingkuan Song. 2022. Frequency domain model augmentation for adversarial attack. In European conference on computer vision. Springer, 549–566.

[22] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017).

[23] Muzammal Naseer, Salman Khan, Munawar Hayat, Fohad Shabbaz Khan, and Fatih Porikli. 2020. A self-supervised approach for adversarial robustness. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 262–271.

[24] Alex Serban, Eick Poll, and Joost Visser. 2020. Adversarial examples on object recognition: A comprehensive survey. ACM Computng Surveys (CSUR) 53, 3 (2020), 1–38.

[25] Satya Narayan Shukla, Anit K Jam. 2020. Adversarial attacks on object recognition: A comprehensive survey. ACM Computng Surveys (CSUR) 53, 3 (2020), 1–38.

[26] Satya Narayan Shukla, Anit K Jam. 2020. Adversarial attacks on object recognition: A comprehensive survey. ACM Computng Surveys (CSUR) 53, 3 (2020), 1–38.

[27] Ali Ismail, Alexandros Laskaridis, and Patrick McDaniel. 2017. Ensemble adversarial training: Attacks and defenses. arXiv preprint arXiv:1705.07204 (2017).

[28] Kunyu Wang, Xuanran He, Wenxuan Wang, and Xiaosen Wang. 2023. Boosting adversarial transferability by block shuffle and rotation. arXiv preprint arXiv:2308.10299 (2023).

[29] Xiaosen Wang and Kun He. 2021. Enhancing the transferability of adversarial attacks through variance tuning. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 1924–1933.

[30] Xiaosen Wang, Xuanran He, Jingdong Wang, and Kun He. 2021. AdmitX: Enhancing the transferability of adversarial attacks. In Proceedings of the IEEE/CVF International Conference on Computer Vision. 16158–16167.

[31] Xiaoming Wang, Qunming Yao, James T Kwok, and Lionel M Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. ACM Computing Surveys (CSUR) 53, 3 (2020), 1–34.

[32] Peter M Williams. 1995. Bayesian regularization and pruning using a Laplace prior. Neural computation 7, 1 (1995), 117–143.

[33] Chiang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. 2017. Mitigating adversarial effects through randomization. arXiv preprint arXiv:1711.01991 (2017).

[34] Chiang Xie, Zhishuai Zhang, Yuin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L Yuille. 2019. Improving transferability of adversarial examples with input diversity. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2730–2739.

[35] Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K Jam. 2020. Adversarial attacks and defenses in images, graphs and text: A review. International Journal of Automation and Computing 17, 2 (2020), 151–178.

[36] Jianping Zhang, Jen-tse Huang, Wenxuan Wang, Yichen Li, Weibin Wu, Xiaosen Wang, Yuxin Su, and Michael R Lyu. 2023. Improving the transferability of adversarial samples by path-augmented method. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 8173–8182.

[37] Jianping Zhang, Jen-tse Huang, Wenxuan Wang, Yichen Li, Weibin Wu, Xiaosen Wang, Yuxin Su, and Michael R Lyu. 2023. Improving the transferability of adversarial samples by path-augmented method. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 8173–8182.

[38] Zhengyu Zhai, Zhuoran Liu, and Martha Larson. 2021. On success and simplicity: A second look at transferable targeted attacks. Advances in Neural Information Processing Systems 34 (2021), 6115–6128.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009