# PatchAttack: A Black-box Texture-based Attack with Reinforcement Learning

Chenglin Yang, Adam Kortylewski, Cihang Xie, Yinzhi Cao, and Alan Yuille

Johns Hopkins University
{chenglin.yangw,cihangxie306,alan.l.yuille}@gmail.com
{akortyl1,yinzhi.cao}@jhu.edu

**Abstract.** Patch-based attacks introduce a perceptible but localized change to the input that induces misclassification. A limitation of current patch-based black-box attacks is that they perform poorly for targeted attacks, and even for the less challenging non-targeted scenarios, they require a large number of queries. Our proposed *PatchAttack* is query efficient and can break models for both targeted and non-targeted attacks. PatchAttack induces misclassifications by superimposing small textured patches on the input image. We parametrize the appearance of these patches by a dictionary of class-specific textures. This texture dictionary is learned by clustering Gram matrices of feature activations from a VGG backbone. *PatchAttack* optimizes the position and texture parameters of each patch using reinforcement learning. Our experiments show that PatchAttack achieves $> 99\%$ success rate on ImageNet for a wide range of architectures, while only manipulating 3% of the image for non-targeted attacks and 10% on average for targeted attacks. Furthermore, we show that PatchAttack circumvents state-of-the-art adversarial defense methods successfully. The code is publicly available here.

**Keywords:** Adversarial Machine Learning; Black-box Attack

## 1 Introduction

Computer vision models have achieved strong performance on image recognition tasks, however, they are known to be vulnerable against adversarial examples [49]. Adversarial examples are modifications of images crafted to induce misclassification. Understanding the vulnerability of computer vision models to adversarial attacks has emerged as an important research area, providing opportunities for understanding and improving computer vision models.

Recent works have introduced very successful attacks in the white-box setting [20,34,9], where both the network architecture and parameters are available to the attacker. In real-world applications, a more common attacking scenario is that the attacker only has access to the model's input and the predicted output, *e.g.* , attacking popular image analysis APIs [2,1,25,26,22,21] or self-driving cars [6,39,50,12,40,48,15,28]. This *black-box* scenario is challenging because adversarial modification of the input must be computed without access to the loss gradient of the model.

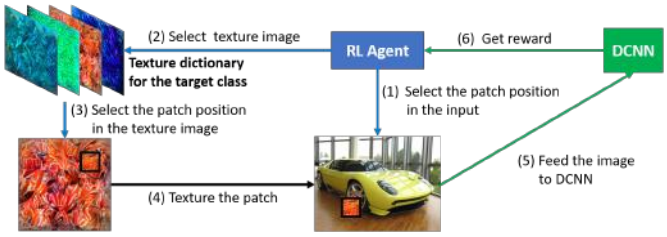


**Fig. 1.** Illustration of our black-box texture-based patch attack via reinforcement learning (RL): (1) The RL agent selects the patch position in the input image and (2) selects a texture image for the target category (lionfish in this example) from our learned texture dictionary. (3) The agent selects a patch position in the texture image to extract a texture patch, (4) which is then superimposed on the input image. (5) The adversarial image is fed into the deep convolutional network (DCNN). (6) The output scores of the DCNN is used to calculate the reward for the optimization of the agent. This six step process is repeated until the DCNN is attacked successfully.

Two paradigms have emerged for black-box attacks. *Perturbation-based* methods introduce imperceptible changes to the image that are constrained to have a small norm but are typically applied to the whole input image [37,38,10,51,5,4]. Recently, several defense methods have shown that perturbation-based attacks can be successfully defended [20,31,34,53].

In this paper, we study a complementary type of adversary, *Patch-based black-box attacks*, introducing a perceptible (large norm) but localized change to the input. In their pioneering work, Fawzi et al. [16] show that superimposing monochrome black patches onto images generated by random search can successfully induce misclassificataions. However, a major limitation of current patch-based black box attacks is that they perform poorly on targeted attacks and require large amounts of queries for non-targeted attacks (Experiments 4.2).

In this work, we introduce *PatchAttack*, a patch-based black-box attack that is query efficient and achieves very high success rates for both targeted and non-targeted attacks. Our main contributions are two fold: 1) We formulate the search over the position and shape of adversarial patches as a reinforcement learning problem. Hence, we define the attack as a decision-making process of an agent that interacts with its' environment (the model) by taking actions (placing patches in the image) and observing rewards (misclassification rates). In this way, the parameter search is formulated as an optimization problem that is much more effective compared to random search strategies in terms of query efficiency. 2) Our experiments show that attacks with monochrome patches do not succeed as targeted attacks. The intuition is that monochrome patches can remove information from the image, but do not add any information, which is critical to confuse a model in a targeted manner. We overcome this limitation by introducing texture to the adversarial patches. To parameterize the texture efficiently, we learn a dictionary of class-specific textures by clustering Gram matrices of feature activations from a VGG backbone. The texture dictionary

enables a query-efficient search over the patch appearance and leads to very high success rates at targeted and non-targeted attacks, while also strongly reducing the image area that needs to be corrupted for a successful attack.

Our results on ImageNet [42] show that PatchAttack achieves considerably higher success rates for targeted and non-targeted attacks compared to related work, while being more efficient in terms of the number of queries and the size of the attacked image area (on average only 3% of the image needs to be modified for non-targeted and $< 10\%$ for targeted attacks respectively). Furthermore, we show that PatchAttack can successfully overcome Feature Denoising [53], a state-of-the-art defense for perturbation based attacks. Finally, we perform experiments with shape-based DCNNs [19] which were designed to overcome the texture bias of DCNNs trained on ImageNet, and hence should be more robust to PatchAttack. Interestingly, we cannot observe any increased robustness of shape-based DCNNs, although PatchAttack is texture-based and the object shape is largely preserved in the adversarial images.

## 2 Related Work

Sparked by the seminal works of Szegedy et al. [49] adversarial machine learning has emerged as an important research area for understanding and improving deep neural networks. In recent years, two complementary paradigms have emerged for black-box attacks, perturbation-based and patch-based attacks. In this paper, we focus on patch-based black-box attacks, but we also provide a short review of perturbation-based black-box attacks as the search strategies for both attack types are related.

**Perturbation-based black box attacks.** While we focus on black-box attacks with access to model output scores, attacks with even more limited access to model decisions only have been explored [7,29,11]. Such approaches often require lots of queries and are therefore difficult to apply in real-world applications. Early work on perturbation-based black box attacks with access to prediction scores proposed to estimate model gradients with finite differences [5,29,52,51,30,10]. In particular, these iterative attacks estimate gradients via sampling from a noise distribution around the feature point. While this approach is successful it requires large amounts of model queries. Other approaches use evolutionary algorithms [3,30] or random search strategies [23,4], but still often require many queries to be successful. A complementary approach is to compute transferable adversarial examples based on the gradient of substitute networks, [37,38,33,14,54,32,56,45,35].

The success of perturbation-based attacks has sparked an arms race between adversarial attacks and corresponding defense mechanisms [20,31,34,53]. A particularly successful defense method is feature denoising [53], where the features in a neural network are denoised using non-local means during adversarial training. To the best of our knowledge, this defense mechanism has not been successfully broken yet. In our experiments, we show that our patch-based attack can defeat this defense successfully.

**Patch-based white box attacks** Tom et. al. [8] proposed adversarial patch as a white-box attack to cause classification errors. They craft adversarial examples by superimposing a patch onto the input image. Given a deep network, the pattern of the patch is optimized using gradient descent. The trained patch performs well but overfits to the network architecture. As shown in their experiments, the patches trained from four different networks are still not able to confuse a fifth network with a high success rate when the patch area is less than 10%. We perform transferability experiments in Appendix E.

**Patch-based black box attacks.** The seminal work of Fawzi et al. [16] introduced patch-based black box attacks. They don't optimize the pattern of the patches, and instead use the monochrome patches. The position and shape of the rectangular patches was searched using Metropolis-Hastings sampling. We refer their attack as Hastings Patch Attack (HPA). While their approach is successful, the random search strategy requires many queries. Furthermore, our experimental results show that using monochrome patches only to craft adversarial examples leads to very low success rates and even then requires to cover more than 70% of the image(see Experiments 4.2).

We introduce a patch-based black-box attack using textured patches that is optimized with reinforcement learning. Our approach is significantly more query efficient, achieves $> 99\%$ success rates on targeted and non-targeted attacks and modifies only very small areas of the input image.

## 3 Methods

In this section, we first discuss the mathematical framework for patch-based adversarial attacks (Section 3.1). In Section 3.2 we introduce our reinforcement learning (RL) framework for patch-based black-box attacks. Finally, we discuss how the texture of adversarial patches can be optimized efficiently using RL by parametrizing the appearance of the patch with a class-specific texture dictionary learned by clustering Gram matrices of feature activations from a DCNN backbone (Section 3.3).

### 3.1 Mathematical Framework

We denote a deep neural network as a function $\mathbf{y} = \mathbf{f}(\mathbf{x}; \boldsymbol{\theta})$, where $\mathbf{x}$, $\boldsymbol{\theta}$ and $\mathbf{y}$ denote the input image, model parameters, and output score of the model after softmax. To perform an adversarial attack we optimize an objective function:

$$\mathcal{L}\left(\mathbf{y}, y'\right), \quad \text{where} \quad \mathbf{y} = \mathbf{f}\left(\mathbf{g}\left(\mathbf{x}\right); \boldsymbol{\theta}\right), \tag{1}$$

where $\mathcal{L}$ is the loss between the output of the neural network $\mathbf{y}$ and a target class $y'$ with $y$ denoting the ground truth label. $\mathbf{g}(\mathbf{x})$ denotes the adversarial example obtained by perturbing $\mathbf{x}$. For targeted attacks, the goal is to induce a high confidence score for the class $y'$ while non-targeted attacks, it is only to induce misclassifications. In perturbation-based attacks, $\mathbf{g}(\cdot)$ modifies $\mathbf{x}$ at every pixel

and the perturbation is constrained to have a small norm. In contrast, the only constraint for patch-based attacks is that the perturbation must be localized in a small region $\mathcal{E}$:

$$\mathbf{g}(\mathbf{x}): \begin{cases} x_{u,v} = \mathbf{T}(x_{u,v}), & \text{if} \quad (u,v) \in \mathcal{E} \\ x_{u,v} = x_{u,v}, & \text{otherwise} \end{cases} \tag{2}$$

$$\mathcal{E} = \mathbf{s}(\mathbf{x}, \mathbf{f}(\cdot,\theta), \mathcal{S}) \subseteq \{(u,v) \,|u \in [0,H), v \in [0,W)\} \tag{3}$$

$H$, $W$ are the height and width of a image, $u, v$ are the pixel coordinates. $\mathbf{T}(\cdot)$ is the transformation function applied to pixels inside $\mathcal{E}$. To determine $\mathcal{E}$, a search mechanism $\mathbf{s}(\cdot)$ is defined over a search space $\mathcal{S}$ of potential image areas. The optimal region $\mathcal{E}^*$ depends on the input image $\mathbf{x}$ and the neural network $\mathbf{f}(\cdot,\theta)$. HPA uses Metropolis Hastings sampling to search the space $\mathcal{S}$ defined in Eq 4.

### 3.2 Patch Search with Reinforcement Learning

In this section, we propose our Monochrome Patch Attack (MPA). In general, this black-box attack uses monochrome rectangular patches which do not have patterns but have variable sizes and positions. We formulate the search over the position and size of adversarial patches as a reinforcement learning problem. The environment consists of $\mathbf{x}$ and $\mathbf{f}(\cdot,\theta)$, and an agent $\mathbb{A}$ is trained to sequentially place monochrome patches in the input image. The search space is defined as:

$$\mathcal{S} = \{(u_1^1, v_1^2, u_1^3, v_1^4, \cdots, u_C^1, v_C^2, u_C^3, v_C^4)\} \tag{4}$$

where $C$ is the number of patches and each element in this set represents the coordinates of $C$ pair of opposite corner points with each pair determining one rectangular region. $\mathcal{S}$ has $4C$ dimensions therefore we set the agent to take $4C$ actions in sequence to generate $\mathbf{a} \in \mathcal{S}$. We formulate the attack in the following:

$$\mathbb{A}(\theta_{\mathbb{A}}): P(a_t|(a_1,\cdots,a_{t-1}), \mathbf{f}(\cdot;\theta), \mathbf{x}) \qquad t = \{1,\cdots,4C\} \tag{5}$$

$$\mathbf{r} = \begin{cases} \ln y' - \mathbf{A}(\mathbf{a})/\sigma^2, & \text{target attack} \\ \ln(1-y) - \mathbf{A}(\mathbf{a})/\sigma^2, & \text{non-target attack} \end{cases} \tag{6}$$

$$\text{MPA}: \begin{cases} \mathcal{E} = \mathbf{J}(\mathbf{a}) \\ \mathbf{T}(x_{u,v}) = 0 \\ \mathcal{L} = -\mathbf{r} \cdot \ln \mathbf{P} \end{cases} \tag{7}$$

Similar to [41,46], we define $\mathbb{A}$ to be a combination of an LSTM and a fully connected layer which represents a policy network with $\theta_{\mathbb{A}}$ being its parameters. At step $t$, the environment state is determined by previous actions, the deep network and the input. The agent outputs the probability distribution over the possible actions for step $t$ as shown in Eq 5. Then it samples one action and records the probability of the sampled action. In the end, this agent generates an action sequence $\mathbf{a}$ and the probability sequence $\mathbf{P}$ recording sampling these

actions at each step. $\mathbf{J}(\cdot) : \mathbf{a} \rightarrow \mathcal{E}$ is the function transferring $\mathbf{a}$ to the areas formed by the $C$ patches. The values of pixels in $\mathcal{E}$ are changed to 0 as shown in Eq 6. Since $\mathbf{x}$ is normalized, the patch color is gray. The reward $\mathbf{r}$ for the agent is defined in Eq 6, where $\mathbf{A}(\cdot)$ calculates the area of $\mathcal{E}$ and $\sigma$ controls the penalty on this area. The loss function to optimize $\boldsymbol{\theta}_{\mathbb{A}}$ is shown in Eq 7.

Based on this framework, we further extend the search space to

$$\mathcal{S} = \{(u_1^1, v_1^2, u_1^3, v_1^4, R_1^5, G_1^6, B_1^7, \cdots, u_C^1, v_C^2, u_C^3, v_C^4, R_C^5, G_C^6, B_C^7)\} \quad (8)$$

where $R, G, B$ represent the values of the RGB channels of the patches, splitting MPA into two variants MPA_Gray and MPA_RGB.

### 3.3 Texture-based Patch Attacks

Monochrome Patch Attacks (MPAs) are powerful in non-targeted setting, however, in targeted setting their performance is not satisfying (Experiments 4.2), because MPAs only remove information at some parts of the image. The lack of additional input signals prevents MPAs from performing targeted attacks. However, we observe that MPA_RGB achieves superior performance compared to MPA_Gray (Table 4.2), motivating our texture-based patch attacks.

**A Class-specific Dictionary of Adversarial Textures.** A major challenge when adding texture to patches is to find an efficient parameterization of the texture to retain fast and query efficient attacks. Our solution is to build a class-specific texture dictionary, where the patch patterns can be searched from. Each dictionary element represents a prototypical adversarial texture of a target class. Hence, to attack models trained on ImageNet, we build a dictionary with 1000 different categories, corresponding to the 1000 object classes in ImageNet. Each category has 30 different texture images whose contents are extracted from the ImageNet training set (see Figure 2 for examples of dictionary elements).

We generate the texture dictionary using a four step process: First, we extract class-specific textures from a set of images of the target class. Inspired by style transfer [18,17], we use VGG19 [47] as the backbone for extracting texture information from images. Let $\mathbb{D}$ be the fully convolutional part of VGG19 pretrained on ImageNet and it consists of 5 blocks. Let $\mathbf{F}_i^j$ be the feature maps from the $j$th convolution layer in the $i$th block of $\mathbb{D}$, and $\mathbf{G}_i^j$ be the corresponding Gram matrix of the feature activations. Following the approach of style transfer, we feed each image into $\mathbb{D}$ and compute the following gram matrices: $\mathbf{G}_1^2$, $\mathbf{G}_2^2$, $\mathbf{G}_3^2$ and $\mathbf{G}_4^2$. Subsequently, we flatten all gram matrices and concatenate them into one vector $\bar{\mathbf{G}}$ that encodes the texture information.

In a natural image, not all the regions are equally important for the final classification. Often the backgrounds or other objects are not of interest. Therefore, we need to extract only the texture of relevant objects in images and hence make the extracted $\bar{\mathbf{G}}$ more semantically meaningful and increase the transferability among different deep networks. In order to locate the relevant information in each image, we perform Grad-CAM [44] on VGG19 for each image and mask out irrelevant regions of the image before texture extractions.


**Fig. 2.** Examples in our designed texture dictionary.

The third step is to generate the texture embedding. For each category, we conduct k-means algorithms on $\bar{\mathbf{G}}$s and use the 30 calculated clusters as the texture embedding for that category $\left\{\bar{\mathbf{G}}_c^1, \ldots, \bar{\mathbf{G}}_c^{30}\right\}$, in order to increase the generalization property while maintaining the diversity of the embedding. The fourth part is to generate texture images from the $30 \times 1000\bar{\mathbf{G}}$s to build the dictionary. For each $\bar{\mathbf{G}}$, we optimize a texture image $\mathbf{t}$ starting from random noise according to the objective function $\mathcal{L}_{\text{texture}} = \lambda(\bar{\mathbf{G}} - \mathbf{G_t})^2$, where $\mathbf{G_t}$ and $\lambda$ denote the feature embedding of $\mathbf{t}$ and the weight. See details in Section 4.1.

**Integrating the Texture Dictionary into Patch Attack** Combining the generated texture dictionary, we propose Texture-based Patch Attack (TPA). Compared with MPA, the patches with the optimized locations in TPA are textured and provide more additional information, making TPA a powerful attack in both the non-targeted and targeted setting.

There are two updates from MPA to TPA. First, the search space is updated:

$$\mathcal{S} = \{(u_1^1, v_1^2, i_1^3, u_1^4, v_1^5, \cdots, u_C^1, v_C^2, i_C^3, u_C^4, v_C^5)\} \tag{9}$$

where $i_c^3$ indexes the texture image in category $y'$ used to texture the $c$th patch. $u_c^1$, $v_c^2$ determines the patch position in $\mathbf{x}$, represented by $\mathbf{J}(\cdot)_t^1$ in Eq. 10. While $u_c^4$, $v_c^5$ denote the patch position in the $i_c^3$th texture image where we crop the patterns as the texture of the attacking patches, represented by $\mathbf{J}(\cdot)_t^2$. Note that in TPA, $C$ instead of 1 agents are trained one after another. The $c$th agent's task is to find a position to put one more textured patch onto the image with $(c-1)$ patches already superimposed by the previous agents. The number of agents required to perform an successful attack $C$ varies for different $\mathbf{x}$. We set a maximum number of the patches allowed to place and stop training new agents if the attack is already successful or $C$ reaches the limit. The second update is that there is no penalty term on the patch area in Eq. 7, since the size of each patch an agent can place and texture is pre-fixed. Different from MPA, the total area of the attacking regions is well controlled.

$$\text{TPA}: \begin{cases} \mathcal{E} = \mathbf{J}_t^1(u_1^1, v_1^2, \cdots, u_C^1, v_C^2) \\ \mathbf{T}\left(x_{u,v}\right) = \mathbf{J}_t^2((i_1^3, u_1^4, v_1^5, \cdots, i_C^3, u_C^4, v_C^5)) \end{cases} \tag{10}$$

## 4 Experiments

We conduct experiments on a challenging dataset, ILSVRC2012 [42], a popular subset of the ImageNet database [13]. It consists of 1.3M training images and 50k testing images with high resolution. There are 1000 object categories in total, which are distributed approximately uniformly in the training set and strictly uniformly in the testing set. The networks against which we perform the attacks include ResNet [24], DenseNet [27], ResNeXt [55] and MobileNetV2 [43]. Since our texture dictionary are built through VGG [47] backbone, we do not involve this network to demonstrate the transferability of the texture images in the dictionary. For MPA and TPA, we conduct baseline subtraction on the rewards for the agents, and adopt early stopping when the difference of $\ln r$ averaged on 3 consecutive iterations is less than $1 \times 10^{-4}$, where $r$ is reward.

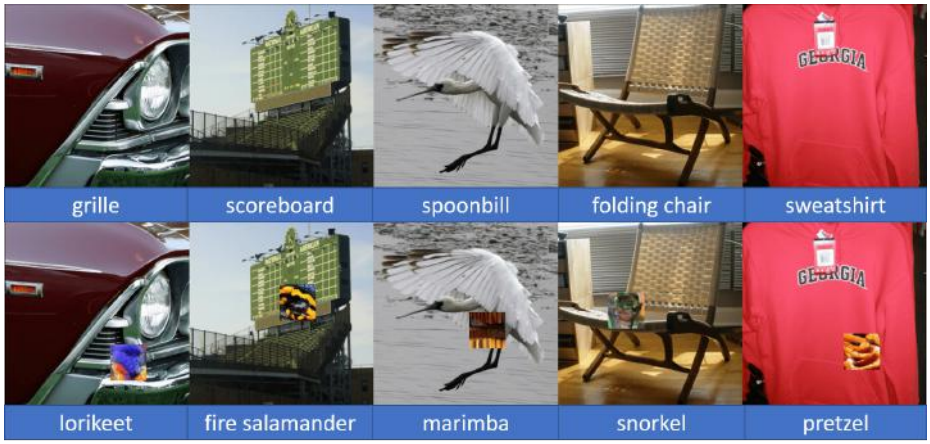


**Fig. 3.** Adversarial examples generated by TPA_N1_4%. The first blue row shows the ground truth labels while the second row the predictions of ResNet50. Each attacking patch is textured by the texture dictionary, taking 4% of the overall area.

### 4.1 Texture Dictionary Setting

The texture dictionary is built upon the training set in ILSVRC2012. All max pooling layers in the extractor $\mathbb{D}$ are replaced by average pooling layers with the kernel and stride sizes both being 2. The Grad-CAM is applied, using the feature map responses of the 5th convolution block of VGG19 to generate a attention map whose values are then normalized between 0 and 1. We consider the region with attention scores larger than the threshold 0.8 as useful. In the generation of texture images, the Adam optimizer is used with the starting learning rate 0.01. The total iteration number is 10000 and the weight $\lambda$ in $\mathcal{L}_{\text{texture}}$ is $1 \times 10^6$. The texture dictionary is constructed with a two-level index structure with first-level key indexing the categories and sub-level indexing 30 texture images of a category. Figure 2 and Figure 3 shows some example texture images and how the texture dictionary is utilized to perform attacks.

### 4.2 Attack Performance

In this section, we demonstrate the effectiveness of our proposed attacks, Monochrome Patch Attack (MPA) and Texture-based Patch Attack (TPA). The baseline is Hastings Patch Attack (HPA) [16]. Both HPA and MPA_Gray superimpose three gray rectangular patches onto the image. Each patch can take an arbitrary aspect ratio and an arbitrary scale. The updates of MPA_RGB is that the patch color is optimized. In TPA, we adopt the square patches and fix their sizes. We set a maximum number of the patches which the algorithm can superimpose. Therefore, the actual number for each image varies. For example, TPA_N4_4% indicates that each patch occupies 4% of the image and the maximum patch

**Table 1.** Experimental results of the non-targeted attacks on a 1000 images randomly selected from the ILSVRC2012 validation set. The maximum allowed query number is 10000. Acc., Avg_area, Avg_qry denotes the classification accuracy, average area percentage occluded by the patches, average queries, respectively

| Network | Attack | Acc. (%) | Avg_area (%) | Avg_qry |
|---|---|---|---|---|
| ResNet50 | – | 72.80 | – | – |
| | **HPA** | 0.40 | 18.05 | 10000 |
| | **MPA**_Gray | 0.00 | 6.57 | 9659 |
| | **MPA**_RGB | 0.00 | 5.41 | 9681 |
| | **TPA**_N4_4% | 0.30 | 5.06 | 1137 |
| | **TPA**_N8_2% | 0.30 | 3.10 | 983 |
| DenseNet121 | – | 74.10 | – | – |
| | **HPA** | 0.10 | 19.82 | 10000 |
| | **MPA**_Gray | 0.00 | 6.87 | 9624 |
| | **MPA**_RGB | 0.00 | 5.73 | 9696 |
| | **TPA**_N4_4% | 0.50 | 5.13 | 1195 |
| | **TPA**_N8_2% | 0.30 | 3.13 | 1001 |
| ResNeXt50 | – | 76.20 | – | – |
| | **HPA** | 0.80 | 19.22 | 10000 |
| | **MPA**_Gray | 0.00 | 7.88 | 9748 |
| | **MPA**_RGB | 0.00 | 6.23 | 9752 |
| | **TPA**_N4_4% | 0.70 | 5.21 | 1280 |
| | **TPA**_N8_2% | 0.50 | 3.25 | 1088 |
| MobileNet-V2 | – | 68.80 | – | – |
| | **HPA** | 0.20 | 16.61 | 10000 |
| | **MPA**_Gray | 0.00 | 5.35 | 9578 |
| | **MPA_RGB** | 0.00 | 4.11 | 9603 |
| | **TPA**_N4_4% | 0.30 | 4.63 | 862 |
| | **TPA**_N8_2% | 0.30 | 2.74 | 756 |

number is 4. This means TPA is able to control the maximum allowed area of an image to be occluded by the patches. There is no such limit on HPA and MPA, which means TPA attacks are better controllable. Additionally, we provide comparisons between Hastings sampling and reinforcement learning in Appendix D.

The experimental results in non-targeted setting are summarized in Table 4.2. In terms of accuracy drops, all the attacks achieve good performances against all the networks, decreasing their classification accuracy down to less than 1%. However, in terms of the average attacked area, HPA occludes 18.05%, 19.82%, 19.22% and 16.61% of the original images against the 4 architectures, while our MPA_RGB only occludes 5.41%, 5.73%, 6.23% and 4.11% respectively. Comparing MPA_RGB with MPA_Gray, we find that optimizing the RGB channel of the patches in MPA decreases the occluded area averaged over all the cases, from 6.67% to 5.37%. This proves that increasing the optimization dimensions and improving the complexity of the patches is beneficial, motivating us to texture these patches. TPA occludes the least area of the image with 3.10%, 3.13%,

3.25% and 2.74% against the different networks. TPA_N8_8% works better than TPA_N4_4%. In terms of query numbers, HPA is inefficient and always uses the whole query budget since it takes 10000 samples and chooses the best one. From MPA to TPA, the algorithm becomes more and more efficient with the query times dropping from 9652 to 957.

For the more challenging targeted setting the experimental results are reported in Table 2. Before performing the attacks, all the networks have a target accuracy not larger than 0.1%. It is observed that although HPA increases target accuracy to 23.05% on average, it occluded 71.54%, 71.68%, 72.57% and 69.45% of the image in the four attacking cases, failing to be considered a successful attack algorithm. For MPA, we use the RGB version since it has been proved to be superior than the gray version in Table 4.2. Although the MPA can only increase the target accuracy to 26.53%, it occludes much less areas than HPA with an average proportion 17.08%. On the contrary, our TPA achieves high performances. TPA_N10_4% is able to increase the target accuracy to 99.70%, 99.90%, 99.70% and 99.90% against the different architectures. The other 2 variant TPA_N10_2% and TPA_N10_10% corresponds to two different requirements for the attack. The first one provides the smaller occlusion area as it uses 7.80%, 7.87%, 7.59% and 7.78% of the areas respectively to increase the target accuracy to 97.70% on average. The second one is more query-efficient as it takes 3747, 3970, 3538 and 4422 queries and obtain an average target accuracy 100%.

In both the non-targeted and non-targeted settings, MPA and TPA are superior to HPA to a large margin. TPA is the best attack among all the perspectives including the accuracy/target_accuracy, occluded areas and query efficiency.

### 4.3 Texture-based Patch Attacks Against Defenses

This section evaluates our attacks against popular defenses. As our MPA and TPA are new types of attacks, we first test them on traditional SOTA defense methods [53]. Another direction is to defend our attack with shape-biased network [19], which is expected to be a good defense against our texture-based patch attack. Additionally, we perform evaluation against the Local Gradients Smoothing (LGS) [36] specifically designed to defend against the patch-based attack in Appendix C.

**Defense 1: Feature Denoising.** In this experimental part, we choose Denoise-ResNet152 [53] to perform MPA and TPA against. It is the SOTA defense against traditional perturbation-based adversarial attacks in a white-box setting. In this scenario, the attacker has access to the architecture and weights of the deep network. This is a strictly easier setting than a black-box one. PGD [34] can only decrease the accuracy to 55.7% and 45.5% after 10 and 100 iterations, respectively. Our experimental results are summarized in Table 3. For non-targeted attacks, both MPA and TPA successfully attack Denoise-ResNet152. MPA reduces the accuracy from 61.6% to 0.00% with the occluded area only being 0.48%, which is even smaller than those of any our previous non-targeted attack

**Table 2.** Experimental results of the targeted attacks on a 1000 images randomly selected from the ILSVRC2012 validation set. The maximum allowed query number is 50000. The target label for each image is difference from its ground truth label. T_acc., Avg_area, Avg_qry denotes the classification accuracy on target labels, average area percentage occluded by the patches, average queries, respectively

| Network | Attack | T_acc. (%) | Avg_area (%) | Avg_qry |
|---|---|---|---|---|
| ResNet50 | – | 0.10 | – | – |
| | **HPA** | 23.20 | 71.54 | 50000 |
| | **MPA**_RGB | 25.90 | 18.45 | 28361 |
| | **TPA**_N10_2% | 97.60 | 7.80 | 15728 |
| | **TPA**_N10_4% | 99.70 | 9.97 | 8643 |
| | **TPA**_N10_10% | 100.00 | 15.36 | 3747 |
| DenseNet121 | – | 0.10 | – | – |
| | **HPA** | 21.50 | 71.68 | 50000 |
| | **MPA**_RGB | 24.90 | 19.38 | 28088 |
| | **TPA**_N10_2% | 97.10 | 7.87 | 15920 |
| | **TPA**_N10_4% | 99.90 | 10.19 | 8953 |
| | **TPA**_N10_10% | 100.00 | 15.84 | 3970 |
| ResNeXt50 | – | 0.00 | – | – |
| | **HPA** | 25.40 | 72.57 | 50000 |
| | **MPA**_RGB | 27.60 | 13.86 | 24738 |
| | **TPA**_N10_2% | 97.60 | 7.59 | 15189 |
| | **TPA**_N10_4% | 99.70 | 9.60 | 8223 |
| | **TPA**_N10_10% | 100.00 | 15.04 | 3538 |
| MobileNet-V2 | – | 0.10 | – | – |
| | **HPA** | 22.10 | 69.45 | 50000 |
| | **MPA**_RGB | 27.70 | 16.64 | 28294 |
| | **TPA**_N10_2% | 98.50 | 7.78 | 15479 |
| | **TPA**_N10_4% | 99.90 | 10.39 | 8948 |
| | **TPA**_N10_10% | 100.00 | 16.85 | 4422 |

on normal networks in Table 4.2. The two versions of TPA decrease the accuracy to 1.6% and 1.3%, respectively, both with the taken queries less than 1000. For targeted attacks, the target accuracy for the network is 0.1%. Although MPA only increases this to 38.3%, it is higher than that of any our previous MPA attacks in targeted settings in Table 2. TPA_N10_4% is able to improve the target accuracy to 94.60%, reflecting the vulnerability of this defense against TPA.

**Defense 2: Against Shape-biased Network.** The textures of the patch play a significant role in magnifying the power of our patch attack, as shown in the comparisons between MPA and TPA in 4.2. According to this dependence on the texture, the best defense against TPA is the model making predictions primarily based on the shapes of objects in the images instead of being largely influenced by their textures. Therefore, we consider the Shape-Network in [19] as the current best potential defense against TPA. The Shape-Network is trained

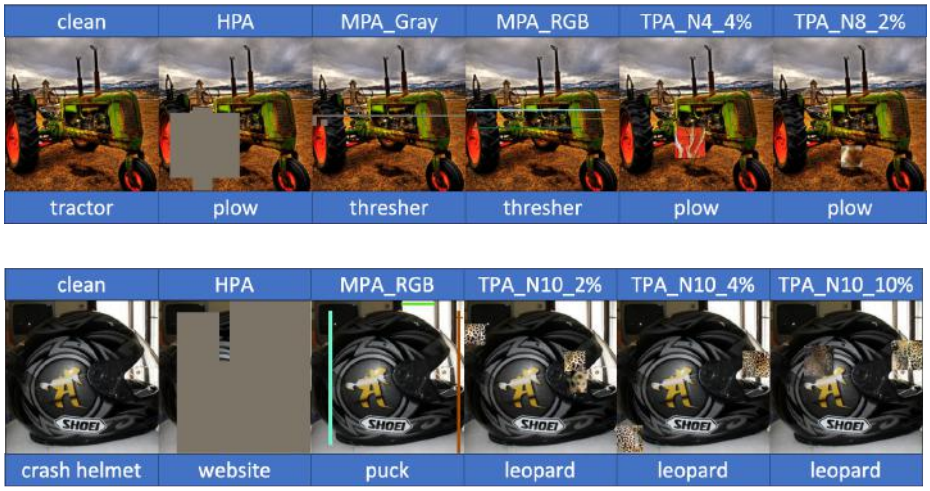


**Fig. 4.** Visualization of patch-attacked examples on ResNet50. The first row corresponds to non-targeted attacks and the second targeted attacks with the target class being leopard. More examples and their attention maps are provided in Appendix A and B.

on the Stylized-ImageNet, which is created by conducting style transfer on the whole training and validation sets of ImageNet, randomly changing object textures while maintaining object shapes in each image. By this design, the ShapeNetwork is supposed to be insensitive to textures but rely more on shapes to make inferences. Note that the construction of our texture dictionary used by TPA is

**Table 3.** Experimental results of the defenses on 1000 images randomly selected from the ILSVRC2012 validation set. The maximum allowed query number is 10000 and 50000 for the non-targeted and targeted settings. Acc., T_acc., Avg_area, and Avg_qry denote the classification accuracy on ground truth and target labels, average area percentage occluded by the patches, average query number, respectively

| Non-target | Attack | Acc. (%) | Avg_area (%) | Avg_qry |
|---|---|---|---|---|
| Denoise_ResNet152 | – | 61.60 | – | – |
| | **MPA**_RGB | 0.00 | 0.48 | 9287 |
| | **TPA**_N4_4% | 1.60 | 4.71 | 919 |
| | **TPA_N8_10%** | 1.30 | 2.91 | 867 |
| Target | Attack | T_acc. (%) | Avg_area (%) | Avg_qry |
| Denoise_ResNet152 | – | 0.10 | – | – |
| | **MPA**_RGB | 38.30 | 6.39 | 27464 |
| | **TPA**_N10_2% | 84.00 | 9.73 | 22196 |
| | **TPA**_N10_4% | 94.60 | 13.40 | 13932 |
| | **TPA**_N10_10% | 99.30 | 20.90 | 6920 |

**Table 4.** Experimental results of the defenses on 1000 images randomly selected from the ILSVRC2012 validation set. The maximum allowed query number is 10000 and 50000 for the non-targeted and targeted settings. Acc., T_acc., Avg_area, and Avg_qry denote the classification accuracy on ground truth and target labels, average area percentage occluded by the patches, average query number, respectively

| Non-target | Attack | Acc. (%) | Avg_area (%) | Avg_qry |
|---|---|---|---|---|
| Shape-Network | – | 73.70 | – | – |
| | **TPA**_N4_4% | 0.50 | 5.19 | 1242 |
| | **TPA_N8_10%** | 0.20 | 3.17 | 1031 |
| Target | Attack | T_acc. (%) | Avg_area (%) | Avg_qry |
| Shape-Network | – | 0.10 | – | – |
| | **TPA**_N10_2% | 96.30 | 8.36 | 17443 |
| | **TPA**_N10_4% | 100.00 | 10.31 | 9229 |
| | **TPA**_N10_10% | 100.00 | 15.52 | 3822 |

also inspired by the style transfer dealing with object textures, as illustrated in Section 3.3. So in principle, the Shape-Network is a very strong defense against our attacks. However, the experimental results in Table 4 show that TPAs easily confuse the Shape-Network with basically no difference as against a normal deep network. In the non-targeted setting, TPAs decrease the network's accuracy from 77.70% to 0.50% and 0.20% with the occluded area being 5.19% and 3.17% for the two variants respectively. The average taken queries is 1137. For the targeted setting, the three variants of TPAs increase the target accuracy from 0.10% to 96.30%, 100.00% and 100.00%, respectively. TPA_N10_2% provides the smallest occluded area 8.36%. TPA_N10_10% is the most query-efficient with only 3822 taken queries but high occluded area 15.52%. TPA_N10_4% is the moderate choice with small occluded area 10.31% and small taken queries 9229.

## 5 Conclusion

In this work, we propose *PatchAttack*, a powerful black-box texture-based patch attack. Our attack shows that even small textured patches are able to break deep neural networks. We model the attacking process as a reinforcement learning problem with an agent that is trained to superimpose patches onto the images in order to induce misclassification. Using monochrome patches only, we achieve a strong performance on non-targeted attack, surpassing previous work by a large margin using less queries and smaller patch areas. After enabling the reinforcement learning agent to also use texture from an adversarial texture dictionary, PatchAttack achieves exceptional performances in both non-targeted and targeted settings. Furthermore, we show that PatchAttack breaks traditional SOTA defenses and shape-based networks.

**Acknowledgements** This work was supported in part by the Johns Hopkins University Institute for Assured Autonomy with grant IAA 80052272, National Science Foundation (NSF) grant BCS-1827427 and NSF grant CNS-18-54000.

## References


1. Clarifai api (2020), https://clarifai.com/
2. Google vision api (2020), https://cloud.google.com/vision/
3. Alzantot, M., Sharma, Y., Chakraborty, S., Zhang, H., Hsieh, C.J., Srivastava, M.B.: Genattack: Practical black-box attacks with gradient-free optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference (2019)
4. Andriushchenko, M., Croce, F., Flammarion, N., Hein, M.: Square attack: a query-efficient black-box adversarial attack via random search. arXiv preprint arXiv:1912.00049 (2019)
5. Bhagoji, A.N., He, W., Li, B., Song, D.: Practical black-box attacks on deep neural networks using efficient query mechanisms. In: European Conference on Computer Vision. Springer (2018)
6. Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., et al.: End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316 (2016)
7. Brendel, W., Rauber, J., Bethge, M.: Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. arXiv preprint arXiv:1712.04248 (2017)
8. Brown, T.B., Mané, D., Roy, A., Abadi, M., Gilmer, J.: Adversarial patch. arXiv preprint arXiv:1712.09665 (2017)
9. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (2017)
10. Chen, P.Y., Zhang, H., Sharma, Y., Yi, J., Hsieh, C.J.: Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security (2017)
11. Cheng, M., Le, T., Chen, P.Y., Yi, J., Zhang, H., Hsieh, C.J.: Query-efficient hard-label black-box attack: An optimization-based approach. arXiv preprint arXiv:1807.04457 (2018)
12. Chernikova, A., Oprea, A., Nita-Rotaru, C., Kim, B.: Are self-driving cars secure? evasion attacks against deep neural networks for steering angle prediction. In: 2019 IEEE Security and Privacy Workshops (2019)
13. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2009)
14. Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., Li, J.: Boosting adversarial attacks with momentum. In: Proceedings of the IEEE conference on computer vision and pattern recognition (2018)
15. Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., Song, D.: Robust physical-world attacks on deep learning visual classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2018)
16. Fawzi, A., Frossard, P.: Measuring the effect of nuisance variables on classifiers. In: British Machine Vision Conference (2016)
17. Gatys, L., Ecker, A.S., Bethge, M.: Texture synthesis using convolutional neural networks. In: Advances in neural information processing systems (2015)
18. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition (2016)

19. Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F.A., Brendel, W.: Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. arXiv preprint arXiv:1811.12231 (2018)
20. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
21. Goodman, D.: Transferability of adversarial examples to attack cloud-based image classifier service. arXiv pp. arXiv–2001 (2020)
22. Goodman, D., Wei, T.: Cloud-based image classification service is not robust to simple transformations: A forgotten battlefield. arXiv preprint arXiv:1906.07997 (2019)
23. Guo, C., Gardner, J.R., You, Y., Wilson, A.G., Weinberger, K.Q.: Simple black-box adversarial attacks. arXiv preprint arXiv:1905.07121 (2019)
24. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition (2016)
25. Hosseini, H., Kannan, S., Zhang, B., Poovendran, R.: Deceiving google's perspective api built for detecting toxic comments. arXiv preprint arXiv:1702.08138 (2017)
26. Hosseini, H., Xiao, B., Poovendran, R.: Google's cloud vision api is not robust to noise. In: 2017 16th IEEE International Conference on Machine Learning and Applications (2017)
27. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition (2017)
28. Huang, L., Gao, C., Zhou, Y., Xie, C., Yuille, A.L., Zou, C., Liu, N.: Universal physical camouflage attacks on object detectors. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020)
29. Ilyas, A., Engstrom, L., Athalye, A., Lin, J.: Black-box adversarial attacks with limited queries and information. arXiv preprint arXiv:1804.08598 (2018)
30. Ilyas, A., Engstrom, L., Madry, A.: Prior convictions: Black-box adversarial attacks with bandits and priors. arXiv preprint arXiv:1807.07978 (2018)
31. Kannan, H., Kurakin, A., Goodfellow, I.: Adversarial logit pairing. arXiv preprint arXiv:1803.06373 (2018)
32. Li, Y., Bai, S., Zhou, Y., Xie, C., Zhang, Z., Yuille, A.: Learning transferable adversarial examples via ghost networks. In: Proceedings of the AAAI Conference on Artificial Intelligence (2020)
33. Liu, Y., Chen, X., Liu, C., Song, D.: Delving into transferable adversarial examples and black-box attacks. arXiv preprint arXiv:1611.02770 (2016)
34. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)
35. Naseer, M.M., Khan, S.H., Khan, M.H., Khan, F.S., Porikli, F.: Cross-domain transferability of adversarial perturbations. In: Advances in Neural Information Processing Systems (2019)
36. Naseer, M., Khan, S., Porikli, F.: Local gradients smoothing: Defense against localized adversarial attacks. In: 2019 IEEE Winter Conference on Applications of Computer Vision (2019)
37. Papernot, N., McDaniel, P., Goodfellow, I.: Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. arXiv preprint arXiv:1605.07277 (2016)
38. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against machine learning. In: Proceedings of the 2017 ACM on Asia conference on computer and communications security (2017)

39. Pei, K., Cao, Y., Yang, J., Jana, S.: Deepxplore: Automated whitebox testing of deep learning systems. In: proceedings of the 26th Symposium on Operating Systems Principles (2017)
40. Ranjan, A., Janai, J., Geiger, A., Black, M.J.: Attacking optical flow. In: Proceedings of the IEEE International Conference on Computer Vision (2019)
41. Ren, Z., Wang, X., Zhang, N., Lv, X., Li, L.J.: Deep reinforcement learning-based image captioning with embedding reward. In: Proceedings of the IEEE conference on computer vision and pattern recognition (2017)
42. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115**(3), 211–252 (2015)
43. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition (2018)
44. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE international conference on computer vision (2017)
45. Shi, Y., Wang, S., Han, Y.: Curls & whey: Boosting black-box adversarial attacks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2019)
46. Shu, M., Liu, C., Qiu, W., Yuille, A.: Identifying model weakness with adversarial examiner. arXiv preprint arXiv:1911.11230 (2019)
47. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
48. Sitawarin, C., Bhagoji, A.N., Mosenia, A., Chiang, M., Mittal, P.: Darts: Deceiving autonomous cars with toxic signs. arXiv preprint arXiv:1802.06430 (2018)
49. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
50. Tian, Y., Pei, K., Jana, S., Ray, B.: Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In: Proceedings of the 40th international conference on software engineering (2018)
51. Tu, C.C., Ting, P., Chen, P.Y., Liu, S., Zhang, H., Yi, J., Hsieh, C.J., Cheng, S.M.: Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence (2019)
52. Uesato, J., O'Donoghue, B., Oord, A.v.d., Kohli, P.: Adversarial risk and the dangers of evaluating against weak attacks. arXiv preprint arXiv:1802.05666 (2018)
53. Xie, C., Wu, Y., Maaten, L.v.d., Yuille, A.L., He, K.: Feature denoising for improving adversarial robustness. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2019)
54. Xie, C., Zhang, Z., Zhou, Y., Bai, S., Wang, J., Ren, Z., Yuille, A.L.: Improving transferability of adversarial examples with input diversity. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (2019)
55. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition (2017)
56. Zhou, W., Hou, X., Chen, Y., Tang, M., Huang, X., Gan, X., Yang, Y.: Transferable adversarial perturbations. In: Proceedings of the European Conference on Computer Vision (2018)

## A Adversarial Examples

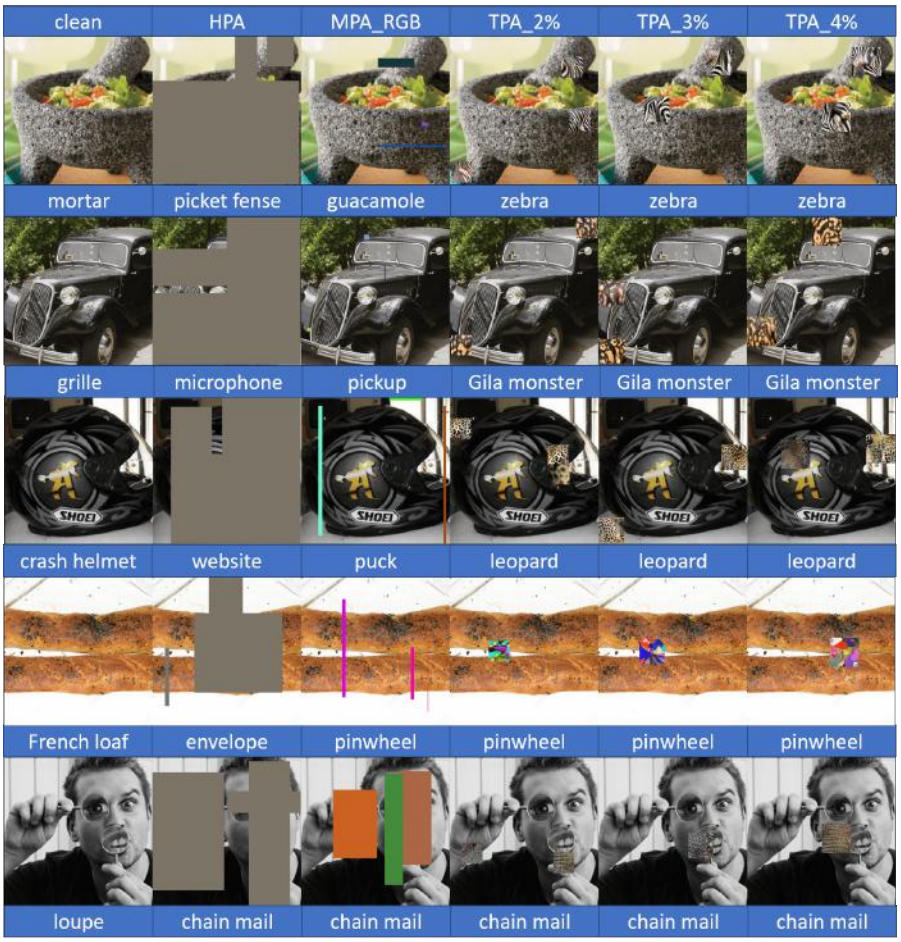


**Fig. 5.** Adversarial examples generated by targeted *PatchAttack* on ResNet50. The images in the same row are attacked with the same target class. The first three columns correspond to clean images, Hastings Patch Attack (HPA) and Monochrome Patch Attack (MPA), and the last three columns Texture-based Patch Attack (TPA) with the single patch area being 2%, 3% and 4%, respectively.

## B Attention Maps of Adversarial Examples

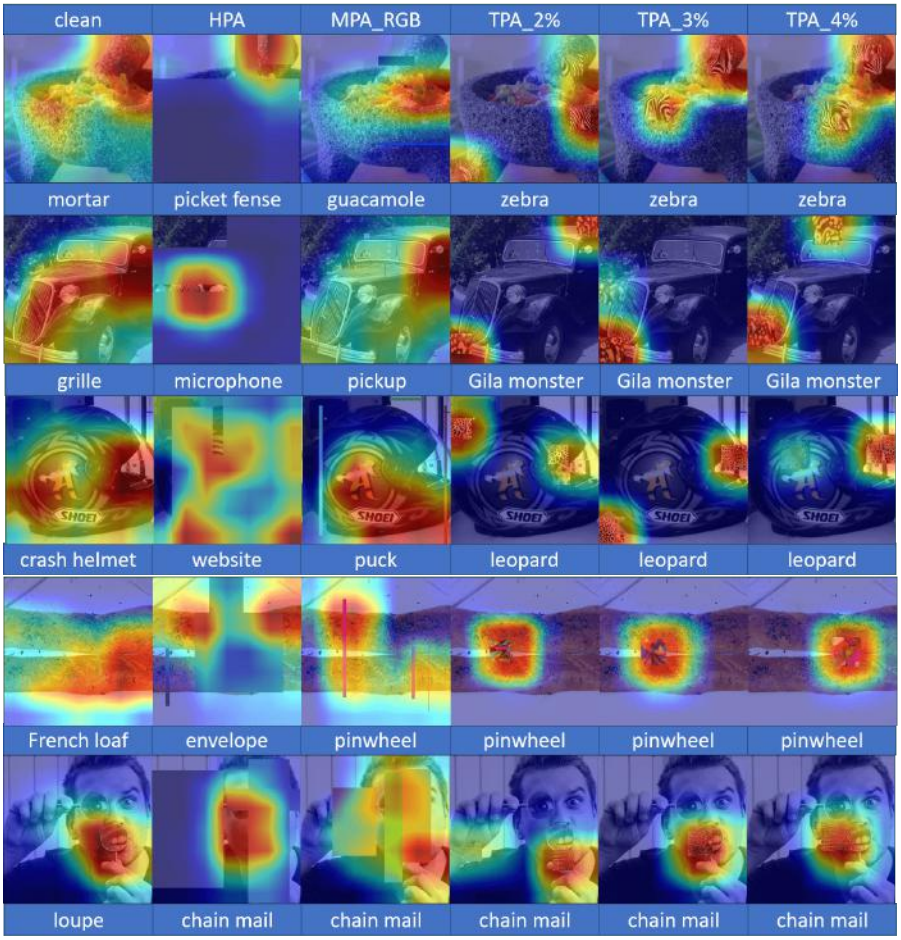


**Fig. 6.** Attention maps of the adversarial examples in Fig. 5 generated by Grad-CAM on ResNet50. The images in the same row are attacked with the same target class. The first three columns correspond to clean images, Hastings Patch Attack (HPA) and Monochrome Patch Attack (MPA), and the last three columns Texture-based Patch Attack (TPA) with the single patch area being 2%, 3% and 4%, respectively.

## C Defense 3: against white-box patch attack defense

We evaluate our Texture-based Patch Attack (TPA) against Local Gradients Smoothing (LGS) [36] which is dedicated to defend against white-box patch attack on ImageNet. We perform the targeted attack on ResNet50 with the same setting in 4.2 and show the result in Table 5. While LGS leads to slightly higher patch area and slightly lower target accuracy, it clearly fails to defend against TPA.

**Table 5.** Experimental results on 1000 images randomly selected from the ILSVRC2012 validation set. T_acc. and Avg_area denote the classification accuracy on target labels and average area percentage occluded by the patches, respectively

| Attack | Defense | T_acc.(%) | Avg_area(%) |
|---|---|---|---|
| **TPA**_N10_4% | – | 99.70 | 9.97 |
| **TPA**_N10_4% | LGS | 97.50 | 13.25 |

## D Comparison between Metropolis-Hastings sampling and Reinforcement Learning

We implement the Hastings Patch Attack (HPA) in the same RGB and texture search space used by Monochrome Patch Attack (MPA) and Texture-based Patch Attack (TPA) to compare this sampling method and Reinforcement Learning method (RL). The experiments are performed on ResNet50 with the standard setup in 4.2.

**Table 6.** Experimental results of the defenses on 1000 images randomly selected from the ILSVRC2012 validation set. The maximum allowed query number is 10000 and 50000 for the non-targeted and targeted settings. Acc., T_acc., Avg_area, and Avg_qry denote the classification accuracy on ground truth and target labels, average area percentage occluded by the patches, average query number, respectively

| Non-targeted | Acc.(%) | Avg_area(%) | Avg_qry |
|---|---|---|---|
| **HPA**_RGB | 0.20 | 16.88 | 10000 |
| **MPA**_RGB | 0.00 | 5.41 | 9681 |
| targeted | T_acc.(%) | Avg_area(%) | Avg_qry |
| **HPA**_RGB | 24.80 | 69.63 | 50000 |
| **MPA**_RGB | 25.90 | 18.45 | 28361 |

It is observed that MPA_RGB is better than HPA_RGB, because it achieves lower accuracy in the non-targeted setting and higher target accuracy in targeted setting, while also using a smaller area and less queries.

**Table 7.** Experimental results of the defenses on 1000 images randomly selected from the ILSVRC2012 validation set. The maximum allowed query number is 10000 and 50000 for the non-targeted and targeted settings. Acc., T_acc., Avg_area, and Avg_qry denote the classification accuracy on ground truth and target labels, average area percentage occluded by the patches, average query number, respectively

| Non-targeted | Acc.(%) | Avg_area(%) | Avg_qry |
|---|---|---|---|
| **HPA**_N4_4% | 1.10 | 5.42 | 3522.5 |
| **TPA**_N4_4% | 0.30 | 5.06 | 1137 |
| targeted | T_acc.(%) | Avg_area(%) | Avg_qry |
| **HPA**_N10_4% | 99.80 | 10.89 | 14345 |
| **TPA**_N10_4% | 99.70 | 9.97 | 8643 |

Here we can observe that RL still is much more query-efficient than the sampling algorithm, however, the methods are comparable in terms of accuracy and occlusion area. This can be attributed to our improved search space for performing the attacks, highlighting the importance of our texture dictionary.

## E Transferability of adversarial patch dictionary generated by white-box method

We implement Adversarial Patch (AP) [8], the white-box patch attack. We first generate an adversarial patch dictionary (AdvPatchDict) consisting of 1000 classes by attacking VGG19 using AP on ImageNet dataset, and then attack the other 4 networks used in our experiments with those patches in the dictionary. The results are shown in the Table 8. In non-targeted settting, AdvPatchDict decreases accuracy to 0.20% on VGG19 but only to $56\% - 66\%$ on the other networks. In targeted setting, it increases target accuracy on VGG to 98.20% but basically fails to increase it for other networks. Clearly, AdvPatchDict generated by the white-box method overfits to the architecture used to generate them, highlighting the superiority of the design of our texture dictionary.

**Table 8.** Experimental results on 1000 images randomly selected from the ILSVRC2012 validation set. Acc., T_acc. and P_area, denote the classification accuracy on ground truth and target labels, area percentage occluded by the adversarial patch, respectively

| AdvPatchDict | Non-targeted Acc.(%) | targeted T_acc.(%) | P_area(%) |
|---|---|---|---|
| VGG19 | 0.20 | 98.20 | 8.95 |
| ResNet50 | 62.50 | 0.00 | 8.95 |
| DenseNet121 | 57.80 | 1.70 | 8.95 |
| ResNeXt50 | 65.30 | 0.10 | 8.95 |
| MobileNet-V2 | 56.00 | 0.10 | 8.95 |