Texture Re-Scalable Universal Adversarial Perturbation

Yihao Huang, Qing Guo, *Member, IEEE*, Felix Juefei-Xu, *Member, IEEE*, Ming Hu, *Member, IEEE*, Xiaojun Jia, Xiaochun Cao, *Senior Member, IEEE*, Geguang Pu, and Yang Liu, *Senior Member, IEEE*

***Abstract*— Universal adversarial perturbation (UAP), also known as image-agnostic perturbation, is a fixed perturbation map that can fool the classifier with high probabilities on arbitrary images, making it more practical for attacking deep models in the real world. Previous UAP methods generate a scale-fixed and texture-fixed perturbation map for all images, which ignores the multi-scale objects in images and usually results in a low fooling ratio. Since the widely used convolution neural networks tend to classify objects according to semantic information stored in local textures, it seems a reasonable and intuitive way to improve the UAP from the perspective of utilizing local contents effectively. In this work, we find that the fooling ratios significantly increase when we add a constraint to encourage a small-scale UAP map and repeat it vertically and horizontally to fill the whole image domain. To this end, we propose texture scale-constrained UAP (TSC-UAP), a simple yet effective UAP enhancement method that automatically generates UAPs with category-specific local textures that can fool deep models more easily. Through a low-cost operation that restricts the texture scale, TSC-UAP achieves a considerable improvement in the fooling ratio and attack transferability for both data-dependent and data-free UAP methods. Experiments conducted on two state-of-the-art UAP methods, eight popular CNN models and four classical datasets show the remarkable performance of TSC-UAP.**

***Index Terms*— Adversarial attack, universal adversarial perturbation, texture scale.**

Manuscript received 30 December 2023; revised 12 April 2024 and 29 May 2024; accepted 6 June 2024. Date of publication 17 June 2024; date of current version 18 September 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62025604; in part by the Nanyang Technological University (NTU)-Desay SV Research Program under Grant 2018-0980; in part by the National Research Foundation, Singapore, and DSO National Laboratories through the AI Singapore Program under AI Singapore (AISG) Award AISG2-GC-2023-008; in part by the National Research Foundation, and the Cyber Security Agency under its National Cybersecurity Research and Development Program under Grant NCRP25-P04-TAICeN; and in part by the Career Development Fund (CDF) of the Agency for Science, Technology and Research (A*STAR), under Grant C233312028. The work of Geguang Pu was supported in part by the National Key Research and Development Program under Grant 2020AAA0107800 and in part by Shanghai Collaborative Innovation Center of Trusted Industry Internet Software. The associate editor coordinating the review of this article and approving it for publication was Prof. Yanjiao Chen. *(Corresponding author: Xiaojun Jia.)*

Yihao Huang, Ming Hu, Xiaojun Jia, and Yang Liu are with the College of Computing and Data Science, Nanyang Technological University, Singapore 639798 (e-mail: jiaxiaojunqaq@gmail.com).

Qing Guo is with the Institute of High-Performance Computing (IHPC) and the Centre for Frontier AI Research (CFAR), Agency for Science, Technology and Research (A*STAR), Singapore 138632.

Felix Juefei-Xu is with the Tandon School of Engineering, New York University, New York, NY 11201 USA.

Xiaochun Cao is with the School of Cyber Science and Technology, Sun Yat-sen University, Shenzhen Campus, Shenzhen 528406, China.

Geguang Pu is with the Software Engineering Institute, East China Normal University, Shanghai 200241, China, and also with Shanghai Industrial Control Safety Innovation Technology Company Ltd., Shanghai 201103, China.

Digital Object Identifier 10.1109/TIFS.2024.3416030

I. Introduction

DEEP learning has shown a broad impact on computer vision tasks, including image classification [1], [2], [3], [4], [5], object detection [6], [7], [8], [9], [10], and image segmentation [11], [12], [13], [14], [15]. However, the susceptibility of deep neural networks (DNNs) to quasi-imperceptible adversarial perturbations has raised concerns about their robustness [16], [17], [18], [19], [20], [21]. As a result, deep learning security [22], [23], [24] has become a popular research area, with adversarial attacks [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37] playing a vital role in the community. However, these attacks are not practical for real-world online applications as they require complex optimization algorithms to generate adversarial examples for each input image, which is not general or efficient enough.

To address this issue, Moosavi-Dezfooli et al. [38] proposed a solution known as universal adversarial perturbations (UAPs), which are designed to misclassify a target CNN model over **any** images, referred to as image-agnostic attacks. For the sake of simplicity, we refer to this method as DeepFool-UAP, as it uses the DeepFool [39] algorithm to generate the UAPs. It is evident that the pre-computed UAPs represent a significant threat to real-time applications, which provides a major advantage over basic adversarial attacks. UAPs have also demonstrated their ability to carry out cross-model and cross-data attacks, which increases the likelihood of employing UAPs in the physical world applications.

The following research focuses on exploring new architectures or new loss functions of the UAP generation method [40]. GAP [41] and NAG [42] utilize generative adversarial networks (GAN) architecture to generate UAP, which attempts to capture the distribution of UAP for a given classifier. However, GAN is a complex architecture that is hard to train, which means the requirement of a generator itself is a drawback of these UAP generation approaches. These methods also do not obtain state-of-the-art fooling ratios. Thus our research does not focus on GAN-based UAPs. Compared to exploring new architectures, the innovation in loss functions is a simple modification towards regular UAP generation methods which can be transferred to other methods, showing better generality. DF-UAP [43] and Cos-UAP [44] propose special loss functions related to output logit vectors of the CNN model. They are the state-of-the-art UAP generation methods and Cos-UAP is a bit better than DF-UAP. Furthermore, due to the elaborate design of the loss functions in DF-UAP and Cos-UAP, they can successfully generate UAPs without any

training samples. We call them data-free UAPs to distinguish them from data-dependent methods such as DeepFool-UAP, GAP, *etc*. Although research on UAP has made good progress, we surprisingly find that rare works attempt to improve UAP from the perspective of texture. However, the most prominent features of UAPs are their special textures, which are significantly different from the adversarial noise in non-universal attacks.

We choose UAP generated from SGD-UAP [45] as an example to introduce such texture patterns. SGD-UAP is a representative UAP generation method with a regular pipeline and the state-of-the-art DF-UAP and Cos-UAP are utilizing the same pipeline as it (only differ in loss functions). As shown in δ of Figure 1, the UAP is classified as 'coral' by ResNet50 and the textures distinctly show the characteristic of coral. Due to this, we conduct experiments and find the "category-specific local" textures can benefit the attack performance of UAP. Based on the observation, we propose a simple yet effective UAP enhancement method with texture scale constraints, called TSC-UAP. It shows excellent performance at a minor cost.

To sum up, our work has the following contributions:

- We are the first to substantiate the significance of category-specific local texture as a new research direction to benefit UAP generation methods.
- We propose TSC-UAP, a simple yet effective way to achieve a higher fooling ratio, attack transferability, and data efficiency with minor costs. It is also general enough to enhance data-dependent and data-free UAP methods.
- The experiment conducted on four classical datasets and eight popular CNNs shows the effectiveness of TSC-UAP in generating better UAPs.

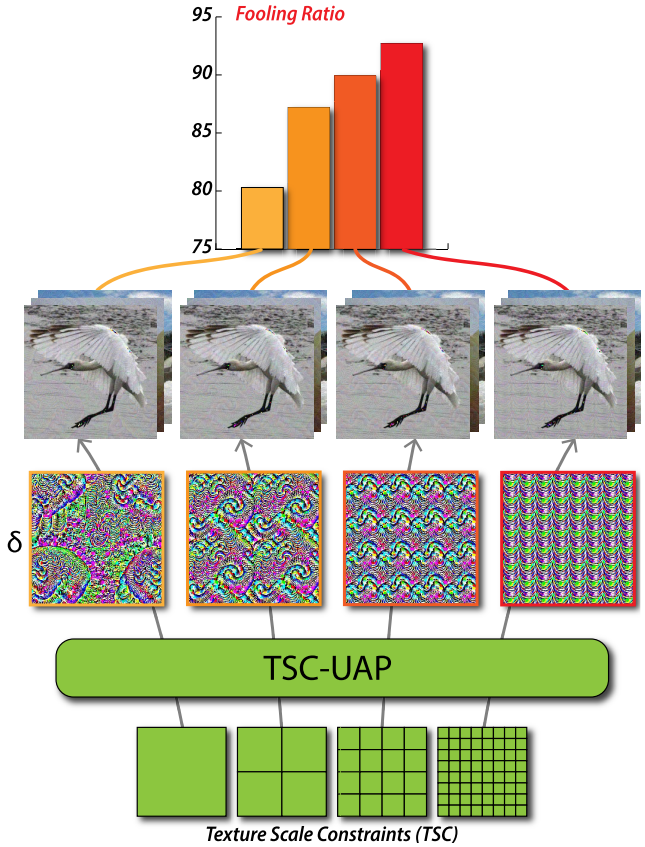

Fig. 1. We propose TSC-UAP, an enhancement method that utilizes texture scale constraints to restrict the UAP. The texture scale constraints facilitate UAP to achieve higher fooling ratios, no matter data-dependent or data-free.

II. Related Work

A. Basic Adversarial Attacks

Szegedy et al. [16] proposed the first adversarial attack method, L-BFGS, which fools neural networks by adding quasi-imperceptible perturbations to input images. Inspired by [16], Goodfellow et al. [25] proposed a simple yet famous method, the Fast Gradient Sign Method (FGSM), which aims to maximize the attack success rate with a restricted perturbation budget (*i.e.*, epsilon (ϵ)) in a single attack step. However, FGSM did not achieve a high attack success rate due to its one-step attack procedure. To improve the attack performance, Iterative FGSM (I-FGSM) [26] and Projected Gradient Descent (PGD) [46] have been proposed to perturb the image in multiple iterations. In each iteration, the images are only allowed to have added noise that is less than a fraction of epsilon.

Different from the FGSM-series methods, other popular attacks, such as DeepFool [39] and C&W [27], approach adversarial attacks from a different perspective. They aim to minimize the perturbation magnitude under the situation that the image is misclassified. DeepFool [39] uses the decision boundaries of the target model as the gradient guidance and updates the gradient. Specifically, it chooses the direction that is orthogonal to the decision hyperplane in each attack iteration. C&W [27] formulates the adversarial attack as an optimization problem and proposes several objective functions to replace highly non-linear classification functions, resulting in better optimization. The generation of adversarial examples through complicated optimization in these basic adversarial attacks is less practical to apply in real-world attack scenarios.

B. Universal Adversarial Attack and Defense

1) Attack: Moosavi-Dezfooli et al. [38] first proposed the concept of Universal Adversarial Perturbations (UAPs). UAP is a single fixed image-agnostic adversarial noise that can fool most of the images from a data distribution with a given CNN model. They proposed iteratively crafting perturbations for training samples by using DeepFool [39] and limiting the perturbation magnitude with a sphere constraint. It is obvious that such UAP can fool images without additional online processes, which shows the practical possibility in real-world applications. To enhance the fooling ratio and attack efficiency, Shafahi et al. [45] proposed a Stochastic Gradient Descent UAP (SGD-UAP) attack. As the inner loop of the DeepFool attack is very time-consuming, they utilized PGD to optimize the perturbation by batches instead of a single image, which also guarantees convergence.

SV-UAP [47] exploited the Jacobian matrices in feature maps to calculate singular vectors for generating UAPs. The method is able to achieve more than 60% fooling ratio with only 64 training samples, which is very data-efficient.

However, since SV-UAP still requires training data, [48] aimed to generate UAPs without any training data, termed data-free UAPs. The data-free methods [44], [49], [50] focus on the neuron values of the target CNN model and increase the model's uncertainty by special loss function for generating UAPs. AT-UAP [51] and TRM-UAP [52] are newly proposed state-of-the-art data-free methods which deliver superior attack performance. AT-UAP introduces a consistency regularizer to analyze the relationship among training data. TRM-UAP redefines the task of generating UAP as a truncated ratio optimization problem, aiming to refine the UAP generation process through the balance of positive and negative activations.

Inspired by Generative Adversarial Networks (GAN), Network for Adversary Generation (NAG) [42] was proposed to generate adversarial noise by simulating the distribution of UAPs. GAP [41] also followed a similar idea. Since GAN-based UAP methods are complex and difficult to train, the use of GAN for attack is a drawback of UAP approaches. They also cannot achieve a high fooling ratio.

In previous works, only PD-UA [50] somewhat mentioned the influence of texture on UAP. They set a texture priori and force the UAP to have a similar style as it. However, the texture priori may be opposite to the generation direction of the UAP category. For example, they said the circle textural pattern achieved the best performance in their experiment. Circle texture priori may be suitable for generating animal-like UAPs for the eyes of animals are of the circle shape. It is not suitable for generating UAP categories that only have line contour, *e.g.*, table, binder. Compared with PD-UA, our method focuses on restricting the scale of texture in generated UAP, which constructs non-prior class-related texture by the attack method automatically.

2) Defense: Akhtar et al. [53] propose the first defense method against UAP. Their method uses a Perturbation Rectifying Network (PRN) as a kind of pre-processing. The PRN is trained with both real and fake perturbations that don't depend on the image. Mummadi et al. [54] demonstrate that adversarial training can effectively prevent UAPs. Shafahi et al. [45] suggest training robust models at a low cost through the optimization of a min-max problem, employing either alternating or simultaneous stochastic gradient approaches.

III. Methodology

A. Problem Statement

Let $\mathcal{X} \in \mathbb{R}^d$ denotes a distribution of training images and $\hat{\mathcal{C}}(\cdot)$ represents a classifier function. The objective of UAP is to seek a fixed single perturbation vector $\delta \in \mathbb{R}^d$ which can fool the $\hat{\mathcal{C}}(\cdot)$ on most of the data samples $x \sim \mathcal{X}$. Please note that as a restricted adversarial attack method, δ should be a tiny noise bounded by ϵ on the l_p-norm. Specifically, UAP aims to seek δ such that

$$\hat{\mathcal{C}}(x+\delta) \neq \hat{\mathcal{C}}(x) \text{ for most } x \sim \mathcal{X}, \text{ s.t. } \|\delta\|_p \leq \epsilon. \quad (1)$$

In previous works, $\epsilon = 10/255$ and $p = \infty$ is a most common choice for image value ranges in [0, 1].

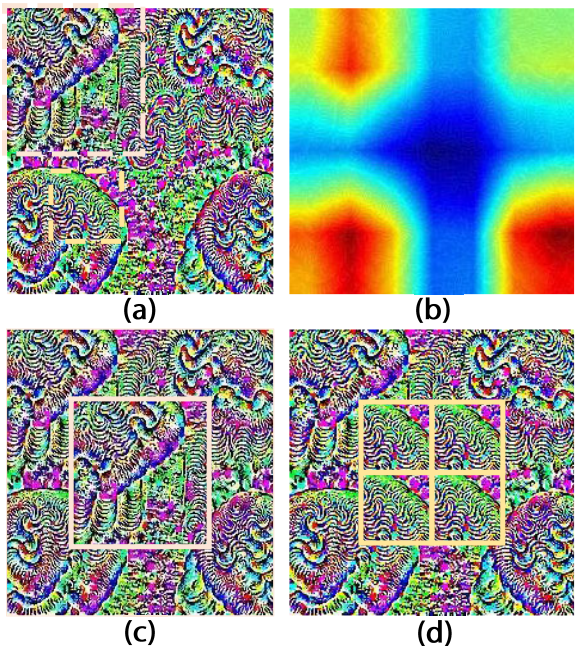

Fig. 2. (a): UAP generated by SGD-UAP. The UAP achieves 80.3% fooling ratio. (b) Attention map of (a). (c): Replace the center of (a) with textures in the top left corner of (a) (*i.e.*, textures in the pink box). The UAP achieves 81.1% fooling ratio. (d): Replace the center of (a) with the replicated local textures in (a) (*i.e.*, textures in the yellow box). The UAP achieves 84.3% fooling ratio.

The most commonly used metric to evaluate the effectiveness of UAP is the *fooling ratio*, which is defined as:

$$\frac{|\{x \in \mathcal{X}_{test} : \hat{\mathcal{C}}(x+\delta) \neq \hat{\mathcal{C}}(x)\}|}{N}, \quad (2)$$

where N is the size of test dataset $\mathcal{X}_{test}$.

B. Motivation

Although previous UAP generation methods have achieved a relatively high fooling ratio (around 80-90%) [40], there is still room for improvement. Here, we use the UAP generated by SGD-UAP [45] as an example to illustrate our observation. The reason for selecting SGD-UAP is that its objective function is easy to understand, and it is representative since the algorithms of state-of-the-art UAP methods, such as DF-UAP [43] and Cos-UAP [44], are based on it. The objective function is

$$\max_{\delta} \mathcal{L} = \frac{1}{N_{\mathcal{X}}} \sum_{i=1}^{N_{\mathcal{X}}} l(x_i + \delta), \text{ s.t. } \|\delta\|_p \leq \epsilon, \quad (3)$$

where $l(\cdot)$ is the loss function (different in SGD-UAP, DF-UAP and Cos-UAP) and $N_{\mathcal{X}}$ is the size of dataset $\mathcal{X}$.

As shown in Figure 2(a), the UAP is generated by SGD-UAP on ResNet50 with ImageNet training samples and achieves an 80.3% fooling ratio on the validation set. There are three observations about this UAP. ❶ We can observe that the textures in UAP are very similar to the pattern of coral. Such obvious category-related texture can not be observed in non-universal adversarial noise. ❷ This UAP, if added to the clean image x, will make the ResNet50 misclassify the fusion image as 'coral' (*i.e.*, $\hat{\mathcal{C}}(\delta + x) =$ 'coral') with a high

probability. ❸ The UAP itself is also classified as 'coral' if directly put UAP into the ResNet50 (*i.e.*, $\hat{\mathcal{C}}(\delta) =$ 'coral'). Please note that the observation ❷ and ❸ are claimed in the paper of DF-UAP and Cos-UAP in a more general form. Due to the category consistency between visualization and the model's prediction, we conclude that the category-related textures highly influence the category of UAP. Since UAP dominates the classification of fusion images, we further conclude that the category-related texture may influence the attack performance of UAP. The conclusions lead us to focus our research on the textures of UAP.

With our naked eye, we can find obvious coral patterns in the four corners of UAP and the textures in the center of UAP are not distinctly coral-like. From the perspective of CNN, it also gives a similar viewpoint (we generate an attention map by Grad-cam [55] to show which area CNN models take as the evidence to classify UAP as coral. The result (*i.e.*, Figure 2(b)) highlights the four corners). Since the category-related textures may influence the attack performance of UAP, it brings an interesting question: what will happen to the fooling ratio if we replace the insipid (less category-related) textures with the textures of strong category features? Figure 2(c) is achieved by cropping the quarter top left corner of UAP (*i.e.*, texture in the pink box of Figure 2(a)) and pasting it to the center of the original UAP. This manually constructed UAP is coarse because it simply moves the distinct category-related textures to the center (not a carefully calculated location) and does not consider the coherence of textures. Surprisingly, such a coarse UAP obtains an 81.1% fooling ratio, which is higher than the original UAP. This observation inspires us to tile category-related textures into the UAP for better performance.

Furthermore, CNN models tend to classify objects according to local textures rather than global shapes [49], thus maybe pasting relatively small local textures into the UAP will achieve a better fooling ratio. To verify this hypothesis, as shown in Figure 2(d), we crop one-sixteenth of UAP (*i.e.*, texture in the yellow box of Figure 2(a)) that catches model attention and tile four same textures into the center of original UAP. This new UAP fools the CNN at a ratio of 84.3%, which is significantly higher than that of the original UAP. This observation inspires us to research the scale of texture to improve the fooling ratio of UAP.

In conclusion, according to the toy examples, *we suggest adding "category-specific local textures" into the UAP and this may benefit the attack performance of UAP*. The explanation of why we use the "category-specific local texture" instead of "category-related local texture" is in Sec. IV. Although we can manually select the category-specific local textures and add them to the UAP, the procedure is subjective because we need to empirically choose local textures. Thus we face a challenge on how to automatically generate UAP including category-specific local textures. To solve the problem, we propose texture scale-constrained UAP.

C. Texture Scale Constrained UAP

To generate UAP with category-specific local textures, an intuitive idea is to segment UAP into many small regions and ensure each region to be a special texture. Since the generation of UAP is based on gradient back-propagation towards full-scale UAP, it requires processing the gradient of UAP in a fine-grained way to fit various regions, which is complex and difficult. Thus we propose to think the other way around and simplify the problem, that is, given a category-specific local texture patch v, how to construct a bigger patch (*i.e.*, UAP) that has the same size as the training images? It is intuitive that tiling the patch v to be a bigger patch is reasonable for that tile is a regular image processing operation with minor cost. To further simplify the operation, for a given training image $\mathbf{I} \in \mathbb{R}^{H\times W\times 3}$, we suggest exploiting a uniform split ratio α which is the common divisor of H and W to segment the UAP $\delta \in \mathbb{R}^{H\times W\times 3}$. Thus the category-specific local texture is of shape $(\frac{H}{\alpha} \times \frac{W}{\alpha} \times 3)$. With the category-specific local texture v and tile function $\mathcal{T}(\cdot)$, we can obtain the UAP with formula

$$\delta = \mathcal{T}(v, \alpha). \tag{4}$$

With the split ratio α, we can flexibly control the shape of category-specific local textures, and we named such an operation texture scale constraint (TSC). The objective function of TSC-UAP can be extended from Eq. (3) to

$$\max_{v} \mathcal{L} = \frac{1}{N_{\mathcal{X}}} \sum_{i=1}^{N_{\mathcal{X}}} l(x_i + \mathcal{T}(v, \alpha)), \text{ s.t. } \|\mathcal{T}(v, \alpha)\|_p \leq \epsilon. \tag{5}$$

Referring to the texture scale constraints, we enhance the common pipeline of UAP generation methods. As shown in Figure 3, the modification is in the green box and the standard UAP generation pipeline is in the gray boxes. Here we choose the pipeline of SGD-UAP since the method is concise and popular. The pipeline can represent both data-dependent and data-free UAP generation methods, which means our texture scale constraints can be applied to both of them, showing the generality of TSC. Furthermore, the state-of-the-art UAP generation methods (DF-UAP [43] and Cos-UAP [44]) share the same pipeline as SGD-UAP (only the loss function is different), which means our texture scale constraints can also improve the SOTA UAP generation methods. To be specific,

Algorithm 1 TSC-UAP

Input: Training dataset $\mathcal{X}$, Classifier $\hat{\mathcal{C}}$, Batch size m, Number of epochs E, Patch v, Perturbation magnitude ϵ, Split ratio α

Output: Perturbation δ

```
v ← 0 ;                                          ▷ initialization
for epoch = 1 to E do
    I ← |X|/m ;                                  ▷ iteration number
    for iteration = 1 to I do
        δ = T(v, α) ;                            ▷ generate UAP
        B ~ X : |B| = m ;                        ▷ randomly sample
        g_v ← E_{x~B} [∇_v L] ;                  ▷ gradient
        v ← Optim(g_v) ;                         ▷ update patch
        v ← min(ε, max(v, −ε)) ;                 ▷ clipping
    δ = T(v, α) ;                                ▷ generate UAP
```

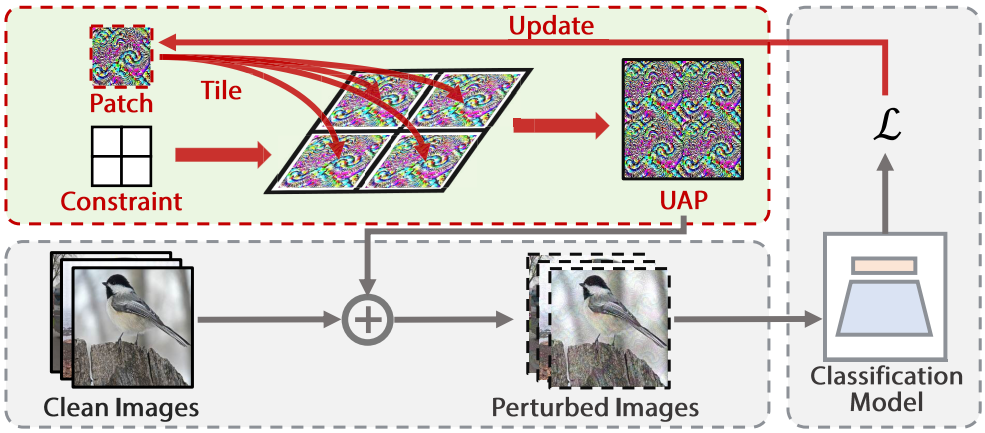

Fig. 3. We choose the pipeline of SGD-UAP since the method is concise. It can represent both data-dependent and data-free UAP generation methods. In the standard UAP generation pipeline, they add UAP to the clean images and put the perturbed images into the classifier to calculate the loss. Our modification is in the green box with red bounds. The difference is that we use the patch (*i.e.*, category-specific local texture) and texture scale constraints (*i.e.*, split ratio α) to obtain the UAP. The gradient back-propagation is applied on the patch, not the UAP.

the loss function of DF-UAP for each data sample is $\mathcal{L} = \max(\mathcal{C}_{gt}(x_i + \delta) - \max_{j \neq gt} \mathcal{C}_j(x_i + \delta), -\kappa)$, where x_i is the sample from the training dataset $\mathcal{X}$, $\mathcal{C}_j$ indicate the j-th value of the output logit, gt indicate the ground truth label of sample x_i, κ is the confidence value set by researcher. The loss function of Cos-UAP for each data sample is $\mathcal{L} = CosSim(\hat{\mathcal{C}}(x_i), \hat{\mathcal{C}}(x_i + \delta))$, where $CosSim$ is cosine similarity, x_i is the sample from the training dataset $\mathcal{X}$. In the standard UAP generation pipeline, they add UAP to the clean images and put the perturbed images into the classifier to calculate the loss. Please note that the loss is calculated for each image batch, not individual images. The update of the UAP is based on the gradient calculated by the loss. In our method, the modification is that we initialize the patch v and texture scale constraints with the split ratio α. Then we tile the patch to obtain the UAP. The gradient back-propagation is applied on the patch, not the UAP, which is computationally efficient. Algorithm 1 outlines the procedure of TSC-UAP. Please note that since L_∞ norm is most commonly used in UAP generation methods, the Algorithm 1 is the implementation of Eq. 5 in L_∞ norm. Specifically, the constraint $\|\mathcal{T}(v, \alpha)\|_{\text{inf}} \leq \epsilon$ forces the largest absolute value of $\mathcal{T}(v, \alpha)$ to be smaller than ϵ. The Line 9 of algorithm is to clip the absolute value within v larger than ϵ, which is equal to $\|\mathcal{T}(v, \alpha)\|_{\text{inf}} \leq \epsilon$ since $\mathcal{T}(v, \alpha)$ is tiled by the patch v.

IV. Explanation on "Category-Specific Local Texture"

The "category-specific local texture" means that we can generate a specific and local universal adversarial texture for each class category (cat, dog, *etc.*) and the local textures of different categories are different. Note the 'specific' property is not exclusive to our local textures, and existing UAPs (*e.g.*, the UAPs generated by SGD-UAP, Cos-UAP, DF-UAP) are also category-specific. The explanation is as follows.

Given a dataset $\mathcal{X}$ and a targeted category y_i, we optimize a targeted UAP δ by solving the following objective function: $\delta_i^* = \text{argmin}_\delta \sum_{x \in \mathcal{X}} l(\hat{\mathcal{C}}(x + \delta), y_i)$, s.t. $\|\delta\|_p \leq \epsilon$. If the dataset $\mathcal{X}$ is fixed and we use the same optimization algorithm for different categories, we have $\delta_i^* \approx \delta_j^*$, if $y_i = y_j$ and $\delta_i^* \neq \delta_j^*$, if $y_i \neq y_j$, which actually means the generated UAPs are category specific. The term '$\approx$' is caused by random operations during optimization like example sampling. In terms of the untargeted attack, although it does not need the targeted category, there is an intriguing phenomenon of untargeted UAP explained in [44]: most images are misclassified to a dominant label, which is the same as the targeted UAP. Hence, untargeted UAP is also category-specific.

The above statement can be extended to our local UAP textures directly. Thus, we say UAPs generated by TSC-UAP contain category-specific local textures. Please note that although the CNNs can recognize category-specific local textures in UAPs and category-specific local textures can benefit the attack performance of UAPs, the shape of textures should not be too small. In specific, we empirically find that when the split ratio $\alpha > 8$ on ImageNet images, the local texture usually may not be human-understandable and fail to enhance the attack performance of UAPs. In order to facilitate the understanding of "category-specific local texture", we select human-understandable UAPs of various categories on different CNNs in Figure 4 with $\alpha = 1/2/4$. We can find that UAPs of different categories have distinct appearances and are related to category patterns. For example, UAPs of the category 'Shield' have shield-like textures. Furthermore, as the split ratio α increases, TSC-UAP is more likely to extract the local texture of objects.

V. Experiment

Datasets and comparative methods: We evaluate the proposed TSC-UAP on a variety of official pre-trained CNN in PyTorch, including GoogleNet [56], VGG [57], ResNet [1], DenseNet [58], MobileNet-v2 [59], *etc*. We use the ImageNet validation set [60] (50,000 images) to evaluate the performance of TSC-UAP. If not specified, the test is done on the entire validation set. With regard to the training dataset of UAP generation, if not specified, the size is 1,000 (by sampling one image for each class (1000 classes) in the ImageNet training dataset). We also carry out experiments on CIFAR10 [61], CIFAR100, and Places-365 [62] for further verification. Places-365 is a scene recognition dataset that has 365 classes.

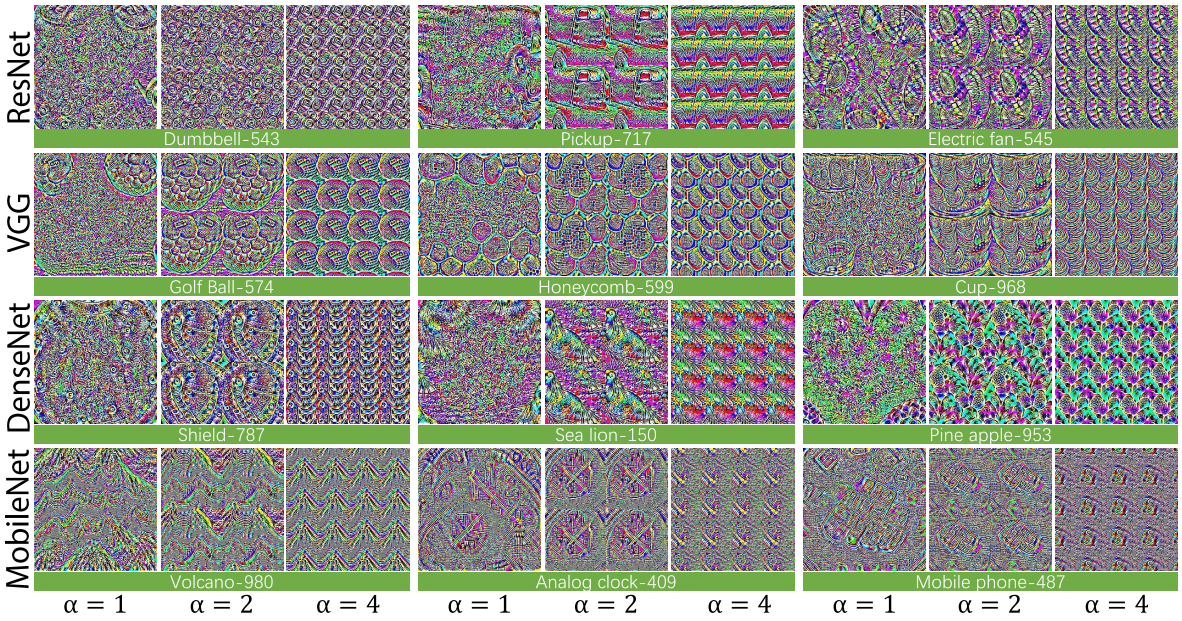

Fig. 4. Here shows the human understandable UAPs generated by attacking ResNet50, VGG19, DenseNet121 and MobileNet-v2 on ImageNet training samples.

We compare the attack performance of TSC-UAP with classical SGD-UAP and SOTA DF-UAP and Cos-UAP. Please note that the DF-UAP and Cos-UAP both do not provide the official code. Since the only difference between them and SGD-UAP is the loss function, we implement these two UAP methods based on the SGD-UAP pipeline and only change the loss function, termed "DF-UAP-rep" and "Cos-UAP-rep" respectively.

Evaluation metric: We use the widely acknowledged fooling ratio metric to quantitatively measure performance.

Implementation details: The maximum perturbation magnitude is set to 10/255 with the pixel range in [0,1]. The attack epoch is 20 and the batch size is 100. The loss function is cross-entropy. We use $tile(\cdot)$ of PyTorch as the tile function $\mathcal{T}(\cdot)$, which is differentiable. For hyper-parameter α, we set it to be one of 1, 2, 4, 8, 16, 32 since they evenly scale the texture size and they are divisible by 224 (the common input shape of CNNs). It is obvious that $\alpha = 1$ means there are no texture scale constraints on the UAP. Since we actually do not add any TSC constraint on SGD-UAP, it is not suitable to regard such a situation as our TSC-UAP. We use $\alpha = 1$ in experiments aiming to make a unified representation ($\alpha = 1, 2, 4, 8, 16, 32$) and facilitate readers to confirm the effects of TSC-UAP under different split ratios. The choice of α values may be an interesting problem, which will be investigated in future work. All the experiments were run on a Ubuntu system with an NVIDIA GeForce RTX 3090 of 24G RAM.

A. Improve Fooling Ratio

We first show the improvement of the fooling ratio on the standard data-dependent UAP generation method (*i.e.*, SGD-UAP). The experiment on SGD-UAP is able to verify the effectiveness of the texture scale constraints on a common UAP pipeline, which shows the wide application range of texture scale constraints.

In Table I, target models (*i.e.*, ResNet50, VGG19, DenseNet121, MobileNet-v2) are in the first row and choices of α are in the first column. For each target model, the highest fooling ratio is in bold. For example, in the second column ('ResNet50'), the performance of baseline is 80.3% (*i.e.*, when $\alpha = 1$). The highest fooling ratio is 92.73%, which is more than 10% higher than the baseline and achieved when α is 8. With respect to VGG, DensetNet, and MobileNet, the highest fooling ratio all outstrip the baseline and the improvements are 10.44%, 19.13%, and 5.11% respectively. Even though the fooling ratio of the baseline is high (*i.e.*, more than 90% in 'MobileNet'), texture scale constraints can still enhance the performance of UAP. Among all four models, when $\alpha \leq 8$ and in the vast majority of cases of $\alpha = 16$, the fooling ratios are higher than the baseline, showing that texture scale constraints can stably improve the UAP.

TABLE I

TARGET MODELS (*i.e.*, RESNET50, VGG19, DENSENET121, MOBILENET-V2) ARE IN THE FIRST ROW AND THE TEXTURE SCALE PARAMETER α IS IN THE FIRST COLUMN. IN EACH CELL, THE HIGHEST FOOLING RATIO IS IN BOLD AND EXCEEDED THE BASELINE BY AN AVERAGE OF 11.77%

α \ Model	ResNet50	VGG19	DenseNet121	MobileNet-v2
1	80.30	82.44	66.05	94.08
2	87.22	90.37	78.83	97.01
4	89.92	**92.88**	82.26	97.25
8	**92.73**	86.57	**85.18**	**99.19**
16	78.47	83.54	79.26	97.53
32	53.96	79.39	51.24	74.65

Furthermore, we find a regular phenomenon that the fooling ratios usually first rise then descend with the increasing of α and peak around $\alpha = 8$. We think this mainly because an overly tiny texture scale means that only a few parameters can be optimized with the gradient. Restricted by perturbation

TABLE II

FIRST TWO COLUMNS REPRESENT THE UAPS GENERATED BY SOURCE MODELS, WHICH IS THE SAME AS THAT IN TABLE I. FIRST TWO ROWS REPRESENT THE TARGET MODELS WHICH NEED TO ATTACK. IN EACH CELL, THERE SHOWS SIX FOOLING RATIO VALUES, CORRESPONDING TO SIX DIFFERENT TEXTURE SCALES (*i.e.*, $\alpha = 1/2/4/8/16/32$). THE HIGHEST FOOLING RATIOS ARE IN RED AND EXCEEDED THE BASELINE BY AN AVERAGE OF 21.44%

FR(α = 1/2/4/8/16/32)(%)		Target			
		ResNet50	VGG19	DenseNet121	MobileNet-v2
Source	ResNet50	-	53.03/59.45/63.83/80.13/61.28/73.13	41.64/49.91/58.12/66.16/51.90/33.97	51.50/57.12/63.99/72.28/55.93/63.98
	VGG19	32.64/43.88/48.61/48.73/42.75/34.61	-	29.18/38.73/43.26/43.99/36.02/28.44	45.95/56.72/63.71/56.40/48.32/55.04
	DenseNet121	42.89/53.29/57.48/60.13/71.58/49.05	50.74/58.44/64.09/75.18/79.54/64.18	-	49.25/54.39/59.18/65.66/66.05/67.78
	MobileNet-v2	26.95/35.32/39.21/34.75/32.89/42.47	38.18/47.85/50.76/41.17/36.01/66.90	23.23/29.51/31.83/22.51/21.47/39.28	-

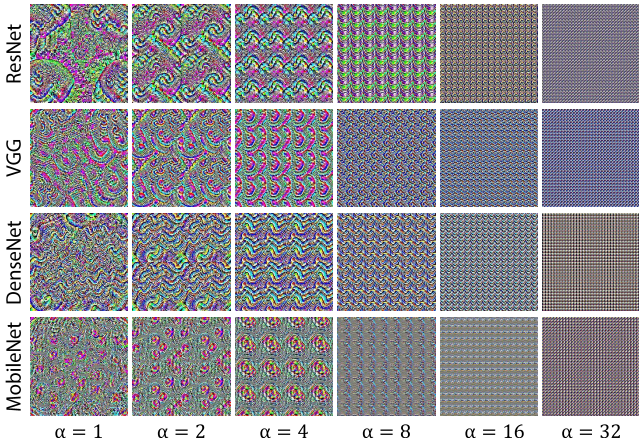

Fig. 5. Untargeted UAPs generated with different texture scale constraints (*i.e.*, α) on ImageNet training samples.

magnitude ϵ, this is a really small search space, which is not enough for generating UAP with a high fooling ratio. Thus we empirically suggest setting α around 8.

The generated UAPs are shown in Figure 5. We can find that with the increase of α, the UAPs include small-scale category-specific textures.

B. Improve Cross-Model Attack Transferability

Attack transferability is also an important metric for adversarial attacks. Due to the fact that CNN models pay more attention to local textures [49], we think the UAPs with relatively small texture scales may have better attack transferability. The results are in Table II. Please note that the values on the diagonal of the table are white-box testing results and are already shown in Table I.

The first two columns represent the UAPs generated by source models, which is the same as that in Table I. The first two rows represent the target models which need to attack. In each cell, there shows six fooling ratio (FR) values, corresponding to six different texture scales (*i.e.*, $\alpha = 1/2/4/8/16/32$). For example, using the ResNet as the source model and VGG as the target model, the fooling ratios of six different UAPs are 53.03%/59.45%/63.83%/80.13%/61.28%/73.13%. We can find that the highest fooling ratio is achieved with $\alpha = 8$ and is 27.1% higher than the baseline (*i.e.*, 53.03%), which is a significant increment. We use red color to mark the highest fooling ratio in each cell. In all the cells, the UAPs generated by TSC-UAP achieve the highest attack transferability, 28.8% increment at most and 21.44% on average. We also find the highest fooling ratios occur most when $\alpha = 8$, achieving 5 in 12 (*i.e.*, 41.6%). Please note that in the vast majority of cases, using texture scale constraints can achieve a higher fooling ratio, which shows that it can stably improve cross-model attack transferability.

C. Improve Targeted Attack

For targeted attacks, we conduct experiments on 5 randomly chosen target labels and for each label, the number of training images is 1,000. We apply two metrics (fooling ratio (FR) and target fooling ratio (TFR)). Compared with FR (as formulated in Eq. 2), TFR additionally requires the prediction label to be the same as the target label. That is,

$$\frac{|\{x \in \mathcal{X}_{test} : \hat{\mathcal{C}}(x+\delta) \neq \hat{\mathcal{C}}(x) \ \& \ \hat{\mathcal{C}}(x+\delta) = y_{target}\}|}{N}. \quad (6)$$

The reason for using FR is to evaluate the attack performance of targeted UAP as a "universal" perturbation. The reason for using TFR is to further evaluate the attack performance of targeted UAP as a "target" perturbation.

As shown in Table III, the first column represents the attack labels. The first and third rows represent the target models. In each cell of the first two rows, there are six fooling ratios, corresponding to six different texture scales $\alpha = 1/2/4/8/16/32$. The highest FRs are in red and exceeded the baseline by an average of 20.68%. Referring to the TFR evaluation, in each cell of the last two rows, there are also six target fooling ratios, corresponding to six different texture scales $\alpha = 1/2/4/8/16/32$. The highest TFRs are in red and exceeded the baseline by an average of 20.3%.

We can find that in all the cells of no matter FR or TFR evaluation, the highest fooling ratio is not achieved by the baseline, which means texture scale constraints also show satisfying effects on improving the targeted UAP task. Please note that in the vast majority of cases when $\alpha \leq 8$, texture scale constraints can stably improve the fooling ratio on targeted attacks, which is the same as on untargeted attacks.

D. Improve Cross-Data Attack Transferability

UAP is also capable of conducting cross-data attacking [50]. We conduct an additional experiment on three different classification datasets (*i.e.*, CIFAR10, CIFAR 100, and Place365) to see whether texture scale constraints improve the cross-data transferability. The UAPs are generated by attacking

TABLE III

WE RANDOMLY SELECT 5 TARGET ATTACK LABELS. THE FIRST AND THIRD ROWS REPRESENT THE TARGET MODELS. IN EACH CELL OF THE FIRST/LAST TWO ROWS, THERE ARE SIX FOOLING RATIOS/TARGET FOOLING RATIOS, CORRESPONDING TO SIX DIFFERENT TEXTURE SCALES α = 1/2/4/8/16/32. THE HIGHEST FOOLING RATIOS/TARGET FOOLING RATIOS ARE IN RED AND EXCEEDED THE BASELINE BY AN AVERAGE OF 20.68% AND 20.3% RESPECTIVELY

FR(α = 1/2/4/8/16/32)(%)	ResNet50	VGG19	DenseNet121	MobileNet-v2
echidna-102	22.23/58.75/47.75/55.99/41.10/37.60	63.45/78.78/72.55/66.46/56.37/70.88	43.50/65.06/61.78/44.55/39.10/32.05	54.14/64.33/67.47/56.78/62.82/57.91
starfish-327	46.63/69.52/64.35/61.43/65.66/40.50	67.40/82.71/83.24/73.14/61.40/63.95	49.13/61.91/66.28/55.79/49.99/42.38	74.55/78.60/69.12/76.63/63.68/62.92
golf ball-574	44.69/65.98/72.00/64.39/44.63/43.17	58.51/72.18/78.33/66.91/74.97/65.71	26.92/54.56/64.92/68.20/51.46/34.67	82.56/85.13/85.49/88.88/78.22/67.60
parachute-701	29.25/64.13/70.24/50.11/37.88/40.61	63.49/81.92/75.74/71.44/61.72/55.87	43.98/63.13/62.92/52.44/35.53/35.82	66.62/74.03/70.07/64.88/68.60/64.29
pineapple-953	51.12/74.07/73.83/69.88/65.83/45.07	52.15/80.03/79.48/73.83/67.94/65.50	35.35/60.27/66.47/60.75/49.15/42.81	87.85/91.29/92.13/91.59/76.98/68.01
TFR(α = 1/2/4/8/16/32)(%)	ResNet50	VGG19	DenseNet121	MobileNet-v2
echidna-102	0.10/30.96/2.82/0.15/0.13/0.10	32.91/50.20/38.63/7.07/0.29/0.08	17.47/43.09/20.27/3.55/0.13/0.22	8.08/17.90/2.31/0.35/0.12/0.12
starfish-327	22.38/37.52/13.80/6.06/1.13/0.18	38.96/55.68/52.00/14.86/4.26/1.36	29.54/36.43/49.40/18.36/0.75/0.15	36.50/47.77/27.18/7.69/3.55/1.17
golf ball-574	14.94/27.28/39.65/24.57/0.04/0.03	29.07/46.97/43.07/0.27/0.23/0.28	7.19/34.99/44.33/41.86/3.06/0.15	33.74/35.63/6.83/1.05/0.28/0.11
parachute-701	5.42/40.92/43.73/2.25/0.21/0.16	33.52/53.23/41.43/17.45/1.61/0.15	22.25/39.36/37.08/9.49/0.21/0.28	18.10/26.40/8.41/2.82/0.65/0.41
pineapple-953	21.11/42.77/35.12/31.03/1.19/0.01	6.13/35.69/29.57/0.80/0.05/0.01	12.81/38.33/43.14/23.73/1.00/0.01	47.82/60.65/40.99/5.38/0.16/0.02

TABLE IV

TARGET MODELS ARE IN THE FIRST ROW AND THE DATASETS ARE IN THE FIRST COLUMN. IN EACH CELL, THERE ARE SIX FOOLING RATIOS, CORRESPONDING TO SIX DIFFERENT TEXTURE SCALES α = 1/2/4/8/16/32. FOR CIFAR10, CIFAR100 AND PLACE365, THE HIGHEST FOOLING RATIO EXCEEDED THE BASELINE BY AN AVERAGE OF 15.32%, 25.37%, AND 17.93% RESPECTIVELY

FR(α = 1/2/4/8/16/32)(%)	ResNet50	VGG19	DenseNet121	MobileNet-v2
CIFAR10	46.52/66.93/75.01/88.92/64.11/6.05	89.33/89.64/89.68/89.04/87.69/88.04	69.43/81.86/84.74/86.41/84.01/6.55	87.58/89.16/88.21/85.38/85.14/82.72
CIFAR100	37.81/55.15/62.51/52.41/38.96/8.85	71.92/85.22/89.93/79.13/81.08/77.80	63.20/78.62/87.99/84.29/63.03/13.13	89.34/90.17/92.52/85.45/84.75/94.21
Place365	39.14/41.83/45.87/40.12/40.99/30.37	-	-	35.81/37.99/37.96/21.30/20.90/64.94

TABLE V

THE TSC-UAP WITHOUT TEXTURE SCALE CONSTRAINTS (*i.e.*, $\alpha = 1$) NEEDS 3789.236S TO PERFORM THE ATTACK WHILE OTHER TSC-UAP WITH α FROM 2 TO 32 NEED SIMILAR TIME AS IT. THE RUNNING TIME IS EVEN SHORTER THAN BASELINE WHEN α IS 8, 16, 32

time(s)	TSC-UAP						DF-UAP-rep	Cos-UAP-rep
	α = 1	α = 2	α = 4	α = 8	α = 16	α = 32		
ResNet	3789.236	3793.574	3795.700	3765.133	3776.235	3769.121	3792.637	3782.797

ImageNet towards four models (*i.e.*, ResNet50, VGG19, DenseNet121, MobileNet-v2) and testing on corresponding models in CIFAR10, CIFAR100, and Place365. Place365 only provides ResNet and MobileNet pre-trained models, thus we only show results on attacking them. Since the shape of the image in ImageNet is not the same as in CIFAR10, CIFAR100, and Place365, thus we generate UAP with ImageNet and resize UAP to the target shape.

As shown in Table IV, the target models are in the first row and the datasets are in the first column. In each cell, the highest fooling ratio is in red. We can find that the UAPs generated by using the texture scale constraints achieve the highest fooling ratios in all the cells. The results show the effect of TSC on improving the cross-data attack performance of UAP. For CIFAR10, CIFAR100 and Place365, the average increments are 15.32%, 25.37% and 17.93%. Please note that in most cases, using texture scale constraints can achieve a higher fooling ratio, which shows that texture scale constraints can stably improve cross-data attack transferability.

E. Time Comparison

To show the influence of texture scale constraints on the efficiency of the method, we evaluate the run time with different α. In Table V, the UAP generation method without texture scale constraints (*i.e.*, $\alpha = 1$) needs 3789.236 seconds to perform the attack while other TSC-UAP with α from 2 to 32 need similar time as it. The running time is even shorter when α is 8, 16, 32. The experiment results show that texture scale constraint is a tiny-cost UAP enhancement approach. We also compare the time with state-of-the-art UAP methods. Since DF-UAP and Cos-UAP do not provide the official code, we realize them based on SGD-UAP by simply adapting their special loss functions, termed DF-UAP-rep and Cos-UAP-rep respectively. We can find that the time of TSC-UAP is similar to DF-UAP-rep and Cos-UAP-rep.

F. Performance on Other Datasets

As shown in Table VI, we apply TSC-UAP on other datasets (*i.e.*, CIFAR10 and CIFAR100) and compare with state-of-the-art UAP methods on four classifiers (*i.e.*, ResNet50, VGG19, DenseNet121 and MobileNet-v2). Each cell shows the FR result. When TSC-UAP achieves the best fooling ratio, the values of α parameter are 8/8/8/2 in the CIFAR10 row and 4/4/4/4 in the CIFAR100 row for the ResNet50/VGG19/DenseNet121/MobileNet-v2. We can find that the α for achieving the best fooling ratio for each dataset is similar. On each model and dataset, we bold the highest fooling ratios and TSC-UAP always achieves the best fooling ratio.

TABLE VI
WE COMPARE TSC-UAP WITH SOTA UAP METHODS ON OTHER DATASETS. TSC-UAP ACHIEVES THE BEST FOOLING RATIO

FR(%)	ResNet50			VGG19			DenseNet121			MobileNet-v2		
	TSC-UAP	DF-UAP-rep	Cos-UAP-rep	TSC-UAP	DF-UAP-rep	Cos-UAP-rep	TSC-UAP	DF-UAP-rep	Cos-UAP-rep	TSC-UAP	DF-UAP-rep	Cos-UAP-rep
CIFAR10	**89.76**	89.72	81.91	**90.75**	90.73	88.56	**89.78**	89.74	87.06	**90.10**	90.05	88.05
CIFAR100	**98.41**	98.04	95.51	**98.84**	98.40	96.89	**98.92**	98.90	97.92	**98.75**	98.45	97.98

TABLE VII
THE INCREMENT OF TSC ON IMPROVING THE SOTA DATA-FREE UAP METHOD COS-UAP, AT AN AVERAGE OF 5.2%

FR(%)	AlexNet	GoogleNet	VGG16	VGG19	ResNet50	ResNet152	DenseNet121	MobileNet-v2
Cos-UAP-rep	96.58	75.51	92.51	85.52	87.75	84.33	76.33	95.38
TSC-Cos-UAP-rep	**96.86**	**90.95**	**92.56**	**92.09**	**91.02**	**88.68**	**84.55**	**98.91**

G. Improve Data-Free UAP Methods

The UAP methods can be classified into two types: data-dependent and data-free, according to whether need training samples. We have conducted experiments on the data-dependent UAP method. For data-free UAP methods, we choose the Cos-UAP as the baseline. To our best knowledge, it is the best data-free UAP method. We add texture scale constraints on Cos-UAP-rep to get TSC-Cos-UAP-rep. In Table VII, we conduct experiments on eight different classifiers. In each cell, there is the FR result and we bold the higher fooling ratios. We can find that the fooling ratios of TSC-Cos-UAP-rep are higher than that of Cos-UAP-rep in all the cases. The experiment shows TSC can also improve the data-free UAP methods, even the state-of-the-art ones. When TSC-Cos-UAP-rep achieves the best fooling ratio, the values of α parameter are 2/8/2/2/4/2/4/8 for the AlexNet/GoogleNet/VGG16/VGG19/ ResNet50/ResNet152/DenseNet121/MobileNet-v2. We can find that the α for achieving the best fooling ratio is among (2, 4, 8). Note that even if not achieving the best fooling ratio, the fooling ratios of α in (2, 4, 8) are still high. For example, in the GoogleNet column, when α is 2, 4, 8, the fooling ratios of TSC-Cos-UAP-rep are 86.31%/85.35%/90.95%. Although not achieving the best fooling ratio when α is 2 and 4, but the fooling ratios 86.31% and 85.35% are still much better than the fooling ratio (75.51%) of Cos-UAP-rep.

H. Improve L2-Norm UAP

Although existing UAP generation methods all follow the L_∞-norm, it would be interesting to explore whether our method can extend to other L_p-norm. Here we use L_2-norm as an example to show the result. As shown in Table VIII, we apply TSC-UAP towards four different target models (*i.e.*, ResNet50, VGG19, DenseNet121, MobileNet-v2) with six different texture scales (*i.e.*, $\alpha = 1/2/4/8/16/32$) under L_2-norm with $\epsilon = 40$. In each cell, there is the FR result and we bold the highest fooling ratios for each classifier. We can find that the TSC can improve the attack performance of the UAPs by an average of 24.83%, which is similar to the improvement under L_∞-norm. It is also interesting to see that most of the optimal value of α is 16, which is different from that under L_∞-norm. This observation not only shows different properties of the UAP under various L_p-norm bounds but also further verifies the effectiveness and generality of TSC.

TABLE VIII
THE PERFORMANCE OF TSC UNDER L2-BOUNDED ATTACK

α \ Model	ResNet50	VGG19	DenseNet121	MobileNet-v2
1	66.49	74.99	52.38	86.92
2	81.41	94.25	60.59	95.49
4	91.82	94.77	86.10	99.04
8	91.96	96.33	86.88	99.16
16	**93.33**	**97.74**	84.71	**99.61**
32	90.66	93.60	**89.42**	95.51

I. Compare to SOTA UAP Methods

DF-UAP [43], Cos-UAP [44], TRM-UAP [52], and AT-UAP [51] are the state-of-the-art UAP methods. To make a fair comparison, we refer to the size of their training dataset. That is, the training dataset is of size 10,000 (by sampling 10 images for each class in the ImageNet training dataset). The comparison is on the ImageNet validation set towards five different models they used (*i.e.*, AlexNet, GoogleNet, VGG16, VGG19, ResNet152).

As shown in Table IX, we bold the highest fooling ratios. We can find that all the UAP methods achieve significantly high fooling ratios, almost achieving more than 90% on all the models. Among them, TSC-UAP obtains the best fooling ratio on every model. When TSC-UAP achieves the best fooling ratio, the value of α parameter are 4/8/4/4/4 for the AlexNet/GoogleNet/VGG16/VGG19/ResNet152. We can find that the α for achieving the best fooling ratio is similar.

J. Data-Efficiency of TSC-UAP

The standard UAP generation methods collect gradient information from a large number of training samples to update the UAP pattern. Since we reduce the update area, similar to sparse representation, there may need fewer training samples to generate UAPs. To verify this, we conduct an experiment to evaluate the data efficiency of the UAP method enhanced by

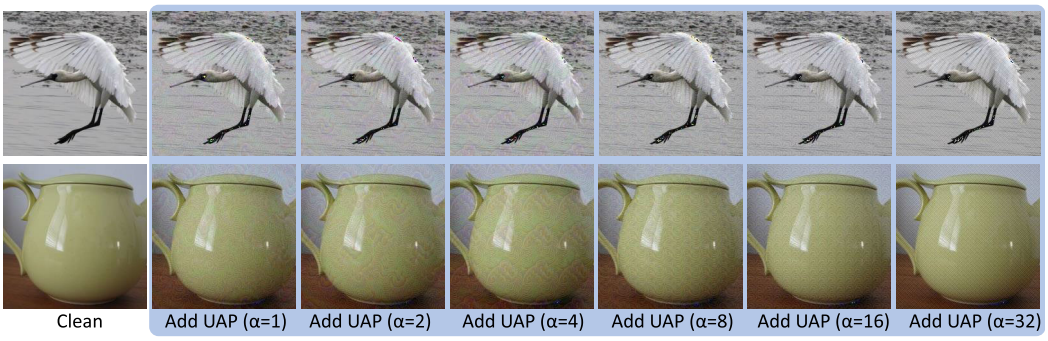

Fig. 6. Two clean images and their perturbed version by adding UAPs of different α on them.

TABLE IX

WE COMPARE TSC-UAP WITH SOTA UAP METHODS. OUR METHOD ACHIEVES BETTER ATTACK PERFORMANCE THAN DF-UAP, COS-UAP, TRM-UAP AND AT-UAP

FR(%)	AlexNet	GoogleNet	VGG16	VGG19	ResNet152	Average
TSC-UAP	96.78	**91.17**	**97.64**	97.51	**92.70**	**95.16**
DF-UAP	96.17	88.94	94.30	94.98	90.08	92.89
Cos-UAP	96.50	90.50	97.40	96.40	90.20	94.20
TRM-UAP	93.53	85.32	94.30	91.35	67.46	86.39
AT-UAP	**97.01**	90.82	97.51	**97.56**	91.52	94.88

texture scale constraints. Please note that for data-free UAP methods, there is no space for improvement, thus we only show the advance in data-dependent UAP methods.

As shown in Table X, we use ResNet 50 as the target model and the first column means the texture scale. The first row means the size of the total training samples. The second row means the classes and number-per-class chosen by us, recorded briefly in format (c,n). For example, when the total number of training samples is 100, we simply exploit three ways to choose the samples. First, we can choose 1 class (*i.e.*, c = 1) with 100 samples (*i.e.*, n = 100), expressed as (1,100). Second, we can choose 10 classes (*i.e.*, c = 10) with 10 samples each (*i.e.*, n = 10), expressed as (10,10). Third, we can choose 100 classes (*i.e.*, c = 100) with 1 sample each (*i.e.*, n = 1), expressed as (100,1).

We can find that the fooling ratios achieve around 80% when the size of the training set is 1,000. However, by using the texture scale constraints, at a tenth of the size we can achieve at most 87.78% fooling ratio. Furthermore, with only ten images, we can improve the fooling ratio of the standard UAP method from 22.03% to 71.72%, which is close to the result of using 1,000 training images with the standard UAP method. The experiment results show the data efficiency of using the texture scale constraints.

K. Visualization of Perturbed Images

We apply UAPs on clean images and visualize them in Figure 6. Here we randomly select two clean images from ImageNet and add UAPs to them. To be specific, we use the six UAPs (with different α) generated with ResNet (*i.e.*, the UAPs in the first row of Figure 5). It is observed that these perturbations do not alter human classification ability, *i.e.*, the crane and cups remain distinctly recognizable.

VI. DISCUSSION

A. Advantages of TSC-UAP

According to the experiment results, we can find that texture scale constraints have four advantages: ❶ Enhance fooling ratio: this is the main metric of UAP and TSC significantly improves the performance. ❷ Enhance attack transferability: TSC-UAP generates UAPs of higher cross-model transferability and cross-data transferability, which further enhance the black-box ability of UAP. ❸ Low cost: TSC is almost no extra overhead and may reduce the calculated amount since the update only applies to local texture, which is smaller than on the whole UAP. ❹ High generality: TSC can be utilized to improve both data-dependent and data-free UAP generation methods. Our work is a practical and preliminary endeavor intended to substantiate the significance of a new research direction in UAP generation through empirical study. The contribution is proposing the huge influence of texture on UAP generation and the proposed TSC is a simple and reasonable way to exploit the issue.

B. Tansferability Between CNNs and ViT

We conduct experiment of TSC-UAP on ViT [63] and evaluate the transferability between CNN and ViT in Table XI and Table XII. In Table XI, we evaluate the transferability of UAPs generated by ResNet50, VGG19, DenseNet121 and MobileNet-v2 to the ViT-B/16 model. We can find that the attack performance of these UAPs is not good on ViT but TSC-UAP can still achieve higher fooling ratios than UAP without TSC constraint (*i.e.*, $\alpha = 1$). In Table XII, we evaluate the transferability of UAPs generated by ViT-B/16 to ResNet50, VGG19, DenseNet121 and MobileNet-v2 models. First, TSC-UAP is effective on ViT since when $\alpha = 2, 4, 8, 16, 32$, the fooling ratios are all higher than UAP without TSC constraint. Also, we can find that the attack performance of these UAPs is high on CNNs, even higher than that on ViT, which means attacking ViT to generate UAP is harder than attacking CNN.

C. Effect of UAP Blocks

The positions of the UAP patches are significant toward attack performance. Here we take TSC-UAP with different α as examples. In Figure 7, we use the TSC-UAP with $\alpha = 2$ and generate four new UAPs (UAP_tl, UAP_bl, UAP_tr, UAP_br) which only reserve one of the four UAP patches. We use the

TABLE X

THE FIRST COLUMN MEANS THE TEXTURE SCALE PARAMETER α. THE FIRST ROW MEANS THE SIZE OF THE TOTAL TRAINING SAMPLES. THE SECOND ROW MEANS THE CLASSES AND NUMBER-PER-CLASS CHOSEN BY US, RECORDED BRIEFLY IN FORMAT (C,N)

FR(%)	1	10		100			1000			
	(1,1)	(1,10)	(10,1)	(1,100)	(10,10)	(100,1)	(1,1000)	(10,100)	(100,10)	(1000,1)
1	19.85	21.44	22.03	22.90	25.77	23.21	78.98	75.00	76.60	80.30
2	19.30	22.02	23.33	28.24	70.56	26.45	86.70	87.30	88.07	87.22
4	21.60	24.77	41.80	74.41	78.81	82.11	90.52	91.65	90.22	89.92
8	23.83	28.82	71.72	87.71	87.78	80.01	93.36	93.28	93.85	92.73
16	28.25	55.93	37.56	81.33	75.61	75.05	80.26	81.25	73.58	78.47
32	31.60	56.18	43.03	46.48	44.28	44.28	52.11	46.49	52.54	53.96

TABLE XI

TRANSFERABILITY FROM UAPS GENERATED BY CNNS TO VIT (*i.e.*, RESNET50, VGG19, DENSENET121, MOBILENET-V2, VIT-B/16) ARE IN THE SECOND ROW AND THE TEXTURE SCALE PARAMETER α IS IN THE FIRST COLUMN. EACH CELL SHOWS THE FOOLING RATIO

Model / α	Source	Target	Source	Target	Source	Target	Source	Target
	ResNet50	ViT-B/16	VGG19	ViT-B/16	DenseNet121	ViT-B/16	MobileNet-v2	ViT-B/16
1	80.30	17.34	82.44	16.97	66.05	18.96	94.08	14.20
2	87.22	18.82	90.37	18.53	78.83	20.94	97.01	15.36
4	89.92	20.99	92.88	20.74	82.26	22.22	97.25	16.49
8	92.73	23.14	86.57	18.78	85.18	23.53	99.19	10.73
16	78.47	16.97	83.54	18.42	79.26	20.49	97.53	10.18
32	53.96	14.81	79.39/	17.31	51.24	23.87	74.65	22.30

TABLE XII

TRANSFERABILITY FROM UAPS GENERATED BY VIT TO CNN (*i.e.*, RESNET50, VGG19, DENSENET121, MOBILENET-V2, VIT-B/16) ARE IN THE SECOND ROW AND THE TEXTURE SCALE PARAMETER α IS IN THE FIRST COLUMN. EACH CELL SHOWS THE FOOLING RATIO

Model / α	Source	Target			
	ViT-B/16	ResNet50	VGG19	DenseNet121	MobileNet-v2
1	21.43	21.69	31.05	22.03	31.65
2	76.78	35.93	50.28	36.55	49.17
4	42.63	43.82	55.60	43.80	55.78
8	37.21	48.70	61.56	50.48	56.37
16	31.32	34.99	57.19	35.83	55.75
32	24.53	36.62	57.84	26.14	50.78

same experimental setting as in Table I of our original paper except replacing UAP with new UAP. In Table XIII, we can find that each patch achieves similar attack performance. We also conduct experiments on TSC-UAP with $\alpha = 4$. In Figure 8, we either reserve the center or the round of the UAP to generate some new UAPs. In Table XIII, we can find that UAP with only center patches reserved achieves higher FR than around UAP patches which have the same perturbation quantity. We think this is because the objects in images usually exist in the center and UAP_center is more likely to mask the object in the image, which makes the classifier focus on the UAP patch and misclassify the image.

D. TSC-UAP Against Defense Method

We conduct experiment against UAP defense methods [53]. It is the classical UAP defense method and according to our investigation, it is the only one that open-sources the code and provides the defense model. Please note that the defense model

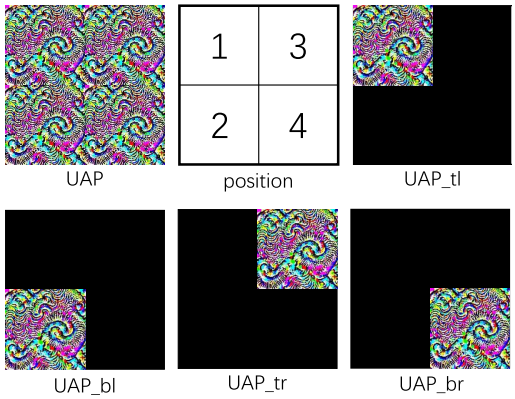

Fig. 7. Influence of different UAP blocks (corners). Here we choose TSC-UAP with $\alpha = 2$ and clip it into four UAP blocks.

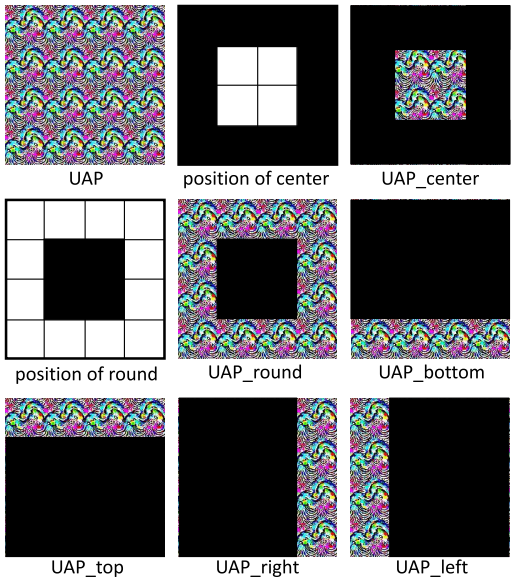

Fig. 8. Influence of different UAP blocks (round and center). Here we choose TSC-UAP with $\alpha = 4$ and clip it into different UAP blocks.

provided by [53] can choose to turn on or off the defense. Thus fooling ratios before/after defense are both achieved on the same defense model. We use the UAPs generated in Table I as input since the UAPs with $\alpha = 1$ can represent the UAPs without TSC while the others are generated with our TSC-UAP. As shown in Table XIV, UAP source models (*i.e.*, ResNet50, VGG19, DenseNet121, MobileNet-v2) are in the first row and the texture scale parameter α is in the first

TABLE XIII
EVALUATION OF UAP PATCHES (DIFFERENT POSITIONS) ON VARIOUS CNNS

FR (%)	ResNet50	VGG19	DenseNet121	MobileNet-v2
UAP (α = 2)	87.22	90.37	78.83	97.01
UAP_tl (α = 2)	23.28	36.88	21.67	31.42
UAP_bl (α = 2)	19.88	31.10	19.52	29.89
UAP_tr (α = 2)	21.30	37.39	21.60	30.28
UAP_br (α = 2)	18.53	31.41	19.36	28.17
UAP (α = 4)	89.92	92.88	82.26	97.25
UAP_center (α = 4)	33.46	43.36	26.55	48.17
UAP_round (α = 4)	75.20	81.18	65.53	79.09
UAP_top (α = 4)	21.43	34.01	19.73	25.26
UAP_right (α = 4)	23.07	30.75	19.70	28.18
UAP_bottom (α = 4)	19.29	31.13	18.38	24.27
UAP_left (α = 4)	23.55	33.53	20.08	30.65

TABLE XIV
FOR THE PERFORMANCE OF TSC-UAP AGAINST DEFENSE METHODS, UAP SOURCE MODELS (*i.e.*, RESNET50, VGG19, DENSENET121, MOBILENET-V2) ARE IN THE FIRST ROW AND THE TEXTURE SCALE PARAMETER α IS IN THE FIRST COLUMN. IN EACH CELL, THE FOOLING RATIOS BEFORE/AFTER DEFENSE ARE ON THE LEFT/RIGHT AND THE DECREMENT IS IN THE PARENTHESIS

α \ Model	ResNet50	VGG19	DenseNet121	MobileNet-v2
1	40.05/30.02(10.03)	33.60/27.32(6.28)	43.08/35.27(7.81)	30.38/22.65(7.73)
2	45.02/36.51(8.51)	41.05/31.64(9.41)	51.01/41.67(9.34)	34.44/25.46(8.98)
4	45.62/37.08(8.54)	46.92/35.32(11.60)	52.57/43.92(8.65)	39.03/27.88(11.15)
8	63.80/41.36(22.44)	43.34/34.76(8.58)	62.57/51.18(11.39)	23.97/19.29(4.68)
16	48.54/36.37(12.17)	35.49/29.47(6.02)	59.76/46.66(13.10)	21.74/18.88(2.86)
32	33.63/22.47(11.16)	35.90/26.04(9.86)	57.41/50.92(6.49)	44.67/32.45(12.22)

column. In each cell, the fooling ratios before/after defense are on the left/right and the decrement is in the parenthesis. We can observe two points. ❶ From the attack performance of UAPs on the model provided by [53] without defense, we can find that the TSC constraint is effective. ❷ We can find that for UAPs generated by the same source model with different α, the decrement is similar. To summarize, the performance of TSC-UAP against the defense model is similar to the UAP without TSC constraint. However, we have to emphasize that TSC-UAP can achieve higher fooling ratios, thus with similar decrement, TSC-UAP can still achieve better attack performance than UAP without TSC constraint. Since TSC-UAP is not designed against defense methods, we think the result is reasonable and we will focus on designing the UAP generation method against the defense model in future work.

E. Potential Application

It would be very interesting if UAP could be applied to a potential application in other domains (*e.g.*, adversarial patch, malware detection). Here we list some potential ideas for applying UAP in different domains. The HARP [64] is a newly proposed model-side defense method against adversarial patches by only inserting lightweight CNN modules into the pre-trained object detectors. Compared with the adversarial patches mentioned in the paper, we think UAP is different from the two points. ❶ The size of UAP (*i.e.*, same size as the input image) is much bigger than the adversarial patch and would fully mask the object. ❷ The perturbation of UAP is not as big as that in the adversarial patch. It would be interesting to evaluate and enhance the robustness of object detectors against UAP. BagAmmo [65] is a newly proposed novel black-box attack method against Function Call Graph (FCG) based Android malware detection systems. They proposed a new malware manipulation called "try-catch trap". We think taking malware manipulation as a kind of perturbation, maybe designing a "universal" malware manipulation method that can be effective at any place of FCG is interesting and practical. Reference [66] comprehensively survey malicious PDF detection in adversarial environments, inspiring us that maybe designing a "universal" content that can fool malicious PDF detectors is an interesting application.

F. Theoretical Analysis of TSC-UAP

Taking UAP generation as a search problem, then the gradient for optimizing UAP is the guidance for the search. If directly optimizing all the pixels in UAP with gradient, the search space is too big. It is hard to force all the meaningless noises into category-specific textures and usually parts region of the UAP still exist meaningless noises (see images with $\alpha = 1$ (without TSC constraint) in Figure 4). By using split ratio $\alpha > 1$ to reduce the region in UAP that needs optimization (see our method Figure 3), the difficulty of optimization is reduced (search space becomes small) since our method only needs to search category-specific texture in the local area. With generated category-specific texture in the local area, the problem becomes how to make it exist in the entire UAP. To address the problem, patch tiling is a good choice for this goal. However, please note that patch tiling is not the only choice since other operations (*e.g.*, flip) can also achieve the goal of making category-specific texture exist in the entire UAP and thus enhance UAP.

G. Limitations of TSC-UAP

Although we point out the huge influence of texture on the attack performance of UAP, the way of exploiting such an issue may not be most appropriate. It would be better to give a theoretical analysis that can further promote the development of research on how to exploit the texture of UAP. However, we firmly believe that this practical study is essential and serves as a valuable starting point for theoretical analysis. Given the complexity of the matter, which cannot be fully elucidated by solely focusing on texture scale since other factors (*e.g.*, texture category, the architecture of the target model, and training dataset) are also important, we consider theoretical analysis needs more observations and more appropriate for the future work.

VII. CONCLUSION

In this paper, we research the textures in universal adversarial perturbation and propose texture scale constraints to improve the UAP methods. The proposed texture scale constraints not only achieve a higher fooling ratio but also higher

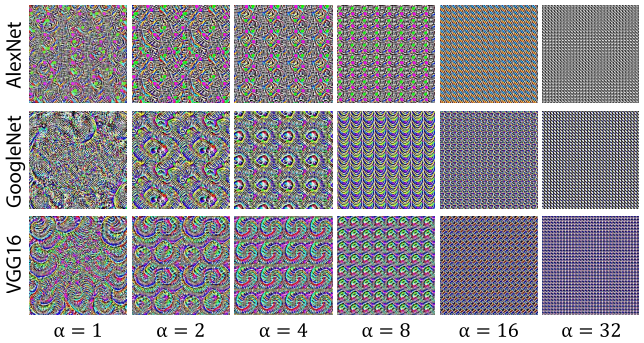

Fig. 9. Here shows the UAPs generated by attacking Alexnet, GoogleNet and VGG16 on ImageNet training samples.

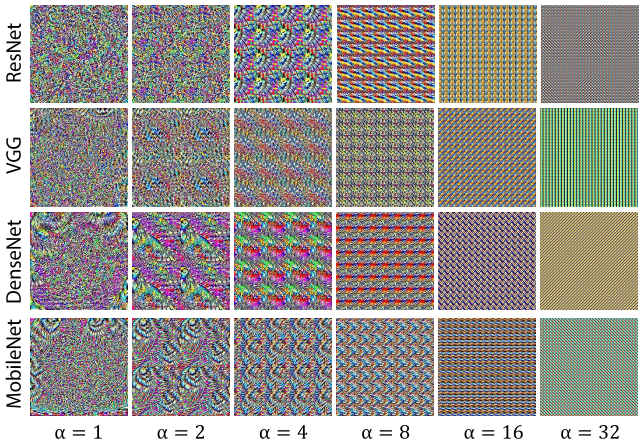

Fig. 10. Here shows the UAPs generated by attacking ResNet50, VGG19, DenseNet121 and MobileNet-v2 on ImageNet training samples with target label sea lion.

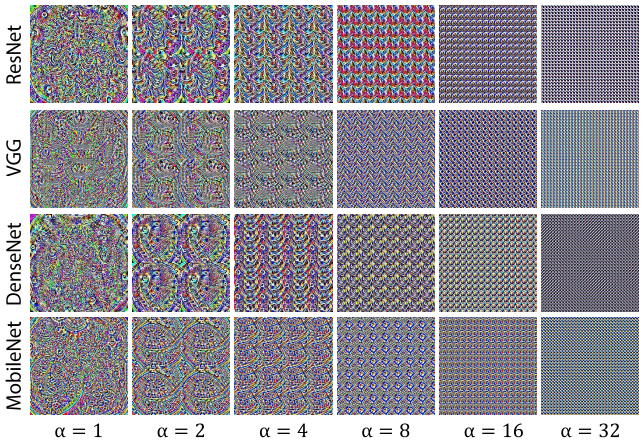

Fig. 11. Here shows the UAPs generated by attacking ResNet50, VGG19, DenseNet121 and MobileNet-v2 on ImageNet training samples with target label shield.

attack transferability with minor computational costs. It is also general enough to be applied to both data-dependent and data-free UAP methods. In future work, we aim to study the relationship between the texture scale and the receptive field of CNNs, which may be instructive for model explainability.

APPENDIX

A. Visualization of Untargeted UAPs Across Different Models

We show the untargeted UAPs on more CNNs (*i.e.*, AlexNet, GoogleNet, VGG16) in Figure 9. We can find that, the texture on different models all seems special when α is not big and shows some semantic in the texture.

B. Visualization of Targeted UAPs Across Different Models

We show the targeted UAPs in Figure 10 and Figure 11 by attacking ResNet50, VGG19, DenseNet121, and MobileNet-v2 on ImageNet training samples with randomly chosen target labels (sea lion and shield) respectively. We can find that there exist texture of sea lion and shield in Figure 10 and 11 respectively, especially when α is 2.

ACKNOWLEDGMENT

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Cyber Security Agency of Singapore.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[2] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.
[3] X. Zheng, H. Sun, X. Lu, and W. Xie, "Rotation-invariant attention network for hyperspectral image classification," *IEEE Trans. Image Process.*, vol. 31, pp. 4251–4265, 2022.
[4] Y. Dong, Q. Liu, B. Du, and L. Zhang, "Weighted feature fusion of convolutional neural network and graph attention network for hyperspectral image classification," *IEEE Trans. Image Process.*, vol. 31, pp. 1559–1572, 2022.
[5] H. Touvron et al., "ResMLP: Feedforward networks for image classification with data-efficient training," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 4, pp. 5314–5321, Apr. 2023.
[6] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.
[7] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10781–10790.
[8] C.-Y. Wang, A. Bochkovskiy, and H.-Y.-M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 7464–7475.
[9] Y. Zhou et al., "MMRotate: A rotated object detection benchmark using PyTorch," in *Proc. 30th ACM Int. Conf. Multimedia*, Oct. 2022, pp. 7331–7334.
[10] G. Cheng et al., "Towards large-scale small object detection: Survey and benchmarks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 11, pp. 13467–13488, Nov. 2023.
[11] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2961–2969.
[12] Z. Liu et al., "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 10012–10022.
[13] T. Lüddecke and A. Ecker, "Image segmentation using text and image prompts," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 7076–7086.
[14] N. Kim, D. Kim, S. Kwak, C. Lan, and W. Zeng, "ReSTR: Convolution-free referring image segmentation using transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 18145–18154.
[15] Y. Ji et al., "Amos: A large-scale abdominal multi-organ benchmark for versatile medical image segmentation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 36722–36732.
[16] C. Szegedy et al., "Intriguing properties of neural networks," in *Proc. Int. Conf. Learn. Represent.*, 2014, pp. 1–10.
[17] W. Nie, B. Guo, Y. Huang, C. Xiao, A. Vahdat, and A. Anandkumar, "Diffusion models for adversarial purification," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2022, pp. 1–22.

[18] C. Schlarmann and M. Hein, "On the adversarial robustness of multi-modal foundation models," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2023, pp. 3677–3685.

[19] Z. Yang, T. Pang, and Y. Liu, "A closer look at the adversarial robustness of deep equilibrium models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 10448–10461.

[20] X. Jia et al., "Improving fast adversarial training with prior-guided knowledge," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 9, pp. 6367–6383, Sep. 2024.

[21] X. Jia, J. Li, J. Gu, Y. Bai, and X. Cao, "Fast propagation is better: Accelerating single-step adversarial training via sampling subnetworks," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 4547–4559, 2024.

[22] Y. Huang et al., "AdvFilter: Predictive perturbation-aware filtering against adversarial attack via multi-domain learning," in *Proc. 29th ACM Int. Conf. Multimedia*, 2021, pp. 395–403.

[23] Y. Huang, F. Juefei-Xu, Q. Guo, G. Pu, and Y. Liu, "Natural & adversarial bokeh rendering via circle-of-confusion predictive network," *IEEE Trans. Multimedia*, vol. 26, pp. 5729–5740, 2024.

[24] Y. Huang et al., "Personalization as a shortcut for few-shot backdoor attack against text-to-image diffusion models," in *Proc. AAAI Conf. Artif. Intell.*, 2024, vol. 38, no. 19, pp. 21169–21178.

[25] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.

[26] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," 2016, *arXiv:1611.01236*.

[27] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 39–57.

[28] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 2206–2216.

[29] Y. Bai, J. Mei, A. L. Yuille, and C. Xie, "Are transformers more robust than CNNs?" in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 26831–26843.

[30] W. Feng, N. Xu, T. Zhang, B. Wu, and Y. Zhang, "Robust and generalized physical adversarial attacks via meta-GAN," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 1112–1125, 2024.

[31] A. Goel and P. Moulin, "Fast locally optimal detection of targeted universal adversarial perturbations," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 1757–1770, 2022.

[32] Y. Wang, M. Zhao, S. Li, X. Yuan, and W. Ni, "Dispersed pixel perturbation-based imperceptible backdoor trigger for image classifier models," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 3091–3106, 2022.

[33] S. Zhao, T. Xu, X.-J. Wu, and J. Kittler, "Pluggable attack for visual object tracking," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 1227–1240, 2024.

[34] H. Zhang, Y. Avrithis, T. Furon, and L. Amsaleg, "Walking on the edge: Fast, low-distortion adversarial examples," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 701–713, 2021.

[35] Y. Huang et al., "ALA: Naturalness-aware adversarial lightness attack," in *Proc. 31st ACM Int. Conf. Multimedia*, Oct. 2023, pp. 2418–2426.

[36] D. Han, X. Jia, Y. Bai, J. Gu, Y. Liu, and X. Cao, "OT-attack: Enhancing adversarial transferability of vision-language models via optimal transport optimization," 2023, *arXiv:2312.04403*.

[37] S. Gao, X. Jia, X. Ren, I. Tsang, and Q. Guo, "Boosting transferability in vision-language attacks via diversification along the intersection region of adversarial trajectory," 2024, *arXiv:2403.12445*.

[38] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1765–1773.

[39] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2574–2582.

[40] C. Zhang, P. Benz, C. Lin, A. Karjauv, J. Wu, and I. S. Kweon, "A survey on universal adversarial attack," in *Proc. 13th Int. Joint Conf. Artif. Intell.*, Z.-H. Zhou, Ed. Aug. 2021, pp. 4687–4694, doi: 10.24963/ijcai.2021/635.

[41] O. Poursaeed, I. Katsman, B. Gao, and S. Belongie, "Generative adversarial perturbations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4422–4431.

[42] K. R. Mopuri, U. Ojha, U. Garg, and R. V. Babu, "NAG: Network for adversary generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 742–751.

[43] C. Zhang, P. Benz, T. Imtiaz, and I. S. Kweon, "Understanding adversarial examples from the mutual influence of images and perturbations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 14521–14530.

[44] C. Zhang, P. Benz, A. Karjauv, and I. S. Kweon, "Data-free universal adversarial perturbation and black-box attack," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2021, pp. 7868–7877.

[45] A. Shafahi, M. Najibi, Z. Xu, J. P. Dickerson, L. S. Davis, and T. Goldstein, "Universal adversarial training," in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, pp. 5636–5643.

[46] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*.

[47] I. Oseledets and V. Khrulkov, "Art of singular vectors and universal adversarial perturbations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8562–8570.

[48] K. R. Reddy, U. Garg, and V. B. Radhakrishnan, "Fast feature fool: A data independent approach to universal adversarial perturbations," in *Proc. Brit. Mach. Vis. Conf.*, 2017, pp. 1–12.

[49] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "ImageNet-trained CNNs are biased towards texture; Increasing shape bias improves accuracy and robustness," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–22.

[50] H. Liu et al., "Universal adversarial perturbation via prior driven uncertainty approximation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2019, pp. 2941–2949.

[51] M. Li, Y. Yang, K. Wei, X. Yang, and H. Huang, "Learning universal adversarial perturbation by adversarial example," in *Proc. AAAI Conf. Artif. Intell.*, 2022, vol. 36, no. 2, pp. 1350–1358.

[52] Y. Liu, X. Feng, Y. Wang, W. Yang, and D. Ming, "TRM-UAP: Enhancing the transferability of data-free universal adversarial perturbation via truncated ratio maximization," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2023, pp. 4762–4771.

[53] N. Akhtar, J. Liu, and A. Mian, "Defense against universal adversarial perturbations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3389–3398.

[54] C. K. Mummadi, T. Brox, and J. H. Metzen, "Defending against universal perturbations with shared adversarial training," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4927–4936.

[55] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 618–626.

[56] C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.

[57] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[58] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 4700–4708.

[59] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[60] O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[61] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," M.S. thesis, Univ. Toronto, Toronto, ON, Canada, 2009.

[62] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1452–1464, Jun. 2018.

[63] A. Dosovitskiy et al., "An image is worth 16×16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–22. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy

[64] J. Cai, S. Chen, H. Li, B. Xia, Z. Mao, and W. Yuan, "HARP: Let object detector undergo hyperplasia to counter adversarial patches," in *Proc. 31st ACM Int. Conf. Multimedia*, Oct. 2023, pp. 2673–2683.

[65] H. Li et al., "Black-box adversarial example attack towards FCG based Android malware detection under incomplete feature information," in *Proc. 32nd USENIX Secur. Symp.*, 2023, pp. 1181–1198.

[66] D. Maiorca, B. Biggio, and G. Giacinto, "Towards adversarial malware detection: Lessons learned from PDF-based attacks," *ACM Comput. Surv.*, vol. 52, no. 4, pp. 1–36, Jul. 2020.

Yihao Huang received the B.S. and Ph.D. degrees from East China Normal University, China, in 2017 and 2022, respectively. He is currently a Research Fellow with Nanyang Technological University, where he works on the security of multi-modal content. His research interests include computer vision and AI security, especially adversarial attacks and generative AI. He was a recipient of the Best Paper Award at the ECCV 2022 AROW Workshop.

Qing Guo (Member, IEEE) received the Ph.D. degree in computer application technology from the School of Computer Science and Technology, Tianjin University, China. He is currently a Senior Research Scientist and the Principal Investigator (PI) with the Center for Frontier AI Research (CFAR), A*STAR, Singapore. He is also an Adjunct Assistant Professor with the National University of Singapore (NUS), Singapore, and a Senior PC Member of AAAI. Before that, he was a Wallenberg-NTU Presidential Post-Doctoral Fellow with Nanyang Technological University. His research interests include computer vision, AI security, and image processing.

Felix Juefei-Xu (Member, IEEE) received the B.S. degree in electronic engineering from Shanghai Jiao Tong University (SJTU), Shanghai, China, the M.S. degree in electrical and computer engineering and the M.S. degree in machine learning from CMU, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University (CMU), Pittsburgh, PA, USA. Currently, he is a Research Scientist with GenAI at Meta, New York, where he works on robust perception and efficient learning problems in the domain of generative AI. He is also affiliated with New York University as an Adjunct Professor. Previously, he was a Research Scientist with Alibaba Group, Sunnyvale, CA, USA. He was a recipient of multiple best/distinguished paper awards, including IJCB 2011, BTAS 2015 and 2016, ASE 2018, and ACCV 2018.

Ming Hu (Member, IEEE) received the B.E. and Ph.D. degrees from the Software Engineering Institute, East China Normal University, Shanghai, China, in 2017 and 2022, respectively. He is currently a Research Scientist with Singapore Management University. Previously, he was a Research Fellow with Nanyang Technological University (NTU), Singapore. His research interests include the area of design automation of cyber-physical systems, federated learning, trustworthy AI, and software testing.

Xiaojun Jia received the Ph.D. degree from the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, and the School of Cyber Security, University of Chinese Academy of Sciences, Beijing. He is currently a Research Fellow with the Cyber Security Research Centre @ NTU, Nanyang Technological University, Singapore. His research interests include computer vision, deep learning, and adversarial machine learning.

Xiaochun Cao (Senior Member, IEEE) received the B.S. and M.S. degrees in computer science from Beihang University, Beijing, China, and the Ph.D. degree in computer science from the University of Central Florida, Orlando, FL, USA. After graduation, he spent about three years at ObjectVideo Inc., as a Research Scientist. He is currently with the School of Cyber Science and Technology, Sun Yat-sen University, Shenzhen Campus, Shenzhen, China. He has authored and coauthored more than 100 journals and conference papers. He is a fellow of IET. His dissertation was nominated for the University of Central Florida's University-Level Outstanding Dissertation Award. In 2004 and 2010, he was a recipient of the Piero Zamperoni Best Student Paper Award at the International Conference on Pattern Recognition. He is on the editorial boards of IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE TRANSACTIONS ON MULTIMEDIA, and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY.

Geguang Pu received the B.S. degree in mathematics from Wuhan University in 2000 and the Ph.D. degree in mathematics from Peking University in 2005. He is currently a Professor with the Software Engineering Institute, East China Normal University. He has published over 100 publications on the topics of software engineering and system verification (including ICSE, FSE, ASE, and CAV). His research interests include program testing and reliable AI systems. He served as a PC member for more than 20 international conference committees.

Yang Liu (Senior Member, IEEE) received the Bachelor of Computing (Hons.) and Ph.D. degrees from the National University of Singapore (NUS) in 2005 and 2010, respectively. He started his post-doctoral work at MIT. In 2012, he joined Nanyang Technological University (NTU), where he is currently a Full Professor and the Director of the Cybersecurity Laboratory. He specializes in software engineering, cybersecurity, and artificial intelligence. He has more than 400 publications in top-tier conferences and journals. His research interests include the theory and practical usage of program analysis, data analysis, and AI to evaluate the design and implementation of software for high assurance and security. He has received several prestigious awards, including the MSRA Fellowship, the TRF Fellowship, the Nanyang Assistant Professor, the Tan Chin Tuan Fellowship, the Nanyang Research Award in 2019, an ACM Distinguished Speaker, the NRF Investigatorship, and 15 best paper awards and one most influence system award in top software engineerings conferences, such as ASE, FSE, and ICSE.