CAA: Toward Camouflaged and Transferable Adversarial Examples

Yipeng Zou, Qin Liu, *Member, IEEE,* Jie Wu, *Fellow, IEEE,* Tian Wang, *Member, IEEE,* Guo Chen, *Member, IEEE,* Tao Peng, *Member, IEEE,* Guojun Wang, *Member, IEEE*

***Abstract*—Transferable adversarial examples (AEs) are visually indistinguishable from benign images, but can successfully mislead unknown deep neural networks. However, existing AEs normally vary considerably from benign images in the feature space, making them hard to pass label checking and adversarial detection. Therefore, how to make AEs camouflaged, disguising as benign images during detection is still an open problem. In this paper, we propose a novel camouflaged adversarial attack (CAA), which produces camouflaged adversarial examples (CAEs) for the first time. Our main idea is to make CAEs' adversarial properties keep "dormant" state until the target model inadvertently triggers the "activated" state. To this end, we craft *attack* and *camouflage* perturbations, so that CAEs are visually and feature/label-wise indistinguishable from benign images at first, but will implicitly turn into AEs once being triggered. Specifically, we exploit two common preprocessing operations, image scaling and JPEG compression, as the trigger, and propose a two-stage optimization strategy. As the preprocessing details of target models are unknown, the first stage trains a well-designed generative adversarial network under varying scaling/compression parameters to enhance the robustness of attack perturbations. The second stage uses feature (dis)similarities and contrastive distances to improve the transferability of camouflage perturbations. Extensive experiments on ImageNet dataset validate the effectiveness of CAA. Especially for robust models, the average fooling rate after preprocessing could reach 96.3% outperforming the state-of-the-art adversarial attack by 13.5%.**

***Index Terms*—Deep neural networks, adversarial attack, camouflaged adversarial example, transferability, imperceptibility.**

I. Introduction

DEEP neural networks (DNNs) are highly vulnerable to adversarial examples (AEs) [1], which can mislead models by adding imperceptible perturbations on benign images. In particular, AEs are cross-model transferable, posing security threats to real-world applications, e.g., face recognition [2], autonomous driving [3], or conversely enabling privacy-preserving inference [4]. An effective defense against AEs is to deploy adversarial detectors like feature squeezing [5] at the front end to filter out abnormal samples before feeding models. Due to the adversarial properties, AEs normally show large differences from benign images in the feature space, and can be easily caught by label checking or adversarial detection [6]. Only after passing detection can AEs carry out attacks, and thus enabling AEs to conceal adversarial properties and disguise as benign images during detection is crucial to the accomplishment of attacks. However, existing work focuses only on imperceptibility or transferability, and *how to make AEs camouflaged is still an open problem.*

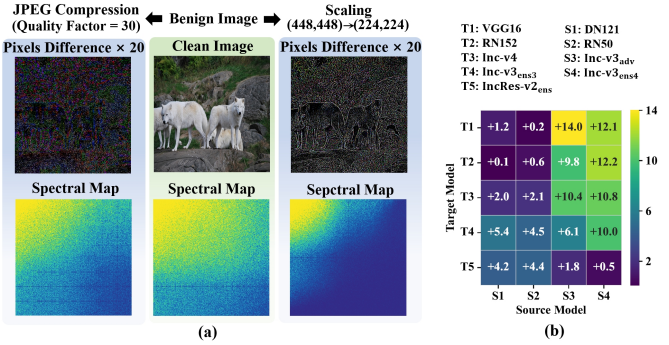


Fig. 1. (a) Changes of spectral energy maps and pixel differences after scaling/JPEG compression. In spectral energy maps, the top-left region corresponds to low frequency, while the bottom-right represents high frequency, with brighter areas indicating stronger spectral energy. For better visual effect, pixel differences are amplified by 20 times. (b) Increased attack success rates (%) when confining MI-FGSM perturbations to low-frequency component.

In this paper, we propose a novel camouflaged adversarial attack (CAA), which produces camouflaged adversarial examples (CAEs) for the first time. At a high-level view, the adversarial properties of CAEs will keep "dormant" state until target models inadvertently trigger the "activated" state. To make the activation event difficult to detect, we exploit two common preprocessing operations, image scaling and JPEG compression, as the trigger, which have been integrated into mainstream deep learning frameworks such as Caffe, TensorFlow and PyTorch [7]. Our design goal is to make CAEs visually and feature/label-wise indistinguishable from benign images at first, but turn into AEs after preprocessing. To achieve this, our initial idea is applying spectrum transformation [8] on input images to craft low-frequency attack perturbations and high-frequency camouflage perturbations, misleading target models and deceiving detection models, respectively. The inspiration comes from two key observations: ① *As shown*

This work was supported in part by the National Natural Science Foundation of China under Grants 62272150, 62222204, 62372121, and 62472113; in part by the Natural Science Foundation of Guangdong Province of China under Grant 2023A1515012358; in part by Sichuan Science and Technology Program under Grants 2024ZDZX0011 and 2025ZNSFSC1472; in part by the Postdoctoral Fellowship Program of CPSF under Grant Number GZC20251074.

Corresponding author: Qin Liu.

Yipeng Zou, Qin Liu, and Guo Chen are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan 410082, China (e-mail: {cszyp, gracelq628, guochen}@hnu.edu.cn).

Jie Wu is with the China Telecom Cloud Computing Research Institute, China; the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122 USA (e-mail: jiewu@temple.edu).

Tian Wang is with the Institute of Artificial Intelligence and Future Networks, Beijing Normal University & UIC, Zhuhai 519000, China; the College of Computer and Data Science, Fuzhou University, China; and the Computer Science and Mathematics, Fujian University of Technology, Fuzhou 350118, China (e-mail: cs_tianwang@163.com).

Guojun Wang and Tao Peng are with the School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou 510006, China (e-mail: {pengtao, csgjwang}@gzhu.edu.cn).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org., provided by the authors. The material includes an appendix.

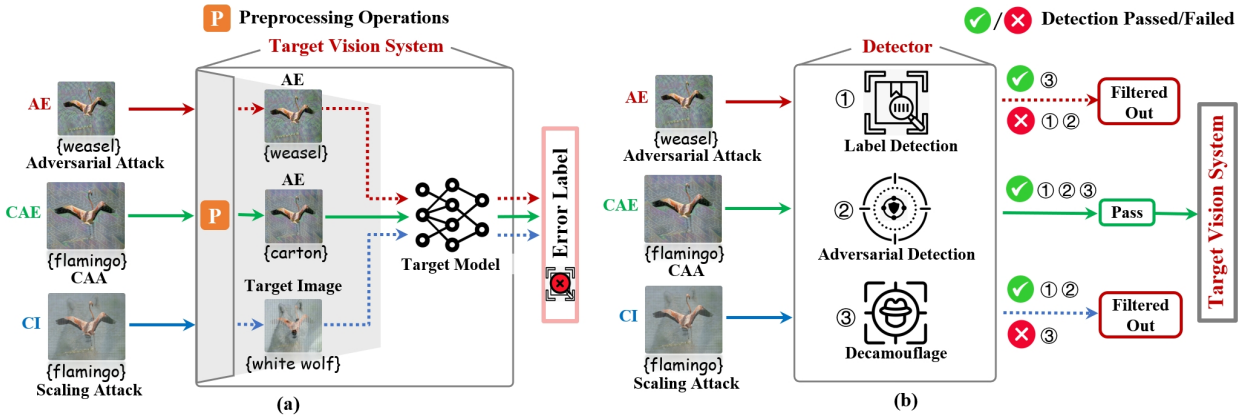


Fig. 2. The innovation of our work. (a) The main difference from existing attacks. In adversarial attack, AE is visually indistinguishable from benign image and can mislead target model, but lacks camouflage. In scaling attack, CI is visually similar to benign image and has camouflage property, but will become a different target image after scaling. In our CAA, CAE is visually indistinguishable from benign image and has camouflage property, which will turn into AE after scaling/compression. (b) The major advantage compared to existing attacks. When three kinds of defenses are deployed at the front end, only CAE can pass detection and reach the back-end target vision system, which performs routine preprocessing operations to trigger adversarial attack.

in Fig. 1-(a), both scaling and JPEG compression tend to drop high-frequency information; ② *As shown in Fig. 1-(b), low-frequency perturbations yield high transferability, especially for adversarially trained models*. The above observations are also experimentally proven in Refs. [8], [9]. We expect that the high-frequency camouflage perturbations play a leading role at first, but after preprocessing, they are filtered out and the low-frequency attack perturbations begin to take effect.

However, merely manipulating frequency-domain perturbations leads to unsatisfactory performances in practice. The main reason is that the exact scaling/compression algorithms and relative parameters adopted by target models are unknown. Thus, it is impossible to determine what information will be filtered out after preprocessing. The unfiltered residual high-frequency camouflage perturbations may impair adversarial properties, causing low fooling rates. Besides, high-frequency perturbations normally lack transferability, making the camouflage perturbations hard to deceive unseen detection models. Therefore, *the main challenge lies in how to improve attack robustness as well as camouflage transferability at once*.

To address this challenge, we formalize the design objective as a bi-level optimization problem, and solve it by a two-stage optimization strategy. In the first stage, we design a novel generative adversarial network (GAN) that is integrated with high-frequency robustness (HFR) and preprocessing modules, while training the GAN under varying scaling/compression parameters to learn robust attack perturbations. In the second stage, we design the feature-level losses and triplet loss [10], utilizing feature (dis)similarities and contrastive distances to improve the transferability of camouflage perturbations. It is worth noting that our design is inspired by the scaling attack [11], in which the camouflage image (CI) crafted by injecting a target image into the benign image can mislead DNNs after scaling. But the CI will show a totally different appearance after scaling, and such dramatic dissimilarity can be easily detected by human eyes. The innovation of our work is illustrated in Fig. 2. From this figure, we can see that while both AE and CAE are visually indistinguishable from benign image, only CAE can disguise as benign before preprocessing so as to pass label checking and adversarial detection; While both CI and CAE have camouflage properties, CI visually retains the trace of the target image, and can be easily detected by decamouflage methods [12] (as shown in Fig. 11).

Our contributions are summarized as follows.

- We find that current AEs lack camouflage, making them hard to pass label checking or adversarial detection. To address this limitation, we propose a novel CAA attack which hides CAEs' adversarial properties behind preprocessing, simultaneously achieving camouflage, imperceptibility, and transferability for the first time.
- We design a two-stage optimization strategy to craft attack and camouflage perturbations, making CAEs visually and feature/label-wise indistinguishable from benign images at first, but implicitly turn into AEs after triggered.
- Empirical evaluations on ImageNet dataset validate the effectiveness. Especially for robust models, the average camouflage success rate (CSR) before preprocessing exceeds 84%, while the average attack success rate (ASR) after preprocessing could reach 96.3%, surpassing the state-of-the-art (SOTA) adversarial attack by 13.5%.

Paper Organization. We review the related work in Section II before introducing the preliminaries in Section III. After describing the threat model in Section IV, we detail the proposed CAA in Section V and provide the performance evaluation in Section VI. Finally, we present a discussion in Section VII before concluding this paper in Section VIII.

II. Related Work

A. Transferable Adversarial Attacks

TAI [13] provided a systematized review of transferable attacks, and suggested the respective optimal hyperparameter settings for fair comparison. To improve transferability, existing researches can be classified into the following categories.

Input Transformation. This kind of approaches modify input images before gradient calculation, showing superior

performance. To enhance input diversity, DIM [14] resized the image into random sizes, TIM [15] translated the image into a set of images, SIM [16] calculated the gradient of several scaled images, and BSR [17] split the image into blocks, which were then randomly shuffled and rotated.

Feature Manipulation. Observing that different models yield similar shallow feature representations, this type of approaches launch attacks at models' intermediate layers, aiming to maximize the distance between the low-level features of benign and adversarial examples. ATA [18] computed model attention over extracted features to regularize the search of additive noises. BIA [19] trained a GAN to disrupt low-level features of input images to enhance transferability. ILPD [20] introduced the intermediate-level perturbation decay to obtain effective adversarial directions. Attention-SA [21] proposed to manipulate latent semantic representations for enhanced adversarial attacks. FAUG [22] proposed injecting dynamic random noise into intermediate features to reduce surrogate-model overfitting.

Frequency Manipulation. For an image, the low-frequency components generally contain the semantic information, whereas the high-frequency components are almost imperceptible to humans. Observing that low-frequency perturbations yield high transferability, recent studies began to inject perturbations into the low-frequency components [8]. Besides, several studies showed that adversarial perturbations were not necessarily confined to high- or low-frequency regions [23]. Based on this, MFI [24] replaced high-frequency components to improve adversarial transferability. SSA [25] proposed the frequency-domain model augmentation by using spectrum transformation; GE-AdvGAN [26] edited gradients based on frequency domain during the generator training to improve adversarial transferability; HFA [27] enhanced the transferability by scaling gradients and combining high-frequency features.

Apart from the above approaches, adversarial transferability can be enhanced by improving loss functions [28], [29], involving momentum into gradient calculation [30], attacking multiple source models simultaneously [31], and modifying source models [32]. Besides, UMI-GRAT [33] focused on transferable attacks against segmentation models. Their work considers a different threat model, wherein the attacker is unaware of the downstream model's tasks and the associated training data. UCG [34] proposed a universal generator that improved adversarial transferability across both architectures and domains by incorporating attention transfer, roughness abatement, and integrated transformation techniques. However, all existing transferable AEs lack camouflage.

B. Imperceptible Attacks

Imperceptible attacks focus on improving adversarial invisibility to deceive human vision system. Initial work improved imperceptibility by manipulating high-frequency components. For instance, AdvDrop [35] generated AEs by removing high-frequency information from benign images; SSAH [36] used a low-frequency constraint to limit perturbations within the imperceptible high-frequency components. As the high-frequency components normally lack transferability, later work employs generative models to improve the naturalness of AEs. AdvAD [37] modeled adversarial attacks as a non-parametric diffusion process by using target model guidance and pixel-level constraint; DiffAttack [38] optimized adversarial perturbations in the latent space of diffusion models. The AEs of imperceptible attacks are more visually deceptive, but vary considerably from benign images in the feature space, thus still cannot pass label checking and adversarial detection.

C. Camouflaged and Conditional Adversarial Patches

While some previous work [39], [40] claimed to achieve adversarial camouflage, their approaches primarily focused on crafting adversarial patches with natural styles that appear legitimate to human observers, rather than evading label-based checking or adversarial detection. The only work that seems somewhat similar to ours is that on conditional adversarial patches [41], which remain benign under normal circumstances but can be triggered to launch an attack by injecting acoustic signals towards cameras. However, their approach focused on adversarial patches, requiring the attacker to actively inject specific signals to launch physical-world attacks. In summary, our CAA differs from conditional adversarial patches in the following aspects: ① *Trigger Mechanism:* Conditional adversarial patches rely on specific external triggers (e.g., lighting or acoustic signals), whereas our CAA leverages common image preprocessing as natural triggers, which can be inadvertently triggered by target models without noticing. ② *Perturbation Form:* Conditional adversarial patches are typically visible to human eyes, in contrast, CAA generates imperceptible perturbations under the l_p-norm constraint.

D. Adversarial Defenses

To mitigate the threat posed by adversarial examples, a wide range of defense techniques [42] have been proposed. Adversarial training has been extensively studied as a robust defense strategy, where models are trained using both clean and adversarially perturbed inputs. TRADES [43] enhanced this approach by introducing a regularization term to balance standard accuracy and robustness. TATA [44] trained models with trigger activation, so that models generated accurate predictions only when triggered data was taken as input. To improve black-box resilience, ensemble adversarial training [45] decoupled the generation of adversarial examples from the target model itself. In contrast to model-centered defenses, input transformation [46] techniques focus on mitigating adversarial perturbations through preprocessing before inference. FD [47] suppressed high-frequency adversarial noise using JPEG-based frequency compression. Randomized resizing and padding (R&P) [48] applied geometric transformations to disrupt structured perturbations. HGD [49] utilized a U-Net decoder to recover clean semantic features, while NRP [50] leveraged self-supervised learning to purify corrupted representations. More recently, DiffPure [51] used diffusion models for purification, demonstrating strong defensive performance. Beyond empirical defenses, certified methods provided formal guarantees of robustness against bounded perturbations. Representative examples include randomized smoothing (RS) [52],

interval bound propagation (IBP) [53]. In this paper, we adopt the advanced defenses to assess the robustness of our adversarial examples.

III. Preliminaries

A. Adversarial Attacks

Adversarial attacks, a critical research area in machine learning security, aim to deceive DNN models through carefully crafted imperceptible perturbations. Formally, given a DNN classifier $f : \mathbb{X} \rightarrow \mathbb{Y}$, an input sample $x \in \mathbb{X}$, and its true label $y \in \mathbb{Y}$, the goal of an adversarial attack is to generate an adversarial example $x' = x + \delta$ such that:

$$f(x') \neq y, \ \ s.t. \|\delta\|_p \leq \epsilon, \tag{1}$$

where δ denotes the adversarial perturbation, and ϵ represents the perturbation budget under l_p-norm constraints. Based on the attacker's knowledge of the target model, adversarial attacks are categorized into white-box and black-box attacks.

B. Generative Adversarial Networks (GANs)

GANs [54] are a class of deep learning models that consist of two neural networks: a generator $\mathcal{G}$ and a discriminator $\mathcal{D}$, which are trained simultaneously in a game-theoretic framework. The generator creates fake data, while the discriminator distinguishes between real and fake samples. The objective function of a GAN can be formulated as a minimax game between the generator and the discriminator:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log \mathcal{D}(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - \mathcal{D}(\mathcal{G}(z)))], \tag{2}$$

where $\mathcal{G}(z)$ is the generator output for a noise vector z, $\mathcal{D}(x)$ is the discriminator's probability that input x is a real sample, and p_{data} and $p_z(z)$ represent the data and noise distributions, respectively. The generator aims to produce realistic data, and the discriminator tries to classify data as real or fake.

C. Contrastive Learning

Contrastive learning [55] is a technique where a model learns to distinguish between similar and dissimilar pairs of samples. The objective is to bring similar samples closer in the feature space and push dissimilar samples apart. Triplet loss [10] is a key loss function used in contrastive learning. The loss aims to preserve the relative distances between instances in the learned embedding space. Triplets are formed from an anchor, a positive sample (similar to the anchor), and a negative sample (dissimilar to the anchor). The goal is to ensure the distance between the anchor and positive is smaller than that between the anchor and negative by a specified margin. Given an anchor A, a positive P, and a negative N, the triplet loss function can be expressed as:

$$\mathcal{L}_{triplet} = \max(d(A, P) - d(A, N) + \alpha, 0), \tag{3}$$

where $d(A, P)$ is the distance between the anchor and positive, $d(A, N)$ is the distance between the anchor and negative, and α is a margin enforced between positive and negative pairs.

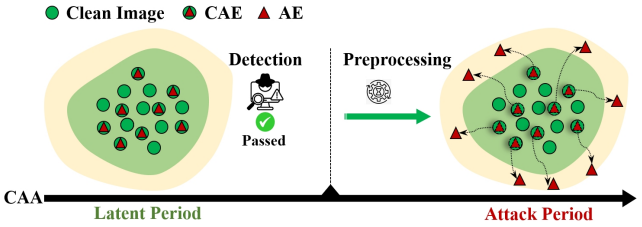


Fig. 3. The unique two-period attack flow of CAA.

IV. Threat Model

Victim and Adversary. In this work, we consider a stronger victim model, which is equipped with defense tools, such as label consistency checking or adversarial detection, and thus can filter out potentially malicious inputs before inference. Accordingly, the adversary setting is more challenging and constrained, where the generated CAEs can evade detection mechanisms at first and fool target models after triggered.

Adversarial Knowledge. Our CAA belongs to the transferable black-box attack, where the attacker is assumed to have no prior knowledge about the target model. Instead, the attacker leverages a surrogate model to craft adversarial examples that transfer to the target model. Besides, the attacker is unaware of the detection mechanisms adopted by the victim. The precise preprocessing details, including the exact algorithms and relative parameters, remain unknown to the attacker.

Attack Goal. Let $f_t : \mathbb{X} \rightarrow \mathbb{Y}$ be a target model, let $f_s : \mathbb{X} \rightarrow \mathbb{Y}$ be a source model, and let $f_d : \mathbb{X} \rightarrow \mathbb{Y}$ be a label detection model. Given a benign image $x \in \mathbb{X}$ with ground-truth label $y \in \mathbb{Y}$, and source model f_s, our goal is to craft the CAE x_{cae} with the following properties: ① **Camouflage**: It resembles benign image in the feature space, which can not only pass adversarial detection, but also deceive the label detection model, i.e., $f_d(x_{cae}) = y$; ② **Transferability:** It fools the target model after preprocessing, i.e., $f_t(\mathcal{P}(x_{cae})) \neq y$, where $\mathcal{P}(\cdot)$ is the preprocessing function, including scaling and compression. ③ **Imperceptibility:** The added perturbations are small enough so that the CAE is visually similar to the benign image before and after preprocessing.

Practical Attack Scenarios. The attacker first creates CAEs of large sizes (e.g., 448×448) and high resolutions (e.g., 3MB), and then performs two types of attacks: ① *Direct Attack:* The attacker directly sends the CAEs to target models; ② *Indirect Attack:* The attacker spreads the CAEs to the Internet making more models affected indirectly. For example, he can inject the CAEs into open-source datasets to poison the models using public training sets. To reduce security risks, victims will employ a detector to inspect and filter abnormal samples before using samples coming from outside.

As shown in Fig. 3, our CAA consists of a "latent" period and an "attack" period. In the latent period, the CAEs disguise as benign images and can pass the detection. Once scaling or JPEG compression is executed, CAA enters the attack period and CAEs become AEs, misleading target models or achieving data poisoning. The unique two-period attack flow creates the possibility of evading detection. If the attacker locally

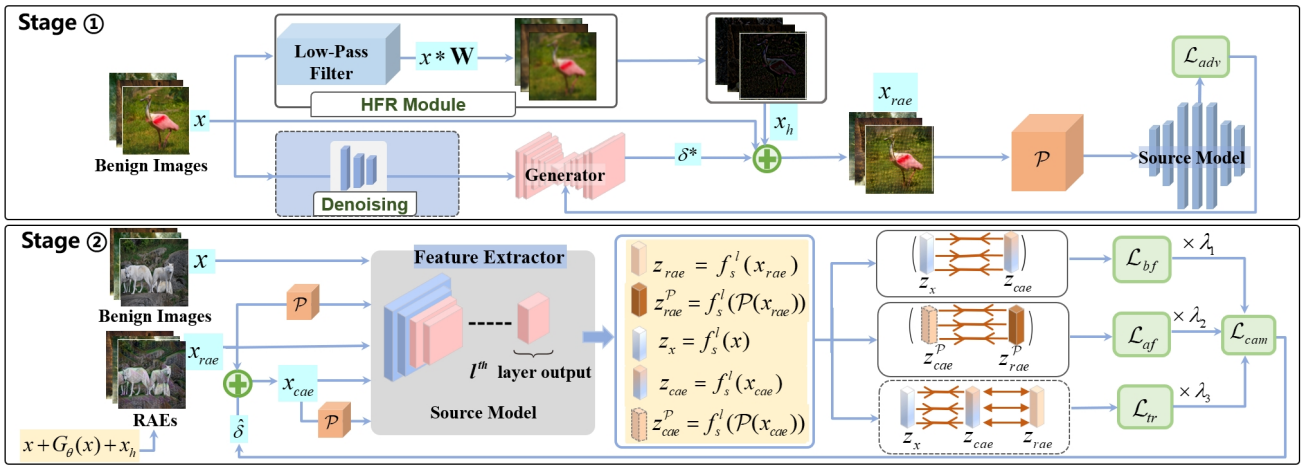


Fig. 4. The overview of CAA. The first stage takes the benign image x as input to learn the attack perturbations δ^*, and the second stage takes the benign image x and the RAE x_{rae} as input to craft the camouflage perturbation $\hat{\delta}$, outputting the CAE $x_{cae} = x_{rae} + \hat{\delta}$. All perturbations are clipped.

performs preprocessing, skipping latent period, CAA achieves the same effect as traditional adversarial attacks.

Choice of Triggers. Modern vision systems almost universally perform input preprocessing before inference. As summarized in Table XIII of the Appendix, mainstream deep learning frameworks usually carry out scaling, JPEG compression, padding, normalization, cropping, color jitter, and filtering. Among various preprocess operations, CAA chooses scaling and JPEG compression as the trigger because of the following reasons: ① *High probability of occurrence.* As shown in Table XIV of the Appendix, DNN models require fixed input sizes (e.g., 224×224), thus scaling becomes a necessary operation in deep learning pipeline. And JPEG compression is universally used for digital image transmission and storage. In addition, prior work has validated the feasibility of employing scaling and JPEG as the trigger to launch stealthy attacks. For example, Ref. [11] proposed camouflage attacks on image scaling algorithms, while Ref. [56] utilized JPEG as the trigger to launch conditional backdoor attacks. ② *Combined effect.* As shown in Fig. 1-(a), both scaling and JPEG compression tend to drop high-frequency information, and thus can supplement each other. Fig. 13 shows that our CAA yields the best performance when triggering them together. Notably, CAA does not rely on specific scaling/compression algorithms or parameters (Fig. 8-Fig. 10), and can successfully attack real-world cloud-based vision models, where the preprocessing details are hidden (Table VI).

V. Methodology

A. Overview

To achieve our attack goal, we plan to craft attack perturbation δ^* fooling target model f_t, and camouflage perturbation $\hat{\delta}$ deceiving detection model f_d, such that $\hat{\delta}$ suppresses δ^* at first, but δ^* plays a leading role after preprocessing. Actually, we adopted a single-stage optimization to craft perturbations at the beginning. However, from the experiment results shown in Table XVI of the Appendix, neither of GAN-based training nor iterative optimization can achieve satisfactory performance by using one-stage optimization strategy. Therefore, we formalize the objective as a bi-level optimization problem.

$$\begin{aligned} \min_{\hat{\delta}} \quad & \mathcal{L}\big(f_d(x+\delta^*+\hat{\delta}), y\big) \\ s.t. \quad & \delta^* = \arg\max_{\delta^*} \mathcal{L}\big(f_t(\mathcal{P}(x+\delta^*+\hat{\delta})), y\big), \\ & \|\delta^*+\hat{\delta}\|_p \le \epsilon, \end{aligned} \tag{4}$$

where the total perturbations are within ϵ under l_p-norm, and function $\mathcal{L}$ is used to determine the distance between the output of target model $f_t(\cdot)$ (or the output of detection model $f_d(\cdot)$) and the ground-truth label y.

As shown in Fig. 4, we solve this problem by two stages:

① **High-Frequency Robust Generative Learning.** This stage trains a GAN under varying scaling/compression parameters to learn robust attack perturbation δ^*. The GAN consists of a HFR module, a preprocessing function, a generator G_θ, and a pre-trained source model f_s. Specifically, the HFR module is used to extract the high-frequency component x_h from a benign image x, f_s is considered as the frozen discriminator, and G_θ guided by loss $\mathcal{L}_{adv}$ aims to fool f_s with the preprocessed robust adversarial example (RAE), $\mathcal{P}(x_{rae}) = \mathcal{P}(x+x_h+\delta^*)$. Once trained, G_θ can craft the attack perturbations robust against high-frequency information.

② **Feature-Level Camouflage.** This stage iteratively generates camouflage perturbation $\hat{\delta}$, where a random compression ratio and a random scale factor is adopted in the preprocessing module. Given a benign image x, we first invoke the generator G_θ and the HFR module to produce the RAE $x_{rae} = x+x_h+G_\theta(x)$. Then, we launch feature-level attacks on the source model f_s, and design the benign feature similarity loss $\mathcal{L}_{bf}$, the adversarial feature distance loss $\mathcal{L}_{af}$, and the triplet loss $\mathcal{L}_{tr}$ to learn camouflage perturbation $\hat{\delta}$. The CAE is generated by adding $\hat{\delta}$ into x_{rae}, i.e., $x_{cae} = x_{rae} + \hat{\delta}$.

To show the effectiveness of CAA, we employ class activation mapping (CAM) [57] to visualize the important features of benign image and the CAE before and after preprocessing. From the second row of Fig. 5, we can observe that the attention heatmap of CAE is largely consistent with that of benign image. But after preprocessing, the attention heatmap

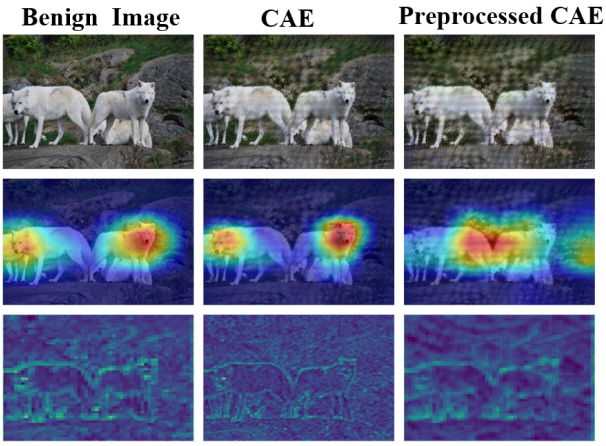


Fig. 5. The heatmaps and feature maps of the benign image, the CAE, and the preprocessed CAE generated on ResNet50 model.

of CAE is significantly disrupted making the model focus on entirely different areas compared with the benign images. This indicates that once the high-frequency camouflage perturbations are removed by preprocessing, the low-frequency attack perturbations start to take effect. Besides, from the third row of Fig. 5 and Fig. 19 of the Appendix, we can see that the feature maps of benign images and CAEs, though not identical, both preserve clear object contours and consistent semantic structures. Compared to benign feature maps, the feature maps of CAEs show sharper high-frequency texture details; but the feature maps of preprocessed CAEs become more blurred and structurally inconsistent. This is because high-frequency camouflage perturbations mainly impact texture details, but low-frequency attack perturbations destroy semantic structures.

B. High-Frequency Robust Generative Learning

Given a benign image x with ground-truth label y, our initial idea is to let the generator G_θ fool the source model f_s with $\mathcal{P}(x + G_\theta(x))$. Our empirical studies show that current attack perturbations lack robustness, which are easily suppressed by the high-frequency information survived after image preprocessing. Thus, we incorporate the HFR module into the GAN, employing an approximate yet simple Gaussian low-pass filter $\mathbf{W}$ to move out high-frequency information of input images [58]. Specifically, $\mathbf{W}$ is a $(4k+1) \times (4k+1)$ kernel matrix, and can be mathematically expressed as:

$$\mathbf{W}_{i,j} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{i^2+j^2}{2\sigma^2}\right), \tag{5}$$

where $\sigma = k$ determines the width of the filter $\mathbf{W}$. As σ increases, more high-frequency components will be filtered out. By tuning the value of σ, we can obtain the high-frequency component x_h of an input image x with Eq. (6):

$$x_h = x - \mathbf{W} * x. \tag{6}$$

The goal of the generator is changed to fool the f_s with $\mathcal{P}(x_{rae})$, where the RAE x_{rae} is crafted by Eq. (7):

$$x_{rae} = \text{Clip}_{x,\epsilon}\left(x + x_h + G_\theta(x)\right), \tag{7}$$

Algorithm 1 The working process of CAA

Input: Benign image X, low-pass filter $\mathbf{W}$, source model $f_s(\cdot) : \mathbb{X} \to \mathbb{Y}$, with $\mathbb{X}$ being ImageNet training set and $\mathbb{Y}$ being ground-truth labels, image preprocessing function $\mathcal{P}(\cdot)$, number of iteration T, and perturbation budget ϵ

Output: Camouflaged adversarial example X_{cae}

{Stage 1: *Generate attack perturbations and RAEs* }

Randomly initialize generator G_θ
repeat
  Sample mini-batch of samples (x, y) from $\mathbb{X}$ and $\mathbb{Y}$
  Calculate the loss $\mathcal{L}_{adv}$ by Eq. (8)
  Optimize $\delta^* = G_\theta(x)$ by maximizing $\mathcal{L}_{adv}$
until model converges

{Stage 2: *Craft camouflage perturbations and CAEs* }

Initialize the camouflage perturbation $\hat{\delta}$
for $i = 1$ to T **do**
  Extract high-frequency component X_h by Eq. (6)
  Construct the RAE $X_{rae} = \text{Clip}_{x,\epsilon}(X + X_h + G_\theta(X))$
  Construct the CAE $X_{cae} = \text{Clip}_{x,\epsilon}(X_{rae} + \hat{\delta})$
  Calculate the total loss $\mathcal{L}_{cam}$ by Eq. (9)-Eq. (12)
  Optimize $\hat{\delta}$ by minimizing $\mathcal{L}_{cam}$
end for
return $X_{cae} = \text{Clip}_{x,\epsilon}(X_{rae} + \hat{\delta})$

where ϵ is the perturbation budget and $\text{Clip}_{x,\epsilon}(\cdot)$ is a clip function to bound x_{rae} within the range of $[x-\epsilon, x+\epsilon]$. Next, we define the following loss to guide the generator:

$$\mathcal{L}_{adv} = \mathbf{CE}\big(f_s(\mathcal{P}(x_{rae})) - f_s(x_h), \mathbb{1}_y\big), \tag{8}$$

where **CE** is the relativistic cross-entropy loss [28] aiming to increase the "fooling gap" $(f_s(\mathcal{P}(x_{rae}))_y - f_s(x_h)_y)$ between $\mathcal{P}(x_{rae})$ and x_h. This loss would be higher when $\mathcal{P}(x_{rae})$ is scored significantly lower than x_h response for the ground-truth label, i.e., $f_s(\mathcal{P}(x_{rae}))_y \ll f_s(x_h)_y$. To illustrate the optimization behavior, Fig. 23 of the Appendix compares the training dynamics, showing that relativistic cross-entropy achieves higher loss values and maintains high gradient norms to ensure more effective optimization compared to standard cross-entropy. In this way, the generator G_θ learns a contrastive signal robust against variable high-frequency references. Table XVIII of the Appendix shows that relativistic cross-entropy loss improves the average ASR and CSR by 2.4% and 1.5%, respectively, compared to simple cross-entropy loss. Also, recent work shows that low-frequency perturbations are highly effective against adversarially trained models [8]. Hence, we perform denoising on input images before feeding to the generator, so that more attack perturbations exist in the low frequency.

C. Feature-Level Camouflage

Given a benign image x and the RAE $x_{rae} = x + x_h + G_\theta(x)$, this stage aims to add the camouflage perturbation $\hat{\delta}$ on the RAE to form the CAE. Following [59], we use one intermediate layer to compute feature-level losses. As shown in Figure 22 of the Appendix, this setting enables decent attack and camouflage performance. Let $f_s^l(\cdot)$ denote the features

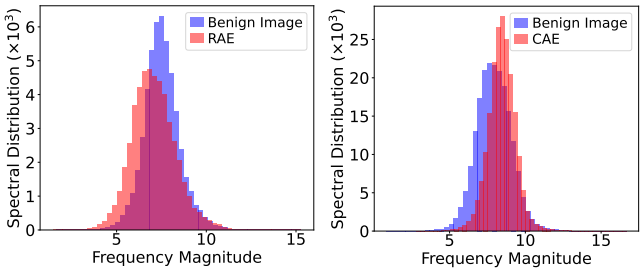


Fig. 6. Comparison of frequency spectrum histograms (benign images vs. RAEs and benign images vs. CAEs).

extracted from the l-th layer of the source model. First of all, we define the benign feature similarity loss $\mathcal{L}_{bf}$ making the CAE resemble the benign image in the feature space:

$$\mathcal{L}_{bf} = -\mathbf{CS}(f_s^l(x_{cae}), f_s^l(x)), \tag{9}$$

where **CS** is the cosine similarity score between the features of CAEs and the benign features. The next thing is to let CAEs show adversarial properties after preprocessing. Our empirical studies show that directly maximizing $\mathcal{L}(f_s(\mathcal{P}(x_{cae})), y)$ would make the preprocessed perturbations deviate from the adversarial features, often leading to local optimization (performance fell by more than 20%). Besides, l_1 norm captures the absolute differences of feature maps and is more sensitive to detailed variations. Hence, we define the adversarial feature distance loss $\mathcal{L}_{af}$, so that after preprocessing, the CAEs resemble the RAEs in the feature space:

$$\mathcal{L}_{af} = \| f_s^l(\mathcal{P}(x_{rae})) - f_s^l(\mathcal{P}(x_{cae})) \|_1, \tag{10}$$

where l_1 norm is used to determine the feature distance between the preprocessed RAE and the preprocessed CAE.

Adopting loss $\mathcal{L}_{bf}$ only makes CAEs close to benign samples in feature space, without considering the distance from RAEs. To further enhance transferability of camouflage perturbations, we introduce the triplet loss to calculate the contrastive distance among CAEs, RAEs, and benign images. Let $[\cdot]_+$ denote $\max(0, \cdot)$. $\mathcal{L}_{tr}$ is defined as:

$$\mathcal{L}_{tr} = 0.5\left(\|z_x - z_{cae}\|_2^2 + [\alpha - \|z_{rae} - z_{cae}\|_2]_+^2\right), \tag{11}$$

where $z_x = f_s^l(x)$, $z_{cae} = f_s^l(x_{cae})$, $z_{rae} = f_s^l(x_{rae})$, and α is the expected margin between the features of CAEs and those of RAEs. Here, the benign image acts as the anchor, and the CAE and RAE act as the positive and negative samples, respectively. As shown in Fig. 21 of the Appendix, applying $\mathcal{L}_{tr}$ makes the CAEs close to benign images but far from RAEs in the feature space, enabling in-class compactness and inter-class separability. Finally, the total loss is defined as:

$$\mathcal{L}_{cam} = \lambda_1(\mathcal{L}_{bf} + \mathcal{L}_{tr}) + \lambda_2\mathcal{L}_{af}, \tag{12}$$

where λ_1 and λ_2 are hyperparameters to trade off each item. The detailed working process of CAA is depicted in Alg. 1.

Remark 1. Although we do not directly constrain perturbations within particular frequency domains, the defined loss $\mathcal{L}_{adv}$ that requires attack perturbations to survive after preprocessing, and the defined loss L_{cam} that makes camouflage perturbations no longer work after preprocessing can achieve a similar effect. From Fig. 6, we can see that compared with benign images, the RAEs have more low-frequency information, but the CAEs have more high-frequency information. This means that the camouflage and attack perturbations are mostly concentrated in the high- and low-frequency components of input images, respectively. This conclusion is also supported in Figs. 17-18 and Fig. 20 of the Appendix. As high-frequency components are almost imperceptible to humans, existing high-frequency-based attacks [35], [36] normally relax the constraint on the size of high-frequency perturbations. Inspired by this, we can provide a camouflage-enhanced version by omitting the clip operations in Line 11 and Line 15 of Stage 2. From Fig. 7 and Table V, we observe that the CAEs crafted by the camouflage-enhanced CAA are also visually similar to the benign images and have fairly good imperceptibility.

Remark 2. The quantization process in JPEG compression includes rounding operations, which are non-differential and cannot be optimized via back-propagation. Following existing work [60], we solve this problem by utilizing tangent function to approximate rounding operations gradually.

VI. Performance Evaluation

The attack performance is assessed on 4 source models and 22 target models, and compared with 14 adversarial attacks. Due to limited space, this section only presents the representative results on 2 source models. More comprehensive experimental results are provided in the Appendix for completeness. In the following evaluation, CAA includes the denoising module, and the preprocessed module includes both JPEG compression and downsampling unless otherwise stated. To make a distinction, we use CAA and CAA_τ to denote the standard and camouflage-enhanced CAAs, respectively. For CAA_τ, we omit the clipping operations in Line 11 and Line 15 of Stage 2 to allow more high-frequency camouflage perturbations. Consequently, $\|\delta^* + \hat{\delta}\|_p$ might exceed ϵ in some cases. To be fair, we solely compared CAA (with clipping) with baselines and used CAA_τ as a reference only in evaluating attack performance (i.e., Tables I, II, VI, and VII).

A. Experimental Settings

Models and Datasets. Source models are ResNet50 and DenseNet201. The target models include 14 normally trained models [61]–[67]: VGG16, VGG19, ResNet50, ResNet152, DenseNet121, DenseNet169, DenseNet201, Visformer, Swin, ViT-B, PiT-B, CaiT-S, LeViT, and HaloNet; 8 robust models [68], [69]: Inception-v3, Inception-v4, IncRes-v2, ResNet-v2-101, Inc-v3$_{ens3}$, Inc-v3$_{ens4}$, IncRes-v2$_{ens}$, and Inc-v3$_{adv}$ Among them, 12 models are selected for label detection. Following [17], we use ImageNet as benchmark dataset, and pick one random image from each class to form the test set, ensuring each testing sample can be correctly classified by almost all models.

Implementation Details. All experiments are conducted using PyTorch on an NVIDIA RTX 4090 GPU. As in [28], we construct the generator based on ResNet. In Stage 1, we adopt the Adam optimizer with a learning rate of $2e-4$ for 10 epochs. In Stage 2, we set the iteration number to $T = 100$.

TABLE I
THE ASRS (%) ON NORMALLY TRAINED MODELS, WHERE "*" INDICATES WHITE-BOX ATTACKS. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD AND THE SECOND-BEST RESULTS ARE UNDERLINED. (ALL THE TABLES IN THIS PAPER ADOPT THE SAME MARKUP METHOD.)

Model	Attack	VGG16	VGG19	ResNet50	ResNet152	DenseNet121	DenseNet169	DenseNet201	Visformer	Swin	ViT-B	PiT-B	CaiT-S	LeViT	HaloNet	AVG.
ResNet50 (RN50)	DIM	85.0	84.0	99.9*	91.1	92.8	91.7	91.3	44.4	42.1	19.8	29.5	33.5	48.3	62.1	65.4
	TIM	56.0	56.2	99.9*	57.4	63.1	58.2	58.1	12.5	10.2	6.0	6.9	9.4	19.1	34.9	39.1
	SSA	95.0	95.3	100.0*	97.7	96.5	96.9	96.7	58.0	57.0	26.2	37.4	41.0	64.5	72.2	73.9
	CDA	94.9	92.8	99.7*	93.0	94.7	96.0	94.4	**70.7**	60.1	16.6	17.4	33.3	50.7	54.7	69.2
	PGN	91.9	91.1	100.0*	97.4	95.9	96.6	95.6	52.6	51.8	26.3	34.6	42.9	61.2	68.2	71.9
	HFA	93.3	92.1	99.1*	90.6	93.1	91.5	90.8	54.0	49.2	26.8	40.5	46.0	65.8	76.2	72.1
	GE-AdvGAN	93.6	91.5	97.2*	69.7	59.9	62.6	57.1	14.9	12.5	3.5	4.6	8.8	24.1	34.2	45.3
	ILPD	94.4	95.1	100.0*	98.2	97.5	97.9	97.3	60.5	**63.5**	24.2	41.3	38.6	72.5	82.9	76.0
	BSR-MI	**98.3**	**97.1**	100.0*	97.1	99.1	97.9	97.4	**70.7**	57.0	31.4	41.2	46.1	71.3	85.1	77.8
	BSR-TI-DIM	97.5	96.3	100.0*	93.4	97.9	97.1	96.9	57.5	44.0	26.3	31.4	44.1	66.0	85.7	73.9
	BSR-SI-TI-DIM	94.0	94.1	99.4*	89.3	97.4	96.4	96.5	49.1	36.0	11.9	14.2	22.4	48.9	71.5	65.8
	BIA	94.2	91.9	98.5*	94.1	93.6	91.8	60.6	69.6	62.3	17.8	19.8	31.9	53.6	61.5	67.2
	FAUG	86.4	86.5	100.0*	90.0	93.9	93.8	92.8	52.0	51.5	16.6	21.3	27.4	53.5	65.5	66.5
	CAA	96.0	95.8	99.5*	**98.5**	**99.4**	**99.3**	**99.0**	66.7	60.8	**33.2**	**41.7**	**46.3**	**77.6**	**93.6**	**79.1**
	CAA_τ	96.9	97.2	99.6*	98.7	99.5	99.4	99.1	73.0	65.1	34.8	43.9	47.6	81.5	95.1	80.8
DenseNet201 (DN201)	DIM	81.2	79.0	88.4	83.6	94.4	95.4	100.0*	53.8	44.5	21.3	33.4	37.2	53.9	68.3	66.7
	TIM	52.3	52.4	57.7	46.6	72.9	74.6	100.0*	18.1	13.1	6.5	8.6	12.8	24.7	38.9	41.4
	SSA	93.3	92.1	96.7	95.3	98.9	98.7	100.0*	76.3	70.8	40.2	54.6	59.7	74.9	83.2	81.1
	CDA	95.7	96.4	95.8	90.1	95.1	98.6	99.3*	65.6	39.4	16.2	11.6	34.5	53.5	69.3	68.7
	PGN	90.7	91.2	96.7	94.3	99.1	98.7	99.9*	71.8	68.0	42.0	53.0	60.7	73.7	81.7	80.1
	HFA	91.7	90.7	91.9	88.3	94.4	94.4	99.4*	69.7	65.0	41.4	54.9	58.8	73.9	82.5	78.4
	GE-AdvGAN	97.1	96.4	91.3	86.0	93.9	93.3	95.9*	45.7	21.8	3.7	5.4	23.9	48.5	76.1	62.8
	ILPD	85.6	85.7	94.9	91.4	97.0	97.6	99.5*	74.0	66.4	39.3	51.4	54.9	65.1	74.2	76.9
	BSR-MI	96.6	96.9	**97.8**	93.9	**99.6**	99.0	100.0*	74.2	60.8	32.5	48.7	53.0	74.1	91.0	79.9
	BSR-TI-DIM	94.3	93.5	95.0	87.3	98.9	97.2	99.9*	54.6	37.1	24.3	33.1	42.5	66.1	87.7	72.3
	BSR-SI-TI-DIM	89.6	88.4	90.1	80.8	97.1	96.0	99.7*	44.7	31.5	20.4	22.7	33.2	65.3	86.2	67.6
	BIA	96.3	96.8	92.5	91.2	94.8	94.2	96.6*	56.8	28.8	20.3	12.8	33.3	55.8	66.2	66.9
	FAUG	82.5	80.9	89.6	83.4	95.0	96.6	99.8*	60.6	46.3	24.1	28.3	38.3	58.4	71.7	68.3
	CAA	**98.9**	**98.6**	97.6	**97.3**	98.9	**99.1**	99.3*	**85.8**	**72.4**	**46.5**	**55.1**	**66.6**	**89.6**	**97.1**	**85.9**
	CAA_τ	99.1	98.7	98.0	97.9	99.0	99.2	99.4*	87.2	74.3	49.7	57.1	69.8	90.7	97.3	87.0

Following [59], we select layer3 of ResNet50, denseblock3 of DenseNet201, the second InceptionC module of Inc-v3, and the third InceptionB module of Inc-v4 to extract features. For compression, we use the standard JPEG algorithm, and set quality factors to $[20, 80]$. For scaling, we use bilinear interpolation and set the image sizes to $[400, 500]$. The default input size and quality factor is 448×448 and 50, respectively.

Baselines. We compare our CAA with 14 adversarial attacks under perturbation budget $\epsilon = 16/255$: ① Input transformation-based attacks: BSR [17], DIM [14], and TIM [15]; ② Frequency-based attacks: SSA [25], HFA [27], and GE-AdvGAN [26]; ③ Loss-based attacks: CDA [28] and PGN [29]; ④ Feature-based attacks: BIA [19], ILPD [20] and FAUG [22]; ⑤ Imperceptible attacks: SSAH [36], AdvDrop [35], and AdvAD [37]. For fair comparison, the hyper-parameters of baselines follow the optimal settings in respective literature. For instance, BSR generates 20 transformed input copies to compute the averaged gradient, DIM adopts the transformation probability of 0.5, and TIM utilizes the Gaussian kernel with the size of 15×15. Besides, GAN-based baselines train the generators for 10 epochs, while all iterative baselines equipped with the momentum term $\mu = 1.0$ are run for 10 iterations, except ILPD, which uses 100 iterations as recommended in its original implementation.

Metrics. We use the ASR to evaluate the attack performance, where the ASR is the ratio of the traditional AEs or our preprocessed CAEs being misclassified by target models. And we use the CSR and detection evasion rate to evaluate the camouflage effect, where the CSR is the ratio of the CAEs or AEs being classified into correct labels by label detection models, and the detection evasion rate denotes the ratio of the CAEs or AEs being determined as benign samples by adversarial detection models. Moveover, we report the average results of 5 trials, with a standard deviation of less than 0.8%.

B. Effectiveness of CAA

Transferability. We compare CAA with 11 transferable baselines against both normally trained and robust models. From Table I, we can see that our CAA shows excellent adversarial transferability on normally trained models. The performance of CAA surpasses the SOTA BSR attack. For example, when DN201 is the source model, CAA outperforms BSR by 4.8%. This indicates that our CAA exhibits superior adversarial transferability even while maintaining camouflage. From Table II, we can observe that CAA shows the best attack performance on robust models, significantly surpassing BSR. This can be attributed to the HFR and denoising modules that make attack perturbations mostly concentrate in the low-frequency domain, showing high transferability.

Camouflage. CAA is the first work on camouflage, and the CSR of each baseline can be directly calculated by subtracting the ASR in Table I from one. Hence, we only evaluate the CSRs of our CAA in the experiments, and use both normally and adversarially trained models for label detection. From Table III, we can observe that the white-box CSRs reach over 97%, and the average CSRs on normally and adversarially trained models are more than 63.0% and 84.3%, respectively. Adversarially trained models are robust against perturbations, which can output the correct labels with high probabilities. In other words, they are more easily "deceived" by CAEs than normally trained models. Besides, CAA_τ does not restrict

TABLE II
THE ASRS (%) ON ROBUST MODELS, INCLUDING FOUR ADVERSARIALLY TRAINED MODELS: Inc-v3$_{ens3}$, Inc-v3$_{ens4}$, IncRes-v2$_{ens}$, AND Inc-v3$_{adv}$.

Models	Attack	Inception-v3	Inception-v4	IncRes-v2	ResNet-v2	Inc-v3$_{ens3}$	Inc-v3$_{ens4}$	IncRes-v2$_{ens}$	Inc-v3$_{adv}$	AVG.
RN50	DIM	73.0	69.8	66.7	77.1	53.3	50.0	39.8	57.6	60.9
	TIM	29.6	26.3	20.2	32.1	26.7	28.4	19.4	24.9	26.0
	SSA	89.0	84.6	83.2	90.9	68.0	63.6	50.6	73.9	75.5
	CDA	92.7	87.4	78.5	91.1	76.2	61.7	50.2	71.7	76.2
	PGN	84.3	77.9	76.8	85.5	69.5	68.8	54.5	72.4	73.7
	HFA	83.3	80.1	80.5	84.1	65.8	62.6	49.5	71.5	72.2
	GE-AdvGAN	63.9	55.7	44.9	69.1	51.1	42.2	27.4	58.7	51.6
	ILPD	91.3	90.8	91.3	91.1	72.5	69.0	55.3	77.8	79.9
	BSR-MI	96.2	90.3	86.7	90.5	78.0	77.2	66.4	81.0	82.8
	BSR-TI-DIM	86.1	84.5	76.0	83.8	75.1	76.5	65.2	76.7	78.0
	BSR-SI-TI-DIM	84.8	82.0	76.9	83.7	76.6	78.3	67.6	78.6	78.6
	BIA	93.6	88.7	75.6	89.5	78.3	63.4	48.7	78.3	76.4
	FAUG	75.2	68.0	68.7	76.7	55.3	53.7	42.5	59.1	62.4
	CAA	**97.6**	**96.8**	**93.3**	**98.9**	**97.1**	**93.8**	**96.3**	**96.8**	**96.3**
	CAA_τ	98.1	97.6	95.4	99.0	98.0	95.4	97.4	97.9	97.4
DN201	DIM	72.3	71.4	66.3	76.0	54.6	54.2	43.6	61.4	62.5
	TIM	32.8	29.6	24.3	31.8	28.9	31.3	23.2	28.8	28.8
	SSA	91.8	89.6	87.3	91.5	78.6	76.3	68.4	82.4	83.2
	CDA	89.3	89.7	78.4	94.9	80.7	69.5	51.7	74.7	78.6
	PGN	87.5	85.2	83.8	87.9	78.6	77.1	67.9	80.8	81.1
	HFA	87.1	84.5	82.6	87.3	74.5	72.4	62.0	77.0	78.4
	GE-AdvGAN	86.0	88.0	78.6	81.6	55.8	48.7	39.1	67.2	68.1
	ILPD	81.7	81.1	79.7	83.0	53.2	49.9	37.8	62.0	66.1
	BSR-MI	93.3	92.3	87.6	91.8	76.6	77.4	66.4	82.3	83.5
	BSR-TI-DIM	86.5	83.5	75.0	81.9	74.4	74.6	65.0	76.4	77.2
	BSR-SI-TI-DIM	78.5	77.1	68.0	76.7	73.7	76.0	63.9	74.0	73.5
	BIA	91.2	90.8	76.3	91.8	83.2	72.5	48.6	76.9	78.9
	FAUG	78.2	74.5	72.0	79.9	58.3	57.0	42.9	63.1	65.7
	CAA	**98.3**	**98.3**	**94.7**	**97.5**	**88.8**	**87.8**	**81.9**	**89.2**	**92.1**
	CAA_τ	98.4	98.3	95.6	97.8	90.4	89.3	83.9	91.5	93.2

TABLE III
THE CSRS (%) ON NINE NORMALLY TRAINED MODELS AND THREE ADVERSARIALLY TRAINED MODELS.

Models	Attack	VGG16	VGG19	RN50	RN152	DN121	DN169	DN201	Visformer	Swin	Inc-v3$_{ens3}$	Inc-v3$_{ens4}$	IncRes-v2$_{ens}$
RN50	CAA	54.5	49.0	**98.1***	75.5	56.0	68.4	72.1	66.9	84.5	82.3	84.3	86.2
	CAA_τ	**62.1**	**60.5**	**98.1***	**94.0**	**82.3**	**86.2**	**87.6**	**67.1**	**84.7**	**82.9**	**85.0**	**87.2**
DN201	CAA	49.3	50.8	68.0	71.0	73.8	73.2	**97.7***	34.1	49.4	79.2	85.8	90.8
	CAA_τ	**58.5**	**65.3**	**87.8**	**90.5**	**91.1**	**92.4**	**97.7***	**53.3**	**55.1**	**92.6**	**91.2**	**93.9**

TABLE IV
COMPARISON OF EVASION RATES (%) AGAINST ADVERSARIAL DETECTION.

Models	DIM	TIM	SSA	CDA	PGN	HFA	GE-AdvGAN	ILPD	BSR-MI	BSR-TI-DIM	BSR-SI-TI-DIM	BIA	CAA	CAA_τ
RN50	21.8	50.2	13.2	13.3	17.9	20.1	39.7	13.6	11.6	12.9	19.8	36.9	**84.3**	85.4
DN201	22.3	45.2	12.3	13.9	9.7	14.7	22.6	14.7	11.6	18.5	21.2	18.6	**83.6**	89.2

the size of high-frequency camouflage perturbations, thus exhibiting better performance than CAA. Apart from CSRs, we compare our CAA and baselines in terms of the evasion rate against adversarial detection [6]. From Table IV, we can see that CAA can achieve the detection evasion rate of 89.2% which is about 4 times higher than all baselines. Thus, our CAA is more likely to successfully launch attacks against computer vision systems equipped with adversarial detection mechanisms. In contrast, traditional AEs can be easily detected and filtered out before reaching the target model.

Visualization of CAEs. Let CAE and CAE_τ denote the CAEs crafted by CAA and CAA_τ, respectively. Fig. 7 visualizes five randomly selected CAE and CAE_τ, respectively. Here, we observe that both CAE and CAE_τ are visually similar to the benign images. In addition, from the right side of Fig. 7, we can see that CAE_τ contains more high-frequency information compared to CAE. The reason is that CAA_τ removes the clipping operations in Stage 2, so that more camouflage perturbations can be injected into the high-frequency domain. Besides, we use PSNR, SSIM, and LPIPS to measure the imperceptibility to the human visual system. As shown in Table V, although not as good as imperceptible attacks, the imperceptibility of CAA is comparable to SOTA transferable attacks, like ILPD, HFA, and BSR.

Real-World Attack. In a real-world scenario, we first create a CAE of a large scale size (448× 448) and a large storage size (≥ 330 KB), and then send it to remote Vision APIs, including Baidu Cloud[1], Alibaba Cloud[2], and Tencent Cloud[3]. In our local machine, the source model is DN201, and the average

[1]https://cloud.baidu.com/product/imagerecognition/general
[2]https://vision.console.aliyun.com
[3]https://console.cloud.tencent.com/tiia/detectlabel

Fig. 7. Visualization and frequency spectrum histograms under source model RN50. **Left:** The 1st, 2nd, and 3rd rows show the benign images, CAE, and CAE_τ, respectively; **Right:** The frequency spectrum histograms of benign image, CAE, and CAE_τ.

TABLE V
COMPARISON OF IMPERCEPTIBLE METRICS. SOURCE MODEL: RN50.

Attacks		PSNR ↑	SSIM ↑	LPIPS ↓
Transferable Attacks	DIM	24.97	0.71	0.47
	TIM	24.93	0.81	0.43
	SSA	24.80	0.70	0.48
	PGN	24.61	0.72	0.47
	ILPD	23.72	0.64	0.53
	HFA	23.35	0.60	0.52
	BSR-MI	24.89	0.71	0.49
	BSR-TI-DIM	24.71	0.74	0.48
	BSR-SI-TI-DIM	24.63	0.73	0.48
	FAUG	23.72	0.64	0.51
	CDA	23.92	0.68	0.54
	BIA	23.88	0.69	0.55
	GE-AdvGAN	23.58	0.66	0.54
	CAA	24.64	0.69	0.41
	CAA_τ	23.20	0.64	0.40
Imperceptible Attacks	AdvDrop	28.73	0.93	0.27
	SSHA	30.31	0.98	0.18
	AdvAD	30.45	0.98	0.17

TABLE VI
COMPARING OF ASRS (%) AGAINST REAL-WORLD MODELS.

APIs	SSA	CDA	PGN	HFA	ILPD	BSR-MI	CAA	CAA_τ
Baidu	77	74	73	70	65	78	**81**	82
Alibaba	68	72	69	68	55	77	**88**	80
Tencent	69	81	68	67	66	67	**89**	82

TABLE VII
THE ASRS (%) AGAINST DEFENSE METHODS. **TOP:** SOURCE MODEL RN50. **BOTTOM:** SOURCE MODEL DN201.

Attack	Bit-RD	R&P	HGD	NIPS-r3	JPEG	FD	DiffPure	OSCP	AVG.
DIM	78.7	79.9	60.2	66.6	75.4	79.5	27.4	22.5	61.3
TIM	50.0	52.8	24.8	20.2	54.9	64.3	17.7	17.4	37.8
SSA	88.3	84.9	69.8	84.4	86.3	88.3	32.8	28.1	70.4
CDA	86.9	80.4	86.9	82.9	73.6	85.3	31.9	35.2	70.4
PGN	86.0	84.3	70.7	77.3	86.0	88.3	38.7	33.3	70.6
HFA	84.5	83.5	64.6	80.8	82.8	87.6	42.8	39.1	70.7
GE-AdvGAN	62.3	49.3	39.7	51.0	57.4	69.1	14.2	18.9	45.2
ILPD	83.0	89.4	78.1	90.8	89.6	75.5	46.3	41.9	74.3
BSR-MI	**89.3**	**92.2**	82.8	91.2	88.7	90.9	47.5	37.5	77.5
BSR-TI-DIM	85.8	91.0	81.6	81.0	87.5	91.7	49.0	40.4	76.0
BSR-SI-TI-DIM	67.2	90.5	78.4	75.6	86.1	**92.0**	**49.1**	38.3	72.2
BIA	87.8	79.5	84.8	81.6	75.5	87.2	33.1	31.2	70.1
FAUG	80.5	80.5	47.0	69.7	78.0	83.3	27.0	17.6	60.5
CAA	87.8	88.4	**98.8**	**93.4**	**91.0**	85.1	47.9	**53.4**	**80.7**
CAA_τ	89.6	90.4	98.9	95.4	92.3	88.6	51.5	57.6	82.9
DIM	77.9	80.9	61.6	65.9	77.7	81.8	28.9	30.8	63.2
TIM	52.7	59.6	28.4	23.3	58.3	67.5	23.3	21.1	41.8
SSA	91.0	90.0	82.6	87.6	90.0	91.2	48.2	44.4	78.1
CDA	85.4	84.7	89.9	81.4	78.5	85.9	30.4	33.6	71.2
PGN	89.9	89.6	80.6	83.6	90.1	91.5	54.4	48.4	78.5
HFA	84.5	87.6	75.8	82.2	86.5	89.0	56.2	49.9	76.5
GE-AdvGAN	80.2	71.4	62.1	82.4	80.2	80.1	31.9	36.0	65.5
ILPD	87.2	85.0	66.5	76.9	83.6	84.7	34.4	35.9	69.3
BSR-MI	90.3	92.6	82.9	88.6	90.2	92.2	39.1	44.5	77.6
BSR-TI-DIM	83.3	90.7	78.7	74.7	87.1	91.8	44.3	42.1	74.1
BSR-SI-TI-DIM	62.7	89.1	73.8	67.3	84.9	91.9	44.6	40.6	69.4
BIA	87.2	83.5	88.3	78.9	79.6	87.4	33.8	35.6	71.8
FAUG	82.7	85.1	64.2	72.3	82.1	86.2	36.7	35.7	68.1
CAA	**93.4**	**93.2**	**94.3**	**93.3**	**91.0**	**92.9**	**60.8**	**60.5**	**85.3**
CAA_τ	94.2	93.9	94.6	93.8	93.6	94.2	63.6	62.9	86.9

CSR on 9 normally trained models (as shown in Table III) is higher than 70%. That is, the CAEs is visually similar to benign images while behaving normally. However, once preprocessing is executed in the cloud-based deep learning pipeline, the CAE turns into an AE, misleading the classifier to make a wrong prediction. To have a better comparison we choose 6 well-performing baselines to craft AEs and feed them to the above APIs. We consider an attack successful if the ground-truth label of a clean sample is not present in the top-5 list of the APIs' predictions. To reduce deviation, we randomly select 100 AEs or CAEs generated by each attack for testing. From Table VI, we can see that CAA always performs best among all competitors. This further validates our insight, revealing the potential security risks introduced by data preprocessing operations in adversarial attacks. Besides, we find that CAA achieves a higher ASR than CAA_τ. This is possibly because the camouflage features of CAA_τ are not fully filtered out.

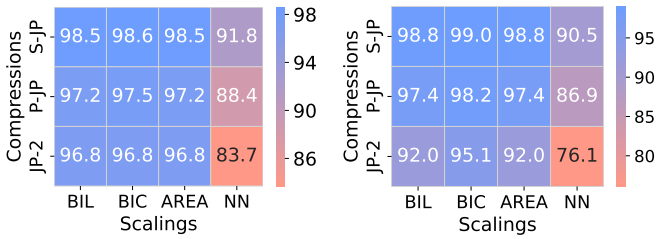


Fig. 8. Average ASRs (%) under varying JPEG/scaling algorithms (source model DN201). S-JP: Standard JPEG, P-JP: Progressive JPEG, JP-2: JPEG 2000; BIL: Bilinear Interpolation, BIC: Bicubic Interpolation, AREA: Area Interpolation, NN: Nearest Neighbor Interpolation. The source model is DenseNet201. **Left:** Results on CAA and **Right:** Results on CAA_τ.

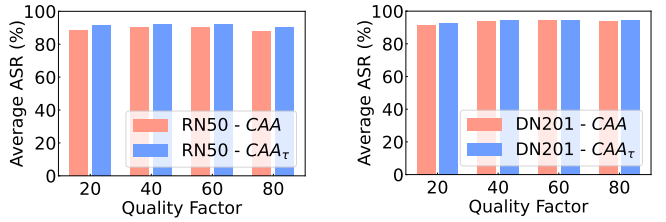


Fig. 9. Average ASR (%) under varying quality factors (image scale is fixed to 448×448). **Left:** Source model RN50. **Right:** Source model DN201.

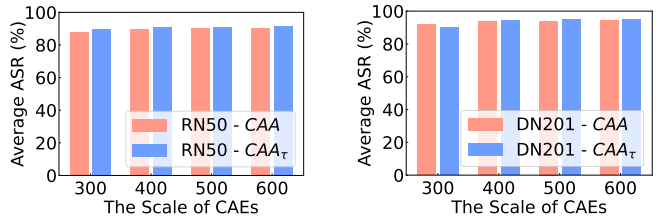


Fig. 10. Average ASR (%) under varying input scales (quality factor is fixed to 50). **Left:** Source model RN50. **Right:** Source model DN201.

C. Robustness

Adversarial Robustness. We compare the ASRs between CAA and transferable baseline attacks against 8 defense methods: Bit-RD [5], R&P [48], HGD [49], NIPS-r3 [70], JPEG [46], FD [47], DiffPure [51], and OSCP [71]. Table VII shows that CAA has superior robustness against defense methods. Especially for HGD, CAA achieves the best ASR of 94.3%. This is because the attack perturbations mainly lie in the low-frequency domain, which are hard to be removed by denoising-based defenses like HGD. Besides, target models may adopt different scaling or compression algorithms and parameters, and we further test the average ASRs of CAA against 9 normally trained models (as shown in Table III) under varying settings. From Fig. 8-Fig. 10, we can see that CAA maintains strong attack performance across different scaling/compression algorithms, and the ASRs show negligible fluctuations with the change of quality factors or input scales. This benefits from the use of random compression ratios and random input sizes during the generator training and iterative optimization stages.

Camouflaged Robustness. Apart from scaling and compression, the deep learning pipeline may perform other preprocessing operations. Therefore, we evaluate the robustness of CAEs against the most common preprocessing operations adopted by popular deep learning frameworks. Specifically, we evaluate the attack performance when extra preprocessing operations are applied before the scaling and compression. Table VIII shows that CAA keeps an ASR over 89.0% when cropping, padding, filtering, sharpening, contrast enhancement (CE), or normalization is executed together with scaling and compression, further validating its camouflaged robustness.

TABLE VIII
AVERAGE ASRS (%) WHEN ADDITIONAL PREPROCESSING OPERATIONS ARE EXECUTED BEFORE TRIGGERING.

Models	Attack	Cropping	Padding	Filtering	Sharpening	CE	Normalization	AVG.
RN50	CAA	87.5	86.9	93.6	88.2	95.0	99.0	91.7
	CAA_τ	85.5	86.6	95.3	91.3	95.8	98.7	92.2
DN201	CAA	85.7	86.4	93.1	98.5	94.9	98.4	92.8
	CAA_τ	72.2	78.3	89.3	99.2	96.0	98.9	89.0

D. Comparisons with Other Attacks

Imperceptible Attacks. At the high-level view, the purpose of Stage 2 is similar to that of imperceptible attacks, i.e., changing the labels of input samples by manipulating high-frequency components. The only difference is that Stage 2 aims to change the fake labels back to ground-truth labels, while imperceptible attacks aim to change the ground-truth labels to fake labels. Therefore, we compare the CSRs of our CAA with the ASRs of imperceptible attacks. As shown in Table IX, AdvDrop, SSHA, and our CAA all demonstrate excellent performance in white-box attacks, yet the transferability of the contrastive methods is significantly lower than that of our CAA. This is because those methods aim to ensure the imperceptibility of adversarial samples without limiting the strength of high-frequency perturbations, and thus easy to overfit the source model. In contrast, our CAA adopts feature-level camouflage and uses triplet loss to enhance the high-frequency transferability. Besides, recent AdvAD employed diffusion model to generate natural AEs under unconstrained perturbations, but still lack adversarial transferability.

Scaling Attacks. We first compare our CAA with scaling attack in terms of the visual effects after image scaling. From the first two rows of Fig. 11, we can see that after scaling, the camouflage image generated through SA is completely transformed into an entirely different image, whereas the CAE generated by our CAA exhibits no noticeable visual changes. For SA, such dramatic dissimilarity before and after scaling can easily be detected by human eyes. But our CAA can effectively resist human inspection. Next, we compare our CAA with SA in terms of the robustness against camouflage detection methods. Decamouflage [12] as the SOTA detection method against camouflage attacks proposed two independent detection strategies: *filter detection* and *steganalysis detection*. From their experimental results, we can know that: (1) For benign images, there is no significant visual difference before and after filter operations; (2) There is only a central spectral point when benign samples are transformed into Fourier spectra by frequency-domain-based steganalysis. From the third and forth rows of Fig. 11, we can observe that both the maximum and minimum filtering can expose triggers hidden in the camouflage image generated by the SA, resulting in significant

TABLE IX
THE ASRS (%) OF IMPERCEPTIBLE BASELINE ATTACKS AND THE CSRS (%) OF OUR CAA ON NINE NORMALLY TRAINED MODELS.

Models	Attack	VGG16	VGG19	ResNet50	ResNet152	DenseNet121	DenseNet169	DenseNet201	Visformer	Swin	AVG.
RN50	AdvDrop	4.2	4.3	96.8*	2.9	4.5	3.8	3.0	0.9	0.6	13.4
	SSHA	0.9	0.9	97.3*	0.6	1.0	0.4	0.6	0.1	0.1	11.3
	AdvAD	2.2	3.9	99.8*	2.7	3.5	3.2	2.6	0.4	0.5	13.2
	CAA	54.5	49.0	98.1*	75.5	56.0	68.4	72.1	66.9	84.5	69.4
	CAA_τ	**62.1**	**60.5**	98.1*	**94.0**	**82.3**	**86.2**	**87.6**	**67.1**	**84.7**	**80.3**
DN201	AdvDrop	4.2	4.4	4.0	3.3	7.2	7.7	95.9*	1.4	1.1	14.4
	SSHA	7.0	3.5	5.0	2.5	7.3	6.5	99.1*	0.4	0.9	14.7
	AdvAD	2.2	3.5	3.5	2.6	8.3	8.9	99.7*	0.2	0.6	14.4
	CAA	49.3	50.8	68.0	71.0	73.8	73.2	97.7*	34.1	49.4	63.0
	CAA_τ	**58.5**	**65.3**	**87.8**	**90.5**	**91.1**	**92.4**	97.7*	**53.3**	**55.1**	**76.9**

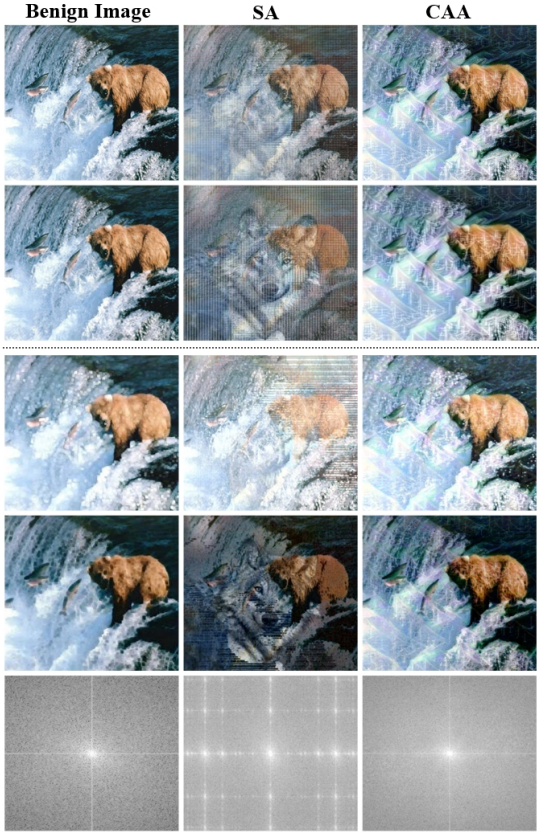


Fig. 11. Comparison results of scaling attack (SA) and CAA. The 1st row shows the benign image, the camouflage image of SA, and the CAE generated by CAA. The 2nd row depicts the results after image scaling. The 3rd and 4th rows show the filtering detection results using maximum and minimum filtering respectively. The 5th row shows the steganography detection results.

visual differences from benign image. In contrast, even the most effective minimum filtering cannot detect anomalies from the CAE generated by our CAA. From the last row of Fig. 11, we can see that after steganalysis, the camouflage image generated by the SA present multiple central spectral points, but those by our CAA present only one central point. Hence, our CAA can successfully evade camouflage detection.

E. Ablation Studies

In the ablation studies, we adopt RN50 as the source model and use the average ASRs and CSRs on 9 normally and 3 adversarially trained models (as shown in Table III) to assess the effectiveness of key components and hyper-parameters. The detail analyses are shown in Table XVII of the Appendix.

TABLE X
ABLATION STUDY ON THE HFR MODULE AND TRIPLET LOSS.

Attack	Normally trained models				Adversarily trained models			
	w/o HFR		w/o $\mathcal{L}_{tr}$		w/o HFR		w/o $\mathcal{L}_{tr}$	
	ASR	CSR	ASR	CSR	ASR	CSR	ASR	CSR
CAA	↓4.3	↑4.3	↓0.5	↓6.5	↓26.6	↓1.6	↓0.5	↓1.0
CAA_τ	↓0.7	↓2.8	↓1.6	↓4.0	↓17.8	↓10.9	↓0.8	↓0.4

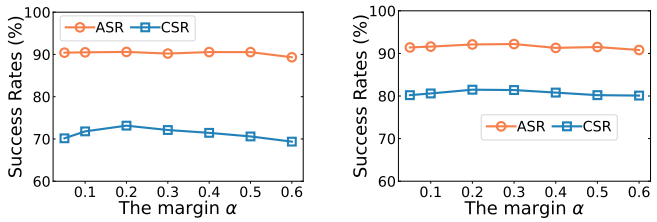


Fig. 12. Ablation study on margin α of triplet loss defined in Eq. (11). **Left:** Results on CAA. **Right:** Results on CAA_τ.

The HFR Module. Table X shows that the HFR module can significantly improve the ASRs on both normally and adversarially trained models. Especially for adversarially trained models, this module plays a key role, improving ASRs and CSRs dramatically. Besides, we also observe that for normally trained models, this module plays a negative influence on CSRs. The reason is that the HFR module may inhibit the amount of high-frequency camouflage perturbations, which also explains why CAA_τ is less affected.

The Triplet Loss. From Table X, we can see that the triplet loss can significantly improve the CSRs on normally trained models while well maintaining the ASRs. This confirms that the triplet loss not only contributes to promoting the transferability of camouflage perturbations, but also helps to strike a balance between camouflage and transferability. But the triplet loss has minor effects on the CSRs of adversarially trained models, which have more robust decision boundaries than the normally trained ones. From Fig. 12, we can see that our CAA consistently yields the best performance when $\alpha = 0.2$.

The Impact of Denoising Module. From Table XI, we can observe that this module can largely improve both the ASRs and CSRs against adversarially trained models. But for normally trained models, the denoising module will cause a

TABLE XI
ABLATION STUDY ON THE IMPACT OF DENOISING FUNCTION.

Attack	Normally trained models w/o denoising		Adversarially trained models w/o denoising	
	ASR	CSR	ASR	CSR
CAA	↓10.1	↑9.7	↓14.3	↓17.2
CAA_τ	↓ 5.8	↓3.5	↓ 9.2	↓29.2

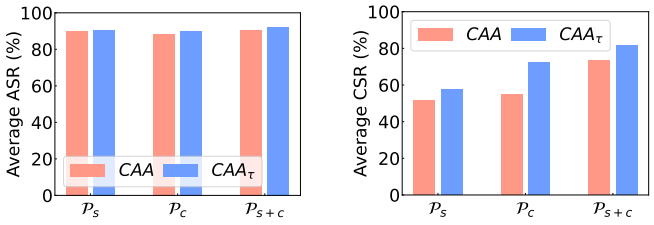


Fig. 13. Ablation study on JPEG/Scaling operations.

large decline in CSRs. The reason is that after denoising the gradients of input images are more smooth, which facilitates the effects of HFR module. For camouflage effects, this module has a similar function as the HFR module.

JPEG/Scaling Operations. Let $\mathcal{P}_s$, $\mathcal{P}_c$ and $\mathcal{P}_{s+c}$ denote the scaling, JPEG compression, and scaling plus compression operations, respectively. Each operation type will be simultaneously performed on source and target models. From Fig. 13, we can observe that $\mathcal{P}_{s+c}$ allows our CAA to achieve the best attack and camouflage performances. This occurs because compared with a single preprocessing operation, $\mathcal{P}_{s+c}$ can drop more high-frequency information, leaving more room for injecting camouflage perturbations. For a similar reason, different JPEG/scaling operations mainly impact the CSRs.

Hyperparameters. λ_1 controls the contributions of benign feature similarity loss and triplet loss, while λ_2 regulates adversarial feature distance loss. Therefore, λ_1 contributes to camouflage effects and λ_2 is related to attack effects. From Fig. 14, we can observe that these hyperparameters significantly affect the performance of CAA. For instance, λ_2 increases, the ASR shows an increasing trend, but the CSR significantly decreases. As shown in Fig. 15, we can observe that the hyperparameters affect the ASRs and CSRs of CAA_τ. For example, when λ_1 remains constant, the CSR of CAA_τ shows a relatively noticeable variation compared to ASRs. Compared to CAA, the ASRs and CSRs of CAA_τ show relatively smoother trends. This is because the unrestricted camouflage perturbations reduce the influence of hyperparameters. As a trade-off, we set $\lambda_1 = 150$, and $\lambda_2 = 1.0$ in the experiments. Note that $\mathcal{L}_{bf}$ and $\mathcal{L}_{tr}$ play the same role of making CAE close to benign samples. Thus, binding them together in the optimization helps to stabilize gradients and avoid overfitting. Fig. 24 of the Appendix further validated the effectiveness of this setting.

F. *Efficiency Analysis.*

To evaluate the practicality of CAA, we compare the runtime and resource consumption with 11 baselines. As shown in Table XII, Stage 1 of CAA requires approximately

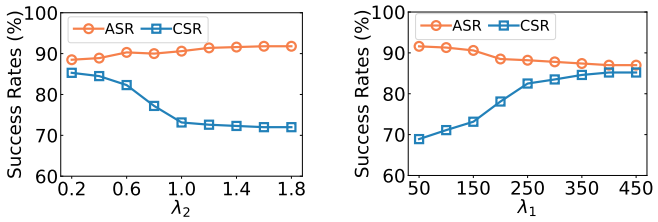


Fig. 14. Ablation study on the hyperparameters of CAA. **Left:** λ_1 is fixed to 150. **Right:** λ_2 is fixed to 1.0.

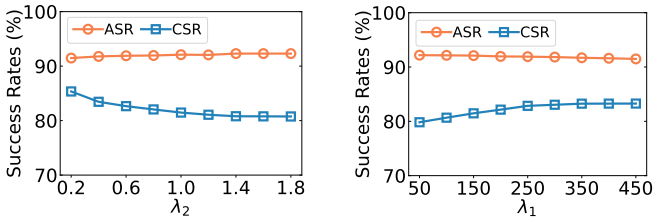


Fig. 15. Ablation study on the hyperparameters of CAA_τ. **Left:** λ_1 is fixed to 150. **Right:** λ_2 is fixed to 1.0.

TABLE XII
COMPARISON OF RUNTIME AND RESOURCE CONSUMPTION.

Attack	GPU Memory (GB)	Training Time (h)	Iteration Time (s / sample)	Inference Time (ms / sample)
DIM	2.4	–	0.03	–
TIM	2.4	–	0.03	–
SSA	2.5	–	0.34	–
PGN	2.5	–	0.41	–
HFA	3.8	–	0.80	–
ILPD	2.5	–	0.22	–
BSR-MI	20.7	–	0.25	–
BSR-TI-DIM	20.7	–	0.26	–
BSR-SI-TI-DIM	20.7	–	0.26	–
FAUG	4.9	–	0.10	–
CDA	6.2	12.3	–	5.2
GE-AdvGAN	5.1	25.3	–	6.7
BIA	7.5	15.6	–	5.5
CAA	10.0 (Stage 1) 11.9 (Stage 2)	15.1	1.6 (Stage 2)	5.3 (Stage 1)

batch size = 10. source model: RN50. Among all baselines, only CDA, GE-AdvGAN, and BIA are GAN-based methods.

15.1 (h), which is notably more efficient than GE-AdvGAN (25.3 h) and comparable to BIA (15.6 h). Once the GAN is trained, the inference time is negligible. Stage 2 of CAA iteratively optimizes each input and takes about 1.6 (s) per sample, which is higher than iteration-based baselines. The main reason is that Stage 2 requires more iterations ($T = 100$) to find the balance between camouflage and transferability, while the iteration-based baselines requires only 10 iterations to optimize perturbations. It is worth noticing that the more iterations in Stage 2, the lower the ASR, but higher the CSR. As shown in Fig. 16, when $T = 10$ (running time is 0.24 s/sample), the average ASR rises to 91.6%, but the CSR drops to 37.2%; when $T = 0$, CAA performs only Stage 1 and degrades into traditional adversarial attacks. At this time, the average ASR rises to 92.8%, but the CSR drops to 7.2%.

In terms of GPU memory usage, Stage 1 of CAA consumes more resources than GAN-based baselines. This is because CAA takes images of larger size (448×448) as input compared with baselines' input size (224×224). For the same reason, Stage 2 of CAA also consumes more resources than most of the iteration-based baselines. But Stage 2 of CAA is

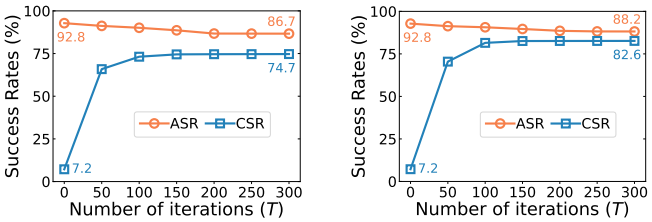


Fig. 16. Effect of iteration number in Stage 2. $T = 0$ means running Stage 1 only. **Left:** Results on CAA. **Right:** Results on CAA_τ. Source model: RN50.

significantly more efficient than the SOTA iterative method, BSR, which requires over 20 GB of memory to maintain multiple transformed inputs. Overall, the comparison results indicate that CAA achieves great camouflage while preserving superior adversarial transferability at moderate costs, thus demonstrating its feasibility for real-world applications.

VII. Discussion

A. Difference from Backdoor Attacks.

While CAA shares the notion of "trigger activation" with backdoor attacks, the two consider different threat models:

- **Adversarial Knowledge.** Unlike CAA launching black-box attacks, backdoor attacks have white-box access to the training data, training processes, and model weights.
- **Attack Goals.** Unlike CAA aiming to achieve camouflage and transferability at once, backdoor attacks focus on making the poisoned model predict the attacker-chosen labels only when getting inputs with a specific trigger.
- **Attack Phase.** Backdoor attacks take place in the training phase, requiring modifying the model parameters. Instead, our CAA launches attacks in the inference phase.

Compared with our CAA, backdoor attacks have the following characteristics: *(1) Lacking Transferability.* Backdoor attacks are effective only for poisoned models, and cannot transfer to unseen models. *(2) Lacking Camouflage.* In backdoor attacks, the benign samples and poisoned counterparts have differentiable characteristics in the feature space, and thus can be detected by feature-space defenses [72]. *(3) Visible Trigger.* Traditional backdoor attacks typically use simple patterns of fixed-sizes as their triggers, which can be easily caught by human inspection and input detection [73], [74].

B. Attack Feasibility

In practical scenarios, the target model may or may not execute the specified preprocessing operations. If the target model does not perform specified preprocessing operation, CAEs will not be activated, failing to achieve an attacking effect. Following scaling attack [11], we propose Alg. 2 to infer the preprocessing operations adopted by target models to increase ASR. In a nutshell, this algorithm infers the executed preprocessing operations by analyzing the prediction consistency between benign samples and customized CAEs. We can set the accuracy threshold γ below the average CSR (e.g., $\gamma = 0.5$) to avoid missing potential preprocessing operations. It is worth noticing that the "inference granularity" can be adjusted by setting the preprocessing operation set $\mathbf{P}$. A larger

Algorithm 2 Preprocessing Operations Inference

Require: Target model f_t (e.g., cloud vision API), preprocessing operations $\mathbf{P} = \{p_1, p_2, \dots\}$, benign images $X = \{x_i\}_{i=1}^N$, accuracy threshold γ
Ensure: Inferred preprocessing algorithm $\mathbf{p}^*$

```
p* ← ∅
for p_j ∈ P do
    Generate camouflaged adversarial examples X_cae^{p_j} = {x̃_i^{p_j}}_{i=1}^N from X via CAA tailored to p_j
    ACC_j ← (1/N) Σ_{i=1}^N 𝕀[arg max f_t(x_i) == arg max f_t(x̃_i^{p_j})]
    if ACC_j < γ then
        p* ← p_j   ▷ Add candidate preprocessing algorithm
        γ ← ACC_j
    end if
end for
return p*
```

set size means more inference details and higher inference cost. For example, we can rapidly infer which preprocessing operations (scaling, JPEG compression, or both) are performed by the target model, or spend more time to infer the specific scaling/compression algorithm details. Given the output of Alg. 2, we can choose the optimal configurations to craft the CAEs. Specifically, we mainly consider the following cases:

(1) The target model executes scaling or JPEG compression or both ($\mathbf{p}^* \neq \emptyset$). In this case, CAA uses the optimal preprocessing operations in $\mathbf{p}^*$ to craft CAEs.

(2) The target model does not execute scaling nor JPEG compression ($\mathbf{p}^* = \emptyset$). In this case, we can locally perform preprocessing on the CAEs and send the preprocessed CAEs to target models. In this way, CAA achieves the same effect as traditional adversarial attacks. Another more effective way is to just run the first stage of CAA and send the preprocessed RAEs to target models directly. From Fig. 16, we can see that running only Stage 1 achieves an average ASR of over 92%, yielding stronger attacker performance than running two stages.

VIII. Conclusion

In this paper, we introduce a novel camouflaged adversarial attack, which hides the adversarial properties behind image preprocessing to pass label checking and adversarial detection. We design a two-stage optimization strategy, making CAEs disguise as benign images, but implicitly turn into AEs after preprocessing. Extensive experiments validate the effectiveness. This work is the first attempt to exploit the feasibility of using preprocessing as a trigger to craft camouflaged AEs. We expect our work can draw attention to the potential threat caused by integrating preprocessing into adversarial attacks.

References

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *Proc. of ICLR*, 2014.
[2] K. Sun, T. Yao, S. Chen, S. Ding, J. Li, and R. Ji, "Dual contrastive learning for general face forgery detection," in *Proc. of AAAI*, 2022.
[3] C. Choi, J. H. Choi, J. Li, and S. Malla, "Shared cross-modal trajectory prediction for autonomous driving," in *Proc. of CVPR*, 2021.

[4] S. Wang, Q. Liu, Y. Xu, H. Jiang, J. Wu, T. Wang, T. Peng, and G. Wang, "Protecting inference privacy with accuracy improvement in mobile-cloud deep learning," *IEEE Transactions on Mobile Computing*, vol. 23, no. 6, pp. 6522–6537, 2023.

[5] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *Proc. of NDSS*, 2018.

[6] G. Vacanti and A. Van Looveren, "Adversarial detection and correction by matching prediction distributions," *arXiv preprint arXiv:2002.09364*, 2020.

[7] F. Pérez-García, R. Sparks, and S. Ourselin, "Torchio: a python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning," *Computer Methods and Programs in Biomedicine*, vol. 208, p. 106236, 2021.

[8] Y. Sharma, G. W. Ding, and M. A. Brubaker, "On the effectiveness of low frequency perturbations," in *Proc. of IJCAI*, 2019.

[9] K. Sayood, *Introduction to data compression*. Morgan Kaufmann, 2017.

[10] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proc. of CVPR*, 2015.

[11] Q. Xiao, Y. Chen, C. Shen, Y. Chen, and K. Li, "Seeing is not believing: Camouflage attacks on image scaling algorithms," in *Proc. of USENIX Security*, 2019.

[12] B. Kim, A. Abuadbba, Y. Gao, Y. Zheng, M. E. Ahmed, S. Nepal, and H. Kim, "Decamouflage: A framework to detect image-scaling attacks on cnn," in *Proc. of DSN*, 2021.

[13] Z. Zhao, H. Zhang, R. Li, R. Sicre, L. Amsaleg, M. Backes, Q. Li, Q. Wang, and C. Shen, "Revisiting transferable adversarial images: Systemization, evaluation, and new insights," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–16, 2025.

[14] C. Xie, Z. Zhang, Y. Zhou, S. Bai, J. Wang, Z. Ren, and A. L. Yuille, "Improving transferability of adversarial examples with input diversity," in *Proc. of CVPR*, 2019.

[15] Y. Dong, T. Pang, H. Su, and J. Zhu, "Evading defenses to transferable adversarial examples by translation-invariant attacks," in *Proc. of CVPR*, 2019.

[16] J. Lin, C. Song, K. He, L. Wang, and J. E. Hopcroft, "Nesterov accelerated gradient and scale invariance for adversarial attacks," in *Proc. of ICLR*, 2020.

[17] K. Wang, X. He, W. Wang, and X. Wang, "Boosting adversarial transferability by block shuffle and rotation," in *Proc. of CVPR*, 2024.

[18] W. Wu, Y. Su, X. Chen, S. Zhao, I. King, M. R. Lyu, and Y.-W. Tai, "Boosting the transferability of adversarial samples via attention," in *Proc. of CVPR*, 2020.

[19] Q. Zhang, X. Li, Y. Chen, J. Song, L. Gao, Y. He, and H. Xue, "Beyond imagenet attack: Towards crafting adversarial examples for black-box domains," in *Proc. of ICLR*, 2022.

[20] Q. Li, Y. Guo, W. Zuo, and H. Chen, "Improving adversarial transferability via intermediate-level perturbation decay," in *Proc. of NeurIPS*, 2023.

[21] Q. Li, Q. Hu, H. Fan, C. Lin, C. Shen, and L. Wu, "Attention-sa: Exploiting model-approximated data semantics for adversarial attack," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 8673–8684, 2024.

[22] D. Wang, W. Yao, T. Jiang, X. Zheng, and J. Wu, "Improving the transferability of adversarial examples by feature augmentation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 9, pp. 17 212–17 226, 2025.

[23] Y. Tsuzuku and I. Sato, "On the structural sensitivity of deep convolutional networks to the directions of fourier basis functions," in *Proc. of CVPR*, 2019.

[24] Y. Qian, K. Chen, B. Wang, Z. Gu, S. Ji, W. Wang, and Y. Zhang, "Enhancing transferability of adversarial examples through mixed-frequency inputs," *IEEE Transactions on Information Forensics and Security*, 2024.

[25] Y. Long, Q. Zhang, B. Zeng, L. Gao, X. Liu, J. Zhang, and J. Song, "Frequency domain model augmentation for adversarial attack," in *Proc. of ECCV*, 2022.

[26] Z. Zhu, H. Chen, X. Wang, J. Zhang, Z. Jin, K.-K. R. Choo, J. Shen, and D. Yuan, "Ge-advgan: Improving the transferability of adversarial samples by gradient editing-based adversarial generative model," in *Proc. of SDM*, 2024.

[27] Z. Zhu, X. Wang, Z. Jin, J. Zhang, and H. Chen, "Enhancing transferable adversarial attacks on vision transformers through gradient normalization scaling and high-frequency adaptation," in *Proc. of ICLR*, 2024.

[28] M. Naseer, S. H. Khan, M. H. Khan, F. S. Khan, and F. Porikli, "Cross-domain transferability of adversarial perturbations," in *Proc. of NeurIPS*, 2019.

[29] Z. Ge, H. Liu, W. Xiaosen, F. Shang, and Y. Liu, "Boosting adversarial transferability by achieving flat local maxima," in *Proc. of NeurIPS*, 2023.

[30] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proc. of CVPR*, 2018.

[31] B. Tang, Z. Wang, Y. Bin, Q. Dou, Y. Yang, and H. T. Shen, "Ensemble diversity facilitates adversarial transferability," in *Proc. of CVPR*, 2024.

[32] R. Wang, Y. Guo, and Y. Wang, "Ags: Affordable and generalizable substitute training for transferable adversarial attack," in *Proc. of AAAI*, 2024.

[33] S. Xia, W. Yang, Y. Yu, X. Lin, H. Ding, L. Duan, and X. Jiang, "Transferable adversarial attacks on sam and its downstream models," in *Proc. of NeurIPS*, 2024.

[34] Z. Li, W. Wang, J. Li, K. Chen, and S. Zhang, "Ucg: A universal cross-domain generator for transferable adversarial examples," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 3023–3037, 2024.

[35] R. Duan, Y. Chen, D. Niu, Y. Yang, A. K. Qin, and Y. He, "Advdrop: Adversarial attack to dnns by dropping information," in *Proc. of ICCV*, 2021.

[36] C. Luo, Q. Lin, W. Xie, B. Wu, J. Xie, and L. Shen, "Frequency-driven imperceptible adversarial attack on semantic similarity," in *Proc. of CVPR*, 2022.

[37] J. Li, Z. He, A. Luo, J.-F. Hu, Z. J. Wang, and X. Kang, "Advad: Exploring non-parametric diffusion for imperceptible adversarial attacks," *Proc. of NeurIPS*, 2024.

[38] J. Chen, H. Chen, K. Chen, Y. Zhang, Z. Zou, and Z. Shi, "Diffusion models for imperceptible and transferable adversarial attack," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[39] R. Duan, X. Ma, Y. Wang, J. Bailey, A. K. Qin, and Y. Yang, "Adversarial camouflage: Hiding physical-world attacks with natural styles," in *Proc. of CVPR*, 2020.

[40] Y.-C.-T. Hu, B.-H. Kung, D. S. Tan, J.-C. Chen, K.-L. Hua, and W.-H. Cheng, "Naturalistic physical adversarial patch for object detectors," in *Proc. of ICCV*, 2021.

[41] W. Zhu, X. Ji, Y. Cheng, S. Zhang, and W. Xu, "{TPatch}: A triggered physical adversarial patch," in *Proc. of USENIX Security*, 2023.

[42] K. Uddin, Y. Yang, T. H. Jeong, and B. T. Oh, "A robust open-set multi-instance learning for defending adversarial attacks in digital image," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 2098–2111, 2023.

[43] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *Proc. of ICML*, 2019.

[44] Y. Yu, S. Xia, X. Lin, C. Kong, W. Yang, S. Lu, Y.-P. Tan, and A. C. Kot, "Towards model resistant to transferable adversarial examples via trigger activation," *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 3745–3757, 2025.

[45] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *Proc. of ICLR*, 2018.

[46] C. Guo, M. Rana, M. Cisse, and L. van der Maaten, "Countering adversarial images using input transformations," in *Proc. of ICLR*, 2018.

[47] Z. Liu, Q. Liu, T. Liu, N. Xu, X. Lin, Y. Wang, and W. Wen, "Feature distillation: Dnn-oriented jpeg compression against adversarial examples," in *Proc. of CVPR*, 2019.

[48] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille, "Mitigating adversarial effects through randomization," in *Proc. of ICLR*, 2018.

[49] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, "Defense against adversarial attacks using high-level representation guided denoiser," in *Proc. of CVPR*, 2018.

[50] M. Naseer, S. Khan, M. Hayat, F. S. Khan, and F. Porikli, "A self-supervised approach for adversarial robustness," in *Proc. of CVPR*, 2020.

[51] W. Nie, B. Guo, Y. Huang, C. Xiao, A. Vahdat, and A. Anandkumar, "Diffusion models for adversarial purification," in *Proc. of ICML*, 2022.

[52] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *Proc. of ICML*, 2019.

[53] S. Gowal, K. D. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, R. Arandjelovic, T. Mann, and P. Kohli, "Scalable verified training for provably robust image classification," in *Proc. of ICCV*, 2019.

[54] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. of NeurIPS*, 2014.

[55] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proc. of CVPR*, 2006.

[56] Q. Duan, Z. Hua, Q. Liao, Y. Zhang, and L. Y. Zhang, "Conditional backdoor attack via jpeg compression," in *Proc. of AAAI*, 2024.

[57] 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. of ICCV*, 2017.

[58] A. Dogra and P. Bhalla, "Image sharpening by gaussian and butterworth high pass filter," *Biomedical and Pharmacology Journal*, vol. 7, no. 2, pp. 707–713, 2014.

[59] Z. Wei, J. Chen, Z. Wu, and Y.-G. Jiang, "Enhancing the self-universality for transferable targeted attacks," in *Proc. of CVPR*, 2023.

[60] Z. Jia, H. Fang, and W. Zhang, "MBRS: enhancing robustness of dnn-based watermarking by mini-batch of real and simulated JPEG compression," in *Proc. of MM*, 2021.

[61] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. of ICLR*, 2015.

[62] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of CVPR*, 2016.

[63] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. of CVPR*, 2017.

[64] Z. Chen, L. Xie, J. Niu, X. Liu, L. Wei, and Q. Tian, "Visformer: The vision-friendly transformer," in *Proc. of ICCV*, 2021.

[65] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. of ICCV*, 2021.

[66] B. Graham, A. El-Nouby, H. Touvron, P. Stock, A. Joulin, H. Jégou, and M. Douze, "Levit: a vision transformer in convnet's clothing for faster inference," in *Proc. of ICCV*, 2021.

[67] A. Vaswani, P. Ramachandran, A. Srinivas, N. Parmar, B. Hechtman, and J. Shlens, "Scaling local self-attention for parameter efficient visual backbones," in *Proc. of CVPR*, 2021.

[68] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. of CVPR*, 2016.

[69] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proc. of AAAI*, 2017.

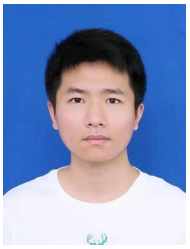
[70] A. Thomas, "Nips: Defense against adversarial attack," https://github.com/anlthms/nips-2017/tree/master/mmd, 2017.

[71] C. T. Lei, H. M. Yam, Z. Guo, Y. Qian, and C. P. Lau, "Instant adversarial purification with adversarial consistency distillation," in *Proc. of CVPR*, 2025.

[72] X. Xu, K. Huang, Y. Li, Z. Qin, and K. Ren, "Towards reliable and efficient backdoor trigger inversion via decoupling benign features," in *Proc. of ICLR*, 2024.

[73] J. Guo, Y. Li, X. Chen, H. Guo, L. Sun, and C. Liu, "Scale-up: An efficient black-box input-level backdoor detection via analyzing scaled prediction consistency," in *Proc. of ICLR*, 2023.

[74] X. Mo, Y. Zhang, L. Y. Zhang, W. Luo, N. Sun, S. Hu, S. Gao, and Y. Xiang, "Robust backdoor detection for deep learning via topological evolution dynamics," in *Proc. of S&P*, 2024.



Yipeng Zou is working toward the Ph.D. degree with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. His research interests include artificial intelligence security, trustworthy machine learning, and deep adversarial learning.

Qin Liu (Member, IEEE) received her B.Sc. in Computer Science in 2004 from Hunan Normal University, China, received her M.Sc. in Computer Science in 2007, and received her Ph.D. in Computer Science in 2012 from Central South University, China. She has been a Visiting Student at Temple University, USA. Her research interests include artificial intelligence security, and security and privacy issues in cloud computing. Now, she is an Associate Professor in the College of Computer Science and Electronic Engineering at Hunan University, China.

Jie Wu (Fellow, IEEE) is the Chair and a Laura H. Carnell Professor in the Department of Computer and Information Sciences at Temple University, Philadelphia, PA, USA. His current research interests include mobile computing and wireless networks, network trust and security, and routing protocols. He serves on several editorial boards, including *IEEE Transactions on Services Computing*, and *Journal of Parallel and Distributed Computing*. He is a CCF Distinguished Speaker. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.

Tian Wang (Member, IEEE) received his BSc and MSc degrees in Computer Science from the Central South University in 2004 and 2007, respectively. He received his Ph.D. degree in City University of Hong Kong in 2011. Currently, he is a professor at the Institute of Artificial Intelligence and Future Networks, Beijing Normal University & UIC, China. His research interests include internet of things and edge computing.

Guo Chen (Member, IEEE) received the Ph.D. degree from Tsinghua University in 2016. He was a Researcher with Microsoft Research Asia from 2016 to 2018. He is currently a Professor with Hunan University. His current research interests include networked systems and with a special focus on data center networking.

Tao Peng (Member, IEEE) received the B.Sc. in Computer Science from Xiangtan University, China, in 2004, the M.Sc. in Circuits and Systems from Hunan Normal University, China, in 2007, and the Ph.D. in Computer Science from Central South University, China, in 2017. Now, she is an Associate Professor of School of Computer Science and Cyber Engineering, Guangzhou University, China. Her research interests include network and information security issues.

Guojun Wang (Member, IEEE) received his Ph.D. degree in Computer Science, at Central South University, China in 2002. He is a Pearl River Scholarship Distinguished Professor of Higher Education in Guangdong Province, and a Doctoral Supervisor of School of Computer Science and Cyber Engineering, Guangzhou University, China. His research interests include artificial intelligence, big data, cloud computing, and blockchain. He is a Distinguished Member of CCF, a Member of IEEE, ACM and IEICE.