Constructive Noise Defeats Adversarial Noise: Adversarial Example Detection for Commercial DNN Services

Meng Shen*, Jiangyuan Bi*, Hao Yu†, Zhenming Bai*, Wei Wang‡, Liehuang Zhu*
*Beijing Institute of Technology, †National University of Defense Technology, ‡Xi'an Jiaotong University
*{shenmeng, jiangyuan, zmbai, liehuangz}@bit.edu.cn, †csyuhao@gmail.com, ‡wei.wang@xjtu.edu.cn

***Abstract*—Commercial DNN services have been developed in the form of machine learning as a service (MLaaS). To mitigate the potential threats of adversarial examples, various detection methods have been proposed. However, the existing methods usually require access to details or the training dataset of the target model, which is commonly unavailable in MLaaS scenarios. Their detection accuracy experiences a significant drop in a setting where neither the details nor the training dataset of the target model can be acquired.**

In this paper, we propose Falcon, an adversarial example detection method offered by a third party, which achieves accuracy and efficiency simultaneously. Based on the disparity in noise tolerance between clean and adversarial examples, we explore constructive noise that cannot affect the model's output labels when added to clean examples while causing noticeable changes in model outputs when added to adversarial examples. For each input, Falcon generates constructive noise with a specific distribution and intensity and achieves detection through differences in the output of the target model before and after adding constructive noise. Extensive experiments are conducted on 4 public datasets to evaluate the performance of Falcon in detecting 10 typical attacks. Falcon outperforms SOTA detection methods with the highest True Positive Rate (TPR) of adversarial examples and the lowest False Positive Rate (FPR) of clean examples. Furthermore, Falcon achieves a TPR of about 80% with an FPR of 5% on 6 well-known commercial DNN services, which outperforms the SOTA methods. Falcon can also maintain its accuracy although the adversary has complete knowledge of the detection details.

I. Introduction

Recent years have witnessed the prosperity of Deep Neural Networks (DNNs) applied in the domain of computer vision, such as face recognition [1], [2] and image classification [3]. Commercial DNN services, such as AWS [4] and Azure [5], have been developed in the form of machine learning as a service (MLaaS) to enable the widespread application of DNNs. However, these DNNs are susceptible to adversarial examples, which add imperceptible noise to an original image to mislead DNNs [6], [7], [8], [9]. Thus, adversarial example detection is crucial to assess whether an input image is adversarial before it is processed by commercial DNNs.

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.230250
www.ndss-symposium.org

Adversarial example detection, offered by a third party independent of the owner of commercial DNNs as illustrated in Fig. 1, provides a flexible and convenient solution. In this scenario, the detector relies only on predicted labels and confidence scores via APIs, without access to model details or the training dataset of commercial DNNs. Due to intellectual property concerns, commercial DNN services in MLaaS scenarios are typically models with access granted only through query APIs [10], [11]. Model owners probably do not have adequate resources or knowledge to detect adversarial examples themselves. Hence, they ought to ask a third party for detection services [12]. However, to safeguard the privacy and intellectual property of the commercial model and prevent model extraction attacks, the detector is commonly prohibited from utilizing details or the training dataset of the model [13]. Therefore, it is imperative to develop advanced detection services in the setting with limited access to the model and data [12], [14]. For practical deployment in commercial DNN services, detection should achieve high accuracy while maintaining cost efficiency with only predicted labels and confident scores served by APIs. It should identify the single submitted adversarial example without historical queries while avoiding mislabeling clean examples as adversarial [15]. Furthermore, the detector should limit the number of queries to commercial DNN services for less detection time overhead.

Although recent studies have proposed several solutions to adversarial example detection [16], [17], [15], [18], they rely on strong assumptions of model and data accessibility, which are impractical. Detections based on discrepancies in the target model's intermediate outputs require access to the target model [17], [19], [16], which are unavailable in MLaaS scenarios. Detections based on training auxiliary model or prediction inconsistency require access to the training dataset of the target model [20], [21], [15], [18]. They suffer a significant drop in detection accuracy when data is limited, as presented in Table I. In this paper, we focus on the setting where only query access to the model through APIs is available. Blacklight [22] and SD [23] are designed specifically for detecting adversarial examples through historical queries [23], [22], which cannot detect whether a single query is adversarial [8].

In this paper, we propose Falcon, an adversarial example detection method that achieves accuracy and cost efficiency simultaneously. Clean examples and adversarial examples exhibit different *tolerances* to noise: adversarial examples are more likely to be affected by additional noise. This motivates us to find a certain type of noise (referred to as *constructive*

TABLE I: A comparison of existing methods of adversarial example detections that can serve detection with only predicted labels and confident scores served by APIs. Query magnitude measures the number of queries on the target model.

Detections	Detection Performance		Detection Overhead	
	Detection Accuracy	Detection on Adaptive Attacks	Detection Time Overhead (s)	Query Magnitude
FS [21]	✗	✗	≈ 0.02	4
MagNet [20]	✗	✗	≈ 0.18	2
RS [24]	✗	✓	≈ 4.00	1.E+3
Falcon (ours)	✓	✓	≈ 0.20	2

noise), when added to an input image, it can lead to significant changes of the target model's output on adversarial examples while not affecting the output of clean examples. As a result, we can achieve an accurate detection by leveraging the differences induced by constructive noise.

We design three modules in Falcon to efficiently generate the distribution and intensity of constructive noise, which is then used for adversarial example detection. First, we design a noise distribution generator to obtain a targeted noise distribution that minimally impacts the regions critical for the correct classification of clean examples, while perturbing the distribution of adversarial noise. Second, we design a noise intensity generator based on a self-supervised learning network to obtain an appropriate noise intensity from the distribution determined in the first module. Intuitively, a higher intensity is prone to mislabeling clean examples (i.e., causing false positives), whereas a lower intensity may fail to induce obvious changes to adversarial examples (i.e., causing false negatives). We design two mutually constrained loss functions to balance the trade-offs. Finally, we design an adversarial example detector by measuring differences in the output of the target model after adding the constructive noise determined by the above two modules. Note that Falcon is designed and trained without prior knowledge of model details, output labels, or the adversary.

To comprehensively evaluate the performance of Falcon, we conduct extensive experiments on 4 public datasets under two scenarios (see more details in Sec. III-B), including CIFAR-10, ImageNet-10, ImageNet-1000 and CelebA. Following previous studies [15], [18], we select ResNet50 [3] as the target model and generate adversarial examples using 10 typical adversarial attacks (e.g., AutoAttack [25] and DSA [8]). We evaluate the performance of Falcon and 7 SOTA detection methods (e.g., CNet [15]) in detecting adversarial examples. The experimental results demonstrate that Falcon achieves a higher true positive rate (TPR) of adversarial examples while reducing the false positive rate (FPR) of clean examples. Falcon achieves its maximum TPR improvement of 21.42% on the ImageNet-1000 dataset when detecting SOTA attack DSA compared to other methods. Specifically, it achieves a TPR improvement of over 7% when detecting adversarial examples generated from ImageNet-1000.

In addition, we use Falcon to serve detection for 6 well-known commercial DNNs in the real world, including Baidu [26], Tencent [27], AWS [4], Azure [5], Google [28] and Alibaba [29]. We select four attacks where the adversary cannot get full knowledge of the target model to generate adversarial examples. The experimental results demonstrate that Falcon significantly outperforms compared detection methods, achieving an improvement in TPR of over 13.00%.

We further evaluate the performance of Falcon against adaptive attacks. We assume that the adversary has full knowledge of Falcon and can manipulate the generated noise to evade the detection by Falcon. We evaluate Falcon's TPR and FPR using three different adaptive attack strategies. The experimental results demonstrate that Falcon can achieve a TPR of over 50% while other methods almost fail completely.

We summarize the main contributions as follows:

- We demonstrate that clean and adversarial examples exhibit different tolerances to noise and further explore constructive noise that does not affect the model's output labels when added to clean examples while causing noticeable changes in the model's output when added to adversarial examples.
- We propose Falcon, an adversarial example detection that achieves accuracy and cost efficiency simultaneously. Falcon generates constructive noise with a certain distribution and intensity and detect by the differences in the output of the target model after adding constructive noise.
- We evaluate the performance of Falcon and 7 representative detection methods in detecting 10 typical adversarial attacks. Falcon achieves higher TPR and lower FPR under two scenarios based on the prior knowledge of the detector.
- We conduct real-world evaluations on detecting attacks on 6 well-known commercial DNN services. Falcon can achieve approximately 80% TPR and 5% FPR, which significantly outperforms other detection methods (e.g., MagNet [20]).
- We evaluate the performance of Falcon in detecting adaptive attacks, where the adversary has full knowledge of Falcon. The experimental results show that Falcon maintains a TPR of over 50% on four datasets.

II. Background and Related Work

A. Background

The adversarial example is generated by adding well-designed noise, which is imperceptible to human eyes, to the input to mislead the target DNN model [6]. The target model can be defined as $f : \mathbb{R}^d \rightarrow \{1, \cdots, K\}$, where d is the dimension of the input and K is the number of classes. Given a clean example X, the targeted and untargeted attacks can be formulated as an optimization problem in Eq. (1),

$$\begin{cases} \min \|\delta\|_p, & \text{s.t. } f(X+\delta) \neq f(X) \quad (untargeted) \\ \min \|\delta\|_p, & \text{s.t. } f(X+\delta) = f(X_t) \quad (targeted) \end{cases} \tag{1}$$

where δ is the adversarial noise, X_t is the image of targeted label and $\|\cdot\|_p$ represents the l_p norm (e.g., l_2, l_∞) of δ.

In this paper, we focus on adversarial attacks which aim to minimize the generated perturbation to mislead the target model (see Eq. (1)). Here, we introduce 10 attacks that are typical or SOTA, including PGD, CW, AutoAttack, VNIFGSM, HJSA, HybridAttack, DSA, SSAE, Kenneth and AT-UAP.

Adversarial examples can be broadly categorized into per-instance and universal attacks, depending on whether the generated perturbation is specific to each input or shared across inputs. Per-instance attacks generate perturbations tailored to

individual input images. This category encompasses a wide range of generation methods, including gradient-based (e.g., PGD [7], DSA [8], AutoAttack [25]), optimization-based (e.g., CW [30]), and GAN-based (e.g., SSAE [31]) approaches. While gradient and optimization-based methods rely on directly manipulating the input to maximize a predefined loss, GAN-based attacks use generative models to learn a distribution of adversarial perturbations that achieve high transferability and low perceptual distortion. On the other hand, universal attacks (e.g., Kenneth [32], AT-UAP [33]) aim to generate a single, input-agnostic perturbation that can be applied across many different original images to induce misclassification. Despite differences in generation mechanisms and perturbation characteristics, these attacks follow the objective in Eq. (1).

In addition to the attacks discussed above, there also exist adversarial attacks in the physical domain [34], [35], [36]. These attacks typically violate the bounded perturbation constraint defined in Eq. (1) and adopt the Expectation over Transformation (EoT) strategy to remain effective under real-world variations such as lighting, viewpoint, and occlusion. Thus, we do not take them into consideration.

B. Related Work

Existing detection methods fall into two categories based on the knowledge accessible to the detector.

Detection methods in the first category require varying degrees of access to the target model's knowledge. When full knowledge of the target model (e.g., its training dataset, architecture and parameters) is accessible, certain detection methods exploit discrepancies in intermediate-layer output or feature attributions for detection [17], [37], [19], [16]. A^2D [38] achieves detection by measuring the difficulty (e.g., required perturbation magnitude) of conducting additional adversarial attacks with full access to the target model. When only prior knowledge of training dataset or all output labels is available, some detection methods train an auxiliary model for detection [15], [18], [39]. CNet [15] detects adversarial examples by training a conditional generative network to reconstruct an image from the target model's predicted label and measuring the differences between the input and its reconstruction. MI-AED [18] detects adversarial examples by training an auxiliary MLP and identifying an input as adversarial when the label predicted by the MLP differs from that of the target model. BEYOND [39] utilizes the robust representation capacity of extra Self-Supervised Learning (SSL) model to detect adversarial examples by examining their proximity to neighbor examples generated by augmentations (e.g., Gaussian noise, crop).

Detection methods in the second category operate under the weakest knowledge assumptions, where only limited output information (i.e., predicted label and confidence score) is accessible via API queries. These methods generate transformed versions of the input and perform detection by comparing the predicted information between the original and transformed inputs [21], [24], [20]. FS [21] uses digital image processing methods such as JPEG compression to generate transformations and calculates the distance between confidence scores of the input and its transformed version. RS [24] applies Gaussian noise to generate transformations and compares the labels of the input with its transformed version. MagNet [20] utilizes deep learning techniques to generate transformations and detect adversarial examples based on the difference (i.e., MSE and KL divergence) between the input and its transformed version. Some methods exploit the differences in historical query sequences caused by the adversary for detection (e.g., SD [23] and Blacklight [22]). However, they fail completely when facing adversarial examples generated without querying the target model, such as VNIFGSM [40].

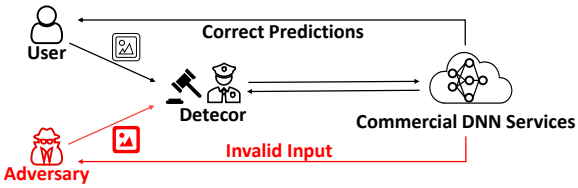


Fig. 1: The illustration of detection services for commercial DNNs. Given an input image submitted by a user, the detector first queries the target model through its API and determines whether the input is adversarial, and then the target model can reply with the correct prediction for clean examples or invalid input for adversarial examples, respectively.

Falcon follows in the same line of research as RS [24] and MagNet [20]. However, Falcon distinguishes itself by modeling the distribution of adversarial noise and generating input-specific constructive perturbations to create a more effective transformation. This results in improved detection accuracy with a lower false positive rate.

In addition to detection-based methods, other SOTA approaches have been developed under different threat models [41], [42], [43]. For instance, Patchcure [43] is dedicated to constructing models robust against adversarial patch attacks in the physical domain. We focus on methods that detect attacks following Eq. (1) in this paper.

III. System Model and Threat Model

A. System Model

In this paper, we focus on providing adversarial example detection for commercial DNN services, as shown in Fig. 1. In this scenario, neither the details nor the training dataset of commercial DNN services can be acquired, while only query access to the model through APIs is available.

Although much less effort has been devoted to this setting, this setting is more realistic in commercial transactions of machine learning services (e.g., AWS [4] and Azure [5]). For example, a lot of organizations (e.g., hospitals and banks) purchase machine learning services that are applied to some safety-critical applications (e.g., face recognition) from the owner of the model [1], [2]. However, these systems are proven to be vulnerable to adversarial examples [8]. The adversary can easily mislead these systems through adding adversarial noise on the input image. Due to the intellectual property, these systems are usually with only query access through APIs [26], [27], based on the typical MLaaS scenario. Access to the details and training dataset of the target model is restricted. Such a setting hinders the organization from detecting whether an input image is adversarial accurately with the existing detection methods [17], [19], [16]. Even if the details of these

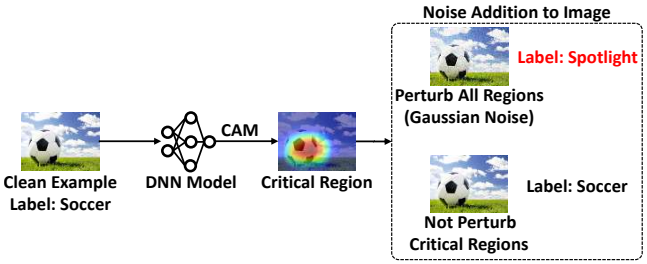


Fig. 2: The effect of noise distribution on the labels produced by the DNN model. We first locate critical regions through CAM. The label remains unchanged when the critical regions are not perturbed while adding Gaussian noise alters the original label of the image.

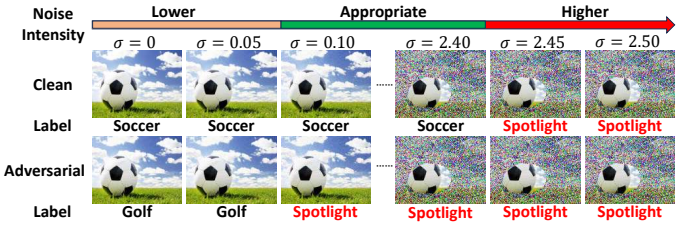


Fig. 3: The classification results of clean example (1st row) and adversarial example (2nd row) when added a varying intensity of noise (σ). When $\sigma \in [0.1, 2.4]$, the label of adversarial example changes, while that of clean example remains unaffected.

systems are available, the organizations probably do not have adequate resources or knowledge to detect potential adversarial examples [12]. Hence, they ought to ask a third party to perform adversarial example detection objectively, which still needs to be conducted in a manner with only information served by APIs due to privacy considerations [44]. Thus, we focus on accurate adversarial example detection with limited details and the training dataset of the target model. It isolates the detection process from the operation of commercial DNNs and makes it easier to update or replace detection methods without affecting the system functionality.

B. Threat Model

In this paper, there are three primary parties: the victim, the detector, and the adversary, as shown in Fig 1.

The *victim* refers to an organization that purchases commercial DNN services. The victim deploys these systems and provides labels and confidence scores for inputs through an API, consistent with the workflow of existing popular commercial DNN services (e.g., Baidu [26] and AWS [4]).

The *adversary* aims to add imperceptible noise to the input to generate an adversarial example that misleads the target model. In real-world scenarios, the adversary misleading commercial DNNs usually has no access to the target model's details [8]. There, we further discuss two types of attacks based on access to the detection strategy.

- *Static attack:* The adversary cannot know the existence of the detection strategy [15].
- *Adaptive attack:* The adversary has full knowledge of the detection strategy [45], [46]. Thus, he can perform an adaptive attack attempting to evade detection.

The *detector* judges whether an input submitted to the victim is adversarial or not in advance. The detector only uses labels and confidence scores served by the victim and cannot gain any prior knowledge of the target model and the adversary. The detector is assumed to construct a shadow dataset and use it to train detection framework. We consider two scenarios based on the constructed shadow dataset:

Scenario#1. Small commercial DNN services are often trained by some public datasets. The detector can access the full set of output labels from these public datasets. This allows the detector to construct a shadow dataset that is approximately identically distributed with the target model's training data. Note that the shadow dataset cannot overlap with the training dataset of the target model.

Scenario#2. Existing large commercial DNN services (e.g., Tencent [27] and Azure [5]) typically train their models on proprietary datasets they construct themselves. The detector cannot access the full set of output labels and can only get a subset of output labels through some queries. Due to the discrepancy in labels, the shadow dataset is not identically distributed with the target model's training data.

C. Design Goals

Based on the knowledge of the detector defined above, it should achieve the following goals.

Accuracy. It should accurately identify individual submitted adversarial example while mislabeling the clean example as adversarial less in a setting where only predicted labels and confidence scores by APIs can be obtained [17], [15], [18].

Robustness. It should not rely on prior knowledge of the adversary and can maintain its accuracy on a wide range of adversarial attacks mentioned in Sec. II-A or adaptive attacks.

Efficiency. It should minimize the number of queries on the target model and achieve detection with a low time overhead.

IV. The Proposed Falcon

A. Existence of Constructive Noise

Adversarial examples generated under the objective in Eq. (1) can mislead the target model with minimal perturbation magnitude and are often located close to the decision boundary. This phenomenon has been consistently demonstrated in prior works [47], [48], [49]. Thus, adversarial examples are more likely to be affected by additional noise. We demonstrate that there exists a certain type of noise in both a clean example and its corresponding adversarial example, the output for the adversarial example might have a significant change (e.g. a sharp drop of confidence score, or even the change of predicted label), whereas that of the clean example might have only a slight change. Now, we investigate where the noise should be added in an image (i.e., noise distribution) and to what extent it should be added (i.e., noise intensity).

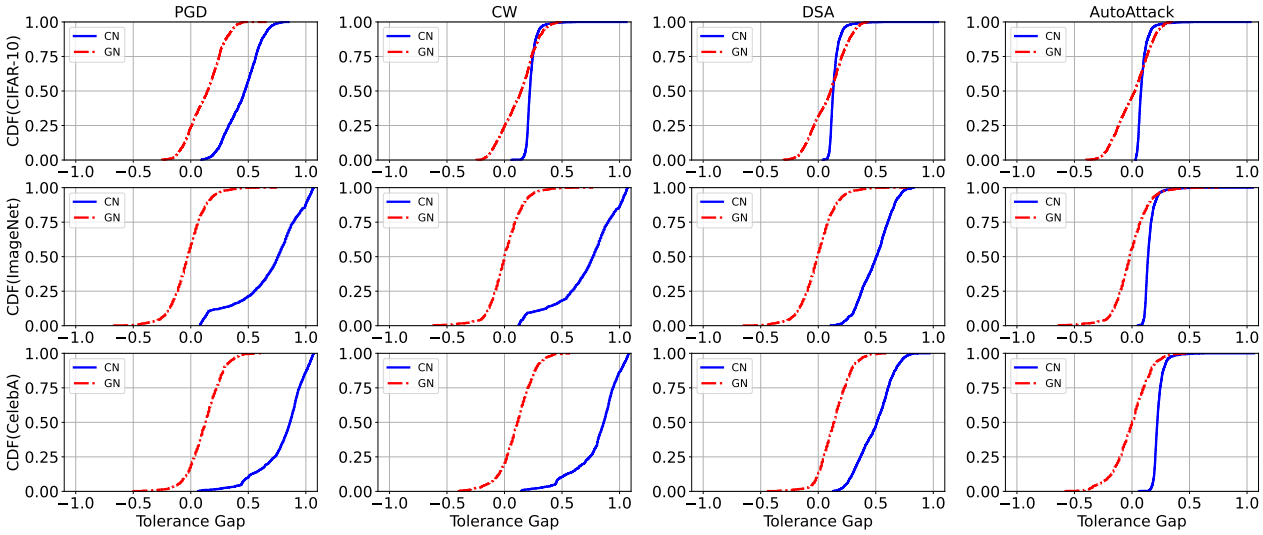


Fig. 4: The CDF of Tolerance Gap (TG) for Constructive Noise (CN) and Gaussian noise (GN) under four adversarial attacks across three datasets. If TG is less than 0, the noise cannot distinguish adversarial examples from clean examples.

Noise Distribution. Gaussian noise is a commonly used random noise that uniformly perturbs all regions of an image. However, Gaussian noise is not suitable for distinguishing between clean and adversarial examples because it is likely to disturb the crucial regions for the classification of clean examples, thus causing a significant change in the label of clean examples. We illustrate this phenomenon in Fig. 2, where the crucial region for the classification of a clean example is visualized using the CAM method [50]. The Gaussian noise perturbs all regions of the clean example and thus leads to the change of its label. However, if we add the same magnitude of noise, avoiding perturbing the critical regions, the resulting label remains the same. Thus, a noise distribution should not perturb critical regions. We will justify this later (see Fig. 4).

Noise Intensity. After determining the noise distribution, we explore the magnitude of the noise added to the image. For ease of illustration, given an image, its *tolerance* to noise is regarded as the maximum magnitude of noise added that keeps its label unchanged. We define the tolerance of clean and adversarial example as σ_{clean} and σ_{adv}. Intuitively, a clean example and its corresponding adversarial example will have different tolerances to additional noise, which is revealed by their different distances from the decision boundary. To have a clear understanding, we randomly select a pair of clean and adversarial examples and vary the intensity of noise denoted by σ, as shown in Fig. 3. We observe that $\sigma = 0.1$ changes the label of the adversarial example (i.e., $\sigma_{adv} = 0.1$) whereas $\sigma = 2.4$ changes the label of the clean example (i.e., $\sigma_{clean} = 2.4$), indicating that the clean example can tolerate larger noise than its adversarial example.

Constructive Noise. The above analysis shows that there exists a range of noise intensities capable of altering the label of adversarial examples without affecting the label of clean examples (i.e., $\sigma \in (\sigma_{adv}, \sigma_{clean})$). We refer to the noise in this range as *constructive noise*, as it can be used to distinguish between clean and adversarial examples. We also refer to the length of this range as the Tolerance Gap (TG).

To justify the effectiveness of constructive noise, we use ResNet50 as the target model, randomly select 3,000 images from CIFAR-10, ImageNet-1000 and CelebA as clean examples, and generate the corresponding adversarial example for each clean example using the PGD [7], CW [30], DSA [8] and AutoAttack [25] attacks, respectively. We plot the CDF of the tolerance gap for both constructive noise and random noise, as shown in Fig. 4. We can find that using constructive noise, the value of TG is always larger than 0, indicating the existence of constructive noise to differentiate clean examples from adversarial examples. In contrast, when using Gaussian noise, the value of TG is not always larger than 0, indicating the failure to distinguish clean and adversarial examples.

To establish a theoretical foundation for TG, we present its formal definition and prove that the value of TG is always larger than zero under certain constraints in Appendix A. Motivated by these constraints, we generate constructive noise by modeling the distribution of adversarial noise while affecting classification accuracy of clean examples less.

Note that we focus on adversarial attacks following the optimization objective in Eq. (1). The effectiveness of constructive noise does not always hold for adversarial examples in the physical domain (e.g., AdvPatch [36]). A more comprehensive discussion will be presented in Appendix D.

B. Overview of Falcon

Based on the disparity in noise tolerance in Sec. IV-A, we propose Falcon, an adversarial example detection that achieves accuracy, robustness and efficiency. Falcon designs a specific noise distribution and intensity for each input, aiming to generate constructive noise that defeats adversarial noise without judging clean examples as adversarial.

Optimizing both the noise distribution and intensity simultaneously is challenging due to the high-dimensional search space, which increases the computational resources required and complicates the optimization task. To address this, we

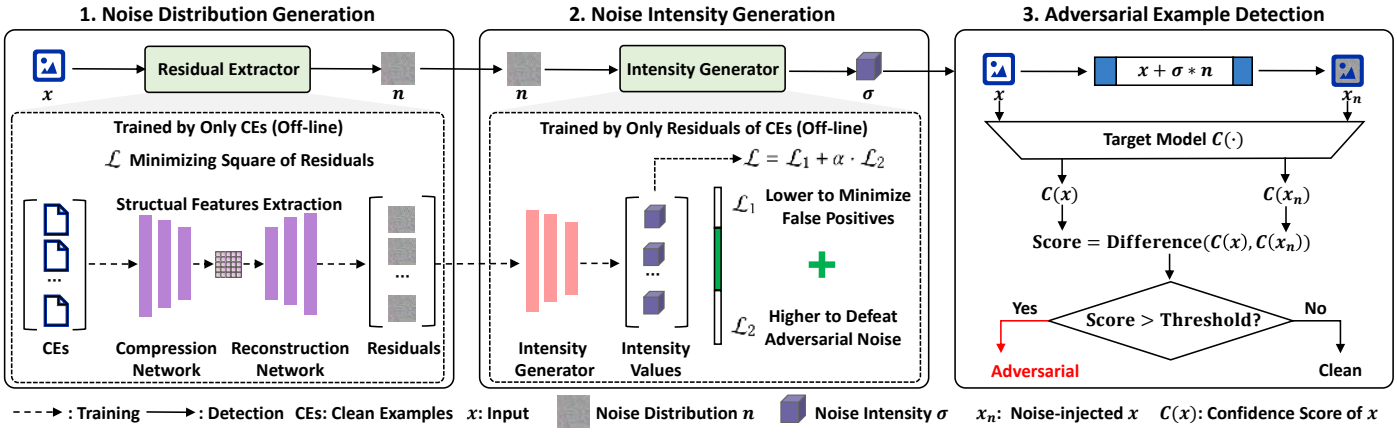


Fig. 5: The overview of Falcon. Only clean examples are used in the training stage of Falcon. The Threshold in the Adversarial Example Detection module is also determined before deployment based on clean examples.

first fix one parameter and then optimize the other, simplifying the optimization process. In Falcon, we prioritize determining the noise distribution before selecting the noise intensity. This strategy is driven by two key considerations. First, the noise distribution shares the same dimensionality as the image, resulting in a large search space. Fixing the noise intensity first and then optimizing the noise distribution would require a large computational cost. Second, the appropriate intensity is influenced by the specific distribution used. The overview of Falcon is shown in Fig. 5, which consists of three components.

Noise Distribution Generation. The Noise Distribution Generation module proposes a residual extractor to create a targeted noise distribution that minimally affects regions critical for correct classification while defeating adversarial noise. Specifically, we design a compression network to map the input into a low-dimensional space and a reconstruction network to reconstruct the image from the low-dimensional representation, extracting key features and filtering out some unnecessary noise. We choose the residual between input and its reconstruction as the noise distribution. The extracted residuals for clean examples are distributed in regions less critical to correct classification. For adversarial examples, they highlight the distribution of adversarial noise.

Noise Intensity Generation. Once the noise distribution is determined, the Noise Intensity Generation module designs a generator based on a self-supervised learning network to generate an appropriate noise intensity from the noise distribution. Appropriate intensities lie within a range: lower intensity fails to counter adversarial noise while higher intensity risks false alarms for clean examples. Our generator optimizes two loss functions: one to enhance noise intensity to defeat adversarial noise, and the other to preserve the classification accuracy of clean examples, thereby facilitating generated intensity to fall within the appropriate range.

Adversarial Example Detection. The Adversarial Example Detection module determines if the input is adversarial by measuring the differences in the target model's outputs after adding the noise combined from the generated distribution and intensity. These differences are assessed by comparing label changes and the similarity of confidence scores.

V. Design Details of Falcon

A. Noise Distribution Generation

The Noise Distribution Generation module is designed to create a targeted noise distribution with two key objectives. First, it aims to minimize perturbations in regions critical for correct classification to maintain the accuracy of clean examples. Second, it tries to target the distribution of adversarial noise, perturbing them to induce significant changes in the outputs of adversarial examples.

The selection of noise distribution typically adopts Gaussian noise, which is one of the most commonly used forms of random noise [24], [51]. However, Gaussian noise uniformly perturbs all regions of an image, potentially disrupting areas critical for correct classification. Thus, directly applying Gaussian noise reduces the classification accuracy of clean examples. Designing an appropriate noise distribution involves two key challenges: 1) The critical regions for correct classification vary across different images, so it is necessary to generate a tailored rather than fixed noise distribution for each image. 2) The generated noise distribution should effectively perturb the distribution of adversarial noise.

To address these challenges, we propose a residual extractor to produce a targeted noise distribution. First, we design a compression network that maps the image into a low-dimensional space. Second, a reconstruction network is used to reconstruct the image from the low-dimensional representation extracted by the compression network. The compress network learns to capture the unique structural features of each image by mapping it into a low-dimensional space, effectively filtering out some unnecessary noise [52], [53]. The reconstruction network then uses this compressed representation to reconstruct the image, ensuring that more critical information for classification is preserved. The residual between the reconstructed image and the input image serves as the final noise distribution.

Our mechanism is designed to retain the key structural features of any image while compressing unnecessary noise, ensuring that each image has a tailored residual. For clean examples, using the residual as the noise distribution does not

disrupt regions critical for correct classification. For adversarial examples, adversarial noise is essentially a type of noise that will be partially filtered out by the compression network. Thus, using the residual as the noise distribution perturbs the areas where adversarial noise is concentrated.

The compression and reconstruction networks in Falcon are trained solely on clean examples. Since the training process does not rely on knowledge of the target model, the generation of the noise distribution remains independent of the target model. Additionally, the noise distribution generation does not depend on the prior knowledge of the adversary because adversarial examples are not involved in the training process.

Due to the proven effectiveness of the Convolutional Neural Network (CNN) in extracting key patterns from images [54], we adopt CNN as the architecture for both the compression and reconstruction networks. To enhance performance, we use a deeper network architecture compared to previous works [51], [55]. The compress network employs 3×3 convolutional layers, with downsampling performed via a stride of 2. This design choice avoids pooling, which could harm image restoration tasks [56]. In the reconstruction network, upsampling is applied, and the layer structure is symmetric to the compression network. Shortcut connections between corresponding layers of the compression and reconstruction network help maintain efficient information flow [3]. To improve stability and convergence, each convolutional or deconvolutional layer is followed by a Batch Normalization layer and a ReLU activation function. These design choices aim to achieve the goal of accurately compressing and reconstructing images while filtering out noise.

We only use clean examples to train both compression and reconstruction networks. The training loss is measured as the Mean Squared Error (MSE) between the input and reconstructed images. Finally, we back-propagate the loss and update all trainable parameters with the Adam optimizer, which has been widely adopted in prior studies [57], [53].

We adopt the residual-based mechanism to generate the noise distribution, rather than directly generating the distribution. Using a generative network to directly generate a noise distribution has several limitations. The noise distribution shares the same dimensionality as the image, which results in an expansive search space for potential solutions. This vast search space complicates the optimization process, as the generative network could produce a wide range of possible noise distributions, making it difficult to control and refine the solution. Furthermore, the compression and reconstruction networks generate a noise distribution through a smoothing operation, which preserves essential structures while identifying and locating unnecessary details. Unnecessary noise is typically concentrated in regions with rich edges and textures, which correspond to high-frequency information [58].

The compression network can partially filter out adversarial noise but cannot eliminate it entirely. Thus, the reconstructed image derived from an adversarial example may still be adversarial. Moreover, for a clean example, the reconstructed image may lose critical information for classification, resulting in a wrong predicted label. As a result, the reconstructed image cannot be used directly to replace the original input.

B. Noise Intensity Generation

After determining the noise distribution, we further investigate the appropriate magnitude at which the noise distribution should be applied to the input (i.e., noise intensity). The Noise Intensity Generation module is designed to generate an adaptive intensity tailored to the noise distribution.

Based on the observation of noise intensity under a given noise distribution in Sec. IV-A, we can find that the appropriate intensity can take multiple values within an appropriate range. The upper bound of this range is the noise intensity that causes label changes in clean examples, while the lower bound is the intensity that alters labels of adversarial examples. Due to the varying positions of each example in the decision space of the target model, their corresponding upper bounds are different. The lower bounds of adversarial examples vary depending on the attack strategy used. Setting a fixed noise intensity would cause two key issues. For some clean examples, the noise intensity might be higher, leading to misclassification. For certain adversarial examples, the intensity could be lower to defeat adversarial noise.

Generating an appropriate noise intensity involves three key challenges: 1) The generated intensity should be below the upper bound for clean examples and above the lower bound for adversarial examples. 2) The appropriate range of noise intensity is influenced by the target model, but detailed information about it is unavailable. 3) The attack strategy affects the suitable range of noise intensity, but prior knowledge of the adversary is inaccessible.

To address these challenges, we propose an intensity generator based on self-supervised learning, where the network processes the noise distribution to determine an appropriate intensity. The noise distribution shares the same high dimensionality as the image while the noise intensity is a single scalar value. To bridge this dimensional gap and establish a relationship between them, we utilize CNN which is effective in extracting key patterns from images [54] as the architecture. Specifically, we employ two convolutional layers with kernel size 3×3, two max-pooling layers with kernel size 2×2, and fully connected layers in the regression network.

This network optimizes two loss functions: one to enhance noise intensity to defeat adversarial noise, and the other to preserve the classification accuracy of clean examples. First, the generated noise intensity should be high enough to cause obvious changes in the outputs of adversarial examples after adding noise. Thus, we design the first loss function as

$$\mathcal{L}_1 = \frac{1}{\sigma}, \tag{2}$$

where σ is the generated noise intensity. Since the noise intensity is positive, we take its reciprocal as part of the loss function to encourage the generation of the largest possible intensity to defeat adversarial noise.

However, adding noise with a higher intensity will alter the predicted labels of clean examples, causing false alarms. To not rely on the target model, we constrain the noise intensity using distance. Thus, we define the second loss function as

$$\mathcal{L}_2 = ||x_n - x||_2, \tag{3}$$

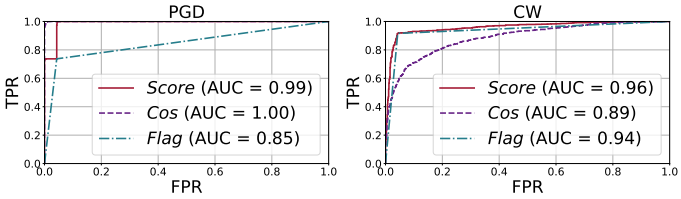


Fig. 6: Effectiveness of Cos, $Flag$, and $Score$ in distinguishing clean examples from adversarial examples. While Cos performs well for PGD adversarial examples, it shows weaker performance for CW adversarial examples.

where x is input and x_n is the image after adding noise with generated distribution and intensity on x. To ensure the generated noise intensity approaches the upper bound without exceeding it, we design a unified loss function as

$$\mathcal{L} = \mathcal{L}_1 + \alpha \cdot \mathcal{L}_2 \tag{4}$$

where α is a hyper-parameter. There we set α as 0.1 because $\mathcal{L}_2$ is approximately an order of magnitude larger than the $\mathcal{L}_1$. Note that, only clean examples are used in the training process, which ensures the generation of noise intensity without dependency on the prior knowledge of the adversary. We only use clean examples to train the regression network. The training loss is $\mathcal{L}$ in Eq. (4). Finally, we back-propagate the loss and leverage the Adam optimizer to update all trainable parameters.

C. Adversarial Example Detection

The Adversarial Example Detection module distinguishes adversarial examples by capturing changes that happened in the model's outputs after adding generated noise combined with the distribution and intensity in the above modules.

Adversarial example detection based on adding generated noise involves two challenges: 1) Only the predicted label and its corresponding confidence score served through API can be obtained due to the limited capability of the target model. 2) The prior knowledge of the adversary cannot be obtained in advance to design a detection mechanism.

To address these challenges, we directly observe the changes in prediction labels and their corresponding confidence scores to examine the differences between clean and adversarial examples. Specifically, we define the output confidence score in the target model of an image as y_1, and the output confidence score of the image with added noise as y_2.

First, we define the first metric $Flag(\cdot)$ to measure whether the output label changes after adding noise as

$$Flag\,(y_1, y_2) = \begin{cases} 0, \text{if } \arg\max(y_1) = \arg\max(y_2) \\ 1, \text{otherwise} \end{cases} \tag{5}$$

where $\arg\max(\cdot)$ computes final predicted label based on the confidence score. If the predicted label changes after adding noise, $Flag(\cdot)$ is set to 1. In contrast, the flag is set to 0.

Second, we define $Cos(\cdot)$ to measure changes in confidence score

$$Cos\,(y_1, y_2) = \frac{y_1 \cdot y_2}{\|y_1\|\|y_2\|} \tag{6}$$

where $\|\cdot\|$ denotes the magnitude (or norm) of a vector.

For adversarial examples, the output predicted label is more likely to change after adding noise, causing the value of $Flag$ to approach 1. Moreover, the changes in their confidence scores after adding noise are more obvious, making Cos closer to 0 and $1 - Cos$ approach 1. Therefore, we define $score$ as

$$score = Flag + (1 - Cos) \tag{7}$$

$score$ further combine $Flag$ and $1 - Cos$ to distinguish adversarial examples from clean examples. We can observe that both $Flag$ and $1 - Cos$ have limitations in distinguishing between attacks of different types (e.g., PGD and CW). However, combining them improves performance, as shown in Fig. 6.

Adding noise to adversarial examples is more likely to induce obvious changes in their outputs in the target model, causing the $score$ of adversarial examples higher than the $score$ of clean examples. To detect adversarial examples without relying on the prior knowledge of the adversary, we set a threshold using only the clean examples. If the $score$ is below this threshold, the input is considered a clean example. If it exceeds the threshold, the input is judged as adversarial. The predicted label and corresponding confidence score are typically secure after adding noise. Thus, the $score$ of clean examples is located around 0. We select a constant value of 0.5 as a threshold guided by the gap in the middle range of scores between 0 and 1.

TABLE II: Summary of Datasets and Target Models.

Datasets	Training Images	Testing Images	Classification Accuracy (%)	
			ResNet50	DenseNet169
CIFAR-10	5,0000	1,0000	94.75	94.00
ImageNet-10	5,0000	1,0000	98.75	98.44
ImageNet-1000	20,0000	2,0000	78.80	76.00
CelebA	4,0000	1,0000	95.19	93.47

VI. Performance Evaluation

In this section, we comprehensively evaluate the performance of Falcon in various experimental settings. Particularly, we would like to study the following research questions:

- Can Falcon accurately identify an individual submitted adversarial example while mislabeling the clean example as adversarial less in the setting where only predicted labels and confidence scores can be obtained?
- Can Falcon maintain its accuracy against various attacks, especially when the adversary has full knowledge of Falcon and conducts adaptive attacks?
- How sensitive is Falcon to changes in data distribution?
- What is the impact of the design components and their hyperparameters on Falcon?

A. Experiment Settings

Datasets. We conduct the experiments using CIFAR-10 [60], ImageNet-10 [15], ImageNet-1000 [61] and CelebA [62] datasets, which have been widely used in previous studies [20], [18]. The CIFAR-10 dataset consists of 10 classes, with each class containing 60,000 images with a size of 32×32×3. The ImageNet-1000 dataset comprises over 14 million images

TABLE III: TPR (%) and FPR (%) for detecting untargeted static attacks (Target Model: ResNet50). **Bolded values** indicate the highest TPR or the lowest FPR.

Datasets	Detections	VNI[40]		HJSA[47]		Hybrid[59]		DSA[8]		PGD[7]		CW[30]		AA[25]		SSAE[31]		Kenneth[32]		AT-UAP[33]	
		TPR↑	FPR↓	TPR↑	FPR↓	TPR↑	FPR↓	TPR↑	FPR↓	TPR↑	FPR↓	TPR↑	FPR↓	TPR↑	FPR↓	TPR↑	FPR↓	TPR↑	FPR↓	TPR↑	FPR↓
CIFAR-10	BEYOND [39]	85.16	5.00	89.92	5.00	85.05	5.00	83.71	5.00	89.46	5.00	88.72	5.00	74.38	5.00	88.68	5.00	84.65	5.00	83.94	5.00
	A^2D [38]	72.96	5.00	73.88	5.00	72.37	5.00	70.08	5.00	78.49	5.00	73.06	5.00	69.38	5.00	74.13	5.00	72.85	5.00	69.53	5.00
	CNet [15]	77.12	5.00	85.36	5.00	74.35	5.00	71.43	5.00	90.27	5.00	88.82	5.00	72.20	5.00	88.57	5.00	72.41	5.00	71.06	5.00
	MIAED [18]	73.48	7.80	74.22	7.80	71.55	7.80	72.38	7.80	81.02	7.80	81.54	7.80	69.18	7.80	81.30	7.80	80.62	7.80	71.46	7.80
	FS [21]	10.35	5.00	16.28	5.00	12.56	5.00	16.24	5.00	46.03	5.00	5.55	5.00	4.73	5.00	12.78	5.00	18.63	5.00	7.92	5.00
	MagNet [20]	57.32	5.00	64.89	5.00	55.10	5.00	1.52	5.00	67.96	5.00	83.84	5.00	42.88	5.00	76.85	5.00	72.60	5.00	52.85	5.00
	RS [24]	77.25	12.24	80.10	12.24	75.68	12.24	76.13	12.24	78.11	12.24	81.87	12.24	67.30	12.24	78.26	12.24	76.85	12.24	67.22	12.24
	Falcon	**90.50**	**5.00**	**94.00**	**5.00**	**91.20**	**5.00**	**88.15**	**5.00**	**92.85**	**5.00**	**91.17**	**5.00**	**85.28**	**5.00**	**91.86**	**5.00**	**90.74**	**5.00**	**90.02**	**5.00**
ImageNet-10	BEYOND [39]	90.37	5.00	91.18	5.00	90.94	5.00	88.35	5.00	90.26	5.00	87.69	5.00	85.08	5.00	91.41	5.00	89.71	5.00	88.25	5.00
	A^2D [38]	72.68	5.00	73.37	5.00	71.96	5.00	69.82	5.00	75.69	5.00	74.66	5.00	67.42	5.00	75.86	5.00	73.09	5.00	70.59	5.00
	CNet [15]	75.09	5.00	78.28	5.00	71.28	5.00	68.87	5.00	94.20	5.00	96.20	5.00	79.86	5.00	95.80	5.00	88.64	5.00	69.58	5.00
	MIAED [18]	72.49	10.24	72.86	10.24	71.06	10.24	69.84	10.14	80.56	10.24	80.78	10.14	70.22	10.14	80.26	10.14	72.86	10.14	71.84	10.14
	FS [21]	61.48	5.00	63.22	5.00	59.68	5.00	23.96	5.00	79.20	5.00	85.90	5.00	57.26	5.00	80.75	5.00	62.49	5.00	61.87	5.00
	MagNet [20]	58.33	5.00	63.12	5.00	56.30	5.00	32.67	5.00	82.10	5.00	71.20	5.00	66.52	5.00	73.82	5.00	65.98	5.00	54.37	5.00
	RS [24]	73.99	9.72	78.76	9.72	72.64	9.72	73.28	9.72	74.82	9.72	77.23	9.72	70.88	9.72	75.80	9.72	76.92	9.72	72.85	9.72
	Falcon	**97.80**	**5.00**	**98.80**	**5.00**	**97.80**	**5.00**	**94.10**	**5.00**	**99.02**	**5.00**	**99.10**	**5.00**	**94.80**	**5.00**	**98.72**	**5.00**	**98.10**	**5.00**	**96.65**	**5.00**
ImageNet-1000	BEYOND [39]	76.15	5.00	74.38	5.00	73.39	5.00	67.58	5.00	81.08	5.00	79.06	5.00	68.75	5.00	80.48	5.00	78.64	5.00	73.69	5.00
	A^2D [38]	70.02	5.00	69.75	5.00	68.36	5.00	67.02	5.00	71.25	5.00	71.86	5.00	66.94	5.00	71.09	5.00	59.87	5.00	66.30	5.00
	CNet [15]	72.42	5.00	73.40	5.00	70.66	5.00	66.72	5.00	84.10	5.00	86.40	5.00	69.55	5.00	85.33	5.00	73.98	5.00	70.24	5.00
	MIAED [18]	69.88	10.56	70.54	10.56	68.12	10.56	68.14	10.56	72.76	10.56	73.12	10.56	66.45	10.56	72.82	10.56	72.96	10.56	71.37	10.56
	FS [21]	56.39	5.00	58.66	5.00	56.10	5.00	17.45	5.00	59.80	5.00	65.40	5.00	55.48	5.00	64.86	5.00	52.46	5.00	49.78	5.00
	MagNet [20]	54.28	5.00	56.33	5.00	54.11	5.00	30.28	5.00	66.20	5.00	58.60	5.00	57.29	5.00	57.62	5.00	56.87	5.00	53.12	5.00
	RS [24]	57.16	10.02	64.20	10.02	56.22	10.02	56.96	10.02	58.16	10.02	62.58	10.02	52.16	10.02	59.87	10.02	62.90	10.02	56.81	10.02
	Falcon	**90.88**	**5.00**	**92.68**	**5.00**	**92.35**	**5.00**	**90.02**	**5.00**	**93.10**	**5.00**	**93.98**	**5.00**	**93.82**	**5.00**	**93.17**	**5.00**	**90.05**	**5.00**	**89.40**	**5.00**
CelebA	BEYOND [39]	70.45	5.00	69.85	5.00	68.50	5.00	64.75	5.00	71.70	5.00	72.10	5.00	66.45	5.00	71.48	5.00	72.50	5.00	63.90	5.00
	A^2D [38]	64.95	5.00	64.50	5.00	63.00	5.00	66.10	5.00	66.25	5.00	62.00	5.00	68.00	5.00	64.29	5.00	62.00	5.00	60.05	5.00
	CNet [15]	68.85	5.00	69.00	5.00	66.78	5.00	66.25	5.00	86.65	5.00	87.65	5.00	75.00	5.00	86.82	5.00	68.48	5.00	64.10	5.00
	MIAED [18]	64.40	10.45	66.50	10.45	63.20	10.45	64.25	10.45	78.40	10.45	78.85	10.45	70.00	10.45	77.25	10.45	66.00	10.45	64.20	10.45
	FS [21]	41.75	5.00	42.50	5.00	40.50	5.00	39.90	5.00	52.25	5.00	47.20	5.00	45.50	5.00	48.91	5.00	42.06	5.00	39.74	5.00
	MagNet [20]	56.25	5.00	58.50	5.00	54.00	5.00	49.95	5.00	67.45	5.00	68.90	5.00	58.80	5.00	66.34	5.00	55.05	5.00	49.58	5.00
	RS [24]	68.00	24.00	70.00	24.00	66.00	24.00	67.00	24.00	65.00	24.00	70.00	24.00	60.00	24.00	66.00	24.00	69.00	24.00	62.00	24.00
	Falcon	**89.50**	**5.00**	**89.80**	**5.00**	**91.00**	**5.00**	**90.00**	**5.00**	**92.50**	**5.00**	**98.00**	**5.00**	**91.00**	**5.00**	**93.00**	**5.00**	**90.80**	**5.00**	**88.75**	**5.00**

and has 1,000 classes of images which are re-scaled to the size of 256×256×3. Following the settings in previous studies [15], [18], we utilize a subset consisting of 10 random classes, which includes 5,000 training images and 1,000 testing images for each class from ImageNet-1000 as ImageNet-10. For the CelebA dataset, we select a subset including 100 different identification randomly. According to previous work [51], we choose 400 training images and 100 testing images for each identification. Commercial DNN services are generally used for high-resolution images [4], [5], thus we focus on evaluating Falcon's performance on ImageNet and CelebA datasets.

Target Models. For each dataset, we select ResNet50 [3] and DenseNet169 [63] as target models that are widely used in existing studies [15], [18], [16]. On CIFAR-10, we use ResNet50 [3] and DenseNet169 [63] with the same architectures and pre-trained weights as in previous work [15]. For ImageNet, we follow prior works [64] and utilize models implemented with pre-trained weights from a HuggingFace repository [65]. Additionally, we train target models on ImageNet-10 and CelebA datasets. The summary of datasets and target models is presented in Table II.

Attacks. Based on the threat model in Sec. III-B, we evaluate the performance of Falcon when detecting static attacks and adaptive attacks. For static attacks, we select ten typical attacks including VNIFGSM [40], HJSA [47], Hybrid [59], DSA [8], PGD [7], CW [30], AutoAttack [25], SSAE [31], Kenneth [32] and AT-UAP [33]. For adaptive attacks, we design three strategies based on SOTA attacks DSA, PGD and CW. More details will be presented in Sec. VI-E.

Metrics. The performance of Falcon focuses on two key metrics: the detection rate of adversarial examples and the false positive rate on clean examples. Thus we use two widely adopted metrics in our evaluation: True Positive Rate (TPR) and False Positive Rate (FPR) [66], [15]. The TPR indicates the fraction of adversarial examples correctly detected, while the FPR reflects the fraction of clean examples incorrectly flagged as adversarial. A lower FPR results in a lower TPR. Thus, there is a trade-off between TPR and FPR. Following previous works [20], [15], we report TPR when fixing FPR as 5%.

Baselines. Six SOTA detections without dependency on full knowledge of the target model serve as baselines, including BEYOND [39], CNet [15], MIAED [18], FS [21], MagNet [20], and RS [24]. For a more comprehensive comparison, we also include A^2D, which requires dependency on the target model. The parameters of baselines are all set to their default values to guarantee its performance. RS is originally designed to directly defend adversarial perturbations and ensure correct classification. Here, we adopt them for detection by examining whether the predicted label of an adversarial example changes for the detection task.

B. Experiments in Detecting Static Attacks

In this subsection, we conduct experiments to evaluate the accuracy of Falcon in detecting static attacks across four datasets, using ResNet50 as the target model. Since the pertur-

TABLE IV: TPR (%) and FPR (%) for detecting untargeted static attacks under ResNet50 and DenseNet169. In this scenario the detector can only get a subset of all potential output labels.

Target Model	Detections	VNI[40]		HJSA[47]		Hybrid[59]		DSA[8]		PGD[7]		CW[30]		AA[25]		SSAE[31]		Kenneth[32]		AT-UAP[33]	
		TPR↑	FPR↓	TPR↑	FPR↓	TPR↑	FPR↓	TPR↑	FPR↓	TPR↑	FPR↓	TPR↑	FPR↓	TPR↑	FPR↓	TPR↑	FPR↓	TPR↑	FPR↓	TPR↑	FPR↓
ResNet50	FS [21]	26.45	5.00	26.88	5.00	25.90	5.00	25.20	5.00	23.40	5.00	23.58	5.00	20.15	5.00	25.17	5.00	20.26	5.00	18.45	5.00
	MagNet [20]	33.62	5.00	33.17	5.00	32.88	5.00	33.50	5.00	34.70	5.00	34.35	5.00	31.20	5.00	34.82	5.00	33.97	5.00	34.06	5.00
	RS [24]	57.16	10.02	64.20	10.02	56.22	10.02	56.96	10.02	58.16	10.02	62.58	10.02	52.16	10.02	58.15	10.02	57.46	10.02	56.11	10.02
	Falcon	**83.50**	**5.00**	**83.68**	**5.00**	**83.40**	**5.00**	**81.90**	**5.00**	**82.34**	**5.00**	**81.97**	**5.00**	**80.50**	**5.00**	**84.72**	**5.00**	**82.05**	**5.00**	**81.13**	**5.00**
DenseNet169	FS [21]	27.69	5.00	27.84	5.00	26.44	5.00	24.50	5.00	24.33	5.00	24.10	5.00	23.58	5.00	26.42	5.00	25.69	5.00	23.88	5.00
	MagNet [20]	34.28	5.00	34.10	5.00	34.66	5.00	33.28	5.00	35.19	5.00	34.87	5.00	32.48	5.00	35.46	5.00	35.68	5.00	33.94	5.00
	RS [24]	59.22	11.30	64.76	11.30	56.60	11.30	57.35	11.30	58.44	11.30	62.58	11.30	52.16	11.30	58.67	11.30	58.00	11.30	56.40	11.30
	Falcon	**84.46**	**5.00**	**85.10**	**5.00**	**83.64**	**5.00**	**82.34**	**5.00**	**84.03**	**5.00**	**83.60**	**5.00**	**81.63**	**5.00**	**84.06**	**5.00**	**86.34**	**5.00**	**82.16**	**5.00**

TABLE V: Summary of Detection Overhead.

Detections	Query Magnitude	Detection Time Overhead(s)		
		Scenario#1	Scenario#2	
		ResNet50	ResNet50	Baidu
CNet [15]	1	0.22	0.24	1.50
MIAED [18]	1	0.15	0.17	1.52
FS [21]	4	0.01	0.03	2.80
MagNet [20]	2	0.18	0.21	1.40
RS [24]	1.E+3	4.00	4.90	700
Falcon	2	0.20	0.22	1.60

bation in untargeted attacks is typically easier to generate with a small budget and more challenging to detect [8], we focus on detecting untargeted attacks in our experiments. We select 10 typical attacks in Sec. VI-A to generate adversarial examples for each clean example. The detector may only get a subset of output labels to construct shadow dataset. Thus, we conduct experiments under two scenarios mentioned in Sec. III-B.

Scenario#1. For each dataset, we randomly select 50% of the training images from each label and divide them equally into two parts: one for training the target model and the other for training Falcon. Note that Falcon determines the threshold for adversarial example detection using only clean examples. As a result, its FPR is independent of the attack strategies and remains the same detecting different attacks. This property also holds for A^2D, BEYOND, CNet, FS and MagNet. MIAED and RS achieve detection through judging whether output label changes. This decision is binary and threshold-free, resulting in a fixed FPR that cannot be tuned to 5%.

As summarized in Table III, Falcon achieves the highest TPR and the lowest FPR across four datasets when detecting ten attacks, outperforming compared baselines. Falcon achieves the best accuracy on ImageNet-10, with a TPR of over 94% when fixing FPR as 5%. Falcon achieves the maximum improvement in detecting SOTA attacks DSA and AutoAttack. When detecting DSA on ImageNet-1000, the TPR of Falcon is higher than that of BEYOND, A^2D, CNet, MIAED, FS, MagNet, and RS by 22.44%, 23.00%, 23.30%, 21.89%, 72.57%, 59.74%, 30.06%, respectively.

The performance of BEYOND, CNet and MIAED significantly declines on dataset containing images of more labels (e.g., ImageNet-1000). This is because a larger number of labels increases the complexity of the target model, making it more difficult for the auxiliary model to match its behavior. A^2D leverages the difficulty of applying additional adversarial attacks for detection. But the perturbation may shift the input in arbitrary directions rather than along the minimal path to the decision boundary, limiting the reliability of this approach. The performance of FS, MagNet and RS declines on dataset containing images of high resolution. Higher image resolution introduces more detailed information, which may act as noise and mislead the detection process of these methods.

Additionally, existing methods struggle to detect DSA and AutoAttack effectively, as these attacks employ lower perturbation budgets and leverage diverse generation strategies. SSAE generates perturbations of low magnitude, thereby limiting the effectiveness of methods like MagNet that partially rely on pixel-level differences. AT-UAP generates universal adversarial perturbations with robustness to common image transformations (e.g., Gaussian noise, rotation), which reduces the effectiveness of detection methods leveraging such transformations for data augmentation (e.g., FS, RS, BEYOND). Falcon maintains its performance against adversarial perturbations generated by different mechanisms. This stems from Falcon's noise distribution generation module, which aims to capture the distribution of adversarial noise. Building on this, Falcon further designs the noise intensity to form constructive noise that counteracts these perturbations.

Scenario#2. In this scenario, the detector can only get a subset of output labels to construct shadow dataset. We use the training images from ImageNet-1000 to train the target model and the training images from ImageNet-10 to train Falcon. The testing images of ImageNet-1000 serve as clean examples, from which adversarial examples are generated using the attacks described in Sec. VI-A. Considering that CNet and MIAED completely fail when handling images whose labels are not covered by ImageNet-10, we choose FS, MagNet and RS as baselines to compare.

As summarized in Table IV, Falcon achieves a TPR of more than 80%, while maintaining a fixed FPR of 5%. In contrast, the TPR of MagNet and FS experiences a decline of over 20%, highlighting their sensitivity to unseen labels. Since these methods rely on patterns learned from labels in training data, they struggle to generalize to inputs from unseen labels, resulting in reduced detection accuracy. In contrast, Falcon leverages the property that adversarial examples tend to lie closer to the decision boundary. Moreover, adversarial perturbations commonly manifest as high-frequency components across datasets of various labels. Falcon generates constructive noise that selectively captures the distribution of these high-frequency components, while preserving features essential for

TABLE VI: TPR (%) and FPR (%) for detecting attacks on commercial DNN services.

Commercial DNN Services		Baidu [26]		Tencent [27]		AWS [4]		Azure [5]		Alibaba [29]		Google [28]	
Attacks	Detections	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
VNIFGSM	FS [21]	26.00	5.00	25.00	5.00	29.00	5.00	22.00	5.00	28.00	5.00	24.00	5.00
	MagNet [20]	33.00	5.00	31.00	5.00	28.00	5.00	32.00	5.00	31.00	5.00	29.00	5.00
	RS [24]	73.00	13.00	71.00	12.00	68.00	12.00	70.00	13.00	66.00	12.00	69.00	11.00
	Falcon	**96.00**▲23.00	**5.00**	**90.00**▲19.00	**5.00**	**85.00**▲17.00	**5.00**	**86.00**▲16.00	**5.00**	**90.00**▲24.00	**5.00**	**88.00**▲19.00	**5.00**
HJSA	FS [21]	46.00	5.00	48.00	5.00	44.00	5.00	47.00	5.00	46.00	5.00	47.00	5.00
	MagNet [20]	58.00	5.00	59.00	5.00	60.00	5.00	57.00	5.00	56.00	5.00	59.00	5.00
	RS [24]	76.00	13.00	72.00	12.00	71.00	12.00	73.00	13.00	72.00	15.00	73.00	14.00
	Falcon	**96.00**▲20.00	**5.00**	**91.00**▲19.00	**5.00**	**86.00**▲15.00	**5.00**	**87.00**▲14.00	**5.00**	**89.00**▲17.00	**5.00**	**90.00**▲17.00	**5.00**
Hybrid	FS [21]	19.00	5.00	22.00	5.00	26.00	5.00	21.00	5.00	18.00	5.00	20.00	5.00
	MagNet [20]	29.00	5.00	28.00	5.00	24.00	5.00	31.00	5.00	26.00	5.00	31.00	5.00
	RS [24]	71.00	13.00	69.00	12.00	64.00	12.00	66.00	13.00	67.00	14.00	71.00	11.00
	Falcon	**95.00**▲24.00	**5.00**	**87.00**▲18.00	**5.00**	**79.00**▲15.00	**5.00**	**79.00**▲13.00	**5.00**	**86.00**▲19.00	**5.00**	**88.00**▲17.00	**5.00**
DSA	FS [21]	9.00	5.00	11.00	5.00	13.00	5.00	6.00	5.00	6.00	5.00	8.00	5.00
	MagNet [20]	16.00	5.00	7.00	5.00	17.00	5.00	9.00	5.00	8.00	5.00	11.00	5.00
	RS [24]	68.00	13.00	65.00	12.00	59.00	12.00	63.00	13.00	64.00	14.00	65.00	13.00
	Falcon	**94.00**▲26.00	**5.00**	**86.00**▲21.00	**5.00**	**79.00**▲20.00	**5.00**	**78.00**▲15.00	**5.00**	**87.00**▲23.00	**5.00**	**85.00**▲20.00	**5.00**

correct classification. This design allows Falcon to maintain robust accuracy for inputs from unseen labels.

C. Detection Overhead

When adversarial example detection is deployed for usage, the inference time will impact its practicality. We define detection overhead in terms of two aspects: query magnitude (i.e., the number of queries to the target model) and the detection time overhead (i.e., the time required to determine whether a submitted input is adversarial).

Table V summarizes the detection overhead of Falcon compared to the baselines under various scenarios. Falcon completes detection within approximately 0.2 seconds, requiring only two queries to the target model. The detection time overhead of Falcon is large when serving detection for Baidu. This stems from the high communication overhead of visiting cloud services. However, by deploying Falcon as an integrated system alongside commercial DNN services, the communication overhead between the detector and the model can be significantly minimized [67]. Although RS achieves a TPR of over 60% when serving detection for commercial DNN services, it requires a large number of queries to the target model, resulting in significant time consumption and limiting its applicability in practice.

D. Experiments on Commercial DNN Services

In this subsection, we conduct experiments to evaluate the accuracy of Falcon when serving detection for commercial DNN services. We choose six well-known commercial DNN services, namely Baidu [26], Tencent [27], AWS [4], Azure [5], Alibaba [29] and Google [28] as the target model, which are widely used in the real world [68], [8]. Query results returned by the APIs consist of labels with confidence scores. Referring to the settings of the previous research [8], We randomly select 100 test images from ImageNet [61], excluding those belonging to ImageNet-1000 labels, to serve as clean examples. Given that the adversary cannot have access to details of these services, we select VNIFGSM, HJSA, Hybrid and DSA to generate adversarial examples, respectively. Falcon is trained on ImageNet-1000 in advance. In addition, the detector cannot get full label set of these services. As this setting aligns with Scenario#2 mentioned in Sec. VI-B, we consider only FS, MagNet, and RS for comparison.

As summarized in Table VI, Falcon obtains a maximum TPR of 96% when fixing FPR as 5%. Compared to other methods, Falcon achieves an improvement of over 13% while maintaining a lower FPR, which indicates that Falcon can serve accurate detection for existing commercial DNN services. Since Falcon relies exclusively on the predicted labels and confidence scores provided by commercial DNN APIs, it can maintain its performance across different platforms.

E. Experiments in Detecting Adaptive Attacks

In this subsection, we evaluate the accuracy of Falcon when detecting more powerful attacks. We assume that the adversary can obtain full knowledge of the target model and Falcon to conduct adaptive attacks. First, we give a detailed adaptive object loss function design to best utilize the knowledge of Falcon. Then, we design three different strategies and apply them to three well-konwn attacks: DSA, PGD and CW. We choose ResNet50 as the target model and use testing images in Sec. VI-A. For attacks based on DSA, we set the query budget as 200 and use its default values [8]. For attacks based on PGD, we set iteration steps as 40 and l_∞ constraint ($l_\infty = 8.0/255$ for all datasets). For attacks based on CW, we set optimization steps as 100 and l_2 constraint ($l_2 = 1.0$ for CIFAR-10 and $l_2 = 16.0$ for other datasets).

Customizable Adaptive Objective Loss Function. Successful adaptive attacks seek to achieve two objectives: bypassing Falcon's detection mechanism and misleading the target model. For the first goal of evading Falcon, it is unnecessary to consider every component of the detection loss. A highly complex objective loss function could hinder optimization, making the attack less effective. Instead, the adversary focuses on the most exploitable weaknesses in the detection process of Falcon [15]. For the second goal of misleading the target model, the classification loss is the key to deceiving the target model. Thus, the adversary must consider this loss.

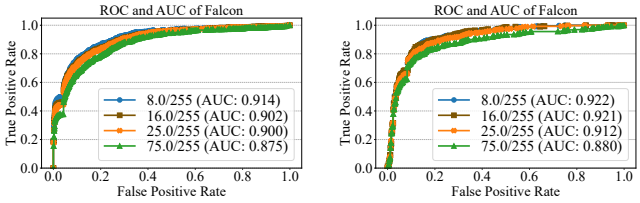
The design details of customizable adaptive objective loss function are presented in Appendix C. Then, we introduce three strategies of adaptive attacks.

TABLE VII: TPR (%) and FPR (%) for detecting Adaptive Attacks (Target Model: ResNet50).

Scenarios	Datasets	Adaptive DSA				Adaptive PGD						Adaptive CW			
		Strategy#1		Strategy#2		Strategy#1		Strategy#2		Strategy#3		Strategy#1		Strategy#2	
		TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
Scenario#1	CIFAR-10	87.44	5.00	81.20	5.00	90.46	5.00	66.49	5.00	68.76	5.00	89.96	5.00	70.80	5.00
	ImageNet-10	93.50	5.00	87.99	5.00	98.40	5.00	68.75	5.00	69.88	5.00	98.52	5.00	69.99	5.00
	ImageNet-1000	88.54	5.00	82.39	5.00	92.54	5.00	60.50	5.00	63.98	5.00	92.76	5.00	63.78	5.00
	CelebA	88.00	5.00	79.60	5.00	90.10	5.00	61.17	5.00	62.50	5.00	97.80	5.00	64.00	5.00
Scenario#2	ImageNet-1000	80.20	5.00	75.48	5.00	81.10	5.00	52.85	5.00	51.93	5.00	80.05	5.00	51.71	5.00

TABLE VIII: TPR(%) and FPR(%) for detecting Orthogonal-PGD attacks.

Detections	Scenario#1				Scenario#2			
	Orth		Select		Orth		Select	
	TPR	FPR	TPR	FPR	TPR	FPR	TPR	FPR
CNet[15]	47.85	5.00	48.66	5.00	N/A	N/A	N/A	N/A
MIAED[18]	10.05	5.00	9.88	5.00	N/A	N/A	N/A	N/A
FS[21]	0.14	5.00	0.08	5.00	0.12	5.00	0.14	5.00
MagNet[20]	1.26	5.00	1.38	5.00	1.18	5.00	1.05	5.00
RS[24]	58.06	10.12	58.06	10.12	57.48	10.85	57.61	10.85
Falcon	64.35	5.00	63.98	5.00	67.84	10.00	67.59	10.00



(a) Strategy#2 (b) Strategy#3

Fig. 7: ROC and AUC of Falcon. ROCs are generated under various adaptive PGD attacks based on Strategy#2 and Strategy#3 with different perturbation budgets.

Strategy#1. Existence of constructive noise in Sec. IV-A is based on minimizing perturbations on regions critical for correct classification. Therefore, we constrain the distribution of adversarial noise to design adaptive attacks, ensuring it blends with these important regions. We identify these critical regions using CAM, as described in Sec. IV-A. Only adversarial noise on these regions are optimized.

Strategy#2. The aim of adaptive attacks is to evade Falcon while misleading the target model. Therefore, we balance classification loss and customizable adaptive loss function to achieve these two goals respectively in the process of optimizing adversarial noise. More details of adaptive attacks with Strategy#2 are presented in Appendix C.

Strategy#3. We employ a rising adaptive attack benchmark Orthogonal Projected Gradient Descent (Orthogonal-PGD [46]) which focuses on breaking detection-based methods, further evaluating the robustness of Falcon. There are two attack strategies in Orthogonal-PGD, Selective strategy(Select) and Orthogonal strategy (Orth).

First, we evaluate the accuracy of Falcon when detecting attacks using three different strategies. As shown in Table VII, Falcon can maintain a TPR of over 50% when fixing FPR as 5% when detecting these attacks. The TPR of Falcon changes slightly compared to relative static attacks (maximum 1.5%) when the adversary uses Strategy#1, indicating that generated constructive noise can still defeat adversarial noise that is added on critical regions. When the adversary uses Strategy#2 and Strategy#3 to conduct adaptive attacks, Falcon maintains a TPR of over 64% in Scenario#1 and 50% in Scenario#2.

Then we further evaluate the robustness of Falcon using Orthogonal-PGD benchmark. The accuracy of Falcon and baselines is summarized in Table VIII. Falcon can maintain a TPR of over 50% while baselines almost fail completely. Although CNet can achieve a TPR of about 50%, it rely on prior knowledge of full set of output labels. CNet will fail completely under Scenario#2. RS maintains a TPR of over 50%. However, it requires a large number of queries to the target model, resulting in significant time consumption. The robustness of Falcon against adaptive attacks stems from its ability to generate input-specific noise distributions and intensities to form constructive noise, enabling it to dynamically capture the distribution of adversarial noise to defeat it.

Finally, we investigate the accuracy of Falcon changes when the perturbation budget is large. Based on the previous works [52], [15], we set various perturbation budgets l_∞ including 8.0/255, 16.0/255, 25.0/255 and 75.0/255. As shown in Fig. 7, the TPR of Falcon cannot change significantly when the perturbation budget is large.

F. Sensitivity of Falcon to Different Training Datasets

In this subsection, we include sensitivity analysis to changes in the distribution. To investigate the sensitivity of Falcon, we conduct experiments across four datasets: CIFAR-10, ImageNet-10, ImageNet-1000, and CelebA. We randomly select 10,000 images from these datasets for clean examples and choose SOTA attack AutoAttack to generate adversarial examples, respectively. In each experiment, one dataset is used as the training dataset, and the trained Falcon is then evaluated on all four datasets to measure its sensitivity to distributional changes. We use TPR as the metric when fixing FPR as 0.05.

The confusion matrix in Fig. 8 shows that Falcon maintains a TPR of at least 0.59 across all training–testing combinations among CIFAR-10, ImageNet-10, and ImageNet-1000. This is because all three datasets focus on image classification and share overlapping object categories and visual patterns. Specifically, Falcon trained on ImageNet-1000 achieves a TPR of 0.84 on CIFAR-10. In contrast, Falcon trained on CIFAR-10 only maintains a TPR of 0.59 on ImageNet-1000. This results from the low resolution and limited class categories of

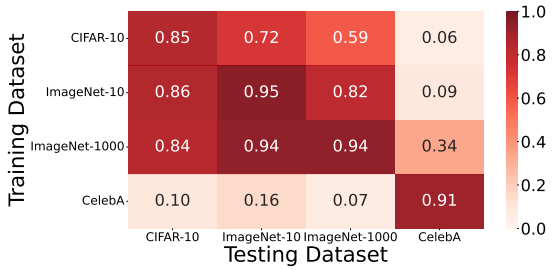


Fig. 8: The TPR of Falcon when training dataset and testing dataset are randomly chosen from four datasets (i.e., CIFAR-10, ImageNet-10, ImageNet-1000, CelebA).

TABLE IX: The contribution of each module in Falcon (%).

Modules	Variations	DSA [8] TPR	DSA [8] FPR	AutoAttack [25] TPR	AutoAttack [25] FPR
Falcon	**Full**	90.02	5.00	93.82	5.00
Noise Distribution	Gaussian Noise	51.08	5.00	47.82	5.00
	2D Gradients	53.50	5.00	51.88	5.00
	Upscaling and Downscaling	60.87	5.00	61.22	5.00
	High-pass Filter	57.41	5.00	56.71	5.00
Noise Intensity	Fixed Value (σ=0.5)	76.58	5.00	75.44	5.00
	Fixed Value (σ=1.0)	84.68	5.00	83.25	5.00
	Fixed Value (σ=1.5)	84.19	5.00	83.76	5.00
	Fixed Value (σ=2.0)	74.52	5.00	74.05	5.00
Detection	w/o Confience Scores	87.25	5.28	88.70	5.28

CIFAR-10, which limits Falcon's noise distribution generation module in capturing the distribution of adversarial noise, resulting in decreased performance. Falcon trained on CIFAR-10, ImageNet-10, or ImageNet-1000 exhibits a sharp decline in TPR, with values below 0.1 when evaluated on CelebA. Similarly, Falcon trained on CelebA achieves a TPR below 0.1 when evaluated on the other three datasets. This performance drop is due to the lack of overlap in both visual content and label categories between the face recognition dataset and the image classification datasets. A notable exception is Falcon trained on ImageNet-1000, which maintains a higher TPR of 0.31 on CelebA due to the presence of a subset of human face images within ImageNet-1000.

G. Ablation Study

In this section, we conduct a comprehensive ablation study to investigate important components of Falcon, i.e., noise distribution generation, noise intensity generation, and adversarial example detection as well as their corresponding contributions. We generate different variants by changing the settings of each component. For each component removal, we perform the following: 1) Substitute methods to generate noise distribution; 2) Add the generated noise distribution with fixed noise intensity; 3) Use only label without confidence score to achieve adversarial example detection.

First, we find that removing any component of Falcon results in a decrease in the TPR of adversarial examples and an increase in the FPR of clean examples, as shown in Table IX. For instance, Falcon without Noise Distribution Generation uses Gaussian noise directly, only achieving a TPR of about 52%. Considering that the core component of noise distribution is extracting key structure of image. We adopt existing methods that can extract the mentioned structure including UNet, low-pass filter and 2D gradient extraction. However, these methods only achieve a TPR of less than 65% when fixing FPR as 5%.

Second, the Noise Intensity Generation module plays a critical role in Falcon. The appropriate intensity varies for different inputs. Therefore, if we add the generated noise distribution directly for all inputs, the intensity may be lower to break some adversarial noise while larger to cause false alarms for some clean examples. The TPR increases by a maximal 15.50% by designing an appropriate intensity for each input.

Third, some adversarial attacks generate adversarial examples with high confidence scores (e.g., 99%). Adding generated noise in Falcon to these examples does not alter their labels but significantly reduces their confidence scores. Relying solely on labels for adversarial example detection results in approximately about 3% decrease in TPR.

VII. Discussion

Training Dataset. Falcon depends on the models in the noise distribution generation and noise intensity generation modules for detection, which are trained on clean examples. Using more clean examples in the training process helps improve the effectiveness of generated noise, thereby enhancing Falcon's performance further.

Ethical Consideration. All experiments conducted on commercial DNN services were strictly limited to our controlled research environment and were neither disclosed nor shared externally, thereby avoiding any potential risk to the platforms involved. Moreover, we select examples from the publicly available dataset, ensuring that no personal privacy or proprietary commercial information was compromised. We have informed all relevant companies of the potential vulnerability via official reporting channels or direct email communications. To date, we have received confirmation from AWS and Baidu.

VIII. Conclusion

In this paper, we proposed Falcon, an adversarial example detection achieving accuracy and cost efficiency simultaneously by leveraging constructive noise to defeat adversarial noise. Falcon leveraged the different noise tolerances of clean and adversarial examples by designing specific noise distributions and intensities to generate constructive noise. For each input, Falcon performed detection based on the differences in the target model's outputs before and after adding constructive noise. In multiple scenarios and datasets, Falcon achieved better performance than the SOTA methods with the same threat model and had a significant performance improvement when serving detection for several commercial DNN services.

In the future, we will investigate techniques to improve Falcon's detection performance when the detector can only obtain predicted labels via APIs, which represents a more challenging scenario. In addition, we plan to explore its applicability in broader domains, such as text classification and image segmentation, where adversarial threats also pose significant challenges.

ACKNOWLEDGMENTS

This work is partially supported by National Key R&D Program of China with No. 2023YFB2703800, NSFC Projects with Nos. U23A20304 and 62222201, and Beijing Natural Science Foundation with No. M23020.

AVAILABILITY

Implementations and data for reproducing our results are available at https://github.com/JiangYuanB/Falcon.

REFERENCES

[1] N. Abudarham, L. Shkiller, and G. Yovel, "Critical features for face recognition," *Cognition*, vol. 182, pp. 73–83, 2019.

[2] P. Nagrath, R. Jain, A. Madan, R. Arora, P. Kataria, and J. Hemanth, "Ssdmnv2: A real time dnn-based face mask detection system using single shot multibox detector and mobilenetv2," *Sustainable cities and society*, vol. 66, p. 102692, 2021.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE CVPR*. IEEE Computer Society, 2016, pp. 770–778.

[4] Amazon, "Aws," https://aws.amazon.com/cn/rekognition/, 2024, accessed 19 March 2024.

[5] Microsoft, "Azure," https://azure.microsoft.com/en-us/products/cognitive-services/vision-services/, 2024, accessed 19 March 2024.

[6] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[7] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[8] M. Shen, C. Li, Q. Li, H. Lu, L. Zhu, and K. Xu, "Transferability of white-box perturbations: Query-efficient adversarial attacks against commercial DNN services," in *USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*, D. Balzarotti and W. Xu, Eds. USENIX Association, 2024.

[9] M. Shen, H. Yu, L. Zhu, K. Xu, Q. Li, and J. Hu, "Effective and robust physical-world attacks on deep learning face recognition systems," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4063–4077, 2021.

[10] S. Peng, Y. Chen, J. Xu, Z. Chen, C. Wang, and X. Jia, "Intellectual property protection of dnn models," *World Wide Web*, vol. 26, no. 4, pp. 1877–1911, 2023.

[11] B. Zheng, P. Jiang, Q. Wang, Q. Li, C. Shen, C. Wang, Y. Ge, Q. Teng, and S. Zhang, "Black-box adversarial attacks on commercial speech platforms with minimal information," in *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*, 2021, pp. 86–107.

[12] Y. Dong, X. Yang, Z. Deng, T. Pang, Z. Xiao, H. Su, and J. Zhu, "Black-box detection of backdoor attacks with limited information and data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16 482–16 491.

[13] Y. Xin, Z. Li, N. Yu, D. Chen, M. Fritz, M. Backes, and Y. Zhang, "Inside the black box: Detecting data leakage in pre-trained language encoders," in *ECAI 2024*, pp. 3947–3955.

[14] B. Yi, T. Huang, S. Chen, T. Li, Z. Liu, Z. Chu, and Y. Li, "Probe before you talk: Towards black-box defense against backdoor unalignment for large language models," in *The Thirteenth International Conference on Learning Representations*, 2025.

[15] Y. Yang, R. Gao, Y. Li, Q. Lai, and Q. Xu, "What you see is not what the network infers: Detecting adversarial examples based on semantic contradiction," in *29th Annual Network and Distributed System Security Symposium, NDSS 2022, San Diego, California, USA, April 24-28, 2022*.

[16] S. Zhang, S. Chen, C. Hua, Z. Li, Y. Li, X. Liu, K. Chen, Z. Li, and W. Wang, "LSD: adversarial examples detection based on label sequences discrepancy," *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 5133–5147, 2023.

[17] X. Ma, B. Li, Y. Wang, S. M. Erfani, S. N. R. Wijewickrema, G. Schoenebeck, D. Song, M. E. Houle, and J. Bailey, "Characterizing adversarial subspaces using local intrinsic dimensionality," in *ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[18] S. Gao, R. Wang, X. Wang, S. Yu, Y. Dong, S. Yao, and W. Zhou, "Detecting adversarial examples on deep neural networks with mutual information neural estimation," *IEEE Trans. Dependable Secur. Comput.*, vol. 20, no. 6, pp. 5168–5181, 2023.

[19] P. Yang, J. Chen, C. Hsieh, J. Wang, and M. I. Jordan, "ML-LOO: detecting adversarial examples with feature attribution," in *AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 6639–6647.

[20] D. Meng and H. Chen, "Magnet: A two-pronged defense against adversarial examples," in *CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, B. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds. ACM, 2017, pp. 135–147.

[21] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *NDSS 2018, San Diego, California, USA, February 18-21, 2018*.

[22] H. Li, S. Shan, E. Wenger, J. Zhang, H. Zheng, and B. Y. Zhao, "Blacklight: Scalable defense for neural networks against query-based black-box attacks," in *USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, K. R. B. Butler and K. Thomas, Eds. USENIX Association, 2022, pp. 2117–2134.

[23] S. Chen, N. Carlini, and D. Wagner, "Stateful detection of black-box adversarial attacks," in *Proceedings of the 1st ACM Workshop on Security and Privacy on Artificial Intelligence*, 2020, pp. 30–39.

[24] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *international conference on machine learning*. PMLR, 2019, pp. 1310–1320.

[25] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *ICML 2020, 13-18 July 2020, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 2206–2216.

[26] Baidu, "Baidu ai," https://ai.baidu.com/tech/imagerecogn-ition/general, 2024, accessed 19 March 2024.

[27] Tencent, "Tencent cloud," https://cloud.tencent.com/product/imagetagging, 2024, accessed 19 March 2024.

[28] Google, "Google vision cloud," https://cloud.google.com/vision, 2025, accessed 5 July 2025.

[29] Alibaba, "Alibaba cloud," https://vision.console.aliyun.com, 2025, accessed 5 July 2025.

[30] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," in *SP 2017, San Jose, CA, USA, May 22-26, 2017*. IEEE Computer Society, 2017, pp. 39–57.

[31] S. Lu, Y. Xian, K. Yan, Y. Hu, X. Sun, X. Guo, F. Huang, and W. Zheng, "Discriminator-free generative adversarial attack," in *MM '21: ACM Multimedia Conference, Virtual Event, China, October 20 - 24, 2021*. ACM, 2021, pp. 1544–1552.

[32] K. T. Co, L. Muñoz-González, S. de Maupeou, and E. C. Lupu, "Procedural noise adversarial examples for black-box attacks on deep convolutional networks," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*. ACM, 2019, pp. 275–289.

[33] M. Li, Y. Yang, K. Wei, X. Yang, and H. Huang, "Learning universal adversarial perturbation by adversarial example," in *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*. AAAI Press, 2022, pp. 1350–1358.

[34] M. Shen, Z. Liao, L. Zhu, K. Xu, and X. Du, "VLA: A practical visible light-based attack on face recognition systems in physical world," *Proc.*

ACM Interact. Mob. Wearable Ubiquitous Technol., vol. 3, no. 3, pp. 103:1–103:19, 2019.

[35] W. Jia, Z. Lu, H. Zhang, Z. Liu, J. Wang, and G. Qu, "Fooling the eyes of autonomous vehicles: Robust physical adversarial examples against traffic sign recognition systems," in *29th Annual Network and Distributed System Security Symposium, NDSS 2022, San Diego, California, USA, April 24-28, 2022*. The Internet Society, 2022.

[36] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," in *NeurIPS Workshops*, 2017.

[37] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *CoRR*, vol. abs/1703.00410, 2017.

[38] Z. Zhao, G. Chen, T. Liu, T. Li, F. Song, J. Wang, and J. Sun, "Attack as detection: Using adversarial attack methods to detect abnormal examples," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 3, pp. 1–45, 2024.

[39] Z. He, Y. Yang, P.-Y. Chen, Q. Xu, and T.-Y. Ho, "Be your own neighborhood: Detecting adversarial examples by the neighborhood relations built on self-supervised learning," in *Proceedings of the 41st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 235. PMLR, 21–27 Jul 2024, pp. 18 063–18 080.

[40] X. Wang and K. He, "Enhancing the transferability of adversarial attacks through variance tuning," in *CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 2021, pp. 1924–1933.

[41] H. Salman, S. Jain, E. Wong, and A. Madry, "Certified patch robustness via smoothed vision transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 15 137–15 147.

[42] C. Xiang, S. Mahloujifar, and P. Mittal, "PatchCleanser: Certifiably robust defense against adversarial patches for any image classifier," in *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA, 2022, pp. 2065–2082.

[43] C. Xiang, T. Wu, S. Dai, J. Petit, S. Jana, and P. Mittal, "Patchcure: Improving certifiable robustness, model utility, and computation efficiency of adversarial patch defenses," in *33rd USENIX Security Symposium (USENIX Security)*, 2024.

[44] X. Yang, K. Zhou, Y. Lai, and G. Li, "Defense-as-a-service: Black-box shielding against backdoored graph models," *arXiv preprint arXiv:2410.04916*, 2024.

[45] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. J. Goodfellow, A. Madry, and A. Kurakin, "On evaluating adversarial robustness," *CoRR*, vol. abs/1902.06705, 2019.

[46] O. Bryniarski, N. Hingun, P. Pachuca, V. Wang, and N. Carlini, "Evading adversarial example detection defenses with orthogonal projected gradient descent," in *ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[47] J. Chen, M. I. Jordan, and M. J. Wainwright, "Hopskipjumpattack: A query-efficient decision-based attack," in *2020 ieee symposium on security and privacy (sp)*. IEEE, 2020, pp. 1277–1294.

[48] M. Shen, C. Li, H. Yu, Q. Li, L. Zhu, and K. Xu, "Decision-based query efficient adversarial attack via adaptive boundary learning," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 4, pp. 1740–1753, 2024.

[49] F. Wang, X. Zuo, H. Huang, and G. Chen, "ADBA: approximation decision boundary approach for black-box adversarial attacks," in *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*. AAAI Press, 2025, pp. 7628–7636.

[50] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, "Grad-cam: Why did you say that?" *arXiv preprint arXiv:1611.07 450*, 2016.

[51] X. Jia, X. Wei, X. Cao, and H. Foroosh, "Comdefend: An efficient image compression model to defend adversarial examples," in *IEEE CVPR*. Computer Vision Foundation / IEEE, 2019, pp. 6084–6092.

[52] P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis," in *CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 2021, pp. 12 873–12 883.

[53] C. Zheng, T. Vuong, J. Cai, and D. Phung, "Movq: Modulating quantized vectors for high-fidelity image generation," in *NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., 2022.

[54] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 2017, pp. 1243–1252.

[55] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE transactions on image processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[56] X. Mao, C. Shen, and Y.-B. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," *Advances in neural information processing systems*, vol. 29, 2016.

[57] M. Shen, K. Ji, Z. Gao, Q. Li, L. Zhu, and K. Xu, "Subverting website fingerprinting defenses with robust traffic representation," in *32nd USENIX Security Symposium (USENIX Security 23)*, Anaheim, CA, 2023, pp. 607–624.

[58] A. Subramanian, E. Sizikova, N. Majaj, and D. Pelli, "Spatial-frequency channels, shape bias, and adversarial robustness," vol. 36, 2023, pp. 4137–4149.

[59] F. Suya, J. Chi, D. Evans, and Y. Tian, "Hybrid batch attacks: Finding black-box adversarial examples with limited queries," in *29th USENIX security symposium (USENIX Security 20)*, 2020, pp. 1327–1344.

[60] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[61] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[62] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, 2015, pp. 3730–3738.

[63] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 2261–2269.

[64] S. Ma, Y. Liu, G. Tao, W. Lee, and X. Zhang, "NIC: detecting adversarial samples with neural network invariant checking," in *NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019.

[65] HuggingFace, "Hugging face," https://huggingface.co/, 2023, accessed 10 January 2024.

[66] A. Aldahdooh, W. Hamidouche, S. A. Fezza, and O. Déforges, "Adversarial example detection for dnn models: A review and experimental comparison," *Artificial Intelligence Review*, vol. 55, no. 6, pp. 4403–4462, 2022.

[67] Y. L. Khaleel, M. A. Habeeb, A. Albahri, T. Al-Quraishi, O. Albahri, and A. Alamoodi, "Network and cybersecurity applications of defense in adversarial attacks: A state-of-the-art using machine learning and deep learning methods," *Journal of Intelligent Systems*, vol. 33, no. 1, p. 20240153, 2024.

[68] Y. Mao, C. Fu, S. Wang, S. Ji, X. Zhang, Z. Liu, J. Zhou, A. X. Liu, R. Beyah, and T. Wang, "Transfer attacks revisited: A large-scale empirical study in real computer vision settings," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1423–1439.

[69] W. He, B. Li, and D. Song, "Decision boundary analysis of adversarial examples," in *International Conference on Learning Representations*, 2018.

[70] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017.

[71] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 2574–2582.

[72] M. Shen, J. Wu, K. Ye, K. Xu, G. Xiong, and L. Zhu, "Robust detection of malicious encrypted traffic via contrastive learning," *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 4228–4242, 2025.

[73] Y. Cao, N. Wang, C. Xiao, D. Yang, J. Fang, R. Yang, Q. A. Chen, M. Liu, and B. Li, "Invisible for both camera and lidar: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks," in *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*. IEEE, 2021, pp. 176–194.

APPENDIX

A. Theoretical Analysis

In this subsection, we give the formal definition of TG and justify that the value of TG is larger than 0 under certain constraints. These constraints motivate us to further design modules to generate constructive noise.

Let $f(\cdot)$ be the target model and $\mathcal{D}(\cdot)$ be the generated noise distribution. Let $d(\cdot)$ be the minimal offset to the decision boundary of the target model (i.e., the minimal perturbation to change the output label of $f(\cdot)$) [24]. For noise distribution $n \in \mathcal{D}(x)$ generated from any input x, we can derive that:

$$\begin{cases} f(x) = f(x+n) & \text{s.t.} \frac{nd(x)}{\|d(x)\|} < \|d(x)\|, \\ f(x) \neq f(x+n) & \text{s.t.} \frac{nd(x)}{\|d(x)\|} > \|d(x)\|. \end{cases} \tag{8}$$

Where $\frac{nd(x)}{\|d(x)\|}$ denotes the projection of the added noise n onto the direction of the minimal perturbation $d(x)$. If $\frac{nd(x)}{\|d(x)\|} > \|d(x)\|$, the noise n pushes x beyond the decision boundary, resulting in a label change.

Then we analyze the maximum intensity $\sigma_{max}(x, \mathcal{D}(x))$ to add any noise distribution $n \in \mathcal{D}(x)$ to clean example x, achieving that the predicted label of the target model will not change after adding noise.

We can define $\sigma_{max}(x, \mathcal{D}(x))$ for any $n \in \mathcal{D}(x)$ as:

$$\sigma_{max}(x, \mathcal{D}(x)) \frac{nd(x)}{\|d(x)\|} \leq \|d(x)\|. \tag{9}$$

For any adversarial example $x+\delta$, we can define $\sigma_{max}(x+\delta, \mathcal{D}(x+\delta))$ for any $n \in \mathcal{D}(x+\delta)$ as:

$$\sigma_{max}(x+\delta, \mathcal{D}(x+\delta)) \frac{nd(x+\delta)}{\|d(x+\delta)\|} \leq \|d(x+\delta)\| \tag{10}$$

Then TG can be defined as:

$$\text{TG} = \sigma_{max}(x, \mathcal{D}(x)) - \sigma_{max}(x+\delta, \mathcal{D}(x+\delta)). \tag{11}$$

Theorem 1. *Let $\mathbb{P}(\cdot)$ refers to the probability of an event. Assume that $\mathbb{P}(\|d(x)\| > \|d(x+\delta)\|) > \gamma$. Let $\mathcal{D}(\cdot)$ be the generated noise distribution. For any clean example x and its any potential adversarial example $x+\delta$, there always exists $\mathbb{P}(TG > 0) > \gamma$ when $max(\frac{\mathcal{D}(x)d(x)}{\|d(x)\|}) < min(\frac{\mathcal{D}(x+\delta)d(x+\delta)}{\|d(x+\delta)\|})$.*

Theorem 1 indicates that minimizing $\frac{\mathcal{D}(x)d(x)}{\|d(x)\|}$ while maximizing $\frac{\mathcal{D}(x+\delta)d(x+\delta)}{\|d(x+\delta)\|}$ facilitates achieving TG>0. Based on this insight, we can design a noise distribution that perturbs regions critical for correct classification less, increasing directional discrepancy between $\mathcal{D}(x)$ and $d(x)$ and making $\frac{\mathcal{D}(x)d(x)}{\|d(x)\|}$ less. In addition, the noise distribution is expected to locate where adversarial noise is added, reducing directional discrepancy between $\mathcal{D}(x+\delta)$ and $d(x+\delta)$ and making $\frac{\mathcal{D}(x+\delta)d(x+\delta)}{\|d(x+\delta)\|}$ larger. When guaranteeing TG>0, we can further select proper noise intensity to generate constructive noise.

Then, we give the proof of Theorem 1 and further analyze why Gaussian noise cannot balance detection performance and low false positives at the same time.

Proof. [Proof of Theorem 1] Assume that $\mathbb{P}(d(x) > d(x+\delta)) > \gamma$. For any clean example x and its any potential adversarial example $x+\delta$, if satisfying $max(\frac{\mathcal{D}(x)d(x)}{\|d(x)\|}) < min(\frac{\mathcal{D}(x+\delta)d(x+\delta)}{\|d(x+\delta)\|})$, then we can derive that the offset caused by added noise distribution n from $\mathcal{D}(x+\delta)$ is closer to the direction of minimal offset to the decision boundary.

We leverage the examples' distance to the decision boundary as the basis for the proof of Theorem 1. $\mathcal{D}(\cdot)$ is the generated noise distribution and $d(\cdot)$ is the minimal offset to the decision boundary (Eq. (2)). First, we define n_c and n_a as

$$\begin{aligned} n_c &= \arg\max_{n \in \mathcal{D}(x)} \left(\frac{nd(x)}{\|d(x)\|} \right), \\ n_a &= \arg\max_{n \in \mathcal{D}(x+\delta)} \left(\frac{nd(x+\delta)}{\|d(x+\delta)\|} \right), \end{aligned} \tag{12}$$

where n_c and n_a represent noise distributions that induce the largest offset toward the decision boundary (i.e., $\frac{n_c d(x)}{\|d(x)\|}$ and $\frac{n_a d(x+\delta)}{\|d(x+\delta)\|}$).

We can derive:

$$\begin{aligned} \sigma_{max}(x, \mathcal{D}(x)) &= \frac{\|d(x)\|^2}{n_c d(x)}, \\ \sigma_{max}(x+\delta, \mathcal{D}(x+\delta)) &= \frac{\|d(x+\delta)\|^2}{n_a d(x+\delta)}. \end{aligned} \tag{13}$$

$\frac{\mathcal{D}(x)d(x)}{\|d(x)\|}$ and $\frac{\mathcal{D}(x+\delta)d(x+\delta)}{\|d(x+\delta)\|}$ represent all possible offsets toward the decision boundary. Thus, we can obtain:

$$\begin{aligned} \frac{n_c d(x)}{\|d(x)\|} &= \max(\frac{\mathcal{D}(x)d(x)}{\|d(x)\|}), \\ \frac{n_a d(x+\delta)}{\|d(x+\delta)\|} &= \max(\frac{\mathcal{D}(x+\delta)d(x+\delta)}{\|d(x+\delta)\|}). \end{aligned} \tag{14}$$

If $\max(\frac{\mathcal{D}(x)d(x)}{\|d(x)\|}) < \min(\frac{\mathcal{D}(x+\delta)d(x+\delta)}{\|d(x+\delta)\|})$ holds, we can have:

$$\begin{aligned} \frac{n_c d(x)}{\|d(x)\|} = \max(\frac{\mathcal{D}(x)d(x)}{\|d(x)\|}) < \min(\frac{\mathcal{D}(x+\delta)d(x+\delta)}{\|d(x+\delta)\|}) \leq \\ \max(\frac{\mathcal{D}(x+\delta)d(x+\delta)}{\|d(x+\delta)\|}) = \frac{n_a d(x+\delta)}{\|d(x+\delta)\|}. \end{aligned} \tag{15}$$

Thus, $\max(\frac{\mathcal{D}(x)d(x)}{\|d(x)\|}) < \min(\frac{\mathcal{D}(x+\delta)d(x+\delta)}{\|d(x+\delta)\|})$ implies $\frac{n_c d(x)}{\|d(x)\|} < \frac{n_a d(x+\delta)}{\|d(x+\delta)\|}$.

The value of TG depends on the projection toward the decision boundary and the minimal offset to the decision boundary. If $\mathbb{P}(\frac{n_c d(x)}{\|d(x)\|} < \frac{n_a d(x+\delta)}{\|d(x+\delta)\|}) = 1$ holds, the value of

TABLE X: Detailed configurations of adversarial examples used in the evaluation. ASR refers to the rate of successful adversarial examples in all generated adversarial examples.

Attack Type	Attack Method	Parameters Introduction	Norm Type	Dataset	Parameters Settings	ASR
Gradient-based	PGD [7]	δ: perturbation budget s: number of update steps to iterate.	l_∞	CIFAR-10 ImageNet-10 ImageNet-1000 CelebA	$\delta = 8.0/255, s = 40$ $\delta = 8.0/255, s = 40$ $\delta = 8.0/255, s = 40$ $\delta = 8.0/255, s = 40$	98.24% 99.40% 97.20% 97.55%
Optimization-based	CW [30]	k: confidence s: binary search steps m: attack iteration	l_2	CIFAR-10 ImageNet-10 ImageNet-1000 CelebA	$k = 1.0, s = 10, m = 100$ $k = 1.0, s = 10, m = 100$ $k = 1.0, s = 10, m = 100$ $k = 1.0, s = 10, m = 100$	97.81% 98.53% 97.83% 99.85%
Adaptive-based	AutoAttack [25]	δ: perturbation budget	l_∞	CIFAR-10 ImageNet-10 ImageNet-1000 CelebA	$\delta = 8.0/255$ $\delta = 8.0/255$ $\delta = 8.0/255$ $\delta = 8.0/255$	100.00% 100.00% 100.00% 100.00%
Query-based	HJSA [47]	δ: perturbation budget m: query magnitude	l_2	CIFAR-10 ImageNet-10 ImageNet-1000 CelebA	$\delta = 1.0, m = 100000$ $\delta = 16.0, m = 100000$ $\delta = 16.0, m = 100000$ $\delta = 16.0, m = 100000$	97.48% 97.87% 96.83% 94.05%
Transfer-based	VNIFGSM [40]	δ: perturbation budget s: number of update steps to perform.	l_∞	CIFAR-10 ImageNet-10 ImageNet-1000 CelebA	$\delta = 8.0/255, s = 10$ $\delta = 8.0/255, s = 10$ $\delta = 8.0/255, s = 10$ $\delta = 8.0/255, s = 10$	85.88% 82.16% 69.91% 77.50%
Hybrid-based	Hybrid [59]	δ: perturbation budget m: query magnitude	l_2	CIFAR-10 ImageNet-10 ImageNet-1000 CelebA	$\delta = 1.0, m = 1000$ $\delta = 16.0, m = 1000$ $\delta = 16.0, m = 1000$ $\delta = 16.0, m = 1000$	93.37% 96.20% 91.78% 91.20%
	DSA [8]	δ: perturbation budget m: query magnitude	l_2	CIFAR-10 ImageNet-10 ImageNet-1000 CelebA	$\delta = 1.0, m = 100$ $\delta = 16.0, m = 100$ $\delta = 16.0, m = 100$ $\delta = 16.0, m = 100$	97.75% 99.12% 98.56% 99.60%
GAN-based	SSAE [31]	δ: perturbation budget	l_∞	CIFAR-10 ImageNet-10 ImageNet-1000 CelebA	$\delta = 8.0/255$ $\delta = 8.0/255$ $\delta = 8.0/255$ $\delta = 8.0/255$	95.14% 94.38% 93.86% 93.88%
Universal	Kenneth [32]	δ: perturbation budget	l_∞	CIFAR-10 ImageNet-10 ImageNet-1000 CelebA	$\delta = 8.0/255$ $\delta = 8.0/255$ $\delta = 8.0/255$ $\delta = 8.0/255$	88.28% 87.25% 87.12% 86.15%
	AT-UAP [33]	δ: perturbation budget	l_∞	CIFAR-10 ImageNet-10 ImageNet-1000 CelebA	$\delta = 8.0/255$ $\delta = 8.0/255$ $\delta = 8.0/255$ $\delta = 8.0/255$	93.19% 94.42% 93.50% 92.80%

$\mathbb{P}(\text{TG} > 0)$ can be presented as:

$$\begin{aligned}
&\mathbb{P}(\text{TG} > 0) \\
&= \mathbb{P}(\frac{\|d(x)\|^2}{n_c d(x)} - \frac{\|d(x+\delta)\|^2}{n_a d(x+\delta)} > 0) \\
&= \mathbb{P}(\frac{n_a d(x+\delta)}{\|d(x+\delta)\|}\|d(x)\| > \frac{n_c d(x)}{\|d(x)\|}\|d(x+\delta)\|) \\
&> \mathbb{P}(\|d(x)\| > \|d(x+\delta)\|)\mathbb{P}(\frac{n_c d(x)}{\|d(x)\|} < \frac{n_a d(x+\delta)}{\|d(x+\delta)\|}) \\
&= \mathbb{P}(\|d(x)\| > \|d(x+\delta)\|) \\
&= \gamma.
\end{aligned} \tag{16}$$

Thus, for any clean example x and its any potential adversarial example $x+\delta$, there always exists $\mathbb{P}(\text{TG} > 0) > \gamma$ when $max(\frac{\mathcal{D}(x)d(x)}{\|d(x)\|}) < min(\frac{\mathcal{D}(x+\delta)d(x+\delta)}{\|d(x+\delta)\|})$. □

To calculate the value of γ, we follow the distance estimation suggested in [69]. We estimate the distance to a decision boundary in a example of random directions in the input space of the target model, starting from a given input point. In each direction, we estimate the distance to a decision boundary by computing the predictions of the targeted model on perturbed inputs at points along the direction and increase the random directions by a magnitude factor (0.002) if the predictions dose not change in any of directions. We perform this search over a set of 1,000 random orthogonal directions. We conduct experiments on CIFAR-10 and ImageNet-1000 using ResNet-50 as the target model and use AutoAttack to generate adversarial examples. $\mathbb{P}(\|d(x)\| > \|d(x+\delta)\|)$ is 96% on CIFAR-10 and 95% on ImageNet-1000.

In addition, we we consider the case where $d(\|x\|) < d(\|x+\delta\|)$ (i.e., extreme case where $\gamma < 0$) and analyze its influence on TG. TG is determined by both the minimal distance to the decision boundary and the statistical properties of the generated noise distribution. If $\frac{\|d(x)\|^2}{n_c d(x)} - \frac{\|d(x+\delta)\|^2}{n_a d(x+\delta)} > 0$, TG is still larger than 0.

Failure of Gaussian Noise. Based on the above analysis, we can conclude that the added noise should make the offset to cross the decision boundary for adversarial examples. For Gaussian noise, the offset caused by it may align with any direction due to its randomness. If added noise has the same direction of adversarial noise, it is difficult to break adversarial noise by added noise under low false positives.

B. Additional Details of Attack Settings

This section provides a detailed description of the attack configurations for the static attacks mentioned in Sec. II-A. To evaluate the detection performance for the static attacks, we select 10 typical attacks, including VNIFGSM [40], DSA [8], PGD [7], BIM [70], CW [30], DeepFool [71], AutoAttack [25], SSAE [31], Kenneth [32] and AT-UAP [33]. The detailed parameter settings for these attacks are summarized in Table X. For iteration based attacks (e.g., PGD and BIM), we set step size as 0.03 and iterations as 40. For optimized attacks (e.g.,

CW), we set confidence as 1.0 and optimization steps as 100. Attack configurations are set following previous works [15], [8]. For VNIFGSM [40], ResNet20 and VGG19 are chosen as substitute models to leverage the transferability of adversarial examples. For DSA [8], ResNet20 and VGG19 are chosen as substitute models. The query times of the target model are limited to no more than 1,000 times when conducting attacks. For the perturbation budget of attacks, both l_2 and l_∞ norms are used as metrics to constrain the perturbation distance. For GAN-based attack SSAE and universal attacks such as Kenneth and AT-UAP, we select l_∞ as the constraint of perturbation. Following the settings in previous works [8], the perturbation budget δ is set to 1.75 for CIFAR-10 and 16.38 for ImageNet and CelebA under the l_2 norm, and 8/255 for all datasets under the l_∞ norm.

C. Designs of Adaptive Attack

Based on the design details of Falcon, we can construct loss function $\mathcal{L}_{det}$ from three aspects to evade the detection. Firstly, Falcon leverages the noise distribution that affects critical regions less but locates adversarial noise. Thus, we design loss function $\mathcal{L}_d$ to make generated noise distribution locate adversarial noise less:

$$\mathcal{L}_d = W(n, \delta), \tag{17}$$

where $W(\cdot)$ refers to the Wasserstein distance that evaluates the differences between two noise distributions.

Secondly, the adversary can try to make generated noise intensity closer to 0 to reduce the effect of constructive noise on adversarial noise. Thus we design loss function $\mathcal{L}_s$ as:

$$\mathcal{L}_s = \sigma, \tag{18}$$

where σ is the generated noise intensity.

Finally, the adversary can consider generated noise distribution and intensity together, just making sure that generated constructive noise cannot defeat adversarial noise. We design loss function $\mathcal{L}_t$ to reduce differences in confidence scores after adding constructive noise:

$$\mathcal{L}_t = CrossEntropy(C(x'), C(x' + \sigma * n)), \tag{19}$$

where $CrossEntropy(\cdot)$ represents cross-entropy loss [72], [52] between output confidence scores before and after noise is added.

We define the final $\mathcal{L}_{det}$ as:

$$\mathcal{L}_{det} = \mathcal{L}_d + \mathcal{L}_s + \mathcal{L}_t, \tag{20}$$

where $\mathcal{L}_d$, $\mathcal{L}_s$ and $\mathcal{L}_t$ the above three components.

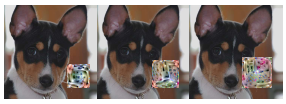
Then, we introduce more details of adaptive attacks based on strategy #2. To conduct the adaptive attack based on DSA, we design the loss function as:

$$\mathcal{L}_{DSA} = \mathcal{L}_T + \mathcal{L}_{det}, \tag{21}$$

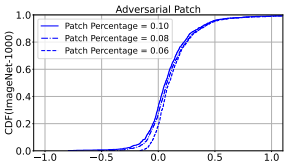
where the $\mathcal{L}_T$ is the same as the original DSA to fool the target model [7] and $\mathcal{L}_{det}$ will try to evade Falcon. Note that $\mathcal{L}_T$ includes no details of the target model.

To conduct the adaptive attack based on PGD, we design the loss function as:

$$\mathcal{L}_{PGD} = \mathcal{L}_{classifier} + \mathcal{L}_{det}, \tag{22}$$



(a) Visualization of successfully generated adversarial examples using the adversarial patch attack with varying patch sizes, corresponding to 0.06, 0.08, and 0.1 of the image area. These perturbations are obvious for human eyes.



(b) The CDF of Tolerance Gap (TG) for Constructive Noise (CN) under adversarial patch attacks. The values of TG are not always larger than 0 when facing adversarial patch attacks.

Fig. 9: Visualization of adversarial patches and the CDF of TG for Constructive Noise under adversarial patch attacks.

where the $\mathcal{L}_{classifier}$ is the same as the original PGD to fool the target model [7] and $\mathcal{L}_{det}$ will try to evade Falcon.

As for adaptive attack based on CW, we keep the same objective loss items as the settings of CW [30]. We design the loss function as:

$$\mathcal{L}_{CW} = ||x' - x||_2 + c \cdot f(x') + \mathcal{L}_{det}. \tag{23}$$

The first item $||x' - x||_2$ minimizes the distance between x' and x. The second item $c \cdot f(x')$ is designed to deceive the target model according to settings in CW [30]. Finally, $\mathcal{L}_{det}$ is to evade the Falcon.

D. Validation of Constructive Noise

The validity of constructive noise does not hold for certain adversarial examples in the physical domain. Physical-domain adversarial attacks [36], [73], [35] typically violate the bounded perturbation constraint and often employ the Expectation over Transformation (EoT) strategy to maintain effectiveness under real-world variations such as lighting changes, viewpoint shifts, and occlusion. Compared to adversarial attacks that follow the objective in Eq. (1), physical attacks tend to introduce obvious distortions. This inherently increases the distance of adversarial examples from the target model's decision boundaries, making them less sensitive to constructive noise. To further investigate this phenomenon, we select AdvPatch [36], a representative adversarial patch attack in the physical domain. Instead of adding imperceptible perturbations on pixels, AdvPatch replaces a localized image region with a generated adversarial patch. As shown in Fig. 9(a), these perturbations are visually obvious.

To simulate the strongest effect of AdvPatch, we apply adversarial patches in the digital domain, where the absence of real-world distortions allows the generated adversarial patch to maintain its maximum impact. We randomly select 3,000 images from the ImageNet validation set and apply adversarial patches with sizes set to 0.06, 0.08, and 0.1 of the image area based on the prior work [62]. As shown in Fig. 9(b), the values of TG are not always larger than zero, indicating that the validation of constructive noise does not always hold for adversarial examples in the physical domain.