# *Jujutsu*: A Two-stage Defense against Adversarial Patch Attacks on Deep Neural Networks

Zitao Chen
University of British Columbia
Vancouver, BC, Canada
zitaoc@ece.ubc.ca

Pritam Dash
University of British Columbia
Vancouver, BC, Canada
pdash@ece.ubc.ca

Karthik Pattabiraman
University of British Columbia
Vancouver, BC, Canada
karthikp@ece.ubc.ca

## ABSTRACT

Adversarial patch attacks create adversarial examples by injecting arbitrary distortions within a bounded region of the input to fool deep neural networks (DNNs). These attacks are *robust* (i.e., physically-realizable) and *universally* malicious, and hence represent a severe security threat to real-world DNN-based systems.

We propose *Jujutsu*, a two-stage technique to detect and mitigate robust and universal adversarial patch attacks. We first observe that adversarial patches are crafted as localized features that yield large influence on the prediction output, and continue to dominate the prediction on *any* input. *Jujutsu* leverages this observation for accurate attack detection with low false positives. Patch attacks corrupt only a localized region of the input, while the majority of the input remains unperturbed. Therefore, *Jujutsu* leverages generative adversarial networks (GAN) to perform localized attack recovery by synthesizing the semantic contents of the input that are corrupted by the attacks, and reconstructs a "clean" input for correct prediction.

We evaluate *Jujutsu* on four diverse datasets spanning 8 different DNN models, and find that it achieves superior performance and significantly outperforms four existing defenses. We further evaluate *Jujutsu* against physical-world attacks, as well as adaptive attacks.

## CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → **Machine learning**;

## KEYWORDS

Adversarial robustness, Security, Deep learning, Neural networks

**ACM Reference Format:**
Zitao Chen, Pritam Dash, and Karthik Pattabiraman. 2023. *Jujutsu*: A Two-stage Defense against Adversarial Patch Attacks on Deep Neural Networks. In *The 18th ACM ASIA Conference on Computer and Communications Security (ASIA CCS 2023), 10–14 July, 2023, Melbourne, Australia* . ACM, New York, NY, USA, 15 pages. https://doi.org/xxxxxx


*Asia CCS'23, 10–14 July, 2023, Melbourne, Australia*

ACM ISBN xxx…$15.00
https://doi.org/xxxxxx

## 1 INTRODUCTION

DNNs are widely used in various application domains, such as autonomous driving [8, 33], facial recognition [31, 39] and healthcare [15, 35]. Unfortunately, DNNs are known to be vulnerable to *adversarial attacks*, which maliciously perturb the inputs to cause the DNNs to misbehave [41]. Different variants of adversarial attacks have been proposed in the literature, including *universal* adversarial attacks that cause misclassification on arbitrary inputs [20, 25, 38]; and *robust* adversarial attacks that can remain adversarial even when translated to the physical world [7, 9, 16].

A sub-category of adversarial attacks are *adversarial patch attacks* that perform arbitrary changes to the input images within a region of bounded size, in order to cause targeted image misclassification in DNNs [9]. These attacks create *robust and universal* adversarial examples - AEs (henceforth referred to as *patch attacks*). They are an important threat as they entail dire consequences for real-world safety-critical systems such as autonomous vehicles. Further, their universal nature drastically lowers the adversary's barrier to launch the attack: an universal adversarial patch can be widely distributed to fool arbitrary DNN systems with little effort.

Patch attacks have been the subject of considerable study, and many techniques have been proposed to detect [12, 21] and mitigate [19, 30, 34, 42, 43] them. For example, SentiNet [12] detects patch attacks based on model interpretability and statistical analysis. LGS [30] mitigates patch attacks by smoothing out the important features in an image based on pre-defined thresholds. Adversarial training has also been adopted for countering patch attacks [34, 42]. Unfortunately, these techniques suffer from one or more of the following limitations, (1) high false-positives rates (FPR) - unable to correctly distinguish between adversarial and benign image features [12, 21, 30, 43]. (2) poor detection performance - unable to reliably locate the region of adversarial patch [12]. (3) low mitigation performance (i.e., robust accuracy on adversarial examples) - unable to allow the DNNs to make correct inference on the adversarial examples as many important features are corrupted [30, 34, 42, 43].

To address the above issues, we propose *Jujutsu*[1] for both detecting and mitigating adversarial patch attacks. We first outline the challenges in attack detection and attack mitigation, then explain the main ideas to address them.

**Attack detection**. The key challenge in accurate attack detection with a low FPR is to identify the unique symptom that characterizes adversarial examples *and* exhibits differences with benign examples. Our solution is based on two insights: (1) the adversarial patch is crafted to constitute localized features in the input, which

[1]Jujutsu is a martial art whose philosophy is to manipulate the opponent's force against him- or herself rather than confronting them with one's own force. Our technique has a similar philosophy, and hence the name.

exerts a large influence on the output in order to manipulate the output; (2) it exhibits the dominant influence on *any* input (input-agnostic). *Jujutsu* is built on both insights to expose this behavior of the patch attacks and distinguish AEs from benign examples.

We leverage the first insight to identify the potential location of the adversarial patch in AEs. Specifically, we propose to locate the adversarial patch region by locating the *salient features* in the saliency map, which are the features that have a large influence on the output (similar to how adversarial patch would behave). However, benign features in AEs may also have large influence on the output, and hence they may be *conflated* with the adversarial patch (undesirable). To resolve this, we propose a robust method to *pre-process* the saliency map, which can highlight the regions associated with the adversarial patch so that *Jujutsu* can correctly identify the region associated with the adversarial patch (instead of benign features). This enables *Jujutsu* to reliably locate the adversarial patch from AEs (Section 3.2).

We build on the second insight to distinguish the adversarial patch from a benign patch, which is important because the incoming input can be either adversarial or benign, and thus the extracted patch can also be adversarial (i.e., from adversarial examples) or benign (i.e., from benign examples). Specifically, we propose a *guided feature transplantation* method to strategically transfer the extracted patch from the original input to a dedicated region (the region where the *least*-salient features reside) in a new input, and determine whether the patch continues to cause misclassification of the new input. If so, it is likely to be an adversarial patch.

**Attack mitigation.** A natural solution for attack mitigation is to simply mask the entire patch region, and let the DNNs perform inference on the remaining features. However, with this approach, the important features in the original images may be corrupted (overridden) by the adversarial patch, and hence it is difficult for the DNNs to make correct predictions on the remaining uncorrupted features (e.g., see Fig. 2).

We make the observation that, *patch attacks only perturb a localized region, and hence the majority of image pixels are uncorrupted* (Section 2.2). These pixels can be used to reconstruct the semantic contents in the pixels corrupted by the attacks. Therefore, we use GANs to perform localized attack mitigation, by reconstructing the uncorrupted contents from the corrupted region, which creates the "clean" images from AEs for correct prediction. In addition to improving robust accuracy, our mitigation technique can also be leveraged to further reduce FPs on benign examples (Section 3.3.3).

Finally, different applications may prioritize different defense goals, and hence a configurable defense technique is important. For example, in some systems, the detection performance should be prioritized as an undetected intrusion might cause severe property damage, and hence higher FPR may be acceptable in those settings. For this purpose, we propose a parametric defense strategy that allows for balancing between detection performance and FPR. We find that the targeted misclassification caused by the adversarial patch often becomes ineffective even *without* completely performing attack recovery on the entire patch, based on which we introduce a parametric attack mitigation strategy (Section 3.3.2).

**Contributions:** The contributions of this work are as follows.

- A novel patch attack detection method that can reliably locate the regions of adversarial patches in adversarial examples and effectively distinguish between adversarial and benign examples.
- A novel attack mitigation technique that leverages the generative power of GANs to allow the DNNs to make correct predictions on AEs (high robust accuracy), and distinguish false detection on benign examples (low FPRs). It further provides configurability to balance between the detection of AEs and FPRs.
- A comprehensive evaluation of *Jujutsu* on four datasets (ImageNet, ImageNette, CelebA and Place365) spanning eight different DNNs. We find that *Jujutsu* achieves superior detection and mitigation performance with low FPRs, and outperforms four existing defenses: LGS [30], SentiNet [12], adversarial training [34, 42] and PatchGuard [43]. *Jujutsu* can further defend against both physical-world attacks and adaptive attacks.

# 2 BACKGROUND

## 2.1 Attack Formulation

We express a DNN as $F_\theta : X \rightarrow Y$, where $X \in \mathbb{R}^n$ and $Y \in \mathbb{R}^m$ denotes the input and output space, and $F$ is parameterized by weights $\theta$ (hereafter omitted for simplicity). $\bar{y_i}$ is the ground truth label and $\hat{y} = \text{argmax} F_\theta(x)$ the prediction label. We call an input $x' \in X$ an adversarial example if

$$x' \in X \wedge \text{argmax}\, F(x') = y^{adv} \wedge \text{argmax}\, F(x) = \bar{y}, \tag{1}$$

where $y^{adv}$ is the target class, $x'$ is the adversarial example generated from the original input $x$. Patch attack replaces a part of the image with an image patch [9], denoted as $\delta \in \mathbb{R}^n$:

$$x' = (1 - m) \odot x + m \odot \delta, \tag{2}$$

where $m \in \{0, 1\}^n$ is a mask used to put the adversarial patch ($\forall m_i \in m, m_i = 1$ is where the patch will be placed), $\odot$ is element-wise multiplication, $\delta$ is the adversarial patch.

To make patch $\delta$ be universal (i.e., input-agnostic), the patch is trained over a variety of images. For each input $x \in X$, patch $\delta$ can be applied in any random location $L$.

To make patch $\delta$ robust (i.e., physically realizable), [9] propose to use a variant of Expectation over Transformation (EOT) framework [7]. EOT is used for a distribution of environmental transforms $T$ that transform $x$ to different physical environments (e.g., translation, rotation, lightness changes), under which the adversarial examples aim to remain robust. Based on the above, the objective function of the patch attack can be formulated as:

$$\delta = \underset{\delta}{\text{argmax}}\, \mathbb{E}_{x \sim X, t \sim T, l \sim L}[\log \Pr(y = y^{adv} | x')], \tag{3}$$

where $T$ is a distribution of transformations over the patch, and $L$ is a distribution over locations in the images. This allows the patch to work regardless of the background.

## 2.2 Threat Model

We assume a white-box attacker with full knowledge of the victim DNN like its structure and parameters. We assume however that the attacker has no knowledge of the exact inputs to the DNN, but instead has access to a surrogate dataset, which follows the same distribution as the legitimate inputs. This is similar to the assumption in other universal attack papers, which have shown that the

knowledge of the input distribution often suffices for the attacker to generate universal adversarial perturbations [20, 25, 38]. As in other defense studies [23, 28, 30, 43], we consider an adversary who replaces a contiguous region of an image with a single adversarial patch - thus, the adversarial patch is localized to a single region of the image and the adversary's goal is to universally cause targeted misclassification on any input[2]. Further, the defender has access to a hold-out set hidden from the attacker, which can be created by randomly sampling a series of images from the data distribution.

# 3 METHODOLOGY

## 3.1 Design Overview

Fig. 1 illustrates attack detection and Fig. 2 attack mitigation.

**Detecting the adversarial patch.** We first identify *suspicious features* that potentially contain the adversarial patch. We observe that the universal and localized nature of adversarial patch induces the perturbations to have a *disproportionately large* influence on the output in order to dominate the prediction on *any* input, which can be exposed by investigating the *salient features* from the saliency map (Step 1 in Fig. 1). These salient features are considered suspicious as they have a large influence on the output, similar to the adversarial patch's behavior. However, salient features may also point to the natural features in the images and hence the adversarial patch region may be *undetected* (see the left of Fig. 3). To avoid this, we *pre-process* the saliency map to highlight the regions that are associated with the adversarial patch, hence we can reliably locate the adversarial patch region (see the right of Fig. 3).

On the other hand, since the input can either be adversarial or benign, the suspicious features can also be adversarial (from adversarial example) or benign (from benign example). Our idea to distinguish them is based on the observation that the adversarial patch, when transplanted to other images, *will continue* to trigger misclassification, which is *different* from how benign examples would behave[3]. Therefore, Step 2 extracts the suspicious features from the original input, and transplant them *to the least-salient feature region* (the region where the *least*-salient features reside) of hold-out input. Step 3 compares the prediction on the original input and the hold-out input implanted with suspicious features. If both predictions lead to the *same* prediction label, the suspicious features are marked as adversarial.

**Mitigating the adversarial patch.** The goal of mitigation is to remove the attacks' effects, and allow the DNN to predict the correct label from the adversarial examples. A straightforward solution is to mask out the suspicious features so that the adversarial patch will not contribute to the final prediction. Unfortunately, masking alone does not work in many situations. For instance, in Fig. 2, masking the suspected feature mitigates the adversarial attack, as the DNN no longer predicts the adversarial example as a "toaster" (the target label determined by the attacker), thus defying the attack. However, the DNN predicts the image with the mask as a "monitor", which is clearly not the correct label for the image. This shows that merely masking the suspected feature is not sufficient, as it also removes

[2]Section 5.1 discusses attack variants (e.g., multi-patch attacks) that are outside our threat model.

[3]There are two potential scenarios that would lead to false positive on benign samples, and they are discussed in Section 3.2.2 and Section 3.3.3, respectively.

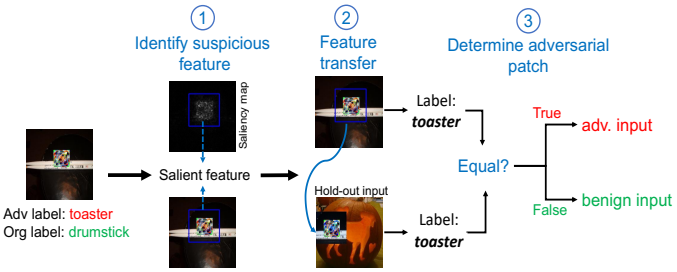


**Figure 1: Attack *detection*. Step 1: Identify suspicious features that may contain adversarial patch. Step 2: Transfer the suspicious features to a hold-out input. Step 3: Determine the maliciousness of the suspicious features based on prediction consistency.**

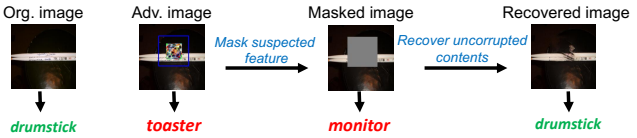


**Figure 2: Attack *mitigation*: (randomly) mask the suspicious features (blue box), and use GAN to recover the uncorrupted contents in the mask.**

the semantic contents in the image, and hence the DNN is *unable* to predict the correct label from the masked images.

Our goal is to remove the effects of the attacks while preserving the semantic contents. We observe that the adversarial patch is confined within a *localized* region, and the majority of the pixels are uncorrupted, which can be used as a rich context to reconstruct the semantic contents corrupted by the attacks. Specifically, we use generative adversarial networks (GAN) [26, 47, 49] to reconstruct the semantic contents from the pixels that are masked, resulting in a "clean" image that is free from corruptions for the DNN to make correct prediction. As shown in Fig. 2, after recovering the contents from the masked regions, the DNN is able to correctly predict the adversarial image as a "drumstick". We use the prediction label of the recovered image as the final output.

Section 3.3.3 discusses how our mitigation technique can also be leveraged to reduce false detection on benign examples..

## 3.2 Detecting the Adversarial Patch

*3.2.1 Robust Suspicious Feature Detection.* We first compute a saliency map that models the contributions of different pixels on the final decision. One common approach is to compute the gradients of the output with respect to the input pixels. Mathematically, the saliency map $M_j(x)$ can be expressed as: $M_j(x) = \partial F(x)_j / \partial x$, where $j$ indicates the class label. $M_j(x)$ represents how much difference a tiny change in $x$ would contribute to the output $F(x)_j$. Thus $M_j(x)$ can highlight the key regions in predicting $F(x)_j$.

We use SmoothGrad [37], which can visually sharpen the gradient-based saliency map and smooth out the noisy gradients (that arise due to the local variations in partial derivatives [37]). Other methods such as Grad-cam [36], Integrated Gradient [40] may also be used. Given the noisy (fluctuating) gradients, SmoothGrad computes a local average of the gradient values, by taking random examples in the neighborhood of an input $x$, and averaging the resulting saliency maps. This operation can be expressed mathematically as:

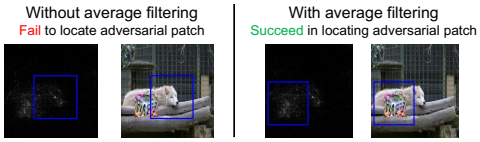


**Figure 3: Extracting suspicious features from saliency map with and without average filtering. The proposed use of average filtering is able to locate the adversarial patch correctly.**

$$\hat{M}_j(x) = \frac{1}{n}\sum_{1}^{n} M_j(x + \mathcal{N}(0, \sigma^2)), \tag{4}$$

where $n$ is the number of examples, and $\mathcal{N}(0, \sigma^2)$ represents the Gaussian noise with standard deviation $\sigma$.

To extract the suspicious features from the saliency map, we first choose the point that has the maximum value in the saliency map, and draw a detection box around it. However, this approach is susceptible to noise and a single large-value pixel outside the adversarial patch could result in a mis-identification. Thus, the detection box would fail to locate the adversarial patch. The reason is that there are many benign features that are uncorrupted in the adversarial examples, which may also have large influence on the outputs, and might be conflated with those of the adversarial patch. In the left-hand side of Fig. 3, the region associated with the benign feature is identified as the salient feature.

Therefore, we perform an *average filtering* [4] to pre-process the saliency map in order to highlight the regions of the adversarial patch, and downplay those of the benign feature. It takes the average of all the pixels under the kernel area (in the saliency map) and uses the average value to replace the central element. Our intuition is that the region of adversarial patch has a higher density than that of the benign feature (because it needs to have a disproportionately large influence on the output to dominate the prediction on any input), and hence by performing average filtering, the adversarial patch will remain salient while the benign feature will become less salient, thereby allowing us to accurately locate the adversarial patch. A visual comparison of the two approaches in identifying the suspicious features is shown in Fig. 3. Our ablation study (Section A.2) also validates that the proposed pre-processing method enables *Jujutsu* to detect much more AEs than without it.

*3.2.2 Guided Feature Transplantation.* As mentioned, identifying the suspicious features by itself is not enough to determine whether the features are coming from adversarial or benign examples. Hence, we transfer the suspicious features from the original input to the hold-out input in order to determine whether they are truly malicious. One way is to *randomly* transplant the suspicious features to the new hold-out input and compare the prediction. However, this may occlude the foreground object in the hold-out input. Should this happen, the prediction labels on the original and hold-out inputs may become the same, leading to a mis-detection of benign input, i.e., False Positive (FP). Fig. 4 shows an example where randomly transplanting the benign features to a hold-out input leads to the same prediction label 'Sloth bear", thus resulting in an FP.

To avoid FP, we propose a guided feature transplantation method to transplant the suspicious features to the *least-salient* regions of the hold-out input, in order to minimize the chances that the suspicious features override the hold-out input's foreground object. The

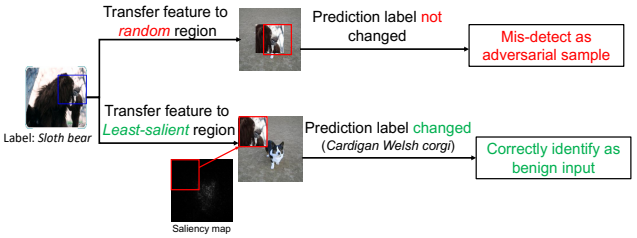


**Figure 4: Different strategies to transfer features. In the top, features are transplanted to a *random* location of the hold-out input which leads to a FP, while those in the bottom are to the *least-salient* region of the hold-out input (our approach), thus avoiding a FP.**

least-salient regions are those regions that have *low* influence on the output according to the saliency map. Only those suspicious features containing the adversarial patch at the least-salient regions will also lead to the same prediction label (due to the patch's universal nature). Fig. 4 shows how this method works and our ablation study (Section A.2) shows that it is able to yield much lower FPRs (compared with random transplantation).

*3.2.3 Prediction Comparison for Attack Detection.* The final step is to compare the prediction labels on the original and hold-out images implanted with suspicious features. The original image is deemed to be adversarial if and only if both images yield the same prediction label. This is because only the suspicious features that contain the adversarial patch will cause (the same) misclassification on the hold-out input. We are also able to identify suspicious features that come from the benign features, by checking whether the prediction labels on the original and hold-out images implanted with suspicious features are *different*. We consider an image to be benign if the prediction labels on the original and hold-out images are different.

## 3.3 Mitigating the Adversarial Patch

*3.3.1 GAN-based Localized Attack Mitigation.* A natural solution to mitigate the attacks is to mask the adversarial patch, and let the DNNs perform inference on the remaining features. However, some of the important features in the original images might have been corrupted (overridden) by the adversarial patch, and hence performing masking alone will result in the loss of semantic contents that are crucial for the DNNs to classify the images correctly.

On the other hand, we also note that patch attacks only perturb a small localized region and a large portion of image pixels are intact, which can serve as the rich context to synthesize the contents that are corrupted (replaced) by the attacks. Based on this observation, we propose to use generative adversarial networks (GAN) [26, 32, 47, 49] to perform localized attack mitigation by reconstructing the contents replaced by the adversarial patch, and to increase the probability that the DNN predicts the correct label. While there are many GAN techniques proposed in the literature. We use PICnet [49], a recent technique that can generate multiple and diverse plausible contents from the masked regions. Although we choose PICnet in this work, other techniques [26, 32, 47] may also be considered for the same.

Formally, let $x$ be the original image, $x_m$ the image with a region of pixels being masked, and $x_c$ the original pixels that are masked. PICnet synthesizes diverse contents from the mask by sampling

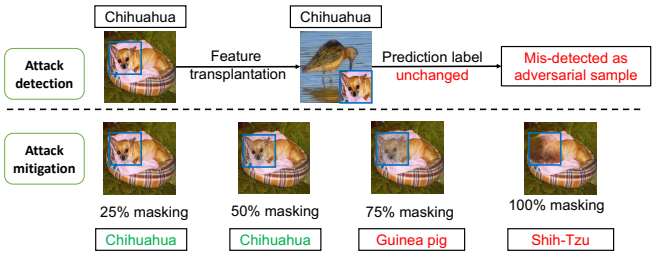


**Figure 5: Upper row: An illustration of a false detection on benign sample. Lower row: How *Jujutsu* may reduce FP in its attack mitigation phase (under different masking percentages).**

a conditional distribution $p(x_c|x_m)$. In the training phase, PICnet uses a *reconstructive* pipeline, in which the missing regions $x_c$ are encoded into the latent space representation in a continuous distribution that can be exampled to rebuild the diverse and plausible $x_c$. The reconstructive pipeline leverages $x_c$ and $x_m$ to reconstruct $x$ in a supervised manner ($x_c$ is the ground truth). In the testing phase, PICnet uses a *generative* pipeline to infer the conditional distribution of $p(x_c|x_m)$, which is exampled to generate $x_c$. The parameters in the reconstructive pipeline are shared with the generative pipeline so that it can reconstruct $x$ from $x_m$ during testing. The resulting recovered images are meant to be free from adversarial perturbations, and thus we use the labels on the recovered images as the final output for mitigation.

*3.3.2 Parametric Attack Mitigation.* We now introduce our parametric mitigation strategy that allows balancing between the detection performance and FPRs. The motivation is that it is often *unnecessary* to mask all the pixels in order to make the target misclassification ineffective, e.g., we find masking 75% of the suspected features is able to change the targeted miclassification in over 99.9% of the adversarial examples.

By partially masking suspected features, *Jujutsu* allows the defender to reduce the FPR - we explain the reason below. We can determine a mis-detection on the benign input (i.e., FP) if the predictions on both the original and recovered images result in the same label (to be discussed in Section 3.3.3). The fewer pixels that are masked, the better is the quality of the resulting recovered image, because more semantic information is preserved in the image.

Fig. 5 shows an example of the recovered images under different masking percentages. The original Chihuahua image is misdetected as adversarial, which can be eliminated if both the original and recovered image have the same label. In this example, if 25% or 50% of the pixels are masked, *Jujutsu* is able to rectify the misdetection. However, if 75% or 100% of the suspicious features are masked, the DNN is unable to generate the correct prediction on the recovered image, thus resulting in an FP. This explains why masking the entire set of suspicious features could be undesirable.

*3.3.3 Reducing FPRs.* Section 3.2.2 explains the first scenario where FPs might occur. We now explain the second scenario where FPs may still arise, and how *Jujutsu* can prevent them.

We use Fig. 5 to illustrate. In the attack detection phase, *Jujutsu* transplants the Chihuahua object (as suspicious features) to the hold-out input. The resulting hold-out input, originally containing a single Dowitcher, is now classified as a Chihuahua by the DNN, which is the same as the original input. This thus results in a FP.

To reduce the above FP, we propose to signal a FP when the prediction label on the original input (that *Jujutsu* originally detected as adversarial) and the recovered input are *identical*, during the mitigation phase. The intuition is that a benign input does not contain an adversarial patch, and hence predictions based on the original and the recovered images should both result in the same prediction label.

The above process to reduce the FPR might also flag some adversarial examples as benign ones, thereby resulting in missed detection. For example, if the masking percentage is low, the adversarial patch will continue to cause misclassification on the recovered images, based on which *Jujutsu* would incorrectly flag the adversarial patch as a benign image patch. We study how different masking percentages would affect the detection performance (Table 2).

*3.3.4 Algorithm.* Algorithm 1 shows the overall algorithm. The inputs are the images to be classified and parameters for *Jujutsu*. For each $x_i$, the output includes the prediction label $y_{x_i}$ and a flag $isAdv_{x_i}$ on whether $x_i$ is adversarial. Lines 4-8 extract the salient features from $x_i$. Lines 10-15 identify the least-salient regions in the hold-out input $x^*$, which will be replaced by the salient features from $x_i$. Lines 16-25 perform feature transfer and compare the prediction labels on the original and implanted images. Lines 29-39 perform attack mitigation by accepting the label from the recovered image, and checking for mis-detection (thereby reducing FPs).

# 4 EVALUATION

We first describe the experimental setup of *Jujutsu*, and then answer the following research questions (RQs) in subsequent sections.
**RQ1:** What's the detection performance of *Jujutsu*?
**RQ2:** What's the mitigation performance of *Jujutsu*?
**RQ3:** How does *Jujutsu* compare with existing techniques?
**RQ4:** Can *Jujutsu* defend against physical-world attacks?
**RQ5:** Can *Jujutsu* defend against attacks targeting different classes?
**RQ6:** Is *Jujutsu* able to thwart the adaptive attackers?

## 4.1 Experimental Setup

*4.1.1 Datasets and Architectures.* We evaluate *Jujutsu* on ImageNet [13], ImageNette [5], CelebA [27] and Place365 [50]. ImageNet is a 1000-class dataset and we use a ResNet-50. ImageNette is a 10-class subset of ImageNet and we use a ResNet-18. CelebA is a facial dataset with diverse faces. We created a 307-classes subset from the original set and train a ResNet-18 model following [1] to perform identity classification. Place365 is a 365-class dataset containing common natural sceneries (e.g., patio, restaurant) and we use a ResNet-50. We evaluate *Jujutsu* on 6 more DNN models in Section 5.2.

*4.1.2 Attack Setup.* The attacker's goal is to synthesize adversarial patches that achieve high attack success rate. As in prior work [23, 28, 30, 43], we consider a square digital patch (we use a circle patch in evaluating physical world attack in Section 4.5), and discussion on other patch shapes is in Section 5.1. For each dataset, we generate patches of different sizes, occupying 5%, 6% and 7% of the pixels. The patch is overlaied to a random position in the image. We use $x$% patch to refer to a patch that occupies $x$% of the pixels of the image. We do not consider patches of smaller size because we find that they are unable to universally cause misclassification, e.g., use of a 4% patch on CelebA degraded the attack success rate by more than

**Algorithm 1** Detect and mitigate patch attacks

**Input:** $X_{test}$: Test images; $X_{hold}$: Hold-out images; $F$: DNN model;
$l$: Length of detection box; $p$: Percentage of pixels to mask
**Output:** $Y_{test}$: Prediction on $X_{test}$; $isAdv_{X_{test}}$: whether $X_{test}$ is adversarial

1: **function** DETECTION($X_{test}$, $X_{hold}$, $F$, $l$)
2: **for each** $(x_i, y_{x_i}, isAdv_{x_i}) \in (X_{test}, Y_{test}, isAdv_{X_{test}})$ **do**
3: $y_j$ = argmax$F(x_i)$
4: // **Extract the salient features** $B_{x_i}$ **from** $x_i$
5: $M_j(x_i)$ = SmoothGrad($x_i$, $y_j$) // Saliency map for ($x_i$, $y_j$)
6: $M_j(x_i)$ = AverageFilter($M_j(x_i)$) // Average filtering over saliency map
7: $(x_{max}, y_{max})$ = MaxLoc($M_j(x_i)$) // point with maximal value
8: Draw a box $B_{x_i}$ around $(x_{max}, y_{max})$ with length $l$ // suspicious features
9: // **Identify the least-salient features** $B_{x^*}$ **from** $x^*$
10: Randomly select $x^* \in X_{hold}$
11: $y_k^*$ = argmax$F(x^*)$
12: $M_k(x^*)$ = SmoothGrad($x^*$, $y_k^*$) // Saliency map for ($x^*$, $y_k^*$)
13: $M_k(x^*)$ = AverageFilter($M_k(x^*)$) // Average filtering over saliency map
14: $(x^*_{min}, y^*_{min})$ = MinLoc($M_j(x_i)$) // point with minimal value
15: Draw a box $B_{x^*}$ around $(x^*_{min}, y^*_{min})$ with length $l$
16: // **Feature transfer and prediction comparison**
17: $x^{**} = x^*$.replace($B_{x^*}$, $B_{x_i}$)
18: $y_k^{**}$ = argmax$F(x^{**})$
19: **if** $y_k^{**} == y_j$ **then**
20: $y_{x_i}, isAdv_{x_i}$ = **MITIGATION**($x_i$, $B_{x_i}$, $F$, $p$) // $x_i$ **is adversarial**
21: **else**
22: $y_{x_i} = y_j$ // $y_j$ is the prediction label from line 3
23: $isAdv_{x_i}$ = False // $x_i$ **is benign**
24: **end if**
25: **end for**
26: **return** $Y_{test}, isAdv_{X_{test}}$
27: **end function**
28:
29: **function** MITIGATION($x$, $B_x$, $F$, $p$)
30: $y_{org}$ = argmax$F(x)$
31: $x_{mask}$ = Randomly mask $p$% of pixels within $B_x$ in $x$
32: $x_{recovered}$ = PICNet($x_{mask}$) // GAN-based recovery
33: $y_{new}$ = argmax$F(x_{recovered})$
34: **if** $y_{org} != y_{new}$ **then**
35: **return** $y_{new}$, True // **Attack mitigation**
36: **else**
37: **return** $y_{org}$, False // **Reduce false positive**
38: **end if**
39: **end function**

45%. We consider 7% as the largest patch size because it is already able to achieve very high attack success rate (average over 99%), and we evaluate *Jujutsu* on larger patch sizes (8%∼10%) in Section 5.3 for completeness. For each patch, we train it for 30 epochs on a training set with 2000 images. We evaluate the attack success rate on a separate test set, and choose the one with the highest success rate. For the attack evaluation on ImageNette and CelebA, we use the entire test set in each dataset; for ImageNet and Place365, we use 10000 images from the validation set for each. Examples of adversarial examples can be found in Fig. 9 (Appendix).

*4.1.3 Defense Setup.* There are three parameters in our defense setup: (1) kernel size for pre-processing (average filtering) the saliency map; (2) size of the detection bounding box; and (3) number of hold-out images used for feature transplantation.

We vary *each* parameter under different values and empirically select the one that strikes a good balance between detection performance and FPRs (e.g., a larger detection bounding box may achieve higher detection performance but with higher FPRs) - a detailed evaluation for each parameter under different values is in Appendix A.1, based on which we choose a kernel size of 51, a bounding box occupying ∼20% of the image pixels and 2 random hold-out images (out of 1000) for feature transplantation (on all dataset)[4].

[4]Our code is publicly available at https://github.com/DependableSystemsLab/Jujutsu.

**Table 1: Detection performance in terms of detection success recall on adversarial examples (AEs) and detection FPR.**

| Dataset | Patch Size | Clean Acy. | Attack Success Rate | Detection Success Recall | Detection FPR[1] |
|---|---|---|---|---|---|
| ImageNet | 7% | 74.17% | 99.81% | 99.74% | 4.24% |
| | 6% | | 98.34% | 97.70% | 4.01% |
| | 5% | | 94.06% | 93.37% | 3.81% |
| ImageNette | 7% | 98.32% | 100.00% | 100.00% | 8.11% |
| | 6% | | 99.00% | 99.94% | 9.18% |
| | 5% | | 98.36% | 99.88% | 8.98% |
| CelebA | 7% | 83.13% | 99.40% | 99.10% | 0.00% |
| | 6% | | 96.62% | 95.16% | 0.00% |
| | 5% | | 83.22% | 69.09% | 0.00% |
| Place365 | 7% | 54.45% | 96.77% | 98.93% | 0.57% |
| | 6% | | 96.90% | 99.29% | 0.46% |
| | 5% | | 97.11% | 98.97% | 0.53% |
| **Average** | N/A | N/A | 96.63% | 95.93% | 3.33% |

[1] The FPR can be further *reduced* as explained in Section 3.3.3 - see Table 2.

## 4.2 RQ1 - Detecting Adversarial Patch Attacks

**Metrics.** We use *detection success recall* to denote the fraction of adversarial examples detected by *Jujutsu*, and *detection FPR* for the fraction of false positives on benign inputs. Benign inputs are the same as the adversarial inputs, except that they do not have the adversarial patch. We consider an image as adversarial if and only if the predicted label for it is identical to that of *both the hold-out images* implanted with the suspicious features.

Table 1 shows *Jujutsu*'s detection performance on all 4 datasets.

*Detection success recall. Jujutsu* is able to consistently detect adversarial examples, with a detection success recall rate of over 93% across patch sizes (in most cases). The detection success recall increases with the size of the patch as a larger patch has a higher attack success rate. On average, *Jujutsu* can detect around 96% of the adversarial examples on all the datasets.

*Detection FPR. Jujutsu* yields an average FPR of 3.3% on the 4 datasets. We find that the FPRs on the two object-recognition datasets (ImageNet and ImageNette) are higher than that on the facial and scenery datasets. This is because the salient features in object-recognition datasets might contain the entire object (e.g., a small bird), which can cause the model to continue to assign the same label to the transplanted image. However, for the facial and scenery datasets, the salient features only contain a fraction of the image pixels (e.g., a partial face), and are hence unlikely to result in the same label on the transplanted image.

Despite the higher FPRs on ImageNet and ImageNette, *Jujutsu*'s mitigation mechanism is able to further reduce FPRs as explained in Section 3.3.3. We evaluate the FPR reduction in the next section.

## 4.3 RQ2 - Mitigating Adversarial Patch Attacks

**Metrics.** We use three metrics for evaluation in this section.

(1) *Robust Accuracy* is the prediction accuracy on all the AEs.
(2) *Mitigation FPR* is the (reduced) FPR from the two-staged combination of detection and mitigation (explained in Section 3.3.3).
(3) *Mitigation success recall* is the detection recall from the combination of detection and mitigation (Section 3.3.3) - we distinguish this from the *detection success recall*, which is the detection recall from the detection technique alone. *Mitigation success recall* gives the final amount of adversarial examples detected.

**Result.** Table 2 shows *Jujutsu*'s mitigation performance on all datasets. The results are averaged across patches of different

**Table 2: Mitigation performance for: 1) GAN-based recovery and 2) masking-alone recovery. Better results are marked in bold.**

| Metric (%) | Approach | ImageNet | | | | ImageNette | | | | CelebA | | | | Place365 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Masking percentage | | | | Masking percentage | | | | Masking percentage | | | | Masking percentage | | | |
| | | 25% | 50% | 75% | 100% | 25% | 50% | 75% | 100% | 25% | 50% | 75% | 100% | 25% | 50% | 75% | 100% |
| **Robust Accuracy** | GAN-based | 41.98 | **69.70** | **73.52** | **77.47** | 21.72 | 85.17 | **94.73** | **95.51** | 34.00 | **51.40** | **51.81** | **64.56** | 16.58 | 74.03 | **75.04** | **81.39** |
| | Masking-alone | **58.15** | 69.32 | 70.48 | 75.24 | **34.60** | **90.98** | 94.53 | 94.19 | **41.91** | 46.00 | 45.27 | 48.12 | **63.39** | **74.47** | 74.53 | 75.12 |
| **Mitigation FPR** | GAN-based | **0.34** | **0.95** | **1.67** | **1.74** | **0.14** | **0.61** | **1.06** | **0.91** | **0.00** | **0.00** | **0.00** | **0.00** | **0.00** | **0.10** | **0.14** | **0.20** |
| | Masking-alone | 1.03 | 1.69 | 2.00 | 1.85 | 0.76 | 1.26 | 1.55 | 1.74 | **0.00** | **0.00** | **0.00** | **0.00** | 0.08 | 0.17 | 0.22 | 0.25 |
| **Mitigation Success Recall** | GAN-based | 53.71 | 93.61 | 96.90 | 96.89 | 21.78 | 88.17 | 99.38 | 99.70 | 57.60 | 87.33 | 87.75 | **87.79** | 18.67 | 98.02 | **99.06** | **99.06** |
| | Masking-alone | **81.56** | **96.83** | **96.93** | **96.91** | **34.78** | **95.75** | **99.66** | **99.72** | **84.74** | **87.68** | **87.79** | **87.79** | **81.86** | **99.05** | **99.06** | **99.06** |

sizes (5% to 7%). The detection performance is higher on larger patches as these patches have higher attack success rates (difference between the largest and smallest patch is about 9%). The mitigation performance is consistent across different patch sizes (differences less than 2%). We consider two mitigation strategies, (1) masking-alone and (2) GAN-based recovery (which first performs masking and then use GAN to recover the contents from the mask) (*Jujutsu*).

*Robust accuracy*: Masking with GAN-based restoration is able to yield higher robust accuracy than masking alone. This is because GAN-based restoration synthesizes the missing semantic contents in the mask for the network to make a correct prediction. The only exception is when 25% or 50% of the pixels are masked, where masking alone has higher robust accuracy than GAN-based recovery. This is because the GAN relies on the regions outside the mask as the context to synthesize the contents. When the masking percentage is small, a large portion of the adversarial pixels remain intact, and thus the GAN cannot reconstruct the contents correctly. In this case, it is better to mask the perturbations to shield their contributions to the prediction, rather than GAN-based recovery as done by *Jujutsu*.

We also notice that the robust accuracy by *Jujutsu* on CelebA is lower than that on the other datasets, which is because the GAN needs to synthesize the correct facial features belonging to a particular celebrity's face to enable correct identity prediction. This is a much more challenging task for the GAN than for the other three datasets, and hence *Jujutsu* yields a lower robust accuracy. *Jujutsu* achieves the highest robust accuracy on ImageNette, as it is a 10-class dataset, and performing correct image classification on this dataset is easier than on the other complicated datasets such as the 1000-class ImageNet.

*Mitigation FPR*: GAN-based recovery achieves low FPR, because the restored inputs are more similar to the original benign inputs than the masked inputs (in the latter case many features are simply masked). Therefore predictions on the original and restored inputs are more likely to be the same, which is not the case for inputs that are merely masked. We also see that *Jujutsu* consistently achieves very low FPRs on all the datasets.

*Mitigation success recall*: While masking alone is able to achieve higher detection recall compared to GAN-based recovery when the masking percentage is small, the difference becomes negligible when the masking percentage increases. This is because when the masking percentage is low, the masked images are more likely to have a label *different* from that of the original image; while the restored images are more likely to have the *same* label as the original image - this is similar to the reason why robust accuracy from masking alone is higher than that from GAN-based recovery for 25% masking. However, when the masking percentage increases, both the masked and restored images are likely to have labels that are different from that of the original image - thus the difference becomes negligible between both approaches. We see that *Jujutsu* is highly effective in detecting adversarial examples on all the datasets.

> The GAN-based recovery strategy outperforms the masking-alone strategy with higher robust accuracy and lower FPRs.

*Trade-off by varying masking percentage*: Our results also show that the proposed parametric masking is able to moderate the balance between different metrics, based on which the defender can adjust *Jujutsu* to prioritize different outcomes. For instance, if the defender's goal is to detect/mitigate adversarial attack while *minimizing* FPR on the benign inputs, he/she can perform recovery on 50% of the suspicious features, which is able to detect over 91% of the adversarial examples, achieve a robust accuracy of over 70% with a FPR of less than 0.5% (all on average). On the other hand, the defender who wants to *maximize Jujutsu*'s performance can perform recovery on 100% of the suspicious features, which on average yields the highest robust accuracy (79.73%) and detection success recall (95.86%) with a slightly higher FPR (0.71%).

> *Jujutsu* is able to balance between different performance metrics, by varying the percentages of the GAN-based recovery.

## 4.4 RQ3 - Comparison with Related Techniques

We consider four related defenses against patch attacks below (and compare with two more trajan-attack defenses in Appendix A.6).

**1. Localized Gradient Smoothing [30].** Naseer et al. propose local gradient smoothing (LGS) to neutralize the effect of adversarial patch pixels. They first perform normalization over the gradient values, and then use a moving window to identify high-density regions (based on certain thresholds), which will be smoothed out to suppress the influence of the adversarial pixels. We follow [30] to set the threshold as 0.1 and smoothing factor as 2.3.

**2. Adversarial training.** Adversarial training (AT) increases the robustness of DNNs by explicitly training the networks to be robust against the patch attack [34, 42]. We adopt the approach from prior work [34, 42] to conduct AT on ImageNette. We first train the models on clean images, which is then used for adversarial training. For each DNN, we train three different models, one for each patch size. We train the models by using the SGD optimizer and varying different hyperparameters such as learning rate, momentum, dropout, number of epochs, batch size.

**3. SentiNet [12].** Chou et al. propose SentiNet for detecting patch attacks. Sentinet first uses a selective search image segmentation to generate a list of class proposals, i.e., input segments corresponding to different classes. It then extracts the salient maps from

**Table 3: Comparison with LGS [30], SentiNet [12] and PatchGuard [43]. Better results are highlighted in bold.**

| Metric (%) | Technique | | | |
|---|---|---|---|---|
| | LGS | SentiNet | PatchGuard | *Jujutsu* |
| **Detection Recall** | N/A | 73.04 | N/A | **96.89** |
| **Robust Accuracy** | 53.86 | N/A | 11.70 | **77.47** |
| **False Positive** | 12.14 | 7.66 | 44.67 | **1.74** |

the class proposals and identifies the unique features, by subtracting the common regions in the saliency maps, which is then overlaid to a set of new images. It then replaces the extracted features with inert patterns such as Gaussian noise in order to distinguish between adversarial and benign features. The final attack detection is based on statistical analysis on two metrics: (1) the number of misclassified images from images with the unique extracted features, and (2) the average confidence values from images with inert patterns. We follow [12] to overlay the salient features from the test image to 100 new images, which is to calculate the statistics for detecting adversarial examples. We randomly sample 400 clean images to compute the detection threshold for detection. SentiNet [12] is the most closely related technique to *Jujutsu*, and we compare both techniques in detail below.

**4. PatchGuard [43].** Xiang et al. [43] propose PatchGuard, a certified defense technique against patch attacks. The main idea is to enforce small receptive field in the DNNs, and secure feature aggregation by masking out the regions with the highest sum of class evidence (as these regions are more likely to be manipulated by adversarial patch to dominate the prediction).

**Result.** We compare *Jujutsu* (GAN-based recovery with 100% masking) with LGS, SentiNet and PatchGuard on ImageNet. Table 3 shows the average results for patches of different sizes.

**1. Comparison with LGS.** LGS achieves an average robust accuracy of 53.86%, which is considerably lower than that of 77.47% by *Jujutsu*. This is because: (1) not all adversarial patch regions would stand out as the high-density region after normalization by LGS; and (2) LGS uses gradient smoothing as the mitigation strategy, which is inferior to GAN-based recovery by *Jujutsu* that can reconstruct the semantic contents from the corrupted regions. LGS also incurs a very high FP of 12.14%, which is because the natural features in the benign examples may also be identified as high-density regions and hence LGS incorrectly perform gradient smoothing on these regions. In contrast, *Jujutsu* incurs an FP rate of only 1.74%.

**2. Comparison with SentiNet.** SentiNet detects 73% adversarial examples while *Jujutsu* detects 96.89%, which yields an improvement of 32.7%. Further, SentiNet has an FPR of 7.66% while *Jujutsu* has only 1.74% (a 77.3% reduction). Hence, *Jujutsu* outperforms SentiNet by *having a higher detection rate on adversarial examples, and achieving a much lower FPR*. We next qualitatively compare both techniques to understand their significant performance difference.

The low detection rate of SentiNet is due to its poor identification of the adversarial patch region. Specifically, SentiNet extracts the adversarial patch region by subtracting the common regions of the saliency maps belonging to different classes. However, the adversarial patch may reside in the common regions of different saliency maps, and thus the patch region will be removed after subtraction, thereby remaining *undetected*.

**Table 4: Comparison with AT in defending against *multiple* target classes. Better results are highlighted in bold.**

| Metric (%) | Adversarial training | | | *Jujutsu* | | |
|---|---|---|---|---|---|---|
| | 1 target | 3 targets | 5 targets | 1 target | 3 targets | 5 targets |
| **Robust Acy.** | 92.29 | 84.49 | 78.97 | **95.34** | **94.18** | **94.15** |
| **False Positive** | 20.23 | 24.22 | 27.05 | **0.72** | **0.87** | **0.88** |

Though *Jujutsu* also uses saliency map in its detection, *Jujutsu* follows a different principle to locate adversarial patch from the saliency map and proposes a robust suspicious feature detection method (e.g., a pre-processing technique to highlight adversarial patch region) that can reliably locate the adversarial patch *and* verify its maliciousness through a guided feature transplantation and prediction comparison. This allows *Jujutsu* to detect substantially more AEs (32.7% more).

The high FPR in SentiNet is because SentiNet overlays the suspicious features to a *random* region of an image, which could cause FPs when the salient features occlude the image's natural features.

Instead, *Jujutsu* achieves low FPR through: 1) strategically transplanting the salient features to the *least-salient* feature region of the image; and 2) using the generative power of GAN to reduce FPR. Both innovations combined allows *Jujutsu* to achieve a much lower FPR than SentiNet (77.3% lower).

**3. Comparison with PatchGuard.** PatchGuard provides provable robust accuracy but has a robust accuracy of 11.7% and an FPR of 44.67%. In contrast, *Jujutsu* has a 77.47% (empirical) robust accuracy with only 1.74% FP. This is because PatchGuard provides a (provable) lower bound of the adversarial robustness and the high FPR is due to the small receptive field enforced by PatchGuard, which causes considerable clean accuracy drop as in [43].

**4. Comparison with adversarial training (AT).** AT requires training for *each* target class, which is challenging as attackers may target diverse classes. In contrast, *Jujutsu* does not require any training, and is agnostic to the target classes of the attack. Therefore, we compare the performance of AT and *Jujutsu* when the attacker targets different classes, by training multiple 7% patches targeting *different* labels on ImageNette.

Table 4 shows that AT's performance degrades as the number of target classes increases, on both robust accuracy and FP. This is because with more target classes, the learning objective for AT becomes increasingly difficult - this is similar to how common DNNs would yield lower accuracy on a 1000-class dataset than on a simple 10-class dataset. On the other hand, we see that *Jujutsu* achieves consistently high performance in terms of both robust accuracy and FP across attacks targeting different classes. Further, *Jujutsu* yields significantly better performance than AT in all cases.

## 4.5 RQ4 - Physical-world Patch Attacks

In this RQ, we evaluate the effectiveness of *Jujutsu* against physical-world patch attacks. We use the printable adversarial patch from [9]. We printed it out, placed it next to the cell phone object (a iPhone 6s device) at various locations, and captured its video.

Fig. 6 shows the video frames in our evaluation. Both videos with and without patches contain around 430 frames. 80% of the frames with patches successfully caused the targeted misclassification. We increase the length of the detection box to 142 as the physical patch occupies more pixels in the images than the digital patch [9].

**Figure 6: Video frames for a cell phone object with (top row) and without (bottom row) a physical adversarial patch.**

**Table 5: *Jujutsu*'s performance on *physical patch attack*.**

| Metric (%) | Masking percentage | | | |
|---|---|---|---|---|
| | 25% | **50%** | 75% | 100% |
| **Robust Accuracy** | 82.46 | **95.32** | 93.86 | 81.87 |
| **Mitigation FPR** | 1.15 | 2.99 | 3.45 | 5.52 |
| **Mitigation Success Recall** | 86.84 | **95.91** | 95.91 | 95.91 |

**Table 6: *Jujutsu*'s performance on *attacks targeting 5 labels*.**

| Metric (%) | ImageNet | ImageNette | CelebA | Place365 |
|---|---|---|---|---|
| **Robust Accuracy** | 79.27 | 94.15 | 71.69 | 77.32 |
| **Mitigation FPR** | 1.61 | 0.88 | 0.00 | 0.20 |
| **Mitigation Succ. Recall** | 99.54 | 98.57 | 96.25 | 93.98 |

As before, we evaluate the effectiveness of *Jujutsu* in terms of robust accuracy, mitigation success recall and mitigation FPR. The results are shown in Table 5.

*Robust accuracy.* Unlike the previous evaluation on digital patch, we can see from Table 5 that a low masking percentage is able to yield a high robust accuracy for the physical attack (this is low for the evaluation on digital patch). This is because the perturbations in the physical patch are more susceptible to masking and recovery compared with the digital patch that is directly applied to the image. Perturbations in the physical patch need to undergo camera transformations, which makes the perturbations more amenable to being mitigated by *Jujutsu*. Thus, even a low masking percentage in *Jujutsu* is able to effectively mitigate the physical patch attack.

In addition, the robust accuracy from 75% and 100% masking is lower than that from 50%, which is unlike the trend in the previous evaluation for digital patches. This is because the detection box is larger, and hence a higher masking percentage means more contents are masked for the GAN to recover, which leads to degraded quality of the recovered images and thus the DNN is unable to infer the correct label. Therefore, 50% masking yields the highest robust accuracy of 95.32%.

*Mitigation FPR* yielded by *Jujutsu* ranges from 1.15% to 5.52%. Similar to the digital patch, the FPR is higher when the masking percentage is higher.

*Mitigation success recall* yielded by *Jujutsu* ranges from 86.84% to 95.91%. The trend is similar to that of digital patch, i.e., the success recall is low when the masking percentage is low (and vice versa).

## 4.6 RQ5 - Attacks Targeting Different Labels

This section evaluates *Jujutsu* against attacks that target different class labels. For each target label, we need to perform training to generate the universal adversarial patches. Training is highly time-consuming, and hence, for each dataset, we train five 7% patches targeting different labels. Note that training is only needed for creating the adversarial patches, and not for *Jujutsu*.

Table 6 shows the results. *Jujutsu* consistently achieves high performance in detecting and mitigating patch attacks targeting different labels, and with very low FPR. On average, *Jujutsu* detects over 97% of the adversarial examples, achieves a robust accuracy of over 80%, with only 0.67% FPR. This high accuracy is because *Jujutsu* works by comparing the prediction label before and after feature transplantation, which is *agnostic* to the exact target label.

## 4.7 RQ6 - Adaptive Attacks

We now evaluate two adaptive adversaries attempting to fool the DNNs even under *Jujutsu* by: (1) evading the detection by manipulating the saliency map; (2) evading the mitigation by generating strong perturbations that can continue to fool the DNN.

*4.7.1 Evading the Detection.* *Jujutsu* uses the saliency map to detect adversarial patches, and the saliency map can be manipulated by an adversary as shown by prior work [18, 48]. However, they consider adversarial perturbations over the entire image, while we consider *localized* perturbations. Therefore, we adapt their approach [18, 48] to manipulate the localized saliency map so that the adversarial patch *will not* be identified as the most salient features (by reducing its influence on the output), hence evading *Jujutsu*.

**Attack setup.** The influence on the output is derived from the saliency map. This attack can be formulated as the following objective function during the generation of the adversarial patch:

$$\delta = \underset{\delta}{\text{argmax}}\, \mathbb{E}_{x\sim X, l\sim L}(\log \Pr(y = y^{adv}|x') - \beta \|\hat{M}_j^*(x) - m_0^*\|_2^2), \quad (5)$$

where $\hat{M}_j^*(x)$ is defined as the saliency map on the region where the adversarial patch resides (not the entire saliency map), and $m_0^*$ is a mask in the same size of the adversarial patch and filled with 0, $\beta$ is a hyperparameter to balance different loss terms. The first term is to cause targeted misclassification, while the second term's goal is to let $\hat{M}_j^*(x)$ have *small* influence to evade detection by forcing the values within $\hat{M}_j^*(x)$ to be close to 0. The adversary stops the optimization once the patch succeeds in evading the detection box.

The second term can be viewed as manipulating the Hessian matrix of $F(x)$, whose values are all zero for DNNs with ReLu activation functions [18, 48]. Therefore, we replace the ReLu function with a parametric softplus function when calculating the gradients [18, 45]: $f(x) = \frac{1}{\alpha}\log(1 + \exp(\alpha x))$, where $\alpha$ is the hyper-parameter to control the shape of the curve and is set as 10 (following [45]). Finally, we only use the parametric softplus for backward propagation, and use ReLU for the normal forward pass.

To be conservative, we consider the 7% patch, which allows the attacker to inject larger perturbations to evade *Jujutsu*. We choose 200 samples for training the adversarial patch, 500 steps per sample and 20 epochs in total. For each dataset, we choose $\beta \in [0.1, 0.5, 1, 5]$ and choose the one yielding the highest attack success rate.

Equation 5 requires several forward and backward passes for calculating the saliency map $\hat{M}_j^*(x)$, which is much more time-consuming than the original optimization (Equation 3). Therefore, we reduce the sampling size $n$ in Equation 4 from 50 to 5 for faster training. We experimentally verified that the smaller sampling size $n$ does not significantly affect the resulting saliency map, and that we can still find all the salient features. Under this setting, it took around 18 days to generate an adversarial patch on Place365 dataset, compared to about 540 days with our previous setup (estimated).

**Result.** We compare the attack success rate of the patches generated from the undefended models and the ones guarded by *Jujutsu*

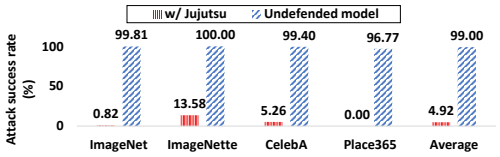


**Figure 7: Attack success rate (ASR) for adaptive attack to evade *Jujutsu*'s *detection*. The lower the better.**

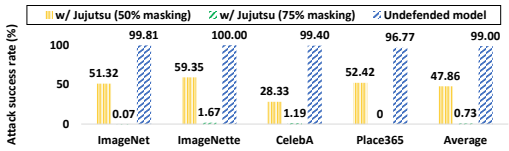


**Figure 8: Attack success rate (ASR) for adaptive attack to evade *Jujutsu*'s *mitigation*. The lower the better.**

in Fig. 7. When *Jujutsu* is used, the adaptive attacker who attempts to evade *Jujutsu*'s detection suffers a significant drop in attack success rate, from 99% to just 4.9% (on average). This is because in Equation 5, the first term aims to *increase* the influence on the final prediction to manipulate the output label; while the second term *reduces* the influence on the output. This equation constrains the adaptive attacker, who cannot evade detection without also significantly degrading the attack's effectiveness in the process.

*4.7.2 Evading the Mitigation.* We now consider a second adaptive attacker who attempts to cause targeted misclassification even if the adversarial examples are detected. Because our masking strategy is parametric, the adaptive attack would be unsuccessful if we perform recovery on the entire set of suspicious features since all of the adversarial perturbations would be removed. Therefore, we study whether the adaptive attack could succeed if the defender masks only 50% or 75% of the suspicious features.

**Attack setup.** The adversary's goal is to generate an adversarial patch that can survive under partial masking (50% or 75% masking). To model the masking of $x$% of the suspicious features, we randomly set $x$% of the values that are non-zero within the mask $m \in \{0, 1\}^n$ to be 0, so that those positions marked with a 0 will not be available for manipulation by the attacker. Therefore, the attacker can use only the remaining perturbations to cause misclassification.

Similar to Section 4.7.1, we consider the 7% patch to maximize the attack's influence. For the masking percentage of 50%, we use 2000 images, a maximal step of 1000 and 30 epochs in total, which is in line with our standard attack generation as in Section 4.1.2. This is because evading the mitigation does not require several forward and backward passes for each step as in Section 4.7.1, and thus we can use more images and optimization steps as well as epochs.

For the 75% masking percentage, we use a maximal epoch of 20, because our evaluation shows that the training is not able to make any progress, and the attack success rate is consistently close to 0%.

**Result.** Fig. 8 shows the percentage of adversarial examples that succeed in causing misclassification despite *Jujutsu*. We can see that the ASRs degrades under the adaptive attack.

When 75% of the perturbations are masked, it is almost infeasible for the attacker to generate a successful adversarial patch, and hence the success rate is near 0% on all datasets.

When 50% of the perturbations are masked, the ASRs are however much higher, ranging from 28% to around 60%. This is because many of the detected adversarial examples will be mis-identified as benign after GAN-based recovery with 50% masking is performed. When the masking percentage is low, the adversarial perturbations will remain intact and they can continue to cause misclassification on the restored images - *Jujutsu* will mis-identify them as benign.

Nevertheless, *the defender can further thwart the attack by increasing the masking percentage to 75% or 100% in* Jujutsu, which only comes at a cost of slightly higher FPRs (0.23% higher) and is able to mitigate most of the adversarial examples. For example, by performing GAN-based recovery on 100% of the suspicious features, the average attack success rate is reduced from 48% to 22%, which is significantly lower than that on the undefended models (99%).

Our results show that *Jujutsu*'s performance can be degraded by the adaptive adversary via deliberately reducing the success rate of patch attacks. For instance, the adaptive patch generated on ImageNet for 50% masking has a success rate of 76.8%, which is lower than the 99% of the non-adaptive patch. This results in 23.5% of adversarial examples going undetected by *Jujutsu*. However, we note that doing so would significantly *constrain* the adversary's ability in fooling the DNN, and hence make the DNN significantly less susceptible to patch attacks. On average, the attack success rate is reduced from 99% to 0.73%∼ 22% across the four datasets.

## 5 DISCUSSION

This section first discusses the limitation of *Jujutsu*, followed by the evaluation of *Jujutsu*'s performance on more DNN models and more different patch sizes. We present an ablation study in Appendix A.2.

### 5.1 Limitation

First, *Jujutsu* incurs overhead in its attack detection and mitigation, and we report its overhead in Appendix A.7 due to space limitations.

Second, *Jujutsu* employs PICnet [49] in its attack mitigation, whose performance also affects *Jujutsu*'s mitigation performance. For example, *Jujutsu* has lower robust accuracy on the CelebA dataset (than other dataset) as it is challenging to synthesize specific human faces with PICNet for the DNN to make correct predictions. Nevertheless, as image inpainting with GAN is an active research area, we believe this issue can be further alleviated by incorporating recent research results. For instance, Li et al. [24] recently introduced a transformer-based model that yields superior inpainting performance on diverse tasks, which may be leveraged by *Jujutsu* to enhance its mitigation performance.

Finally, there are other attack variants outside our threat model. We discuss next how *Jujutsu* may be extended to handle them.

*Multiple patches.* We focus on defense against single-patch attacks, on which existing defenses [12, 30, 34, 42, 43] have very limited success. However, multi-patch attacks are also possible and one potential solution to handle them is to *iteratively* perform detection and mitigation until only the benign features are left in the images. This can be achieved by checking whether the current suspicious features *fail* to cause misclassification on the hold-out inputs, which occurs when the suspicious features are benign.

To validate this, we evaluate the above extension of *Jujutsu* against 2-patch and 3-patch attacks and it is still able to achieve good performance - see Appendix A.3.

*Patches in different shapes.* Similar to prior work [23, 28, 30, 43], we assume a square or circle patch, and conduct extensive evaluation on these attacks. But patches in other shapes such as rectangular one are also possible, which is a limitation of *Jujutsu* (and also of other defenses [23, 28, 30, 43]).

Nevertheless, we evaluate how *Jujutsu* can be extended to defend against rectangular patch (by assuming a rectangular detection box) - see Appendix A.4 for details. We leave the generalization of *Jujutsu* to different patch shapes to future work (e.g., instead of using one single detection bounding box, can we use multiple detection boxes in different shapes to cover different potential patches and flag an attack if any of the suspicious features extracted from different boxes is deemed as adversarial?).

*Untargeted attacks.* We focus on targeted attacks, which allow the adversary to manipulate the DNNs in a controlled manner. Other work [6, 22] has recommended evaluating targeted attacks for large-scale datasets like ImageNet, as untargeted attacks may cause misclassification of very similar classes (e.g., images of two very similar dog breeds).

We tried extending *Jujutsu* to defend against untargeted attack by directly performing attack mitigation on the suspicious features (without the guided feature transplantation and prediction comparison for attack detection), and use the prediction label on the recovered image as the final output, but we had very limited success (see Appendix A.5 for details). Therefore, future work may combine *Jujutsu* with other defenses (e.g., using pre-defined thresholds [30] or small masking on the test image [44]) to first characterize benign and adversarial examples, and then use *Jujutsu*'s mitigation technique to perform attack recovery.

**Table 7: *Jujutsu*'s performance on 6 different DNNs.**

| Metric (%) | Wide-ResNet | Dense-Net121 | Squeeze-Net | VGG16 | ResNet-152 | Google-Net |
|---|---|---|---|---|---|---|
| **Clean Accuracy** | 78.27 | 71.58 | 54.74 | 72.03 | 76.84 | 67.17 |
| **Robust Accuracy** | 82.21 | 76.6 | 69.03 | 78.33 | 79.33 | 73.44 |
| **Mitigation FPR** | 2.13 | 2.39 | 0.67 | 1.91 | 2.38 | 3.47 |
| **Mitigation Succ. Recall** | 99.92 | 98.11 | 98.43 | 99.05 | 97.75 | 99.92 |

## 5.2 Evaluation on More DNN Models

This section evaluates *Jujutsu*'s performance on 6 more DNN models (Wide-ResNet, DenseNet121, SqueezeNet VGG16, ResNet152 and GoogleNet) on ImageNet (all are the pre-trained models from the TorchVision library). We report the results in Table 7.

*Jujutsu* consistently achieves high detection performance (average detecting over 98% AEs), low FPRs (average 2.16%), and high robust accuracy (average 76.49%). The robust accuracy varies between different models because it is also related to the prediction performance of the model, e.g., SqueezeNet has a relatively low clean accuracy, which also leads to lower robust accuracy compared with other models. Our results show that *Jujutsu*'s outstanding defense performance can generalize across different DNN models.

## 5.3 Evaluation on Larger Patches

Section 4 evaluates 5%~7% patches. In this section we evaluate *Jujutsu* against larger patch: 8%, 9% and 10% We report the average results as there is no major variation between different patch sizes.

On average, *Jujutsu* detects 99% of the adversarial examples, achieve a robust accuracy of 81.77% with a FPR of only 0.71%. *Jujutsu* yields slightly better performance on these larger patches (compared with the results in Table 2) because these larger patches are able to achieve slightly higher attack success rate. Therefore, *Jujutsu*'s high detection and mitigation performance and low FPs can generalize to larger adversarial patches.

# 6 RELATED WORK

Defenses against adversarial attack can be divided into certified [11, 23, 28, 43, 44] and empirical defenses [12, 19, 21, 30, 34, 42, 46].

*Certified defenses.* Chiang et al. [11] propose the first certified defense against patch attack by Interval Bound Propagation, which constrains the influence of the adversarial pixels in the hidden layers to compute a lower bound of the robustness. Levine et al. [23] introduce de-randomized smoothing (DS) to build a smoothed classifier whose prediction is based on the ensemble of local prediction on pixel patches. Xiang et al. [43] propose PatchGuard which is based on enforcing small receptive field and robust feature masking. PatchCleanser [44] is a provable defense whose main idea is that clean examples are more robust than AEs under small masking and the AEs can be mitigated via a two-round masking.

*Empirical defenses.* Naseer et al. [30] propose local gradient smoothing (LGS) to mitigate patch attacks by identifying the regions with high gradient magnitude, and smoothing out regions whose gradient values are greater than a certain threshold. Hayes et al. [19] introduce digital watermark (DW), which uses pre-defined thresholds to scan the saliency map to detect and remove important pixels (hence the adversarial pixels are also removed). Jha et al. [21] detect patch attacks by selectively masking the top-$k$% salient features and comparing the prediction labels. While useful in detecting adversarial examples, these techniques also incorrectly flag many benign samples as being adversarial because the natural features in benign samples may also exceed the pre-defined thresholds or remain as the top-$k$% salient features. Hence these techniques suffer from high FPRs. Chou et al. [12] propose SentiNet for detecting patch attacks based on model interpretability and statistical analysis, which provides limited detection performance and does not support attack mitigation. Adversarial training, a standard adversarial defense technique, can also be used to defend against patch attacks [34, 42]. But it also suffers from high FPRs and performance degradation when the adversary targets multiple labels. Unlike prior work, *Jujutsu* is a two-staged defense that provides both attack detection and mitigation, and yields superior detection and mitigation performance with very low FPRs.

# 7 CONCLUSION

This work proposes *Jujutsu*, a technique to detect and mitigate robust and universal adversarial patch attacks against image classification DNNs. *Jujutsu* accurately locates adversarial patch and distinguishes it from benign samplse, and uses generative adversarial networks to reconstruct the "clean" examples from adversarial examples. Our extensive evaluation on four datasets and comparison with four defenses show that *Jujutsu* achieves superior detection and mitigation performance, with low false positives. *Jujutsu* also defends against physical-world and adaptive attacks.

## ACKNOWLEDGEMENTS

This work was funded in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), and a Four Year Fellowship from the University of British Columbia (UBC).

## REFERENCES

[1] [n.d.]. CelebA dataset. https://github.com/ndb796/CelebA-HQ-Face-Identity-and-Attributes-Recognition-PyTorch.
[2] [n.d.]. Code for Februus defense. https://github.com/AdelaideAuto-IDLab/Februus.git.
[3] [n.d.]. Code for STRIP defense. https://github.com/garrisongys/STRIP.
[4] [n.d.]. Image Filtering Median Filtering. https://homepages.inf.ed.ac.uk/rbf/HIPR2/mean.htm.
[5] [n.d.]. ImageNette dataset. https://github.com/fastai/imagenette.
[6] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420* (2018).
[7] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. 2018. Synthesizing robust adversarial examples. In *International conference on machine learning*. PMLR, 284–293.
[8] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* (2016).
[9] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. 2017. Adversarial patch. *arXiv preprint arXiv:1712.09665* (2017).
[10] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. 2018. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*. IEEE, 67–74.
[11] Ping-yeh Chiang, Renkun Ni, Ahmed Abdelkader, Chen Zhu, Christoph Studor, and Tom Goldstein. 2020. Certified defenses for adversarial patches. *arXiv preprint arXiv:2003.06693* (2020).
[12] Edward Chou, Florian Tramer, and Giancarlo Pellegrino. 2020. Sentinet: Detecting localized universal attacks against deep learning systems. In *2020 IEEE Security and Privacy Workshops (SPW)*. IEEE, 48–54.
[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.
[14] Bao Gia Doan, Ehsan Abbasnejad, and Damith C Ranasinghe. 2020. Februus: Input purification defense against trojan attacks on deep neural network systems. In *Annual Computer Security Applications Conference*. 897–912.
[15] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. 2019. A guide to deep learning in healthcare. *Nature medicine* 25, 1 (2019), 24–29.
[16] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1625–1634.
[17] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. 2019. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*. 113–125.
[18] Amirata Ghorbani, Abubakar Abid, and James Zou. 2019. Interpretation of neural networks is fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3681–3688.
[19] Jamie Hayes. 2018. On visible adversarial perturbations & digital watermarking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 1597–1604.
[20] Lifeng Huang, Chengying Gao, Yuyin Zhou, Cihang Xie, Alan L Yuille, Changqing Zou, and Ning Liu. 2020. Universal Physical Camouflage Attacks on Object Detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 720–729.
[21] Susmit Jha, Sunny Raj, Steven Lawrence Fernandes, Sumit Kumar Jha, Somesh Jha, Gunjan Verma, Brian Jalaian, and Ananthram Swami. 2019. Attribution-driven causal analysis for detection of adversarial examples. *arXiv preprint arXiv:1903.05821* (2019).
[22] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. 2018. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373* (2018).
[23] Alexander Levine and Soheil Feizi. 2020. (De) Randomized Smoothing for Certifiable Defense against Patch Attacks. *arXiv preprint arXiv:2002.10733* (2020).
[24] Wenbo Li, Zhe Lin, Kun Zhou, Lu Qi, Yi Wang, and Jiaya Jia. 2022. MAT: Mask-Aware Transformer for Large Hole Image Inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10758–10768.
[25] Zhuohang Li, Yi Wu, Jian Liu, Yingying Chen, and Bo Yuan. 2020. AdvPulse: Universal, Synchronization-free, and Targeted Audio Adversarial Attacks via Subsecond Perturbations. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 1121–1134.
[26] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. 2018. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 85–100.
[27] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.
[28] Michael McCoyd, Won Park, Steven Chen, Neil Shah, Ryan Roggenkemper, Minjune Hwang, Jason Xinyu Liu, and David Wagner. 2020. Minority Reports Defense: Defending Against Adversarial Patches. *arXiv preprint arXiv:2004.13799* (2020).
[29] T Nathan Mundhenk, Barry Y Chen, and Gerald Friedland. 2019. Efficient saliency maps for explainable AI. *arXiv preprint arXiv:1911.11293* (2019).
[30] Muzammal Naseer, Salman Khan, and Fatih Porikli. 2019. Local gradients smoothing: Defense against localized adversarial attacks. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 1300–1307.
[31] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. 2015. Deep face recognition. (2015).
[32] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. 2016. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2536–2544.
[33] Qing Rao and Jelena Frtunikj. 2018. Deep learning for self-driving cars: Chances and challenges. In *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems*. 35–38.
[34] Sukrut Rao, David Stutz, and Bernt Schiele. 2020. Adversarial Training against Location-Optimized Adversarial Patches. *arXiv preprint arXiv:2005.02313* (2020).
[35] Berkman Sahiner, Aria Pezeshk, Lubomir M Hadjiiski, Xiaosong Wang, Karen Drukker, Kenny H Cha, Ronald M Summers, and Maryellen L Giger. 2019. Deep learning in medical imaging and radiation therapy. *Medical physics* 46, 1 (2019), e1–e36.
[36] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*. 618–626.
[37] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. 2017. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825* (2017).
[38] Liwei Song, Xinwei Yu, Hsuan-Tung Peng, and Karthik Narasimhan. 2020. Universal Adversarial Attacks with Natural Triggers for Text Classification. *arXiv preprint arXiv:2005.00174* (2020).
[39] Yi Sun, Ding Liang, Xiaogang Wang, and Xiaoou Tang. 2015. Deepid3: Face recognition with very deep neural networks. *arXiv preprint arXiv:1502.00873* (2015).
[40] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. *arXiv preprint arXiv:1703.01365* (2017).
[41] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).
[42] Tong Wu, Liang Tong, and Yevgeniy Vorobeychik. 2020. Defending Against Physically Realizable Attacks on Image Classification. In *International Conference on Learning Representations*.
[43] Chong Xiang, Arjun Nitin Bhagoji, Vikash Sehwag, and Prateek Mittal. 2020. PatchGuard: Provable Defense against Adversarial Patches Using Masks on Small Receptive Fields. *arXiv preprint arXiv:2005.10884* (2020).
[44] Chong Xiang, Saeed Mahloujifar, and Prateek Mittal. 2021. PatchCleanser: Certifiably Robust Defense against Adversarial Patches for Any Image Classifier. *arXiv preprint arXiv:2108.09135* (2021).
[45] Cihang Xie, Mingxing Tan, Boqing Gong, Alan Yuille, and Quoc V Le. 2020. Smooth adversarial training. *arXiv preprint arXiv:2006.14536* (2020).
[46] Zirui Xu, Fuxun Yu, and Xiang Chen. 2020. LanCe: A comprehensive and lightweight CNN defense methodology against physical adversarial attacks on embedded multimedia applications. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 470–475.
[47] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. 2019. Free-form image inpainting with gated convolution. In *Proceedings of the IEEE International Conference on Computer Vision*. 4471–4480.
[48] Xinyang Zhang, Ningfei Wang, Hua Shen, Shouling Ji, Xiapu Luo, and Ting Wang. 2020. Interpretable deep learning under fire. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*.
[49] Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. 2019. Pluralistic image completion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1438–1447.
[50] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. 2017. Places: A 10 million Image Database for Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).


**Figure 9: Adversarial examples on each dataset.**

# A APPENDIX

## A.1 *Jujutsu*'s Performance under Different Parameter Values

As mentioned in Section 4.1.3, *Jujutsu* has three main parameters in its defense setup: (1) kernel size for pre-processing (average filtering) the saliency map; (2) size of the detection bounding box; and (3) number of hold-out images used for feature transplantation. This section evaluates *Jujutsu*'s performance by varying each of these parameters (all on ImageNet), from which we choose the parameters used in our main evaluation.

*A.1.1 Kernel size for pre-processing (average filtering) the saliency map.* We consider the following different kernel sizes for pre-processing the saliency map: 11, 31, 51 and 71, and the results are in Table 8.

When the kernel size is small, *Jujutsu* has lower detection recall (93.28% for kernel size of 11 vs. over 97% for other larger sizes). Low detection recall means *Jujutsu* erroneously identifies many natural (benign) features as the suspicious features in adversarial examples (hence the actual adversarial patch remains undetected). This is because the natural features would still remain as the salient feature if the kernel size is small (i.e., the kernel area for average filtering). Instead, a larger kernel size can smooth out the regions associated with the benign features, and the regions associated with adversarial patch still remain salient and can be identified by *Jujutsu*. Based on the above analysis, we use a kernel size of 51 in our main evaluation (which has a good balance between different performance metrics).

**Table 8: *Jujutsu*'s performance by using different kernel sizes for pre-processing the saliency map.**

| Metric (%) | Kernel size for average filtering | | | |
|---|---|---|---|---|
| | 11 | 31 | 51 | 71 |
| **Mitigation Success Recall** | 93.28 | 97.28 | 98.35 | 98.04 |
| **Robust Accuracy** | 71.37 | 74.53 | 76.14 | 75.78 |
| **False Positive** | 1.38 | 1.6 | 1.87 | 1.87 |

*A.1.2 Size of the detection bounding box.* We consider the following different sizes of the detection bounding box: 78, 90, 102, 114, 122, which correspond to around 12%, 16%, 20%, 25% and 30% image pixels. We report the results in Table 9.

When the detection bounding box is small, it is more difficult for *Jujutsu* to locate the adversarial patch from the adversarial examples, which leads to lower detection recall as shown. Conversely, we see that detection recall increases as the detection bounding box grows in size.

On the other hand, larger bounding box could also degrade robust accuracy and cause more FP. This is because when the bounding box is large, more contents need to be recovered by the GAN, which is a more challenging task.

Based on the above, bounding boxes in different sizes can be used based on different objectives, e.g., one can use a smaller bounding box to minimize FP while maintaining good detection and mitigation performance. We use a size of 102 in our main evaluation.

**Table 9: *Jujutsu*'s performance by using detection bounding boxes in different sizes**

| Metric (%) | Size of detection bounding box | | | | |
|---|---|---|---|---|---|
| | 78 | 90 | 102 | 114 | 122 |
| **Mitigation Success Recall** | 91.93 | 94.89 | 97.66 | 98.93 | 99.07 |
| **Robust Accuracy** | 81.88 | 80.89 | 78.69 | 70.35 | 63.13 |
| **False Positive** | 0.08 | 0.7 | 1.66 | 4.34 | 7.82 |

*A.1.3 Number of hold-out images used for feature transplantation* . This section shows *Jujutsu*'s performance when using different number of hold-out images for feature transplantation (in attack detection). The results are presented in Table 10.

As shown, detection success recall reduces as the number of images for detection increases. This is because *Jujutsu* determines an adversarial example only if *all* the images implanted with the salient features have the same labels as the original test image. When more images are used for detection, it is more difficult for the adversarial patch to cause the same misclassification on all the hold-out images (it is easier to cause the targeted misclassification on 1 image than on 2 images or more). Hence, *Jujutsu*'s detection performance degrades when more images are used for detection.

We also observe that the FPR reduces as we use more images for detection. The reason is similar to the above. Specifically, a benign image will be mis-detected only if its salient features cause the same prediction labels on all the hold-out images after feature transplantation, which is increasingly difficult as the number of hold-out images increase.

Based on the above, different number of hold-out images can be used based on different objectives, e.g., one can use more images for detection in order to minimize FP. We use 2 images for detection in our main evaluation.

**Table 10: *Jujutsu*'s performance when using different number of hold-out images for detection.**

| Metric (%) | Number of images for detection | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| **Mitigation Success Recall** | 98.79 | 97.58 | 96.40 |
| **Robust Accuracy** | 78.59 | 78.20 | 77.00 |
| **False Positive** | 4.78 | 1.76 | 0.82 |

## A.2 Ablation Study

We consider the following four components in the ablation study.

(1) *Pre-processing the saliency map to identify suspicious features (for attack detection).* We compare *Jujutsu*'s performance with and without the pre-processing component (on ImageNet with 10% patch). Without the pre-processing component, *Jujutsu* detects only 91.93% AEs (Vs. 98.35% with the pre-processing component), because it cannot correctly locate the adversarial patch region from the AEs (instead it identifies many natural features as the suspicious features). The low detection performance also leads to lower robust accuracy of 71.99% Vs. 76.14% with the pre-processing component.

(2) *Guided feature transplantation to reduce FPRs.* We compare the performance of using guided feature transplantation Vs. random feature transplantation. While both approaches achieve similar

detection recall (98.35% and 99.07%) and robust accuracy (75.11% and 76.22%), the proposed guided feature transplantation achieves an FPR of 1.55% Vs. 2.9% by the random transplantation, which constitutes a reduction of FPR by 46.6%.

(3) *GAN-based attack recovery.* We compare the GAN-based and masking-alone attack mitigation strategy, and the detailed results are reported and discussed earlier in Section 4.3. We summarize the main difference here. The proposed GAN-based recovery method outperforms the basic masking-alone strategy (for 75% and 100% masking) with: (1) higher robust accuracy (6.56% higher for 100% masking); and (2) lower FPRs (0.25% lower), because masking alone will cause the loss of semantic contents, which is undesirable for prediction comparison, while the GAN can recover the semantic contents from the masked pixels[5].

(4) *Prediction comparison for reducing FPRs.* Table 1 shows that *Jujutsu* yields an average FPR of 3.33%, which can be reduced to 0.5%~0.71% (depending on different masking percentages used - see Table 2). By comparing the prediction label on the original and recovered inputs using GAN, *Jujutsu* can effectively reduce FP on benign examples, which enables a FPR reduction by 78%~85%.

In summary, the ablation study shows that the above components in *Jujutsu* are crucial in enabling *Jujutsu* to achieve high detection performance, high robust accuracy and low FPRs.

## A.3 Extending *Jujutsu* to Defend Against Multi-patch Attacks

This section evaluated the extension of *Jujutsu* against multi-patch attacks as discussed in Section 5.1. To do so, we modify *Jujutsu* to *iteratively* perform detection and mitigation until only the benign features are left in the images, i.e., the current suspicious features fail to cause misclassification on the hold-out inputs. We consider both 2-patch and 3-patch attacks, and an example is shown below. We report the results in Table 11.

With the above extension, we can see that *Jujutsu* still remains as an effective defense against patch attacks. In both cases, *Jujutsu* detects over 96% adversarial samples with around 2% FPR. For 2-patch attack, *Jujutsu* still achieves a very high robust accuracy of 73.11%, which is slightly lower than that of 77.47% on single-patch attack. For 3-patch attack, however, *Jujutsu* achieves a much lower robust accuracy of 46.98%, which is because mitigating 3 patches in one single image is more difficult than single- or two-patch attacks. Our result therefore demonstrates that *Jujutsu* can be extended to effectively defend against multi-patch attack.

**Table 11: *Jujutsu*'s performance against multi-patch attacks**

| Metric (%) | 2-patch attack | 3-patch attack |
|---|---|---|
| **Mitigation Success Recall** | 96.16 | 96.78 |
| **Robust Accuracy** | 73.11 | 46.98 |
| **False Positive** | 2.26 | 1.93 |

[5]Note that the GAN is *not* always able to recover the correct contents, which explains why the robust accuracy is not as high as the detection recall. Nevertheless, the proposed GAN-based recovery *still outperforms* the basic masking-alone strategy, and is hence adopted by *Jujutsu*.

## A.4 Extending *Jujutsu* to Defend Against Rectangular Patch Attacks

We now evaluate how *Jujutsu* can be extended to defend against rectangular patch attacks. We train the rectangular patches on each dataset using a 7% patch (36*96). As discussed in Section 5.1, we assume the defender is aware of the shape of the patch and hence we use a rectangular bounding box, which occupies around 20% of pixels as before, and it has a width/height ratio of 6:4. Table 12 shows the results on different datasets. On average, *Jujutsu* is able to detect 97.25% adversarial examples, achieve robust accuracy of 78.3% with only 0.57% false positive.

The current extension assumes the defender's knowledge of the potential patch shape, and we leave the improvement of *Jujutsu* to be general to different patch shape in future work. For example, instead of using one single detection bounding box, can we use multiple detection boxes in different shapes to cover different potential patches?

**Table 12: *Jujutsu*'s performance on *rectangular* patch attack.**

| Metric (%) | ImageNet | ImageNette | CelebA | Place365 | Average |
|---|---|---|---|---|---|
| **Robust Accuracy** | 75.44 | 92.42 | 64.58 | 80.76 | 78.3 |
| **False Positive** | 1.54 | 0.67 | 0.00 | 0.06 | 0.57 |
| **Mitigation Success Recall** | 98.43 | 98.84 | 92.43 | 99.30 | 97.25 |

## A.5 Extending *Jujutsu* to Defend Against Untargeted Attacks

We now describe our effort in modifying *Jujutsu* against untargeted attacks. *Jujutsu*'s current attack detection is not designed for untargeted attacks, and hence we modify *Jujutsu* by directly performing attack mitigation on the suspicious features (the procedure to identify the suspicious features is the same), and using the use the prediction label on the recovered image as the final output.

Since we do not use *Jujutsu*'s detection, we focus on the robust accuracy and FPR on benign sample as the evaluation metric. In this setting, *Jujutsu* achieves a robust accuracy of 55.15%, but with an elevated FPR of 44%. *Jujutsu* still achieves over 55% robust accuracy because it is able to successfully identify the adversarial patch region in many AEs (over 68%) and hence it can perform attack recovery on these adversarial patch regions. On the other hand, *Jujutsu* yields a high FPR because *Jujutsu* does not incorporate its attack detection pipeline to distinguish benign samples and directly uses our mitigation strategy to reduce FPR (Section 3.3.3), which can only be reduced to 44%.

Therefore, future work could look into combining *Jujutsu* with other defenses, e.g., recent efforts use pre-defined thresholds [30] or performs small masking on the test image and use the prediction disagreement on the masked images [44] to characterize benign and adversarial examples, which may be combined with *Jujutsu* to facilitate an effective attack detection and mitigation defense.

## A.6 Comparison with 2 Trojan-attack Defenses

In Section 4.4, we compare with 4 existing defenses designed for countering patch attacks. We now evaluate 2 additional defenses built for trojan attacks (STRIP [17] and Februus [14]), and we evaluate whether they are also effective against patch attacks. We choose these two techniques because: (1) STRIP relies on superimposing

two different images to detect trojan attack, which may also be effective for patch attacks if the adversarial examples (corrupted with adversarial patch) continue to cause misclassification after being superimposed with another image. (2) Februus uses pre-defined threshold to scan the saliency map for trojan attack detection, which is similar to other detection techniques for patch attacks [12, 30].

*Comparison with STRIP [17].* Gao et al. [17] propose STRIP to defend against patch-like trojaned adversarial examples by superimposing the entire target image with a number of new images, and detect adversarial examples based on the prediction entropy on the set of new images. The prediction entropy is compared against a detection boundary (derived from benign inputs), and a low entropy indicates that the target image is adversarial. We use the implementation from [3], and we use 2000 images for deriving the detection threshold and construct 100 superimposed examples per testing image, similar to the original paper.

We conduct the evaluation on ImageNet, and found that STRIP only detects around 6% of the adversarial examples while *Jujutsu* detects over 99% (The FPRs by both techniques are both less than 2%). STRIP achieves low detection performance because the adversarial patch is no longer effective after being blended with the new images, thus the prediction entropy is high on the new images.

*Comparison with Februus [14].* Doan et al. [14] propose Februus to defend against patch-like trojaned adversarial examples, which first performs attack detection by identifying the regions that exceed pre-defined threshold in the saliency map, and then performs image restoration on these regions for attack mitigation. We use the original implementation from [2] on VGGFace2 [10].

Under the VGGFace2 dataset, *Jujutsu* achieves a robust accuracy of 37.26%[6], which is significantly higher than that of 0.2%[7] by Februus. This is because Februus relies on the pre-defined threshold to identify the regions associated with adversarial patch. This method would fail to locate the adversarial patch if the patch's influence to the prediction is lower than the threshold, and our experiment validates this.

Our results show that although STRIP [17] and Februus [14] are effective defenses against trojan attack, they are not able to defend against patch attacks.

## A.7 Overhead of *Jujutsu*

*Attack detection. Jujutsu*'s detection involves three steps: (1) locate the salient features from the saliency map; (2) identify the least-salient region of the hold-out inputs and perform feature transplantation and (3) prediction comparison. We perform the evaluation on the ImageNet dataset, and repeat the evaluation 5 times and report the average overhead (on a single Nvidia RTX 3090 GPU).

In step 2, the identification of the least-salient region of the hold-out inputs can be performed offline (since it is independent of the runtime input), hence we evaluate the overhead in performing feature transplantation only. Step 3 requires prediction comparison on 3 images (1 original image and 2 hold-out images implanted with suspicious features), which can be executed in parallel to facilitate faster inference. Step 1 can be decomposed into two steps: computing the saliency map and locating the saliency features from the saliency map. The majority of the overhead by *Jujutsu* is from computing the saliency map using SmoothGrad. The overhead of using SmoothGrad with 15-iteration implementation (i.e., 15 random examples for computing the average gradients) is 340*ms*, and the total overhead by *Jujutsu* is 345.7*ms*. Nevertheless, the overhead in generating saliency map can be optimized by using more efficient saliency map methods, such as [29], which can also accurately generate the salinecy maps (hence not affecting *Jujutsu* detection efficacy) but with a speedup of 1456$x$ over SmoothGrad [29] (this is because [29] requires only a single forward pass through a few of the layers in a network; while SmoothGrad requires multiple forward and backward pass that are more time-consuming). Hence, the overhead by *Jujutsu* can be reduced to 5.93*ms* (estimated), while an inference on the undefended model takes 5.56*ms*, which translates to a 6.7% overhead by *Jujutsu*.

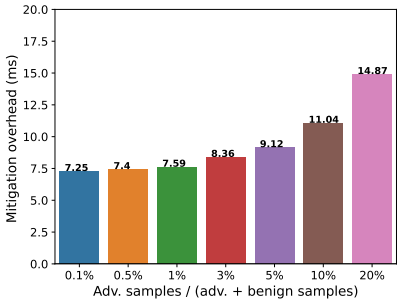


**Figure 10: Overhead of performing attack mitigation by *Jujutsu*, under different ratios of adversarial examples within all the test inputs. One inference pass on the undefended model took 5.56*ms*.**

*Attack mitigation. Jujutsu* involves using a GAN to recover the uncorrupted examples from adversarial examples and prediction comparison for attack mitigation. Note that this process is activated only *after* an attack is detected, hence its overhead is also dependent on the *ratio of adversarial examples in all the test inputs*. For this reason, we plot the overhead under different ratios of adversarial examples, and we show the results in Fig. 10.

When the attack ratio is below 1%, the mitigation overhead by *Jujutsu* (including both performing GAN-based recovery and prediction comparison) took less than 7.6*ms* while inference on the undefended model took 5.56*ms*. As the attack ratio increases, *Jujutsu* incurs higher overhead as it needs to perform more mitigation task on the increasing amount of adversarial examples.

[6] *Jujutsu* has a lower robust accuracy on VGGFace2 than those on the other datasets due to the insufficient performance yielded by the GAN (PICNet [49]). This is because we need to train the PICNet from scratch on VGGFace2, which is very time-consuming as VGGFace2 is a very large dataset. We trained the PICNet on a small subset of the dataset for a week and used it in our evaluation due to time constraint. The performance of *Jujutsu* can be further improved with more resources to train the PICNet (e.g., increase the size of training set and number of epoches).

[7] To ensure the code was implemented correctly, we verified that the code was able to reproduce the results reported in the original paper for trojan attack. We then used the code to evaluate against patch attacks.