Patch Robustness Certification for Transformer via Black-box Testing Approach

by

Yuheng Huang

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering
University of Alberta

© Yuheng Huang, 2023

Abstract

In recent years, deep neural networks (DNNs) have revolutionized the field of artificial intelligence (AI), leading to breakthroughs in areas such as computer vision, natural language processing, and robotics. Despite their superior performance, studies have demonstrated that they are susceptible to input data changes, making them vulnerable to adversarial attacks, where malicious inputs are designed to deceive the DNNs and cause incorrect outputs. This can be a serious weakness for deploying DNNs in safety-critical scenarios, such as autonomous driving, medical diagnosis, face authentication, intruder detection and aircraft control.

More recently, we have witnessed the tremendous success of Transformer as a representative DNN architecture. Vision Transformer (ViT), a particular variant applied for vision tasks, has achieved state-of-the-art performance on various benchmark datasets, demonstrating its effectiveness in learning from large amounts of visual data. However, its robustness is still a major limitation. Its record-breaking performance comes at the cost of extreme sensitivity with respect to the inputs. Relevant studies also demonstrate that its ability to defend against a particular type of attack: physical patch attack, is even weaker than classical convolutional neural networks (CNNs). This poses a serious threat to the deployment of ViT in industries, especially in safety-critical domains.

In this work, we propose PatchCensor, a systematic approach that aims to improve the patch robustness of ViT in a black-box manner. By assuming the attackers have the maximum capability, we design a warning system that can detect the adversary's presence under the worst condition. Our methodology falls into the category of certified defense. Unlike empirical defenses, such approaches can provide a strong guarantee for the inference result. Existing certified defense methods often require substantial efforts in training and usually inevitably sacrifice the base model's performance. To bridge the gap, PatchCensor aims to improve the robustness of the whole system by detecting anomalous inputs rather than relying solely on training a robust model to provide accurate results for all inputs, which could potentially compromise its overall accuracy.

Preface

Some of the research conducted for this thesis forms part of an international research collaboration, supervised by Professor Lei Ma at the University of Alberta, Canada and Professor Yuanchun Li at the Institute for AI Industry Research (AIR) at Tsinghua University, China.

The method, as well as the majority of experiment results, including the preliminary study and four research questions, have been published as Yuheng Huang, Lei Ma, and Yuanchun Li, "PatchCensor: Patch Robustness Certification for Transformers via Exhaustive Testing," *ACM Transactions on Software Engineering and Methodology*, 2023.

Citations will be provided for all non-original images used in this work.

Acknowledgements

I am supported by Canada CIFAR AI Chairs Program, the Natural Sciences and Engineering Research Council of Canada (Funding reference numbers: NSERC No.RGPIN-2021-02549, No.RGPAS-2021-00034, No.DGECR-2021-00019)

I would like to express my sincere gratitude to Professor Lei Ma at the University of Alberta for his invaluable guidance, support, and encouragement throughout my research. His expertise and insight are instrumental in shaping my research project and helping me to develop as a researcher.

I am also deeply grateful to Professor Yuanchun Li at Institute for AI Industry Research (AIR), Tsinghua University, China. His insights and suggestions are critical in helping me to formulate my research questions and methodology and in guiding me toward more robust and meaningful conclusions.

Table of Contents

1 Introduction **1**

1.1 Background 1

1.1.1 Digital Adversarial Attack 1

1.1.2 Physical Adversarial Attack 3

1.1.3 Transformer 5

1.2 Defense of Adversarial Attack 6

1.2.1 Empirical Defense 7

1.2.2 Certified Defense 9

1.3 Our Method 13

1.4 Contribution 16

1.5 Organization of The Paper 16

2 Related Work **18**

3 Preliminary Study **23**

3.1 Evaluation Metrics 23

3.2 Experiment Design and Results 24

4 Method **30**

4.1 Problem Formulation 30

4.2 The Testing-based Defense 32

4.3 Mutation Strategy 35

4.4 Certification Analysis 37

5 Experiment **39**

5.1 RQ1: Performance under Normal Setting 42

5.2 RQ2: Performance under Small ROI 45

5.3 RQ3: Performance under Strong Adversaries 48

5.4 RQ4: Overhead of PatchCensor 54

6 Discussion **58**
6.1 A Special Case for Certified Detection 58
6.2 A Possible Way to Improve Certified Recovery 61

7 Conclusions and Future Work **63**

Bibliography **65**

List of Tables

5.1 The clean and certified accuracy of different certified defenses on ImageNet and CIFAR-10. The numbers of IBP, DS, PG, BC, MR, PG++, and SC are directly copied from their paper. Note that the results of IBP and MR on ImageNet are not available, because they are computationally intensive or even infeasible on high-resolution images. . . . 43

5.2 The clean and certified accuracy of different certified defenses on GTSRB and Food-101. We fine-tune the models on GTSRB for 30 epochs and Food-101 for 60 epochs. All the models are fine-tuned with respect to the given patch size. 44

5.3 The clean accuracy, certified accuracy, and accuracy in trust domain achieved by PatchCensor (PC), PatchGuard (PG) and De-randomized Smoothing (DS) for 2.4%-pixel adversarial patch attack. 46

5.4 Clean accuracy & certified accuracy achieved by PatchCensor and MR+ on **CIFAR-10** under different adversarial patch sizes **without fine-tuning** 51

5.5 Clean accuracy & certified accuracy achieved by PatchCensor and MR+ on **CIFAR-10** under different adversarial patch sizes **with fine-tuning** 51

5.6 Clean accuracy & certified accuracy achieved by PatchCensor and MR+ on **GTSRB** under different adversarial patch sizes **without fine-tuning** 51

5.7 Clean accuracy & certified accuracy achieved by PatchCensor and MR+ on **GTSRB** under different adversarial patch sizes **with fine-tuning** 52

5.8 Clean accuracy & certified accuracy achieved by PatchCensor and MR+ on **Food-101** under different adversarial patch sizes **without fine-tuning** 52

5.9 Clean accuracy & certified accuracy achieved by PatchCensor and MR+ on **Food-101** under different adversarial patch sizes **with fine-tuning** 52
5.10 Clean accuracy & certified accuracy achieved by PatchCensor and MR+ on **ImageNet** under different adversarial patch sizes **without fine-tuning** 53
5.11 Clean accuracy & certified accuracy achieved by PatchCensor and MR+ on **ImageNet** under different adversarial patch sizes **with fine-tuning** 53
5.12 Choosed ViT variants for latency measurement. The base model is what we use for evaluation in RQ1, RQ2, and RQ3. 56

6.1 Adding additional information can improve the performance of PatchSmooth. 62

List of Figures

1.1 Adversarial attack can deceive DNNs with imperceptible small perturbations [8]. 2
1.2 Example of adversarial patch attack [14]. 4
1.3 Adversarial patch can influence the decision-making of autonomous driving [12]. 5
1.4 Mechanism of Vision Transformer (ViT) [20]. 6
1.5 Illustration of certified defense 10

2.1 Example of PatchSmoothing inference 21

3.1 Illustration for image rescaling. The left one is the original image, and the right one is the image after rescaling. 26
3.2 Rescale experiment on De-randomnized Smoothing. The X-axis denotes the difference in value between the top two votes, while the Y-axis represents the frequency of samples yielding the voting result. 27
3.3 Rescale experiment on PatchGuard. The X-axis denotes the difference in value between the top two votes, while the Y-axis represents the frequency of samples yielding the voting result. 28
3.4 Confusion matrix for test instances that are unable to certify but predicted correctly. Y-axis indicates the ranked-1 class and X-axis indicates the ranked-2 class 28
3.5 Illustration for the input used by certified recovery (DS) and PatchCensor. Certified recovery approaches are based on small local regions. Our method is based on the whole image with a small region occluded. 29

4.1 The overall architecture and workflow of PatchCensor. 33
4.2 Number of ViT patches affected by different sizes of adversarial patches. P and W_{adv} represent the widths of the ViT input patch and adversarial patch respectively. 36

5.1 Illustration of the annotation 46

5.2 Certified accuracy and accuracy in the robust domain of PatchCensor, PatchGuard and De-randomized Smoothing on rescaled CIFAR-10. . 47
5.3 Certified accuracy and accuracy in the robust domain of PatchCensor, PatchGuard and De-randomized Smoothing on ImageNet 47
5.4 Certified accuracy and accuracy in the robust domain of PatchCensor, PatchGuard and De-randomized Smoothing on PartImageNet 48
5.5 Certified accuracy and latency for ViT variants of different model sizes 56
5.6 Certified accuracy and latency for ViT variants of different input and patch sizes 56

6.1 An illustration that certified detection may fail at test time - the method may return 'certified' for attacked images. 60
6.2 A real example that can "crack" the certified adversarial patch detection. 61

7.1 Our masking strategy works on abstract feature level and can be applied to both images and text. 64

Abbreviations

AI	Artifical Intelligence.
CNNs	Convolutional Neural Networks.
CV	Computer Vision.
DL	Deep Learning.
DNNs	Deep Neural Networks.
GTSRB	German Traffic Sign Recognition Benchmark.
IBP	Interval Bound Propagation.
LLM	Large Language Model.
NLP	Natural Language Processing.
RNNs	Recurrent Neural Networks.
RQ	Research Question.
SE	Software Engineering.
SOTA	State-of-the-Art.
ViT	Vision Transformer.

Glossary of Terms

Area of Interest (AOI) Area of Interest (AOI) refers to a specific region containing the target object of interest.

Black-box Black-box refers to a system or device whose internal workings are hidden from the user and cannot be easily understood or modified.

Certification We use this term with "verification" interchangeably. Certification denotes a formal and rigorous process of verifying that the target system possesses a specific property.

Classifier This term refers to a machine learning model employed to determine the class to which a given input belongs.

Full-coverage Testing Full-coverage testing is a software testing technique that aims to test all possible inputs and execution paths of a program.

Mutation In software engineering, the term mutation typically refers to a small intentional change made to the source code. In this paper, we employ the term mutation to denote the systematic modification of the input data.

Chapter 1

Introduction

1.1 Background

1.1.1 Digital Adversarial Attack

In 1989, a well-known paper demonstrated the great potential of artificial neural networks by proving that they are theoretically universal approximators [1]. The rapid development of computational capability soon makes it a reality. As computing power grew exponentially, it became increasingly feasible to train and deploy complex deep neural networks (DNNs). Over the last decade, we have witnessed DNNs revolutionize many domains with their superior pattern recognition ability. Learning complex relationships from vast amounts of data and achieving human-level performance across modalities makes it possible to implement complex intelligent systems. Today, DNNs are the key components of many real-world applications, including computer vision, natural language processing, and robotics.

As Deep Learning (DL) techniques continue to advance, there is also a growing trend to deploy DL models in safety-critical scenarios, *e.g.*, intruder detection on surveillance cameras [2], face authentication on smartphones [3], and driving assistance on autonomous cars [4]. Nonetheless, the astonishing learning capability of DNNs also comes with drawbacks. It is well-known that the activation function inside DNNs introduces non-linearity, which is the key property to the success of DNNs but also makes them highly sensitive to input perturbations [5]. This sensitivity, on the one hand, makes it hard for researchers to understand the real underlying mechanism of DNNs, and on the other hand, leaving room for adversarial behaviors. The latter is formalized as adversarial attack [5–7] where related work found that it is possible to modify the prediction of image classifiers using only imperceptible perturbations. Such perturbations can deceive DNNs into confidently identifying an image of a panda as a gibbon (as shown in Fig.1.1).

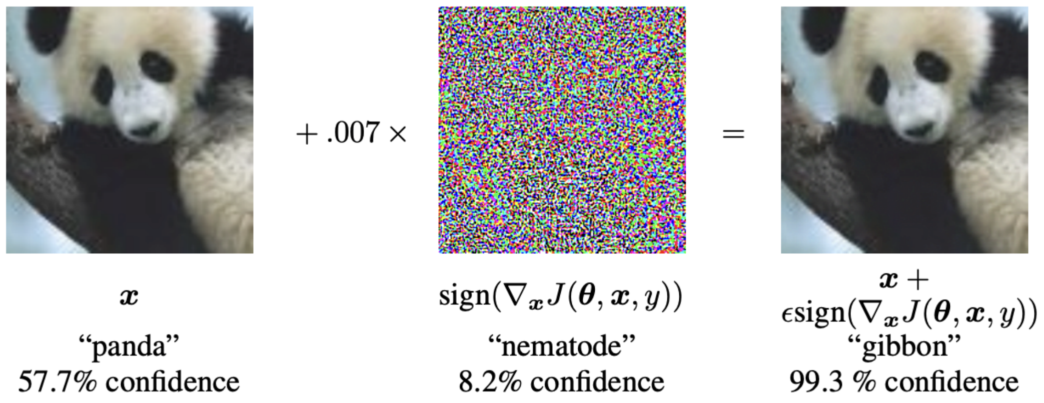

$+ .007 \times$
$=$
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Figure 1.1: Adversarial attack can deceive DNNs with imperceptible small perturbations [8].

Such adversarial attack can be formally defined as an optimization problem [9]:

$$X_* = X + argmin_\delta\{||\delta|| : F(X+\delta) \neq F(x)\} \tag{1.1}$$

Where X stands for the input, F stands for the DNN, and δ represents the perturbation. The goal of attackers is to identify the minimum perturbation that can change the prediction of neural networks. If there is a specific target to alter the prediction, the attack can be categorized as a targeted attack; otherwise, it is considered an untargeted attack. The magnitude of the perturbation δ is generally constrained by the L_p norm, which quantifies the imperceptibility of the perturbation mathematically. The choice of norm constraint can significantly affect how the optimization is performed, and the most popular constraints are the L_0, L_1, L_2, and L_∞ norm. This whole-image, norm-constrained attack is later categorized as the digital attack, and how to improve the corresponding model robustness has received much attention ever since.

1.1.2 Physical Adversarial Attack

More recently, scholars have discovered that executing attacks with small perturbations applied to the entire image in real-world scenarios is difficult. There are four reasons for this: Firstly, noisy physical environments in practical situations can distort digital attacks' perturbations [10]. Secondly, camera sensors' imperfections may cause them to miss the small-scale perturbations [11]. Thirdly, in the real world, it is challenging to execute an attack that modifies the whole background [11]. Fourthly, printing perturbations in small magnitude is less effective [11, 12]. Due to these practical issues, physical attacks [11–17] have been proposed, which is considered to be a real threat and presents a more practical attacking scenario than digital attacks.

The first attempt at physical attacks was to place accessories, such as glasses, on one's face to deceive state-of-the-art face recognition systems into identifying the person as someone else [13]. Later, Brown *et al.* [14] introduce a special kind of physical attack as a patch format, as shown in Fig.1.2. The generation of adversarial patch can be formally defined as:

$$\hat{p} = argmax_p E_{x \sim X, t \sim T, l \sim L}[logPr(\hat{y}|A(p, x, l, t))] \tag{1.2}$$

Here, X are images in the training set, T represents the distribution over transformations of the patch, and L denotes the patch's location in the image. $\hat{p}$ is the generated patch, and the target of the attack is to maximize the probability of a given class $\hat{y}$ (hence this is a targeted attack) across all images, with the relatively high successful attack rate under different possible real-world transformations (*e.g.*, scale, rotation, light condition, *etc.*). The patch size is often an important indicator of how strong the attack is, measured by the proportion of the original image. The

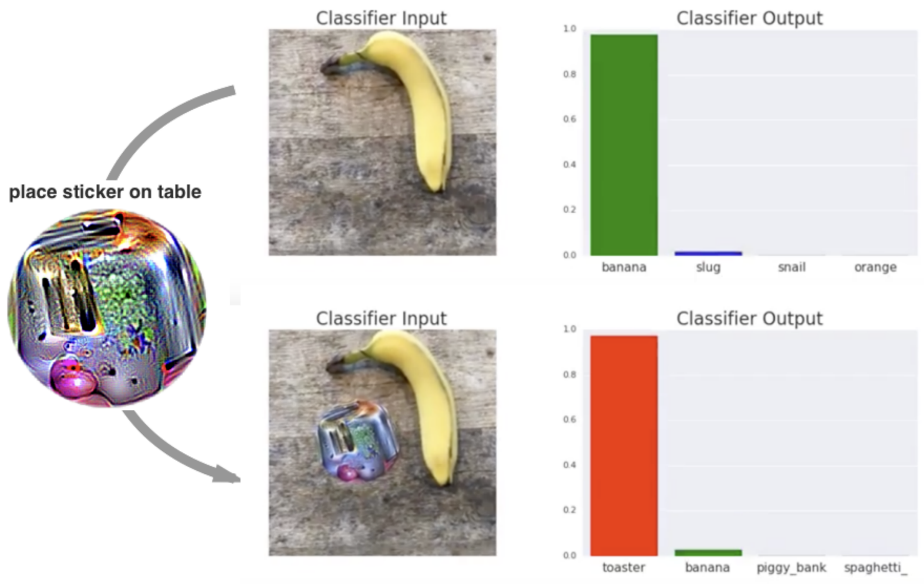


Figure 1.2: Example of adversarial patch attack [14].

attackers' aim is to find a valid patch with a given size to fool the model.

Related patch attacks may relax the constraint here, with an adversarial patch only corresponding to a single instance (instead of an universal patch used for all images in [14]). Still, the adversarial generation process can be roughly summarized as finding a restricted size patch with no magnitude constraint on any position of the input image, so the prediction of the victim DNN will be altered. Following such definition, researchers have discovered that an attacker can deceive the DNN in a self-driving vehicle by affixing a printed sticker onto road signs [11] or bypassing facial authentication by presenting a customized object in front of the camera [18]. An example of such attack is demonstrated in Fig 1.3. Given the challenge posed by physical patch attacks, appropriate defenses are in need. However, since the patch attack has a completely distinct attack model from digital attacks, it is often difficult to directly apply defenses designed for digital attacks. Therefore, it is crucial and pressing to devise a defense mechanism against patch attacks specifically.

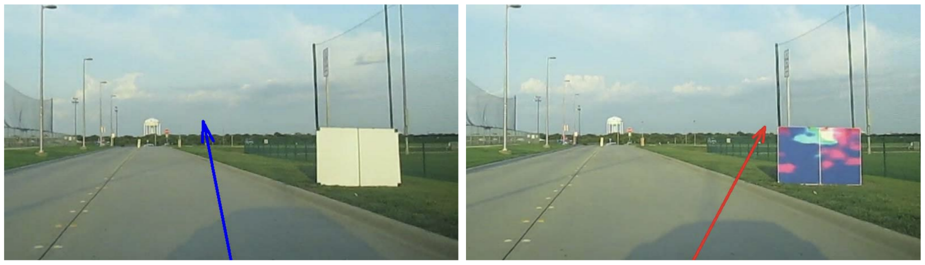


Figure 1.3: Adversarial patch can influence the decision-making of autonomous driving [12].

1.1.3 Transformer

Most adversarial attack research describes neural networks as abstract functions that map input features to output predictions, omitting the actual architecture in mathematical formulations. However, in practice, architecture is a crucial factor when implementing adversarial attacks. For example, convolutional neural networks (CNNs) and recurrent neural networks (RNNs) both have unique computation operators, and their robustness against adversarial attacks may have obvious differences in some cases. The former heavily utilizes convolutional operations, which process the entire input of an instance at one time, while the latter learns the dynamic temporal relation along the timeline, giving output for every point of a sequence. As such, deciding which model architecture to study in particular is important.

This research primarily focuses on a recent state-of-the-art (SOTA) model architecture, the Transformer [19, 20], which is the core of several cutting-edge applications, including Large Language Model (LLM) in the text domain and Vision Transformer (ViT) in the vision domain. We focus on ViT since the adversarial attack research is more mature for computer vision (CV) tasks.

ViT is a model architecture inspired by the success of self-attention-based architectures, particularly the Transformer. In ViT, an image is divided into patches and converted to a sequence of linear embeddings of these patches, which is then fed into a Transformer as the input. Similar to tokens in an NLP application, image patches are treated equally. Recent studies [21–23] have demonstrated that the Transformer architecture can achieve SOTA performance on most CV tasks. The mechanism of ViT is shown in Fig.1.4.

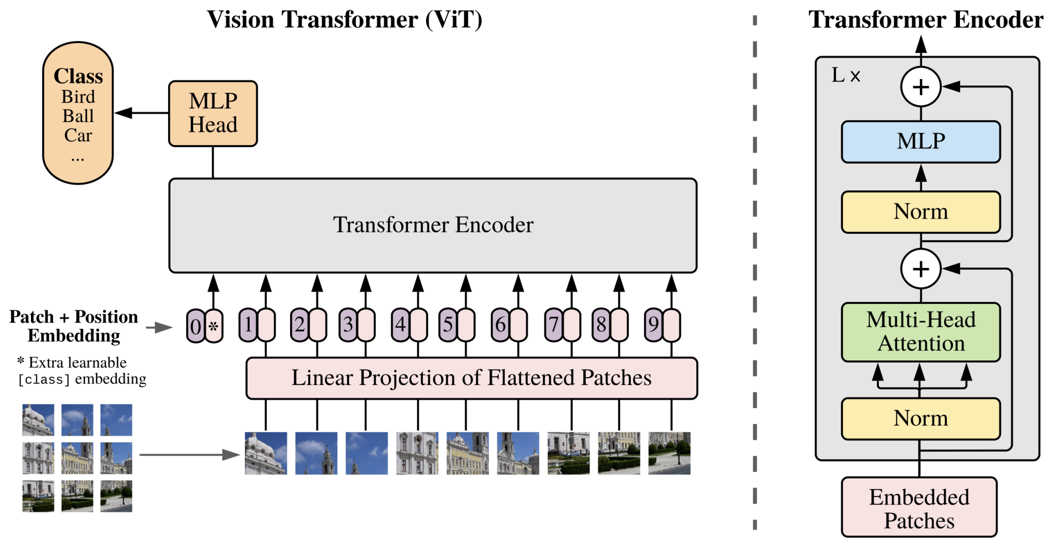


Figure 1.4: Mechanism of Vision Transformer (ViT) [20].

Despite ViT's remarkable success, its resilience against adversarial attacks remains uncertain. According to recent studies [24–26], ViT exhibits robustness to occlusion and natural perturbations. However, it is vulnerable to adversarial attacks [27–30] because of its attention mechanism. Bai *et al.* [27] suggest that transformers are not more robust than CNNs in defending digital-domain adversarial perturbations, whereas Gu *et al.* [28] observe that ViT is considerably more susceptible to physical-domain adversarial patches. Furthermore, Fu *et al.* [30] introduce Patch-Fool, an attack framework that targets ViT's self-attention mechanism specifically. They also report that, in this scenario, ViTs perform worse than CNNs, emphasizing the need to enhance ViT's robustness.

1.2 Defense of Adversarial Attack

This section presents the problem definition for defense of adversarial attack. The major goal of this work is to design a defense mechanism that could improve the robustness of AI-enabled systems so their vulnerability with respect to adversarial attack could be alleviated in a real-world deployment.

The issue of adversarial robustness has been extensively studied in both the Software Engineering (SE) community [12, 31–41] and the Artificial Intelligence (AI) community [42–55] through means of training, testing, and verification. In digital-domain adversarial defense work, a model is considered robust if its prediction remains unchanged in the presence of an adversarially generated *global* L_p-bounded perturbation (where $p = 0, 1, 2, \infty$, and so on). Mathematically, the robustness property can be represented by:

$$\forall \delta. \, ||\delta|| \leq R \rightarrow F(X + \delta) = F(X) \tag{1.3}$$

This corresponds to the formula 1.1 where the predictions of DNNs remain unchanged in the neighborhood of the input X. The range of the neighborhood is categorized by the threshold R, where a higher value indicates greater robustness.

1.2.1 Empirical Defense

There are numerous approaches to improve models' robustness against digital attacks, with one of the most prominent being adversarial training [56]. In this technique, the training loss function is formulated as follows:

$$min_{\theta} \rho(\theta), \text{where } \rho(\theta) = E_{(x,y)\sim D}[max_{\delta \in S} L(\theta, x + \delta, y)] \tag{1.4}$$

where L is the original loss function computed on perturbed input $x + \delta$. The magnitude of δ is restricted and in the above formula the allowed perturbations are denoted as $S \subseteq \mathbb{R}^d$.

Intuitively, the adversarial training first finds the point in the neighborhood of an input that the model performs worst, which is the goal of adversarial attack (inner max), and subsequently optimizes the model's parameters to minimize the loss at that point (outer min). This is the famous min-max equation, and it is the core of adversarial training. Although adversarial training could greatly improve robust accuracy (*e.g.*, from 3.5% accuracy to 45.8% [57]), the normal performance, as well as the model's generalizability, could be substantially reduced [57, 58]. As a result, there is an important tradeoff between models' robustness and performance.

Beyond adversarial training, various empirical defense techniques exist in AI community, including gradient hiding [59], which makes it harder for attackers to solve the optimization in equation 1.1, defense distillation, which aims to train a surrogate model which is harder to be fooled, and feature squeezing, which aims to smooth the input and filter out the influence made by the adversary. However, these heuristic approaches are not capable of ensuring robustness against unforeseen adaptive attacks [47]. All these empirical defense has been broken by stronger attack [60–62]. This dilemma is well-known as Cat and Mouse Game in adversarial attack research, where attackers constantly develop new methods to evade defenses, and defenders constantly adapt to catch up with the latest attacks. It is a back-and-forth struggle between the attacker and the defender.

Improving the quality and robustness of DNNs, as well as guaranteeing their correctness during deployment, is also an important problem within the SE community. Related efforts can be roughly classified into two types : (1) those that seek to evaluate the quality of DNNs before deployment by generating test inputs and (2) those that aim to understand model behaviors with respect to individual input in the deployment stage and present a warning when the system may fail on the input.

The motivation behind the former is that testing data are costly and frequently inadequate for assessing the quality of DNNs. Consequently, some testing techniques have been devised to generate test inputs that can more thoroughly test DNNs [12, 63–68]. This generation process can be guided by some DNN coverage criteria, such as those proposed in DeepXplore [69] and DeepGauge [70]. Once the inputs that DNNs are incapable of processing are identified, model developers can apply repair methods accordingly (*e.g.*, fine-tuning on the error inputs).

Despite the fact that the above methods have been proven successful in many applications, they still require prior knowledge of potential failure cases and thus fall into the category of empirical defense to some extent.

The latter approach aims to explore the limits of a DNN's capabilities with a single input. This analysis of abnormal model behavior seeks to determine when the model might produce erroneous predictions [31–37] and warn the users accordingly. The problem of detecting adversarial inputs is typically formulated as a binary classification task, where the goal is to classify inputs as either natural or adversarial. Various approaches have been proposed to address this problem. One approach involves utilizing a separate machine learning model, such as a deep neural network (DNN), to directly analyze the input and classify it as natural or adversarial [71–73]. Alternatively, some methods extract features from the DNN and use them for detection [74–76]. Another line of research involves analyzing the internals of DNNs to gain insight into their behavior and use this knowledge to develop detection methods [39, 77, 78]. Finally, some techniques focus on the input itself and attempt to detect adversarial inputs through input reconstruction [79, 80] or feature squeezing [81]. However, such empirical attacks are not reliable in safety-critical scenarios and may fail against strong adaptive attacks [47, 82].

1.2.2 Certified Defense

Another line of research seeks to solve the adversarial defense problem once and for all. The related work is called *certified defense*. This approach is based on overapproximation, wherein, under a specified attacker model, the adversary is assumed to possess the maximum capability (*e.g.*, the ability to alter the prediction by modifying just one pixel), which is often extremely hard (if not impossible) to have. If the target model can still be robust under the circumstance, we could say this model is adversary-free. Otherwise we claim that we are uncertain about models' robustness.

An illustration is shwon in Fig. 1.5.

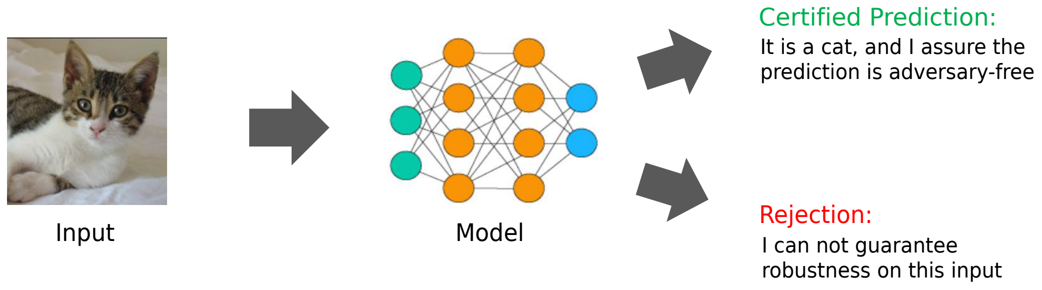

Input
Model
Certified Prediction:
It is a cat, and I assure the prediction is adversary-free
Rejection:
I can not guarantee robustness on this input

Figure 1.5: Illustration of certified defense

The neighborhood of one input has infinite points to cover, and the computation models of DNNs are highly complex. Because of this, providing a certification after enumerating all cases is an impossible mission. To tackle this issue, abstraction and overapproximation are the key ideas in all related work. Abstraction makes infinite points into finite states, and overapproximation simplifies the computation so the black box can be analyzed. As such, an efficient method for computing the overapproximation over abstractions for verification is crucial in certified defense. Related work needs to present a carefully designed test such that when the target model with respect to one particular input passes the proposed test, the model is said to be *surely robust* in the neighborhood of the input. Such certification generally requires rigorous analysis. After designing the test, developers can design an adversarial training method to train a model so its loss under the test can be minimized.

Early attempts on the certified defense of digital domain adversarial attacks rely on traditional software engineering techniques, such as building a linear programming model for DNNs and leveraging satisfiability modulo theories (SMT) solvers to prove the properties of DNNs [83] or reasoning the models' robustness through abstract interpretation [84]. However, these methods are computationally heavy and only capable of small DNNs (approximately 100 to 50000 neurons) with a large amount of time (days to weeks). There is still a huge gap to real-world SOTA models (with billions of neurons).

Later, two other certified defense methods were developed that can scale to relatively larger models. One approach is based on the observation that the non-linear nature of DNNs is the primary obstacle to performing certification computations. To tackle this issue, researchers find it possible to apply linear relaxation for non-linear components (*e.g.*, activation function). After such relaxation, it is possible to propagate input as an interval instead of a single point. Recall from equation 1.3 the robustness property states that the model is robust to a given input if its prediction remains unchanged in the neighborhood of that input. By forwarding an interval instead of a point through DNNs, we are able to verify whether the property holds. This method is called interval bound propagation (IBP), where the propagation is defined mathematically as:

$$\bar{z}^{(k)} = W^{(k)}\frac{\bar{z}^{(k-1)} + \underline{z}^{(k-1)}}{2} + |W^{(k)}|\frac{\bar{z}^{(k-1)} - \underline{z}^{(k-1)}}{2} + b^{(k)} \tag{1.5}$$

$$\underline{z}^{(k)} = W^{(k)}\frac{\bar{z}^{(k-1)} + \underline{z}^{(k-1)}}{2} - |W^{(k)}|\frac{\bar{z}^{(k-1)} - \underline{z}^{(k-1)}}{2} + b^{(k)} \tag{1.6}$$

In the above equations, $z^{(k)}$ is the output of kth layer in the DNN, $\bar{z}^{(k)}$ and $\underline{z}^{(k)}$ correspond to the upper bound and lower bound of $z^{(k)}$. $W^{(k)}$ is the weight and $b^{(k)}$ is the bias of kth layer of the DNN. Such computation makes it possible to estimate the output range given an input interval. If the lower bound of a particular class c is higher than the upper bound of any other classes, we can claim that the prediction of this model will remain unchanged in the input interval. This method can successfully verify an image classifer trained on CIFAR-10 dataset [85]. Nonetheless, this approach has some limitations. Linear relaxation must be implemented for each type of non-linear component, and the quality of the relaxation will significantly impact the final certification outcome.

Another approach takes a probabilistic perspective, aiming to determine the lower and upper bounds of each label's probability via statistical inference. This method is called random smooth, which originates from pixelDP [86], which connects differential privacy with model robustness. Later, Cohen *et al.* [87] provided a more formal definition of the problem with rigorous proof. Roughly speaking, it first adds Gaussian noise to one input image and repeats this process thousands of times. These generated images are sent to the classifier, and statistical inference is performed based on the results. If the lower bound of a class probability is larger than the upper bound of any other classes, the classifier is said to be robust for this input. More images will yield a more accurate estimation. The added Gaussian noise can be regarded as a form of feature squeezing, while the statistical inference provides a rigorous guarantee for the models' robustness. However, this method is also computationally heavy, with nearly 1000x to 10000x overhead because of the repeated inference.

While certified defense against digital domain attack is relatively mature, related study on patch robustness is still in the nascent stage. Although, as we mentioned in section 1.1.2, the physical attack in the patch format is a real threat to the deep learning system. This threat is especially urgent for Transformer as related papers claim their adversarial patch robustness is even weaker than CNN [27–30].

Keen readers may ask whether the digital domain certified defense could be applied to patch attack. The IBP method could be transferred to the patch robustness. However, it will be harder to do the propagation as shown in equation 1.5 and equation 1.6, and we will show later in the related work section this method could perform poorly in certifying patch robustness. For the random smooth, the way it adds noise to the entire image makes it naturally improper to defend the attack in a region with no magnitude restriction.

To address the threat model of patch attacks, researchers have developed several heuristic defenses to improve patch robustness [42, 43]. However, they also suffer from the Cat and Mouse dilemma and could be broken in the future. In contrast, certified defense assumes that attackers possess the utmost capability of a particular type of attack, and if the defense is still effective against such an attack, it is theoretically invulnerable to adversaries and is resilient to unforeseen attacks. Several provable defenses have been recently introduced for adversarial patches. We will discuss them in detail in the related work section. The majority of these defenses [48–51] aim to develop a model that can ensure consistent predictions on images, or determine the presence of adversarial patch. These approaches called **certified recovery**, are based on the insight that prediction can be made by combining local features extracted from small, independent receptive fields. Certified robustness is attained by guaranteeing that the aggregated outcome is not controlled by a small number of possibly adversarial receptive fields. Yet, this secure aggregation paradigm comes at a cost. Related methods seriously compromise the models' accuracy because of the lack of global features. A recent study [48] in this category was able to attain a certified accuracy of only 26.0% on ImageNet [88] with 2%-pixel adversarial patches.

To this end, we raise the major research question of this work:

Is there an efficient way to certify the patch robustness of a complex transformer architecture without much loss on models' performance?

1.3 Our Method

Preserving robustness can be extremely difficult when the perturbation is an unrestricted patch. Models' performance can drop to nearly zero when adding the digital, imperceptible perturbations. It could be more severe when the perturbation is actually visible to human eyes. Furthermore, since DNNs are highly non-linear, the effects of a patch of limited size will quickly propagate to all neurons.

Rather than attempting to make imprecise predictions in the presence of abnormal patches, we argue that it may be more reasonable to alert the user if the input is potentially harmful. For instance, consider a scenario where a DNN is used for medical imaging. In such a case, it would be desirable for the model to warn the doctors that the model is incapable of providing a truthful prediction and it requires human intervention. **Certified detection** is specifically designed for this purpose. Several approaches, such as Minority Reports Defense [53], PatchGuard++ [54], and

ScaleCert [55], aim to detect the presence of an adversarial patch by partially occluding the image around each candidate patch location and analyzing the predictions of all occluded images. However, the Minority Reports Defense is not scalable to high-resolution images like ImageNet due to the challenges of training and the heavy computation overhead it incurs. PatchGuard++ and ScaleCert rely on CNN backbones with small receptive fields, such as BagNet [89], which cannot effectively leverage global features. Therefore, obtaining accurate and scalable robustness certification for complex transformers against adversarial patches remains an open problem.

To this end, we present PatchCensor, *a testing-based method for certifying the robustness of DNNs against adversarial patches that can scale flexibly to complex images of varying sizes.* PatchCensor falls within the category of **certified detection** and aims to alert the user when an abnormal patch appears. As a testing-based method, PatchCensor is zero-shot and directly reuses a pretrained ViT backbone for certification. Furthermore, our approach can be used to detect corrupted natural patches by providing a provable guarantee for worst-case patch attacks.

The reason for PatchCensor being explicitly designed for the Transformer architecture is that:

- The Transformer architecture has demonstrated SOTA performance in tasks related to both Natural Language Processing (NLP) and Computer Vision (CV).
- There is a need to defend Transformers, as suggested from recent studies [28] that they are vulnerable to physical-domain adversarial patches
- Prior work has has indicated that neural network architectures based on Transformers are resilient to the absence of blocks [24, 26], making it more reasonable to design occlusion rules that explicitly mask some blocks.

In the remainder of this paper, we will concentrate on the Vision Transformer (ViT) despite the fact that our defense strategy is intended for Transformers and functions at the abstract feature block level. There are two reasons for this choice: firstly, the majority of relevant research has been designed and assessed in the context of visual tasks, and thus it is more sensible to compare our method with these works in the same domain. Secondly, both adversarial patch attacks and the corresponding defenses are formally defined in the context of visual tasks rather than in the natural language domain.

The ViT architecture divides the input image into small patches, which are subsequently processed by several transformer encoder layers that exchange local information by attending to one another. The final global feature is obtained following the transformer layers and is used to classify the entire image. The attention mechanism employed in ViT allows for an efficient approach to exclude local patches from an inference pass, whereby a patch can be removed by masking the attention of other patches directed towards it. The remaining patches can still interact with one another, thereby obtaining adequate global information for accurate prediction.

PatchCensor examines the input image to determine whether an adversarial patch can control its prediction by testing it with various mutations. Each mutation excludes a region of the image by modifying the attention mask of the ViT, and we verify whether the mutated inference can generate the same result as the original prediction. Exhaustive testing is performed by examining all possible locations of the adversarial patch, and an input is deemed robust if all tests produce a consistent prediction. By designing an optimal strategy for generating the attention mask, PatchCensor can achieve full test coverage with a minimal number of mutations. The robustness certification of PatchCensor is achieved by subjecting samples from a given data distribution to exhaustive testing. Similar to most certification approaches [86, 90], the certified accuracy is calculated as the ratio of correctly predicted samples that are certifiably robust.

1.4 Contribution

In summary, this paper contributes in the following ways:

- The primary contribution of this paper is the introduction of a testing-driven certified defense mechanism against adversarial patch, which leverages the unique characteristics of Transformer-based deep neural network (DNN) architecture.
- We implement and evaluate our proposed method on widely-used datasets, including CIFAR-10, GTSRB, Food-101 and ImageNet, and demonstrate that it achieves SOTA certified accuracy on these datasets.
- Furthermore, we discuss the fundamental distinctions between certified recovery and certified detection, which elucidates the better practicality of our approach.

To the best of our knowledge, PatchCensor represents one of the early attempts to explore the certification of machine learning software security from the perspective of the Transformer architecture and exhaustive testing. We demonstrate that this approach shows considerable promise and practical usefulness compared to other SOTA techniques, which could inspires future research aimed at enhancing the security of data-driven intelligent software.

1.5 Organization of The Paper

In the introduction section, we provide the necessary background information, define the problem, and briefly overview our proposed method. The related work section provides a more comprehensive overview of existing certification techniques for patch robustness. In the preliminary study, we highlight the limitations of current certified recovery approaches. The methodology section presents a formal description of our method. In our evaluation, we apply PatchCensor to commonly-used datasets, including CIFAR-10, GTSRB, Food-101 and ImageNet, and evaluate its performance under adversarial patch sizes ranging from 0.4% to 25%. Our experimental results demonstrate that PatchCensor surpasses existing certified defense methods in both clean accuracy and certified accuracy metrics. We also compare the advantages and disadvantages of PatchCensor with certified recovery approaches to determine whether we have mitigated the issues identified in the preliminary study. We present empirical evidence of the superior ability of PatchCensor in handling images with small regions of interest (ROIs). Moreover, we examine the difference between the statistical certificate obtained by PatchCensor and traditional certificates, a topic that has not been extensively explored in previous literature. Finally, we conclude our work and outline potential directions for future research.

Chapter 2

Related Work

In this section, we discuss related certified defense work in detail. The primary objective of certified defenses is to ensure the robustness of a given model against patch attacks in a rigorous manner.

The primary objective of certified defenses is to ensure the robustness of a given model against patch attacks in a rigorous manner.

Notably, for model robustness we have discussed so far, we mainly discuss adversarial robustness. However, in reality, there is also robustness against natural perturbation. As research continues to explore the robustness of machine learning models against adversarial distribution shifts, efforts are also being made to investigate the natural distributional robustness of deep neural networks (DNNs). The motivation behind this research is the recognition that the worst-case adversarial perturbation is unlikely to occur in the real world, whereas the corruptions commonly encountered in natural environments can have a substantial impact on model performance.

Natural corruption can affect either the entire image or a limited patch region, similar to the adversarial scenario. For example, motion blur and Gaussian noise [91] can be used as natural perturbations applied to the entire image. Alternatively, natural perturbations confined to a patch can be used, as seen in sudden occlusion in object detection [92] and out-of-distribution objects in the open-set video setting [93].

In contrast to whole-image perturbations, where natural and adversarial perturbations are difficult to analyze in a consistent framework because the latter does not impose constraints on perturbations, it is feasible to unify both types of perturbations in the patch setting. More precisely, when considering a fixed patch size, if the model's robustness against the worst-case adversarial scenario is assured, then safety against natural perturbations can also be guaranteed.

At a high level, existing certified defenses can be classified into two categories: *certified recovery* [47–52] and *certified detection* [53–55, 94]. Certified recovery aims to recover the correct prediction on images with adversarial patches, while certified detection aims to detect adversaries with a provable detection rate. In the following, we briefly introduce four representative certification workflows that are closely related to ours: three for certified recovery and one for certified detection. It is worth noting that the success of the patch attack arises from the non-linear property of neural networks, where a small local region can impact the prediction of the global input. As a result, these works share a similarity in that they all aim to isolate or block the influence of the adversarial patch through different approaches.

The first work to address certified defense for patch attacks is the IBP method adapted for the patch scenario [47]. It follows the same basic workflow as introduced in Section 1.2.2. However, instead of imposing a constraint in a l_{inf} ball around the input point (the digital-domain defense approach), the constraint is applied to all images obtained by placing a patch over all possible positions of the original input. Mathematically:

$$\mathbb{B}(x_0) = \{A(x_0, p, l) | p \in \mathbb{P} \text{ and } l \in \mathbb{L}\} \tag{2.1}$$

In the equation above, the operator A places the adversarial patch p on a given image x_0 at location l. To verify the robustness property of a neural network, this method applies IBP over all possible locations and aggregates a lower bound of probability for the target class and an upper bound for other classes. This method is thus computationally heavy and yields relatively poor performance (24.9% certified accuracy on the CIFAR-10 dataset with 2.4% patch size). Nonetheless, this work makes certification on patch robustness possible and can serve as a baseline.

An early endeavor in the domain of certified recovery is (De)Randomized Smoothing (PatchSmoothing) [49]. The primary innovation of this technique involves selecting a specific portion of the image as input and conducting multiple inferences throughout the entire image. Two selection strategies are presented in the paper: block smoothing and band smoothing. In the block smoothing approach, a square of dimensions $s \times s$ is used as input, with all $h \times w$ positions enumerated, where h represents the image's height and w its width. When the block's center reaches the image boundary, PatchSmoothing employs a wrap-around mechanism, utilizing pixels from the other three corners if the center is situated in one corner. Subsequently, the inference results for all squares are tallied, and the difference between the top two classes is computed. If for these two classes we have:

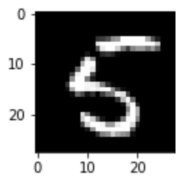
$$\Delta \geq 2(m + s - 1)^2 \tag{2.2}$$

PatchSmoothing proceeds to certify the input with the label of the top-1 class. In this context, Δ denotes the difference, while m signifies the adversarial patch size, and s represents the square size. This threshold is determined by the maximum number of patches the adversary can manipulate. Band smoothing operates in a similar manner; however, it selects a column of size s as the input. The corresponding threshold for this method is expressed as:

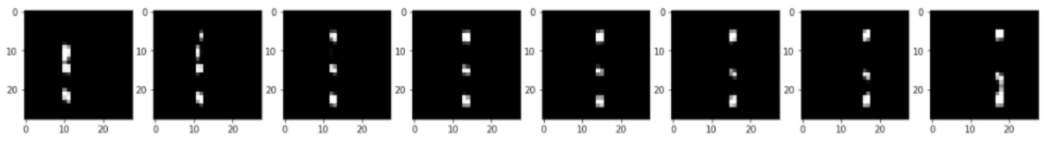
$$\Delta \geq 2(m + s - 1) \tag{2.3}$$

An example for digital number prediction is shown in Fig. 2.1.

PatchGuard [48] advances the field by employing a specific CNN, known as bagnet [95], which possesses small receptive fields. These constrained receptive fields



(a) Input



(b) Input patches sent to model

Figure 2.1: Example of PatchSmoothing inference

serve to limit the impact of adversarial patches. Building on this structure, PatchGuard introduces an element-wise linear aggregation method as a replacement for the original, insecure fully-connected layer, which blends the prediction results from both benign and adversarial patches. The aggregation mechanism functions as follows: It examines the prediction vectors of all sliding windows with a predefined size (adversarial patch size), and if a sliding window with the highest score for a particular class surpasses a predefined threshold, PatchGuard masks that window, employing the remaining windows to predict the class. This approach is motivated by the observation that an adversarial patch significantly increases the score of the target class to surpass the original prediction.

Both certified recovery methods employ a voter mechanism; the former ensures that the majority remains uninfluenced by the adversary, while the latter guarantees that suspicious voters cannot control the outcome. These two parallel methods are examined in detailed in our work because we aim to discuss the distinctions between certified recovery and certified detection, offering both practical considerations and experimental results to support our preference for certified detection.

Conversely, Minority Report (MR) [53] serves as a representative work in certified detection. For each input, MR also conducts multiple inferences, with each masked by an occlusion region of $s \times s$ that is two pixels larger than a predefined adversarial patch. As the occlusion region slides across the image (of size $h \times w$), MR collects results for $(h-s+1)(w-s+1)$ inferences. MR certifies the result only if all predictions yield the same class. In their implementation, however, this requirement is relaxed by initially discarding the lowest score to accommodate outliers and subsequently providing certified results when the supported votes surpass a predefined threshold. This relaxation may result in false negatives but improves the overall performance.

The primary emphasis of this paper is on certified detection. Our proposed system, PatchCensor, distinguishes itself from existing certified detection methods in three distinct aspects:

- Firstly, PatchCensor is founded upon a comprehensive testing strategy without the need for additional efforts to train a model that exhibits robustness against adversarial patches.
- Secondly, the certification process of PatchCensor operates on a range of patch sizes rather than a single fixed size, offering increased flexibility in its application.
- Lastly, PatchCensor is applicable for patch sizes up to 25% (whereas most related work is evaluated on 2%-pixel attacks), rendering it more practical for detecting natural patch perturbations in real-world scenarios.

Chapter 3

Preliminary Study

In this section, our focus lies on examining the potential issue related to the certified recovery approach. The goal of this method is to preserve accurate predictions despite the presence of adversarial inputs. We argue that accomplishing such a goal can be challenging in certain cases. We will first outline the evaluation metrics (which will also be used in following sections), followed by presenting the experimental designs and results.

3.1 Evaluation Metrics

The objective of the defender is to enhance the defended model's quality by improving both clean accuracy and certified accuracy. Let the abstract function of a model as f. Clean accuracy, denoted as acc_{clean}, refers to the accuracy of f on the original dataset $\mathcal{X}$ without considering verification (please note that our clean accuracy definition deviates from previous certified detection work; we report acc_{clean} to enable readers to understand the influence of pre-training on the base model). Meanwhile, certified accuracy, represented by $acc_{\text{certified}}$, corresponds to the images in $\mathcal{X}$ that are both correctly and provably classified. Formally,

$$acc_{\text{clean}} = \mathbf{E}_{x \in \mathcal{X}}[l(f(x), y)] \tag{3.1}$$

$$acc_{\text{certified}} = \mathbf{E}_{x \in \mathcal{X}}[l(f(x), y)\ v(x)] \tag{3.2}$$

where l is the indicator function for correctness. It will return 1 if the result is correct and 0 otherwise. $v(x)$ is the certification result, while 1 indicates certified and 0 otherwise.

$acc_{\text{certified}}$ is usually the most important metric in related study. It can be divided into two components: the proportion of inputs that can be verified, and the accuracy of the model's predictions for those verifiable inputs. The former is defined as:

$$r_{\text{robust}} = \frac{|\mathcal{X}_{trust}|}{|\mathcal{X}|} \tag{3.3}$$

and the model accuracy in the trust (certified) domain is:

$$acc_{\text{in-robust}} = \mathbf{E}_{x \in \mathcal{X}_{trust}}[l(f(x), y)] \tag{3.4}$$

The aim is to maintain high test accuracy and ensure dependability for the majority of inputs ($\mathcal{X}_{\text{robust}}$) while issuing a warning if the input x is potentially malicious ($v(x) = 0$). This concept bears similarity to selective prediction approaches that attempt to incorporate a reject option within neural networks [96]. Such a rejection mechanism proves beneficial in a variety of perception tasks where a fallback solution is available. For instance, in the context of driving assistance models, the human driver may be prompted to take over or execute a conservative action when the model lacks confidence in the current situation.

3.2 Experiment Design and Results

Certified recovery typically relies on a specialized model (e.g., BagNet[95]) that generates predictions using small receptive fields. They require first dividing the input image into smaller regions, each of which is input into the model to produce a local prediction. The defended prediction is determined by voting across all local predictions, and an input sample's certification is achieved if a sufficient number of local predictions vote for the same label, thereby ensuring that an arbitrary adversarial patch will not alter the prediction. In other words, for a given set of $\mathcal{Y}$ classes, an image is certified as $class_A$ if the following condition is met:

$$\begin{aligned} &\text{Vote}_{class_A} - \text{Vote}_{class_B} > threshold \\ &\text{where} \quad \text{Vote}_{class_B} = max_{\forall i \in \mathcal{Y}, i \neq A}(\text{Vote}_{class_i}) \end{aligned} \tag{3.5}$$

Threshold design, which is usually the key, differs between different methods.

While certified recovery is generally anticipated to provide a more rigorous and formal guarantee, we argue that a lightweight and statistical guarantee regarding the presence of an attack may be more applicable in real-world scenarios. The reasons are threefold:

- Firstly, a notable drawback of certified recovery is its limited performance (*e.g.*, a mere 26.0% certified accuracy on ImageNet for a recent work [51]), often rendering such defenses impractical for real-world applications.

- Secondly, the optimization of the model for both clean and adversarial inputs inescapably reduces the model's accuracy across all existing certified recovery approaches, potentially negatively affecting user experience. Nevertheless, attaining moderate certified performance without such optimization proves difficult for current certified recovery techniques. Additionally, this optimization, often realized through re-training or even training from scratch, may take more computational resources and time.

- Lastly, we observe that recovering the correct prediction in the presence of adversarial patches can be difficult or even impossible in certain cases, such as when an adversarial patch conceals the critical region of interest (ROI). Addressing this issue may be particularly challenging for images with a small ROI, where the majority of local voters might need more information to generate the correct prediction.

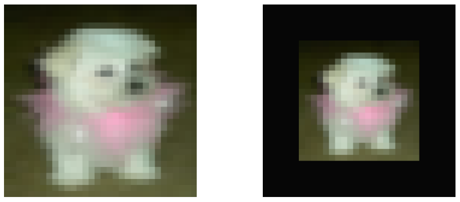
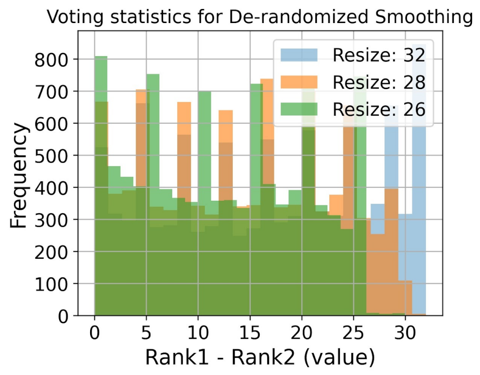


Figure 3.1: Illustration for image rescaling. The left one is the original image, and the right one is the image after rescaling.

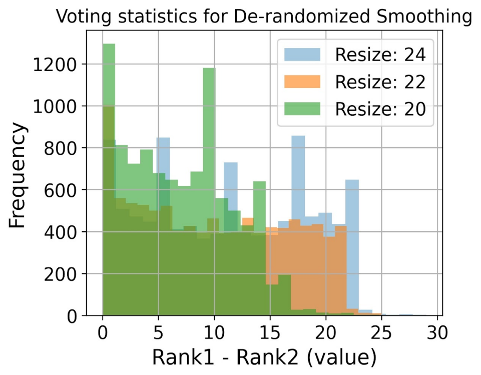
With advancements in camera technology, high-resolution images are becoming increasingly common, resulting in significantly smaller ROIs in practice compared to popular datasets like MNIST and CIFAR-10.

In order to further support our observations and hypotheses, we conduct a preliminary study examining two certified recovery approaches, namely PatchGuard [48] (abbreviated as PG) and De-randomized Smoothing [49] (abbreviated as DS). Specifically, we rescale each image in CIFAR-10 to a smaller size and pad the image to its original dimensions, as demonstrated in Fig. 3.1. This process allows us to control the size of each image's region of interest (ROI). We gradually rescale the image, reducing its size from the original dimensions of 32 to 20 with a 2-pixel interval. Given that the images differ from their original versions after resizing, we need to retrain the models. We train the De-randomized Smoothing ResNet (DS-ResNet) from scratch for 200 epochs, utilizing common settings found in De-randomized Smoothing[49] and PatchGuard [48]. We assess these two methods to certify a 2.4% patch size and record the distribution shift of the voting differences ($\text{Vote}_{Rank1\_class} - \text{Vote}_{Rank2\_class}$). The result for DS is shown in Fig. 3.2, and the result for PG is shown in Fig. 3.3.

It is evident that as the ROI becomes smaller, the margin between Rank-1 voting and Rank-2 voting decreases. If the voting margin for a sample falls below a certain threshold, the model does not provide a certified result for that sample. The threshold is more intuitively understood for Randomized Smoothing (since PatchGuard is based on a weighted sum and the voting value is continuous), in which the threshold is calculated using equation 2.3. For instance, if m (the target size to certify) is 5



(a) Voting distribution on Resize scale from 32 to 26



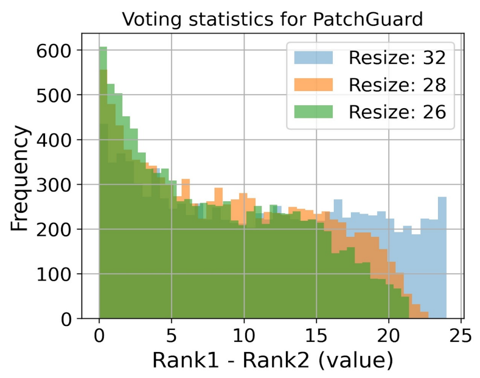
(b) Voting distribution on Resize scale from 24 to 20

Figure 3.2: Rescale experiment on De-randomnized Smoothing. The X-axis denotes the difference in value between the top two votes, while the Y-axis represents the frequency of samples yielding the voting result.

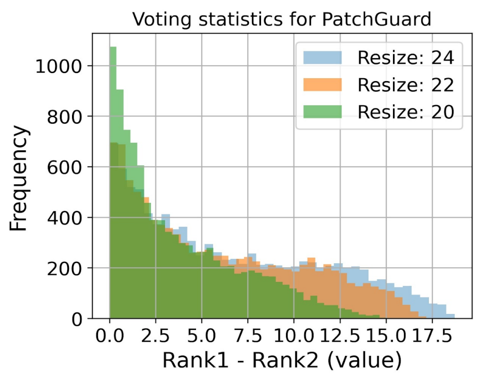
(corresponding to a 2.4% patch size), and s (the column size) is 4 (the default setting), then the threshold is 16. In Figure 3.2b, it is clear that the distribution of De-randomized Smoothing for size 20 has shifted just to the left of 16. Consequently, there is a significant decline in certified accuracy at size 20. Detailed results can be found in Figure 5.2 (refer to section 5).

This scenario of PatchGuard is similar to De-randomized Smoothing, where we can observe a clear distribution shift when the resize scale equals 20, as shown in Fig. 3.3b. Consequently, we can also observe a significant performance drop in Figure 5.2.

We also examine test instances in the CIFAR-10 dataset that are correctly predicted but cannot be certified, demonstrating that relying solely on local features as input may lead to confusion for certified recovery methods. In other words, the difference between the top-2 classes for these instances does not surpass the threshold defined in these algorithms. The confusion matrix for those instances is shown in Fig. 3.4, where the y-axis represents the true class and the x-axis indicates the ranked-2 class. It is evident that for both methods, dog *vs* cat, car *vs* truck, and frog *vs* bird are classes that are challenging to certify. Moreover, these classes can sometimes be difficult to differentiate, even for humans, when given only a small patch of the image, as



(a) Voting distribution on Resize scale from 32 to 26



(b) Voting distribution on Resize scale from 24 to 20

Figure 3.3: Rescale experiment on PatchGuard. The X-axis denotes the difference in value between the top two votes, while the Y-axis represents the frequency of samples yielding the voting result.

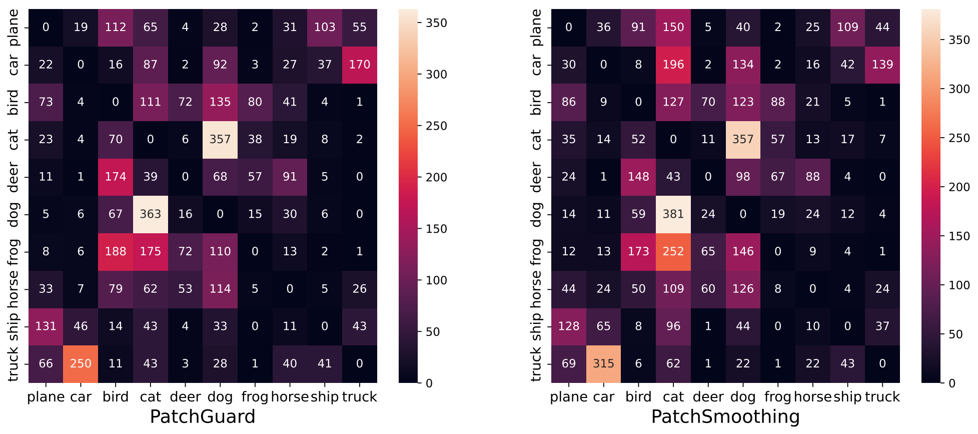


Figure 3.4: Confusion matrix for test instances that are unable to certify but predicted correctly. Y-axis indicates the ranked-1 class and X-axis indicates the ranked-2 class

illustrated in Fig.3.5.

Our preliminary study's findings confirm that certified recovery methods relying on local features may have inherent limitations when dealing with small-ROI images, making it challenging to achieve satisfactory certified accuracy in practice. Conversely, using as many patches as possible during the inference pass can supply sufficient information for the models to make predictions. This can enhance clean accuracy, particularly in the case of a small ROI. However, increasing the number of patches in a single inference will inevitably raise the likelihood of including ma-

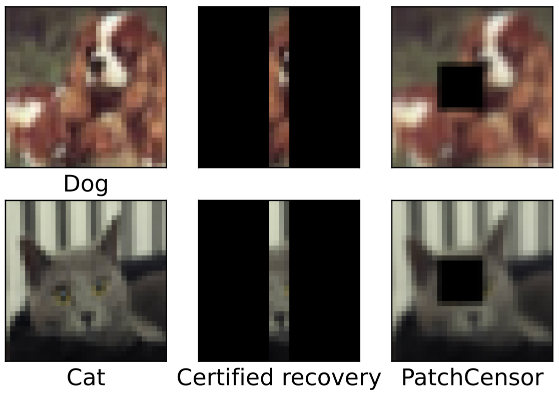


Figure 3.5: Illustration for the input used by certified recovery (DS) and PatchCensor. Certified recovery approaches are based on small local regions. Our method is based on the whole image with a small region occluded.

licious patches and, consequently, increase the defense's difficulty. Our work seeks to address this dilemma by proposing a defense mechanism in a detection-oriented manner, which can utilize more patches to improve certified accuracy.

Chapter 4

Method

This section proposes a certified defense methodology against adversarial patches, which relies on testing and is denoted as PatchCensor. First, the problem is formulated, and the intended properties for certification are formally defined. Subsequently, our approach is introduced, accompanied by a high-level theoretical analysis.

4.1 Problem Formulation

To be general, our work shares the same threat model as existing recent SOTA certified defense work against adversarial patches [47–50] (The attack and defense are both focused on image classification context.). We use $\mathcal{X} \subset \mathbb{R}^{W \times H \times C}$ to denote the distribution of images where each image $x \in \mathcal{X}$ has width W, height H, number of channels C. We take $\mathcal{Y} = \{0, 1, \cdots, N-1\}$ as the label space, where the number of classes is N. We use $f : \mathcal{X} \rightarrow \mathcal{Y}$ to denote the model that takes an image $x \in \mathcal{X}$ as input and predicts the class label $y \in \mathcal{Y}$.

Attacker capability. The attacker has the ability to arbitrarily modify pixels within a constrained region, which may include the target object. For the purpose of our analysis, we assume that all manipulated pixels lie within a square-shaped region, the size of which is conservatively estimated by the defender (*i.e.*, upper bound of the region size). While our proposed technique can be extended to other patch shapes, we have chosen to focus on square-shaped patches for the sake of simplicity. It should be noted that our method is applicable as long as the patch can be contained within a restricted rectangle.

Formally, we assume the attacker can arbitrarily modify an image x within a constraint set $\mathcal{A}(x)$. We use a binary pixel block $p \in \mathcal{P} \subset \{0,1\}^{W \times H}$ to represent the restricted region, where the pixels within the region are set to 1. Then, the constraint set $\mathcal{A}(x)$ can be expressed as $\{x' = (1-p) \odot x + p \odot x''\} | x, x' \in \mathcal{X}, x'' \in \mathbb{R}^{W \times H \times C}, p \in \mathcal{P}\}$, where $\odot$ refers to the element-wise product operator, and x'' is the content of the adversarial patch.

Attack objective. We focus on adversarial patch attacks against image classification models. Given a deep learning model f, an image x, and its true class label y, the goal of the attacker is to find an image $x' \in \mathcal{A}(x) \subset \mathcal{X}$ such that $f(x') = y'$, where y' is an arbitrary incorrect class label defined by the attacker and $y' \neq y$.

We note that the attack objective of inducing misclassification into any wrong class is referred to as an *untargeted attack*. In contrast, when the goal is to misclassify the image to a particular target class $y' \neq y$, it is called a *targeted attack*. Our attack model is defined as untargeted, as it encompasses a broader range of attacks from the defender's perspective. However, it is worth noting that our proposed method is capable of effectively countering both untargeted and targeted attacks.

Defense objective. The role of the defender is to design a defended model $g = (f, v) : \mathcal{X} \to \mathcal{Y} \times \{0,1\}$, where $g(x) = f(x), v(x)$, $f(x) \in \mathcal{Y}$ is the classification result, and $v(x) \in \{0,1\}$ is the verification result indicating whether the prediction $f(x)$ can be verified (1 stands for "verified"). A verified inference means this inference is not influenced by any attacker, and we can trust the prediction made by the model.

The property to be certified. Based on the defense objective, we formulate the property to be certified as follows: Given a model M and a data distribution D, we certify that for any $x \in D$, the prediction $M(x)$ is correct and robust (cannot be falsified by an arbitrary attacker with ability A) with a probability of θ.

The concept of providing statistical guarantees through certification for neural networks is not novel. Numerous certified defense techniques in the digital attack domain [86, 87] offer such assurances (the random smooth, as shown in section 1.2.2). Nevertheless, this formulation has the potential to generate confusion, as the majority of prior certification research related to patch attacks, including certified detection—the central focus of this paper—yield deterministic certification outcomes. We discover that, in actuality, PatchCensor and all preceding certified detection efforts provide a statistical guarantee (discussed in Section 6), and we contend that it is both necessary and crucial to reformulate the problem. Furthermore, we argue that such statistical assurances hold greater practical value in real-world applications, a point to be discussed in more detail later.

The classification and verification result of $\mathcal{D}(x)$ are represented as $\mathcal{D}_y(x)$ and $\mathcal{D}_v(x)$ respectively.

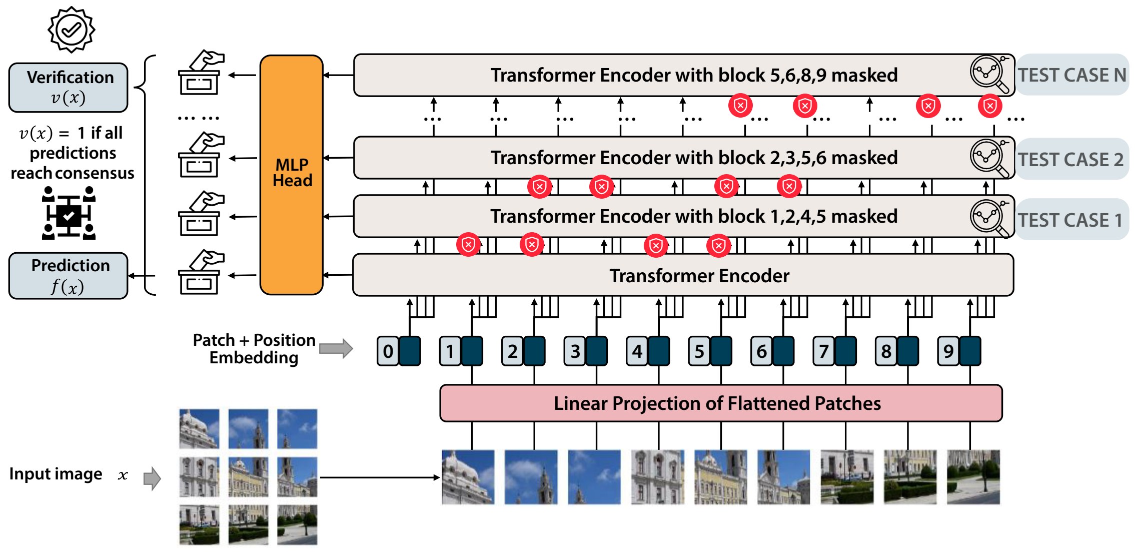
The predictions of PatchCensor should be resistant to adversarial patches in the robust domain, $i.e.$any adversarial example generated by the attacker either is ineffective or can be detected (cannot pass the verification). Formally, for any certified clean data point $x \in \mathcal{X}_{\text{robust}}$ and any adversarial example $x' \in \mathcal{A}(x)$, we ensure either $f(x') = f(x)$ or $v(x') = 0$.

$$(f, g)(x) \triangleq \begin{cases} f(x), & \text{if } g(x) = 1; \\ \text{don't know}, & \text{if } g(x) = 0. \end{cases} \tag{4.1}$$

4.2 The Testing-based Defense

Our adversarial patch defense technique is designed upon the Vision Transformer (ViT) architecture, by utilizing its input partition nature and self-attention mechanism.

The summarized workflow of PatchCensor are shown in Figure 4.1. It can be viewed as a paired function $g = (f, v)$, where f is a pretrained ViT model for prediction, and v is a verification function based on the ViT model. Given an input image x,



Verification $v(x)$
$v(x)$ = 1 if all predictions reach consensus
Prediction $f(x)$
MLP Head
Transformer Encoder with block 5,6,8,9 masked
Transformer Encoder with block 2,3,5,6 masked
Transformer Encoder with block 1,2,4,5 masked
Transformer Encoder
TEST CASE N
TEST CASE 2
TEST CASE 1
Patch + Position Embedding
0
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches
Input image x

Figure 4.1: The overall architecture and workflow of PatchCensor.

PatchCensor produces a pair $(f(x), v(x))$, where $f(x)$ is the predicted class of x and $v(x)$ represents whether the prediction is verified.

In PatchCensor, the input image is split into non-overlapping patches as in vanilla ViT models. Suppose the input patch size is $P \times P$ in pixels, then the input image $x \in \mathcal{X} \subset \mathbb{R}^{W \times H \times C}$ is partitioned into a sequence of patches $\mathcal{P} = \{p_i | i = 1, 2, ..., n\}$, where $n = n_w \times n_h = \frac{W}{P} \times \frac{H}{P}$ is the number of patches. Each patch p_i is then flattened, passed through a linear projection layer, and added a position embedding to generate a patch embedding q_i. A learnable `[class]` embedding q_0 is prepended to the sequence of embeddings, which is used to produce the class prediction after passing through later layers.

Specifically, the patch embeddings are fed into multiple parallel Transformer encoder layers $\mathcal{T} = \{t_0, t_1, ..., t_k\}$ to exchange information between local patches. The encoders share the same weights of the ViT model f, while being used with different attention masks. Each Transformer encoder t_j produces an encoding of the `[class]` node, which is then passed through the MLP head to produce a class prediction y_j. We call the predictions produced with the masked attention maps $(i.e. y_1, y_2, ..., y_k)$ as *masked predictions*.

To test whether there is an adversarial patch, we design a special mutation operator that directly masks the attention maps. The predictions with different masks can be seen as different mutations. The presence of the patch can be detected once we find anomalies among the test results. We call the predictions produced with the masked attention maps ($i.e. y_1, y_2, ..., y_k$) as *mutations*. The class prediction $f(x)$ is produced by the Transformer encoder without mutation ($i.e. y_0$), which is equivalent to a direct inference using the base ViT model. The verification result $v(x)$ is produced by voting over all of the testing results (y_1 to y_k). We ensure that at least one of the mutations is benign (*i.e.*can completely mask the adversarial patch out), which vetoes the adversary's target output even if all other predictions are compromised. The prediction is verified if all Transformer encoders reach consensus (*i.e.*, agreeing on the same class prediction).

As we enumerate all possible positions for the potential adversarial patch in this test generation process before masking, this test can exhaustively cover all corner cases, meaning that we complete a *full-coverage testing*.

Ideally, if testing can be done on all possible situations and these test cases are all passed (producing correct and consistent output), the exhaustive testing would be equivalent to a rigorous verification. However, directly testing all possible adversarial patch positions is computationally infeasible and impractical considering its combinatorial complexity.

Our mask strategy plays a role similar to the abstract set in abstract interpretation [97], providing a convenient (due to the natural robustness of self-attention architecture) and efficient (due to the reduced search space) way to achieve exhaustive full-coverage testing.

As a concrete example, we illustrate our method in Figure 4.1. Suppose we use a ViT model with 30×30 input resolution and 10×10 input patch resolution as the base model.

An input image will be partitioned into 3×3 patches. To defend against adversarial patches with 5×5 resolution, we let each mask exclude 2×2 local patches, *i.e.*, a square region with 20×20 resolution. By sliding the 2×2 mask over all local patches in the image, we can obtain four (2×2) possible mask locations and guarantee at least one of the mask locations can completely hide the 5×5-pixel adversarial patch. A certified prediction is produced if the four masked predictions vote for the same class as the non-masked prediction.

Our technique shares some similarities with Partition Analysis [98], where they first partition the tested domain into sub-domains (just like we partition the input space by enumerating different mask positions) and combine both testing and verification techniques to evaluate program reliability. However, one key difference is that PatchCensor attempts to do full-coverage testing to achieve verification-level guarantee, while *Partition Analysis* complements verification with testing.

4.3 Mutation Strategy

The certification capacity of PatchCensor is realized through testing various mutations, guaranteeing that a minimum of one mutation results in an unaffected prediction (*i.e.*, the associated attention mask can effectively exclude the adversarial patch). This concept bears a resemblance to Byzantine Fault Tolerance [99], in which a benign voter can determine the ultimate outcome.

For inference, each mutation employs a distinct attention mask. The critical aspect of designing these attention masks involves discerning the number of local input patches potentially contaminated by the adversarial patch (i.e., containing pixels from the adversarial patch). Given the fixed input partition plan and the arbitrary location of the adversarial patch within the image, it is necessary to consider the worst-case scenario, which is the maximum number of input patches that could be affected by the adversarial patch.

Our masking strategy is based on the observation that, *in an image that is partitioned into non-overlapping $P \times P$-pixel patches, an adversarial patch with $W_{adv} \times H_{adv}$*

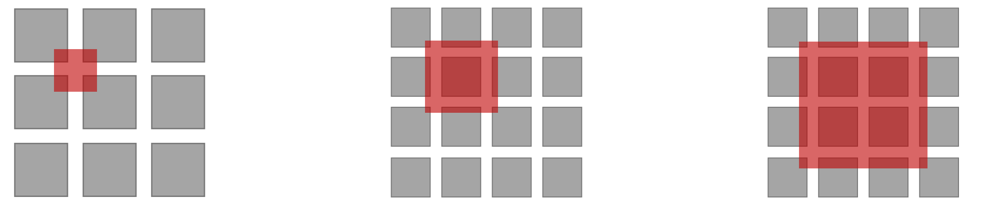

(a) $0 < W_{adv} < P$ **(b)** $P < W_{adv} < 2P$ **(c)** $2P < W_{adv} < 3P$

Figure 4.2: Number of ViT patches affected by different sizes of adversarial patches. P and W_{adv} represent the widths of the ViT input patch and adversarial patch respectively.

resolution can affect at most $N_W \times N_H$ input patches, where $N_W = \lceil \frac{W_{adv}}{P} \rceil + 1$ and $N_H = \lceil \frac{H_{adv}}{P} \rceil + 1$.

For example, as illustrated in Figure 4.2, if the width W_{adv} of a square adversarial patch is smaller than the input patch width P, four input patches may be tainted (if the adversarial patch is at the joint of 2×2 input patches). Similarly, a square adversarial patch with $W_{adv} \in (2P, 3P)$ may affect at most 4×4 input patches.

It is important to highlight that our masking approach guarantees that for any object, irrespective of its location or shape, as long as it can be confined within a rectangle of dimensions $N_W \times N_H$, there must be a masking configuration capable of excluding this object. This adaptability allows for the detection of both adversarial patches and irregular objects in real-world applications, whereas prior research often necessitated fine-tuning the model to accommodate a specific patch size.

Specifically, given the maximum rectangle shape $W_{adv} \times H_{adv}$ of the adversarial patch, we can compute the minimum size $N_W \times N_H$ for the attention mask. By sliding the mask over the whole input patch grid with a stride of 1, we can enumerate all k possible locations of the mask, where

$$k = (\frac{W}{P} - N_W + 1) \times (\frac{H}{P} - N_H + 1) \tag{4.2}$$

We can guarantee that at least one of the k masks can cover the arbitrary adversarial patch.

As we only mask a small proportion of the image and our base model is the powerful ViT, it is relatively easy for all voters to reach a consensus for a correct prediction on clean data.

4.4 Certification Analysis

In this subsection, we provide analysis to show that PatchCensor can achieve the defender's objective.

After obtaining the regular prediction of the ViT base model $f(x)$, the verification result of PatchCensor is obtained by

$$v(x) \triangleq \begin{cases} 1, & \text{if } f_1(x) = f_2(x) = ... = f_k(x) = f(x); \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

where $f_j(x)$ represents the prediction obtained by the ViT base model with the j-th mask position on the attention map.

For any verified clean data point $x \in \mathcal{X}_{trust}$ and any adversarial example $x' \in \mathcal{A}(x)$, we need to ensure the adversarial patch is either ineffective or can be detected. Specifically, assuming x' can pass the verification, we have

$$f_1(x') = f_2(x') = ... = f_k(x') = f(x') \tag{4.4}$$

Based on our masking strategy (Section 4.3), at least one of the masked predictions will exclude the adversarial patch in x', *i.e.*

$$\exists j \in \{1, 2, ..., k\},\ s.t.\ f_j(x') = f_j(x) \tag{4.5}$$

Since x is a verified input, we have $f(x) = f_j(x)$, and thus

$$f(x') = f_j(x') = f_j(x) = f(x) \tag{4.6}$$

meaning the adversarial patch attack on x' is ineffective.

However, it is still possible that the benign voter(s) make mistakes. This will eventually lead to an error as described in Section 6. However, such an error originates from the insufficient capability of the base model and is not influenced by the attacker. This error can also be estimated before model implementation as long as the data distribution for the benign patches (*i.e.*patches that do not contain OOD objects or are not adversarially influenced) is similar before and after implementation. Notice that the similar distribution assumption is one of the key assumptions in statistical machine learning theory [100, 101]

The estimated upper bound of the proportion of instances that the benign voters make mistakes is just $(1 - acc_{\text{certified}})$ on clean data. In other words, the probability of a correct and robust inference θ as we mentioned earlier is $acc_{\text{certified}}$. This estimated confidence can serve as an indicator of how much we can trust the model when the full-coverage testing is passed.

Chapter 5

Experiment

To gain a comprehensive understanding of PatchCensor performance and its relationship with state-of-the-art (SOTA) methods from diverse perspectives, we have devised four research questions.

In RQ1, we first examine the performance of our technique and contrast it with both certified recovery and certified detection methods:

RQ1: How does the proposed system PatchCensor perform in terms of clean accuracy and certified accuracy?

The objective of this research question is to provide a comprehensive comparison of all methods discussed thus far. We choose the CIFAR-10 [102] and ImageNet [103] datasets under varying adversarial patch sizes as our evaluation subjects, as these datasets are extensively employed in prior adversarial patch defense studies. The CIFAR-10 dataset is a collection of 60,000 small images. Each image has 32x32 pixels, and they are classified into ten different classes. The ImageNet dataset is a large-scale image database containing 1,431,167 images with an average resolution of 469×387 in various categories, including animals, people, objects, and scenes. Each image is labeled with one of 1,000 different object categories. In addition to these two default evaluation datasets in related literature, we also conduct experiments on the German Traffic Sign Recognition Benchmark (GTSRB) dataset [104] and Food-101 dataset [105]. The GTSRB dataset is a collection of 51,839 images with 48×48

resolution. Each image is annotated with the class of the traffic sign it represents. The Food-101 dataset is a collection of 101,000 images of food items with 101 different classes. Each image is annotated with the class of the food item it represents, and the maximum side length is 512 pixels. The resolution and number of instances in a dataset often serve as indicators of its difficulty. In order to represent various levels of computer vision tasks, we select these four datasets with varying characteristics.

Following the general evaluation, we aim to validate the observations from our preliminary study (Chapter 3) and examine the distinctions between our method (certified detection) and certified recovery when the area of the region of interest (ROI) is small, as addressed in RQ2:

RQ2: Does the proposed system PatchCensor achieve moderate certified performance even for images with a small area of the region of interest (ROI)?

RQ2 aims explicitly to discuss the rationale behind choosing certified detection over certified recovery, as mentioned in the preliminary study. This discussion is a crucial aspect of this paper, with RQ2 serving as a foundation for further analysis in Section 6. It is also worth mentioning that we have compared PatchCensor performance with that of MR+ in RQ1, demonstrating that PatchCensor achieves state-of-the-art results as a certified detection method. In RQ2, our goal is to compare certified detection with certified recovery at the category level. Consequently, we select the most competitive methods from each category, excluding the second-best MR+ from this research question.

Following the discussion on certified detection and certified recovery, we seek to provide a detailed analysis of certified detection. Our evaluation then shifts to a comparison between PatchCensor and the previous state-of-the-art method, MR+. One advantage of certified detection is its capability to defend against adversarially controlled patches with exceptionally large sizes (e.g., up to 25% of the entire image). Thus, in RQ3, we aim to investigate PatchCensor performance under such formidable adversaries on CIFAR-10, GTSRB, Food-101, and ImageNet:

RQ3: How does the proposed PatchCensor perform under strong adversaries?

Specifically, we increase the adversarial patch size from 0.5% to up 25%. Notice that it is possible to detect natural abnormal objects under such a setting. Also, when the size of dropped patches is large, fine-tuning may greatly influence the models' performance. So we also discuss the influence of fine-tuning in this RQ. We do not report the results of certified recovery because they are incapable of dealing with such large adversarial patches. In our experiments, the training is even hard to converge when the patch size is large.

Lastly, performance improvement is typically accompanied by some cost. Our objective is to comprehend the computational overhead for PatchCensor across various adversarial patch sizes in comparison to the previous SOTA method, MR+. The overhead can be split into two components: training time and inference latency. In this research question, we primarily focus on inference latency, as it is a crucial factor in practical implementations:

RQ4: What is the computational overhead of PatchCensor compared with the state-of-the-art certified detection technique?

It is worth noting that, as the mechanism described in Section 4, the larger the target patch size, the fewer inferences are required for PatchCensor. This relationship presents a trade-off, as a larger patch results in reduced certified accuracy but lower inference latency. To explore this connection, RQ4 encompasses experiments designed to guide the selection of an appropriate target patch size during runtime. We also present experiments demonstrating the impact of different ViT variants. Similar to RQ3, certified recovery methods yield only trivial results when the patch size is large. Additionally, as the fundamental certification objectives differ, we do not conduct evaluations in this research question.

PatchCensor is implemented in Python based on a pre-trained ViT-Base model variant with 16×16 input patch size (ViT-B/16) [106], which achieves 81.8% test accuracy on ImageNet.

To support our large-scale evaluation, experiments were conducted on a Linux server with 4 Nvidia V100 GPUs. All the experiments take around 1,200 GPU hours. In the rest of this section, we summarize the key results for each of the studied research questions.

5.1 RQ1: Performance under Normal Setting

Experimental Settings. In order to assess the performance of the proposed system (PatchCensor), abbreviated as PC, it is compared with eight certified adversarial patch defense techniques within the well-established settings of CIFAR-10 and ImageNet datasets, as they are the default benchmarks in the relevant literature. The comparison encompasses four recovery-based methods, namely Interval Bound Propagation (IBP) [47], De-randomized Smoothing (DS) [49], PatchGuard (PG) [48], and BagCert (BC) [51], as well as four detection-based methods, including Minority Report (MR) [53], PatchGuard++ (PG++) [54], and ScaleCert (SC) [55].

The MR method employs direct training (fine-tuning) of a CNN model to classify images with an obscured square region, resulting in higher clean and certified accuracy. However, the original MR design requires the enumeration of numerous occlusion positions, making it computationally demanding for high-resolution images. Consequently, we implemented an enhanced version of MR, designated as MR+, by incorporating our masking strategy to decrease the number of occlusion positions to the same level as in Equation 4.2. Additionally, the CNN backbone of MR+ was replaced with a pre-trained ResNet50 [106] that exhibits comparable performance (81.1% accuracy on ImageNet) to the ViT backbone (ViT-B/16) employed in PatchCensor.

Subsequent to the CIFAR-10 and ImageNet experiments, evaluations were extended to GTSRB and Food-101 datasets. Four methods were included: De-randomized

Table 5.1: The clean and certified accuracy of different certified defenses on ImageNet and CIFAR-10. The numbers of IBP, DS, PG, BC, MR, PG++, and SC are directly copied from their paper. Note that the results of IBP and MR on ImageNet are not available, because they are computationally intensive or even infeasible on high-resolution images.

Method		ImageNet (2% patch size)		CIFAR-10 (2.4% patch size)	
		acc_{clean}	$acc_{\text{certified}}$	acc_{clean}	$acc_{\text{certified}}$
Recovery	Interval Bound Propogation (IBP) [47]	N/A		47.8	30.8
	De-randomized Smoothing (DS) [49]	44.4	14.0	83.9	56.2
	PatchGuard (PG) [48]	43.6	15.7	84.6	57.7
	BagCert (BC) [51]	45.2	22.9	86.0	60.0
Detection	Minority Reports (MR) [53]	N/A		92.5	77.6
				92.5	62.1
				92.5	43.8
	PatchGuard++ (PG++) [54]	62.96	28.0	91.32	68.9
		62.96	32.0	91.32	71.7
		62.96	35.5	91.32	74.3
		62.96	39.0	91.32	76.3
	ScaleCert (SC) [55]	N/A.[1]	55.4	N/A.[1]	75.3
	Minority Reports Adapted (MR+)	75.5	56.3	97.7	83.3
	PatchCensor (PC, our approach)	**81.8**	**69.4**	**98.7**	**84.1**

Smoothing (DS), PatchGuard (PG), advanced Minority Report (MR+), and PatchCensor. The selection of these methods was based on three criteria: (1) their competitive performance on CIFAR-10 and ImageNet, (2) their representation of methods in the relevant fields, and (3) the availability of their open-source implementations.

Result. We first compare our approach with existing certified defense approaches in terms of clean accuracy and certified accuracy. As shown in Table 5.1 and Table 5.2, our approach is able to obviously outperform the SOTA techniques with a clean accuracy of 81.8% and a certified accuracy of 67.1% on the challenging ImageNet

[1]We are unable to find corresponding clean accuracy with respect to the best certified accuracy claimed by the authors

Table 5.2: The clean and certified accuracy of different certified defenses on GTSRB and Food-101. We fine-tune the models on GTSRB for 30 epochs and Food-101 for 60 epochs. All the models are fine-tuned with respect to the given patch size.

Method		GTSRB (2% patch size)		Food-101 (2% patch size)	
		acc_{clean}	$acc_{\text{certified}}$	acc_{clean}	$acc_{\text{certified}}$
Recovery	De-randomized Smoothing (DS) [49]	52.73	15.97	50.15	17.33
	PatchGuard (PG) [48]	68.64	33.61	76.15	46.57
Detection	Minority Reports Adapted (MR+)	96.36	54.65	**84.68**	64.80
	PatchCensor (PC, our approach)	**99.89**	**83.71**	83.39	**67.61**

with 32×32-pixel adversarial patches. Due to the design of PatchCensor, its clean accuracy remains the same as the base ViT model, which can be even further improved by using better base models.

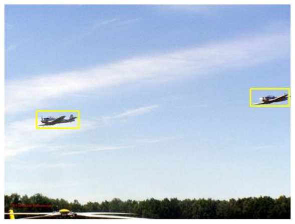
The results of detection-based approaches (including ours) are not directly comparable with the recovery-based approaches because they are designed for different goals. However, we notice that our approach was able to achieve a much higher certified accuracy than recovery-based methods, so it may be more practical to use in the real world. The main reason is that the certified detection is based on voting over predictions with a small region excluded, which can still provide sufficient global information, rather than in recovery-based approaches where each voter is based on a small local patch.

As compared to other SOTA certified detection approaches, PatchCensor could achieve higher certified accuracy on all four datasets. The results of clean accuracy are similar except for MR+ on Food-101, which means ViT-base performs slightly worse than ResNet-50. Notice that here we train both models with the same configuration (randomly occluded patches in the same size), to be relatively fair for comparison. The results indicate that the training configuration might not be optimal for ViT, which could still have some room for further improvement. Even though, under such a non-optimized configuration, the experiment results still confirm the effectiveness of our method in terms of certified accuracy. The result shows that the superior certified accuracy of PatchCensor comes not only from the better base performance of ViT but is also a result of the combination of PatchCensor's defense mechanism and ViT's robustness against the absent patch.

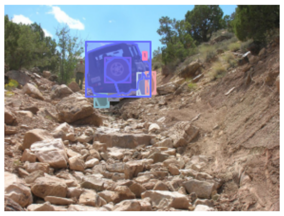
Answer to RQ1: PatchCensor outperforms existing certified recovery methods by a large margin and also achieves higher clean accuracy and certified accuracy compared with other certified detection methods in most settings.

5.2 RQ2: Performance under Small ROI

Experimental Settings. In this research question, we want to evaluate the performance of PatchCensor under small ROI, as compared with the certified recovery methods mentioned in section 3. The adversarial patch size we aim to certify in this RQ is 2.4%, which is 5×5 patch in a 32×32 image. We choose this patch size as it is the default setting in certified recovery work. We first perform rescaling for each image in the CIFAR-10 dataset so the ROI can be controlled. Then we retrain the DS-ResNet on the rescaled image following the default setting in De-randomized Smoothing [49] and PatchGuard [48]. To further validate the PatchCensor can perform well on small ROI, we additionally design two experiments on the ImageNet visual object detection [107] and PartImageNet dataset [108]. The former includes bounding box annotations for the target object (as shown in Fig 5.1a) in the image and could serve as an indicator of how large the Region of interest(ROI) is. We compute the ROI by dividing the bounding box's area by the whole image's area. The latter includes more fine-grained per-pixel part annotations, and each object is partitioned into smaller parts (as shown in Fig 5.1b). As current certified recovery methods usually rely on local features for inference and verification, the area of object parts could also stand for ROI. The ROI of PartImageNet is obtained by dividing the area of the largest part by the area of the whole image. The model architectures we



(a) ImageNet with bounding box



(b) PartImageNet

Figure 5.1: Illustration of the annotation

Table 5.3: The clean accuracy, certified accuracy, and accuracy in trust domain achieved by PatchCensor (PC), PatchGuard (PG) and De-randomized Smoothing (DS) for 2.4%-pixel adversarial patch attack.

Rescaling size	acc_{clean}			$acc_{\text{certified}}$			$acc_{\text{in-robust}}$		
	PC	PG	DS	PC	PG	DS	PC	PG	DS
32 (original size)	**98.8**	83.35	83.35	**88.85**	53.23	51.85	**99.83**	97.87	97.08
28	**98.3**	80.80	80.29	**84.48**	44.66	42.19	**99.85**	96.92	97.06
26	**97.81**	77.47	77.65	**80.44**	38.24	35.94	**99.79**	96.3	96.43
24	**97.79**	77.05	76.85	**77.94**	33.21	30.96	**99.76**	93.79	96.99
22	**97.14**	76.26	75.63	**71.45**	28.70	23.24	**99.68**	92.11	97.44
20	**96.67**	69.38	67.35	**66.56**	10.81	1.03	**99.75**	78.73	83.06

used for all the methods here remain the same with RQ1.

Result. For the rescaled CIFAR-10 experiment, detailed result is shown at Table 5.3. We also plot the changing trend of $acc_{\text{certified}}$ and $acc_{\text{in-robust}}$ for PatchCensor and other two techniques in Figure 5.2.

One interesting observation is that certified recovery techniques (both the PatchGuard and De-randomized Smoothing) experience a quick drop for both r_{robust} and $acc_{\text{in-robust}}$ when the image size scales from 32×32 to 20×20. The r_{robust} of De-randomized Smoothing even drops to nearly 0 (1.24%) with the rescaling size 20, unable to give certified prediction anymore. The reason behind this is the voting mechanism discussed in section 3. While PatchCensor is able to remain a high certified accuracy even with very small ROI.

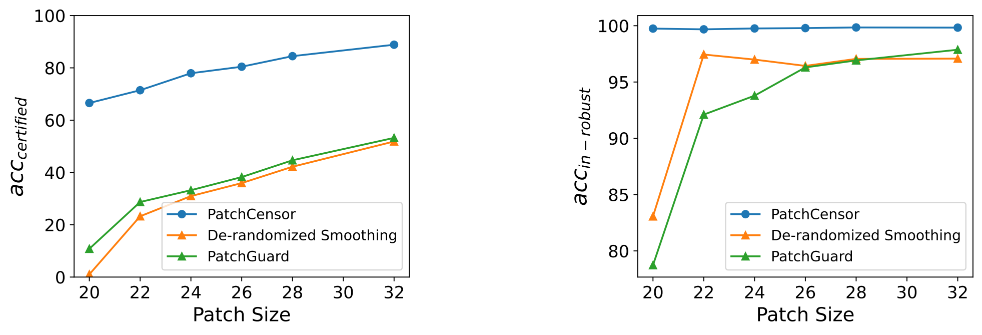


Figure 5.2: Certified accuracy and accuracy in the robust domain of PatchCensor, PatchGuard and De-randomized Smoothing on rescaled CIFAR-10.

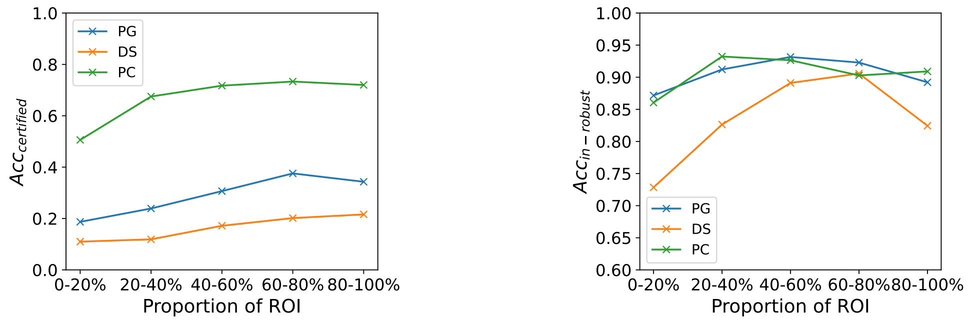


Figure 5.3: Certified accuracy and accuracy in the robust domain of PatchCensor, PatchGuard and De-randomized Smoothing on ImageNet

For experiments on ImageNet object detection and PartImageNet, we first sort the image according to ROI in ascending order and compute certified accuracy in 5-quantiles. The result for the former is shown in Figure 5.3, and the result for the latter is shown in Figure 5.4. It can be observed that PatchCensor can still have 51.0% certified accuracy on ImageNet object detection and 72.6% on PartImageNet in the 0-20% region, while PatchGuard and De-randomized Smoothing can only yield 18.7% (24.2%), and 11.0% (16.2%) certified accuracy, respectively.

It is worth noticing that $acc_{\text{in-robust}}$ of PatchCensor has some overlaps across different ROI on ImageNet object detection and PartImageNet. This may be due to the fact that pre-trained models on ImageNet may utilize other features in the image, such as texture instead of shape, as pointed out by Geirhos *et al.* [109]. In such cases, using only local features may be enough. However, this can be a kind of overfitting.

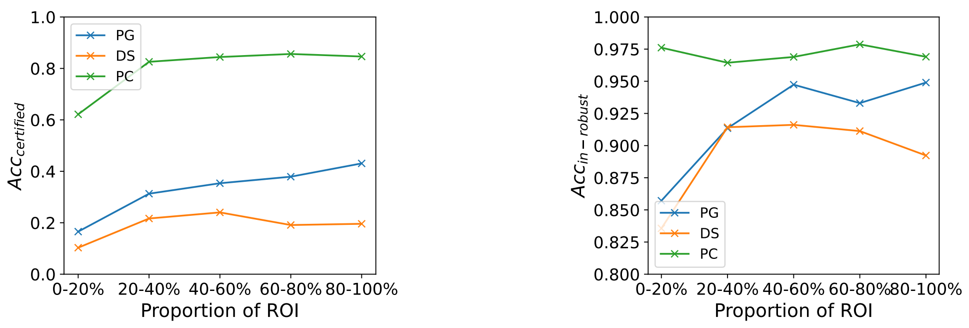


Figure 5.4: Certified accuracy and accuracy in the robust domain of PatchCensor, PatchGuard and De-randomized Smoothing on PartImageNet

Our controlled experiments on CIFAR-10 have no such issue. Despite this, extra information such as texture could provide performance gain for certified recovery, and we will discuss more in Section 6.

Answer to RQ2: PatchCensor is able to achieve moderate performance when ROI is small. It can even maintain relatively good performance under small ROI at more complex and challenging ImageNet dataset.

5.3 RQ3: Performance under Strong Adversaries

Experimental Settings. In this research question, we evaluate the certification performance of PatchCensor and MR+ with large adversarial patches whose size is up to 25.0%. We ignore certified recovery work because they only focus on small patches and are unable to handle large patches. On the one hand, the certification technique yields few certified instances once the patch size becomes large. On the other hand, it is much harder to train the model under such a setting. However, our defense and other SOTA certified detection work focus mainly on masking strategy. It is possible to certify instances even when the adversarial patches are large in size. Being able to certify inputs with large patch sizes would also make it possible to detect abnormal scenarios other than the adversarial setting, such as natural perturbation [110], occlusion in object detection [92] and open-set detection [93]. As natural objects are often larger than commonly evaluated 2.0% patch size. In the experiments, we select CIFAR-10, GTSRB, Food-101 and ImageNet as the subject datasets for evaluation. For the models, ViT-B/16 (the same architecture in RQ1, RQ2) and ResNet50 pre-trained on ImageNet are selected for PatchCensor and MR++ accordingly. Additionally, as we target a large adversarial patch size in this RQ, fine-tuning on randomly dropped patches can potentially improve the certified accuracy for both methods. As a result, we report the performance with/without such fine-tuning to provide a more comprehensive analysis of both methods. This leads to eight groups of experiments in total as shown below.

Result. We tested the defense effectiveness of PatchCensor and MR+ under different adversarial patch sizes on four datasets, with a total of 112 configurations. The results for for CIFAR-10 are summarized in Table 5.4 and Table 5.5, results for GTSRB are summarized in Table 5.6 and Table 5.7, results for Food-101 are summarized in Table 5.8 and Table 5.9, and the results for ImageNet are summarized in Table 5.10 and Table 5.11.

For all four datasets, PatchCensor achieves better performance on the most important certified accuracy metric than MR+ in most of the settings. The maximum difference is 40.73% on CIFAR-10 without fine-tuning, and the minimum difference is 2.57% on GTSRB without fine-tuning. The only exceptions are on the CIFAR-10 dataset with fine-tuning, where MR+ surpasses PatchCensor when the patch size is larger or equal to 18.4%. From the result, we infer that CNN, specifically fine-tuned for patch drop, can be more robust than ViT on a simper dataset such as CIFAR-10.

As for the clean accuracy, PatchCensor is higher than MR+ on nearly all settings (ranging from 1.18% on ImageNet to 8.31% on fine-tuned GTSRB) except for Food-101 with fine-tuning. Still, even under such a circumstance, PatchCensor has higher certified accuracy than MR+ across all adversarial patches on Food-101.

Interestingly, both methods have the worst performance on the GTSRB dataset when the adversarial patch size is large. It can be surprising that the certified accuracy of GTSRB is even lower than that of ImageNet, which is believed to be far more complex than GTSRB. Also, fine-tuning plays an important role in the performance, giving 26.87% improvement for PatchCensor and 19.02% improvement for MR+ on certified accuracy. The reason may be that, compared with ImageNet, the target object in GTSRB occupies most of the image. It is more likely that the dropped patch is inside the target object and thus has more effect on the data distribution, interfering with the model's prediction. When the size of the dropped patch becomes large, important details may be missed, causing a sharp decrease in performance. As such, we suggest fine-tuning the models under a similar situation.

For other subject datasets we evaluated besides the GTSRB dataset, the fine-tuning process does not influence the performance of PatchCensor too much. The average increase after fine-tuning for PatchCensor in certified accuracy across all patches is 5.39%, -0.73 %, and 4.44% for CIFAR-10, Food-101, and ImageNet, respectively. However, MR+ relies heavily on fine-tuning. The corresponding increase in certified accuracy is 33.23%, 19.02%, and 4.72% for CIFAR-10, Food-101, and ImageNet. Furthermore, the certified accuracy of PatchCensor without fine-tuning is higher than that of MR+ with fine-tuning on 16 of 21 cases on these three datasets. We can thus conclude that PatchCensor can yield moderate performance on datasets similar to these three, even without fine-tuning. This means that the pre-trained models can be directly combined with PatchCensor to increase its patch robustness.

Meanwhile, it is interesting to notice that the accuracy of PatchCensor in the trust domain ($acc_{\text{in-trust}}$) remained high (even slightly increases) under larger adversarial patch sizes for all datasets. This means that PatchCensor can retain high usability under strong adversaries - even though PatchCensor may raise more warnings (*i.e.*, report more unverifiable input images) when defending against stronger adversaries, it can still promise a high accuracy when it gives verified predictions.

Table 5.4: Clean accuracy & certified accuracy achieved by PatchCensor and MR+ on **CIFAR-10** under different adversarial patch sizes **without fine-tuning**

Max Adv Patch Size	acc_{clean}		$acc_{\text{certified}}$		r_{robust}		$acc_{\text{in-robust}}$	
	PC	MR+	PC	MR+	PC	MR+	PC	MR+
0.5%	**98.74**	96.80	**94.57**	79.48	**94.89**	79.84	**99.66**	99.55
2.0%	**98.74**	96.81	**90.13**	68.52	**90.29**	68.78	**99.82**	99.62
4.6%	**98.74**	96.80	**84.12**	54.90	**84.24**	55.07	**99.86**	99.69
8.2%	**98.74**	96.79	**76.20**	41.37	**76.28**	41.53	**99.90**	99.61
12.8%	**98.74**	96.81	**67.41**	31.32	**67.46**	31.53	**99.93**	99.33
18.4%	**98.74**	96.81	**58.90**	23.79	**58.97**	24.01	**99.88**	99.08
25.0%	**98.74**	96.79	**48.52**	18.50	**48.57**	18.63	**99.90**	99.30

Table 5.5: Clean accuracy & certified accuracy achieved by PatchCensor and MR+ on **CIFAR-10** under different adversarial patch sizes **with fine-tuning**

Max Adv Patch Size	acc_{clean}		$acc_{\text{certified}}$		r_{robust}		$acc_{\text{in-robust}}$	
	PC	MR+	PC	MR+	PC	MR+	PC	MR+
0.5%	**98.77**	97.61	**95.70**	90.59	**96.05**	90.97	**99.64**	99.58
2.0%	**98.80**	97.73	**92.84**	86.77	**93.06**	87.00	**99.76**	99.74
4.6%	**98.81**	97.70	**88.29**	83.30	**88.42**	83.46	**99.85**	99.81
8.2%	**98.77**	97.51	**82.90**	79.78	**83.04**	79.82	**99.83**	99.82
12.8%	**98.76**	97.35	**75.52**	74.91	**75.99**	75.04	**99.91**	99.83
18.4%	**98.76**	96.90	66.68	**71.03**	66.77	**71.19**	**99.87**	99.78
25.0%	**98.75**	95.57	55.89	**65.60**	55.95	**65.77**	**99.89**	99.74

Table 5.6: Clean accuracy & certified accuracy achieved by PatchCensor and MR+ on **GTSRB** under different adversarial patch sizes **without fine-tuning**

Max Adv Patch Size	acc_{clean}		$acc_{\text{certified}}$		r_{robust}		$acc_{\text{in-robust}}$	
	PC	MR+	PC	MR+	PC	MR+	PC	MR+
0.5%	**97.43**	92.49	**70.26**	36.13	**71.01**	36.32	98.94	**99.48**
2.0%	**97.43**	92.48	**40.89**	22.04	**41.31**	22.06	98.97	**99.93**
4.6%	**97.43**	92.48	**26.04**	11.94	**26.08**	11.94	99.85	**100**
8.2%	**97.43**	92.49	**18.39**	7.19	**18.42**	7.21	**99.87**	99.78
12.8%	**97.43**	92.48	**12.49**	5.65	**12.50**	5.69	**99.87**	99.30
18.4%	**97.43**	92.49	**8.99**	4.73	**9.02**	4.79	**99.74**	98.68
25.0%	**97.43**	92.49	**6.71**	4.14	**6.73**	4.22	**99.76**	98.12

Table 5.7: Clean accuracy & certified accuracy achieved by PatchCensor and MR+ on **GTSRB** under different adversarial patch sizes **with fine-tuning**

Max Adv Patch Size	acc_{clean}		$acc_{\text{certified}}$		r_{robust}		$acc_{\text{in-robust}}$	
	PC	MR+	PC	MR+	PC	MR+	PC	MR+
0.5%	**96.84**	91.31	**83.10**	56.79	**83.97**	58.42	**98.96**	97.21
2.0%	**97.09**	92.53	**70.33**	48.35	**71.21**	49.43	**98.77**	97.81
4.6%	**96.88**	91.09	**61.29**	36.48	**61.88**	37.61	**99.05**	97.01
8.2%	**97.65**	91.23	**49.29**	27.78	**49.89**	28.35	**98.79**	97.96
12.8%	**97.32**	89.96	**43.93**	21.77	**44.36**	22.33	**99.02**	97.48
18.4%	**96.52**	90.15	**35.61**	18.38	**36.08**	18.65	**98.71**	98.51
25.0%	**96.33**	88.02	**28.31**	15.39	**28.86**	15.86	**98.08**	97.05

Table 5.8: Clean accuracy & certified accuracy achieved by PatchCensor and MR+ on **Food-101** under different adversarial patch sizes **without fine-tuning**

Max Adv Patch Size	acc_{clean}		$acc_{\text{certified}}$		r_{robust}		$acc_{\text{in-robust}}$	
	PC	MR+	PC	MR+	PC	MR+	PC	MR+
0.5%	**86.46**	84.45	**77.08**	66.28	**81.04**	68.62	95.11	**96.59**
2.0%	**86.46**	84.46	**72.30**	60.38	**74.91**	61.85	96.52	**97.63**
4.6%	**86.46**	84.46	**66.60**	53.36	**68.26**	54.38	97.56	**98.12**
8.2%	**86.46**	84.46	**60.02**	45.89	**61.15**	46.55	98.15	**98.58**
12.8%	**86.46**	84.46	**52.82**	38.71	**53.61**	39.17	98.52	**98.82**
18.4%	**86.46**	84.46	**44.93**	31.71	**45.52**	32.05	98.70	**98.94**
25.0%	**86.46**	84.46	**36.61**	24.74	**37.09**	25.03	98.73	**98.83**

Table 5.9: Clean accuracy & certified accuracy achieved by PatchCensor and MR+ on **Food-101** under different adversarial patch sizes **with fine-tuning**

Max Adv Patch Size	acc_{clean}		$acc_{\text{certified}}$		r_{robust}		$acc_{\text{in-robust}}$	
	PC	MR+	PC	MR+	PC	MR+	PC	MR+
0.5%	83.28	**84.77**	**72.56**	70.57	**77.66**	73.67	93.44	**95.79**
2.0%	83.39	**84.68**	**67.61**	64.80	**70.93**	66.86	95.32	**96.91**
4.6%	84.02	**84.82**	**63.35**	59.16	**65.42**	60.40	96.84	**97.95**
8.2%	84.80	**84.47**	**58.65**	53.24	**59.96**	54.18	97.81	**98.27**
12.8%	84.90	**85.06**	**53.53**	48.58	**54.42**	49.21	98.37	**98.73**
18.4%	84.97	**85.30**	**47.22**	44.33	**47.92**	44.83	98.55	**98.88**
25.0%	**85.62**	85.33	**42.32**	38.57	**42.81**	39.02	**98.85**	98.84

Table 5.10: Clean accuracy & certified accuracy achieved by PatchCensor and MR+ on **ImageNet** under different adversarial patch sizes **without fine-tuning**

Max Adv Patch Size	acc_{clean}		$acc_{\text{certified}}$		r_{robust}		$acc_{\text{in-robust}}$	
	PC	MR+	PC	MR+	PC	MR+	PC	MR+
0.5%	**81.8**	80.62	**72.0**	62.68	**80.3**	68.27	89.73	**91.81**
2.0%	**81.8**	80.62	**67.2**	56.02	**73.5**	60.24	91.35	**92.99**
4.6%	**81.8**	80.62	**61.9**	49.52	**67.0**	52.75	92.44	**93.88**
8.2%	**81.8**	80.62	**56.4**	42.94	**60.4**	45.46	93.29	**94.46**
12.8%	**81.8**	80.62	**50.5**	36.16	**53.7**	38.22	93.96	**94.62**
18.4%	**81.8**	80.62	**44.1**	30.05	**46.7**	31.70	94.57	**94.78**
25.0%	**81.8**	80.62	**37.1**	24.07	**39.1**	25.37	94.80	**94.88**

Table 5.11: Clean accuracy & certified accuracy achieved by PatchCensor and MR+ on **ImageNet** under different adversarial patch sizes **with fine-tuning**

Max Adv Patch Size	acc_{clean}		$acc_{\text{certified}}$		r_{robust}		$acc_{\text{in-robust}}$	
	PC	MR+	PC	MR+	PC	MR+	PC	MR+
0.5%	**82.70**	75.59	**73.67**	61.84	**81.88**	70.99	**89.98**	87.12
2.0%	**82.73**	75.51	**69.41**	56.31	**75.92**	63.05	**91.43**	89.32
4.6%	**82.67**	75.49	**64.97**	51.23	**70.27**	56.56	**92.45**	90.57
8.2%	**82.66**	76.34	**60.54**	48.17	**64.92**	52.55	**93.26**	91.66
12.8%	**82.55**	76.20	**55.38**	43.39	**59.08**	46.99	**93.73**	92.35
18.4%	**82.51**	75.81	**50.66**	38.77	**53.76**	41.63	**94.23**	93.12
25.0%	**82.49**	75.50	**45.62**	34.79	**48.24**	37.34	**94.56**	93.19

Answer to RQ3: PatchCensor outperforms MR+ in most scenarios on CIFAR-10, GTSRB, Food-101 and ImageNet datasets. It also retains high accuracy in the robust domain ($acc_{\text{in-robust}}$) even with large adversarial patches. Being capable of detecting large abnormal patches makes it possible to use PatchCensor in more complex natural perturbation settings.

5.4 RQ4: Overhead of PatchCensor

Experimental Settings. In this research question, we investigate how much computation overhead PatchCensor may incur. Typically, there are two kinds of overhead for DNN models when conducting the certification. One is overhead for retraining/fine-tuning the model for the designed property. The other is the certification latency.

For training overhead, we have already provided results and analysis with/without fine-tuning on four datasets in RQ3. For CIFAR-10, Food-101, and ImageNet, PatchCensor can yield moderate results without any fine-tuning efforts. Another particular feature of PatchCensor regarding the training overhead we want to emphasize here is that for both fine-tuning and certification, PatchCensor targets adversarial patches whose size is in a specific range (*i.e.*, for all patches whose size is smaller than a threshold). In contrast, MR and most other patch certification methods only target adversarial patches with only one size.

For verification latency, we measured the latency of PatchCensor to verify an input image against different adversarial patch sizes. Here we compare our technique with the previous SOTA MR+, as mentioned in the design goal of RQ4.

Results. We measure the verification latency of PatchCensor to see whether it is acceptable in this RQ. As ViT has many variants, and the latency for them may have a high variance, we independently provide results on each of them. We choose Vit_s16_224 (small), ViT_b16_224 (base), and ViT_l16_224 (large) for investigating the influence of model size. We choose ViT_b16_224 (base), ViT_-b32_224 (base), and ViT_b32_384 (base) for investigating the influence of patch and input size. Model details are shown in table 5.12. We additionally provide the results on clean accuracy and certified accuracy evaluated on 1,000 randomly sampled images from ImageNet without fine-tuning for defense. We hope this can serve as a guide for deployment in practice.

The results of the performance regarding the model sizes are shown in figure 5.5. It is surprising that the smallest model achieves both the best certified accuracy and lowest latency (similar latency with MR+) among all the three variants. The reason may be that this smallest model does not overfit the dataset compared to the other two bigger models and is thus more robust towards masking. A similar result is also reported in [24], where the smallest ViT has a high accuracy for random patch drop when the information loss is high. For the other two variants, stronger backbones produce slightly higher certified accuracy as we expected.

The results of the performance regarding the input and patch sizes are shown in figure 5.6. These three variants have similar certified accuracy with quite different latency. Given the same input size, a higher patch size for the base model will drastically lower the latency. The ViT_b32_224 achieves even lower latency than MR+. However, a smaller base patch will provide a more subtle control of the certified patch size. As PatchCensor certifies the patch *in range*, its certified accuracy is in the format of the step function, such as 0-16, 16-32, 32-48, ... for base patch equals to 16 and 0-32, 32-64, 64-96, ... for base patch equals to 32. If one wishes to certify a patch with a size 47, it is better to choose a small base patch (32-48) rather than a big base patch(32-64) because the former can yield a better certified accuracy.

The results demonstrate that PatchCensor is better when using a smaller model with a large base patch. Under such a scenario, PatchCensor can achieve lower latency than MR+ while providing higher certified accuracy on a complex dataset.

Still, we want to emphasize that PatchCensor is not restricted to any kind of ViT variant. Even with a large ViT with a small base patch, the higher latency incurred by PatchCensor is also a meaningful tradeoff in some realistic scenarios because of the various benefits it provides, such as no need for fine-tuning and higher certified

Table 5.12: Choosed ViT variants for latency measurement. The base model is what we use for evaluation in RQ1, RQ2, and RQ3.

ViT Variants	Number of Parameters	Clean Accuracy	Patch Size	Input size
ViT_s16_224	22,050,664	80.5 %	16	224
ViT_b16_224 (base)	86,567,656	80.9 %	16	224
ViT_l16_224	304,326,632	82.2 %	16	224
ViT_b32_224	88,224,232	80.8 %	32	224
ViT_b32_384	88,297,192	80.8 %	32	384

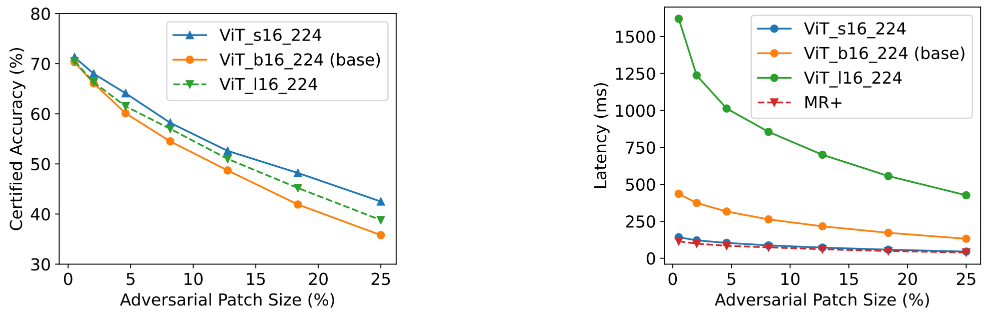


Figure 5.5: Certified accuracy and latency for ViT variants of different model sizes

accuracy. PatchCensor is also a flexible detection framework that can work parallel with other methods. For real-time scenarios where the latency constraint is tight, PatchCensor can be combined and play as a nice complement to other low-latency algorithms, where it focuses on those critical frames that need rigorous analysis.

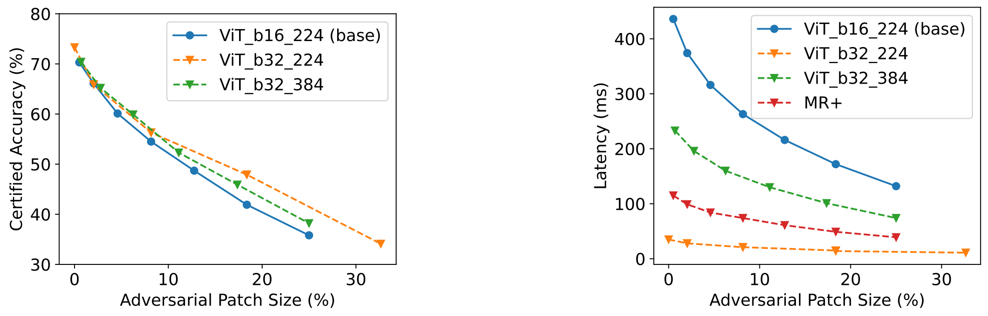


Figure 5.6: Certified accuracy and latency for ViT variants of different input and patch sizes

Answer to RQ4: PatchCensor requires much less effort in the training (fine-tuning) process and can yield good performance even without fine-tuning. PatchCensor can provide high certified accuracy with low latency when the base model is small, or its patch size is large. Even when defending a large model, we argue the higher latency is an acceptable compromise to the advantage it brings.

Chapter 6

Discussion

In this section, we delve deeper into the distinctions between certified detection and certified recovery. First, we examine a situation where the defense of certified detection might be "bypassed." This aspect has been ignored in previous research, potentially leading to misconceptions among readers. Nevertheless, we argue that such exceptional cases will not affect the practicality of certified detection. Subsequently, we explore a potential avenue for enhancing certified recovery.

6.1 A Special Case for Certified Detection

The certified properties of certified recovery and certified detection differ significantly. Certified recovery adheres to the conventional definition found in the digital adversarial defense literature, where a model is considered robust if its inference remains consistent within the vicinity of a given input. The sole distinction lies in the definition of the "neighborhood," which is specified as the L_p norm for a digital adversarial attack and a restricted-size patch for an adversarial patch attack. In order to certify such robustness, certified recovery necessitates the development of intricate strategies to guarantee that the model produces consistent results based on local features alone. In contrast, PatchCensor and other certified detection approaches merely strive to identify the existence of abnormal input by ensuring that at least one benign voter has the ability to influence the inference. The underlying philosophy of certified detection can be encapsulated as follows:

Truth always rests with the minority, and the minority is always stronger than the majority.

However, **What if the minority is wrong**?

This is the point at which the exceptional case arises and the reason why a test-time robustness guarantee cannot be provided for PatchCensor and all other certified detection work thus far. A demonstrative example can be found in Figure 6.1. The prediction is accurate when the majority voting result corresponds to the "correct class," and the other votes unanimously support another class (class B). However, due to the lack of consensus in the voting process, the certified detection method returns a "not certified" outcome. Following the attackers' perturbation of the image, the majority voting may shift to class B entirely. As a result of the original incorrect voting, a consensus is achieved, and the certified detection method erroneously returns a "certified" outcome. This subtle discrepancy has been ignored in most existing certified detection approaches [53–55], as the base classifier is presumed to be sufficiently robust to circumvent such situations. We also found a real example in our experiments in Fig. 6.2. In this example, the original image has 69 votes for class 35 (mud turtle) and 53 votes for class 36 (terrapin). The perturbed image has 122 votes for class 36. The ground truth of this image is class 35. As a result, the model is correct but not certified on the clean image and wrong but certified on the perturbed image. Nonetheless, it is hard to distinguish mud turtles from terrapin even for non-expert humans.

The potential consequences of this issue are significant, as a certified detection method could falsely reassure users of system integrity despite the presence of malicious behavior. However, Before drawing the conclusion that certified detection can be circumvented, we would like to delve deeper into this issue and ask another question: *when will the minority make mistakes?*

The mistakes made by benign voters are not influenced by any attacker or natural

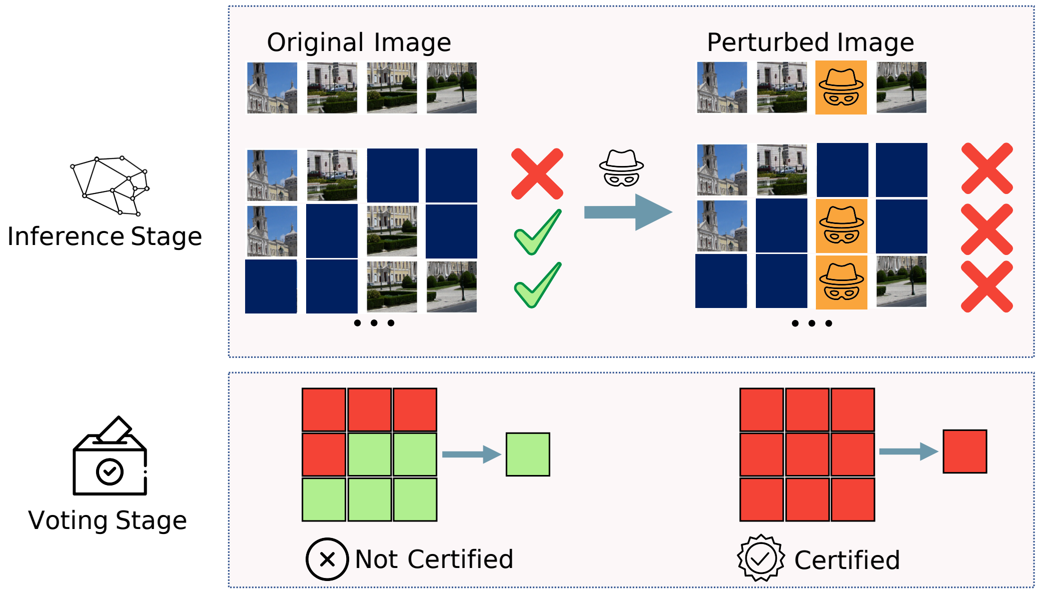


Figure 6.1: An illustration that certified detection may fail at test time - the method may return 'certified' for attacked images.

perturbation, as their impact is entirely masked. Such errors arise solely due to the imperfection of deep learning models and their inability to fit the original distribution accurately. In other words, it is just like other normal errors where the model presents erroneous predictions for a clean image. Furthermore, this type of error can be estimated by assessing the instances in which the detection method returns a certified and correct result on the clean dataset. This evaluation can offer a statistical guarantee, where we can inform the users how likely this model would be correct.

It is also worth noting that in certified recovery techniques, "certified robust" does not necessarily imply "certified to be correct." That is, images that pass the certification might still be incorrect due to the imprecision of the models. From this viewpoint, although the common-sense robustness property (*i.e.*, model prediction remaining unchanged within the vicinity of a given input) can be certified, the guarantee of the model's correctness is accompanied by uncertainty. In fact, it can be observed that the $acc_{\text{in-robust}}$ of PatchCensor consistently surpasses that of the other two SOTA methods in Figure 5.2. It is also noteworthy that in the same experiment when the region of interest (ROI) is small, the accuracy of both certified recovery

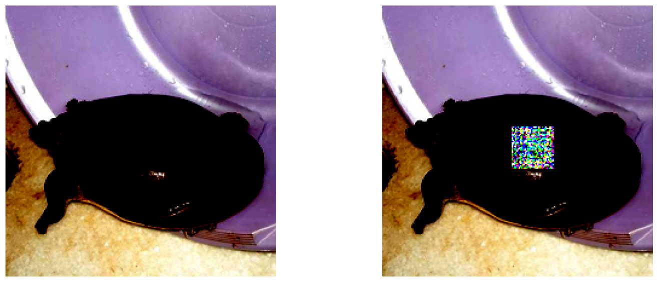

(a) The original image. **(b)** The perturbed image.

Figure 6.2: A real example that can "crack" the certified adversarial patch detection.

techniques experiences a rapid reduce.

In conclusion, while the certification of PatchCensor (along with other certified detection techniques) offers a statistical guarantee for a data distribution, its practical value is enhanced by its superior performance.

6.2 A Possible Way to Improve Certified Recovery

Certified recovery methods strive to utilize only local features to generate reliable predictions. As demonstrated in both Chapter 3 and Chapter 5, this can significantly impact their certified accuracy. However, incorporating additional features such as texture, material, and color of the object might aid the classifier in making accurate predictions, even when solely relying on local features. To validate this hypothesis, we conducted an experiment using a toy example.

MNIST is intrinsically a single-channel image dataset. We expanded these grayscale images into RGB images by adding two channels and assigned different colors to distinct digits. We then assessed the performance of PatchSmooth. The certification was executed with a band size of 2 and an adversarial patch size of 5. The results are presented in Table 6.1.

While this is a toy example, it illustrates a potential avenue for enhancing the performance of certified recovery methods. In real-world scenarios, the robustness of camera-based autonomous driving systems could be improved by diversifying the

Table 6.1: Adding additional information can improve the performance of PatchSmooth.

	Single Channel	Three Channel
Clean Accuracy	96.66 %	100 %
Certified Image Proportion	52.84 %	86.03 %
$Acc_{in-robust}$	99.9 %	100 %

texture of traffic signs or incorporating other supplementary information. Securely integrating additional information into local patches may represent a promising direction for future certified recovery research.

Chapter 7

Conclusions and Future Work

In this paper, we introduce a straightforward yet efficient certified defense against adversarial patches, capitalizing on the novel relationship between adversarial patches and input patches in Vision Transformers. This method can function as a pure testing strategy, delivering moderate performance even without fine-tuning. It also enables the detection of abnormal patches by up to a size of 25% of the original image, significantly exceeding previous certification efforts. This flexibility allows for the detection of issues beyond adversarial attacks (*e.g.*natural corruption in patches or out-of-distribution (OOD) objects). We demonstrate the defense's effectiveness and practicality on CIFAR-10, GTSRB, Food-101, and ImageNet datasets, as well as its adaptability to support varying sizes of adversarial patches. We show that PatchCensor is able to alleviate the issues of a small area of interest (AOI) in certified recovery and its acceptable overhead.

There are several possible directions for future research. First, our application focuses on classification tasks within the computer vision domain. Investigating other applications is a viable option. Given that our masking strategies operate at an abstract level, it is feasible to apply this method to natural language processing (NLP) tasks, as illustrated in Fig. 7.1. Additionally, it would be worthwhile to explore alternative tasks beyond classification, such as regression.

Exploring the scenarios of multiple adversarial patches from both the attacker's

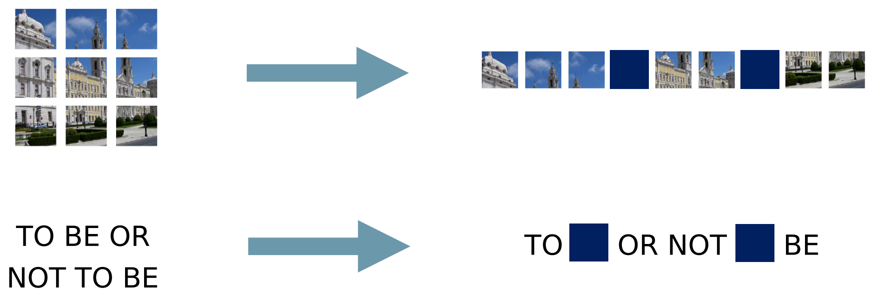


Figure 7.1: Our masking strategy works on abstract feature level and can be applied to both images and text.

and defender's perspectives presents an intriguing area of study. The existing defense framework is tailored to address a single adversarial patch, leaving other patch shapes and multiple patches unaddressed. One potential issue with multiple patches could be the necessity for an excessive number of mutations in order to guarantee that at least one mutation can effectively exclude the adversarial patches. Addressing this issue from the AI perspective by developing more fine-grained attention masking strategies constitutes an interesting direction. Alternatively, adopting a systems perspective and designing a more complex voting mechanism, similar to managing Byzantine faults in distributed systems, is another viable option.

Finally, as natural corruptions (*e.g.*motion blur of cameras) and out-of-distribution (OOD) objects are much more common in practice, there is also an urgent need to develop an efficient detection mechanism for AI-based intelligent systems. Understanding the capability boundary of an AI model and predicting when it would possibly provide erroneous results are vital for the quality assurance of AI enabled systems.

Bibliography

[1] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[2] D. Chung, K. Tahboub, and E. J. Delp, “A two stream siamese convolutional neural network for person re-identification,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1983–1991.

[3] Y. Sun, X. Wang, and X. Tang, “Sparsifying neural network connections for face recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4856–4864.

[4] J. Wei, J. He, Y. Zhou, K. Chen, Z. Tang, and Z. Xiong, “Enhanced object detection with deep convolutional neural networks for advanced driving assistance,” *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[5] C. Szegedy *et al.*, “Intriguing properties of neural networks,” in *2nd International Conference on Learning Representations (ICLR)*, 2014.

[6] B. Biggio *et al.*, “Evasion attacks against machine learning at test time,” in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13*, Springer, 2013, pp. 387–402.

[7] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman, “Sok: Security and privacy in machine learning,” in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2018, pp. 399–414.

[8] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in *3rd International Conference on Learning Representations (ICLR)*, 2015.

[9] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, “Adversarial attacks and defences: A survey,” *arXiv preprint arXiv:1810.00069*, 2018.

[10] J. Lu, H. Sibai, E. Fabry, and D. Forsyth, “No need to worry about adversarial examples in object detection in autonomous vehicles,” *arXiv preprint arXiv:1707.03501*, 2017.

[11] K. Eykholt *et al.*, “Robust physical-world attacks on deep learning visual classification,” in *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 1625–1634.

[12] H. Zhou *et al.*, "Deepbillboard: Systematic physical-world testing of autonomous driving systems," in *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, IEEE, 2020, pp. 347–358.

[13] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016, pp. 1528–1540.

[14] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," in *Conference on Neural Information Processing Systems Workshops (NeurIPS Workshops)*, 2017.

[15] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Artificial intelligence safety and security*, Chapman and Hall/CRC, 2018, pp. 99–112.

[16] D. Song *et al.*, "Physical adversarial examples for object detectors," in *12th USENIX workshop on offensive technologies (WOOT 18)*, 2018.

[17] S. Thys, W. V. Ranst, and T. Goedemé, "Fooling automated surveillance cameras: Adversarial patches to attack person detection," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*, 2019, pp. 49–55.

[18] S. Komkov and A. Petiushko, "Advhat: Real-world adversarial attack on arcface face id system," in *2020 25th International Conference on Pattern Recognition (ICPR)*, IEEE, 2021, pp. 819–826.

[19] A. Vaswani *et al.*, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 6000–6010.

[20] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *9th International Conference on Learning Representations (ICLR)*, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy.

[21] Z. Liu *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," *arXiv preprint arXiv:2103.14030*, 2021.

[22] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16 259–16 268.

[23] H. Zhang *et al.*, *Dino: Detr with improved denoising anchor boxes for end-to-end object detection*, 2022. arXiv: 2203.03605 `[cs.CV]`.

[24] M. M. Naseer, K. Ranasinghe, S. H. Khan, M. Hayat, F. Shahbaz Khan, and M.-H. Yang, "Intriguing properties of vision transformers," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[25] S. Bhojanapalli, A. Chakrabarti, D. Glasner, D. Li, T. Unterthiner, and A. Veit, "Understanding robustness of transformers for image classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 231–10 241.

[26] S. Paul and P.-Y. Chen, "Vision transformers are robust learners," *arXiv preprint arXiv:2105.07581*, 2021.

[27] Y. Bai, J. Mei, A. L. Yuille, and C. Xie, "Are transformers more robust than cnns?" *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[28] J. Gu, V. Tresp, and Y. Qin, "Are vision transformers robust to patch perturbations?" *arXiv preprint arXiv:2111.10659*, 2021.

[29] K. Mahmood, R. Mahmood, and M. Van Dijk, "On the robustness of vision transformers to adversarial examples," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 7838–7847.

[30] Y. Fu, S. Zhang, S. Wu, C. Wan, and Y. Lin, "Patch-fool: Are vision transformers always robust against adversarial perturbations?" In *International Conference on Learning Representations*, 2021.

[31] J. Kim, R. Feldt, and S. Yoo, "Guiding deep learning system testing using surprise adequacy," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, IEEE, 2019, pp. 1039–1049.

[32] A. Stocco, M. Weiss, M. Calzana, and P. Tonella, "Misbehaviour prediction for autonomous driving systems," in *Proceedings of the ACM/IEEE 42nd international conference on software engineering*, 2020, pp. 359–371.

[33] X. Zhang *et al.*, "Towards characterizing adversarial defects of deep learning software from the lens of uncertainty," in *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, IEEE, 2020, pp. 739–751.

[34] X. Zhang, "Uncertainty-guided testing and robustness enhancement for deep learning systems," in *2020 IEEE/ACM 42nd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, IEEE, 2020, pp. 101–103.

[35] H. Wang, J. Xu, C. Xu, X. Ma, and J. Lu, "Dissector: Input validation for deep learning applications by crossing-layer dissection," in *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, IEEE, 2020, pp. 727–738.

[36] Z. Li, X. Ma, C. Xu, J. Xu, C. Cao, and J. Lü, "Operational calibration: Debugging confidence errors for dnns in the field," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 901–913.

[37] W. Ma, M. Papadakis, A. Tsakmalis, M. Cordy, and Y. L. Traon, "Test selection for deep learning systems," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 30, no. 2, pp. 1–22, 2021.

[38] J. Wang, G. Dong, J. Sun, X. Wang, and P. Zhang, "Adversarial sample detection for deep neural network through model mutation testing," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, IEEE, 2019, pp. 1245–1256.

[39] X. Du, X. Xie, Y. Li, L. Ma, Y. Liu, and J. Zhao, "Deepstellar: Model-based quantitative analysis of stateful deep learning systems," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 477–487.

[40] Z. Zhang, Y. Li, Y. Guo, X. Chen, and Y. Liu, "Dynamic slicing for deep neural networks," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 838–850.

[41] Z. Zhao, G. Chen, J. Wang, Y. Yang, F. Song, and J. Sun, "Attack as defense: Characterizing adversarial examples using robustness," in *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2021, pp. 42–55.

[42] M. Naseer, S. Khan, and F. Porikli, "Local gradients smoothing: Defense against localized adversarial attacks," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019, pp. 1300–1307.

[43] S. Rao, D. Stutz, and B. Schiele, "Adversarial training against location-optimized adversarial patches," in *European Conference on Computer Vision Workshops (ECCV Workshops)*, 2020.

[44] Z. Chen, P. Dash, and K. Pattabiraman, "Turning your strength against you: Detecting and mitigating robust and universal adversarial patch attack," *arXiv preprint arXiv:2108.05075*, 2021.

[45] E. Chou, F. Tramer, and G. Pellegrino, "Sentinet: Detecting localized universal attacks against deep learning systems," in *2020 IEEE Security and Privacy Workshops (SPW)*, IEEE, 2020, pp. 48–54.

[46] G. Rossolini, F. Nesti, G. D'Amico, S. Nair, A. Biondi, and G. Buttazzo, "On the real-world adversarial robustness of real-time semantic segmentation models for autonomous driving," *arXiv preprint arXiv:2201.01850*, 2022.

[47] P.-Y. Chiang, R. Ni, A. Abdelkader, C. Zhu, C. Studor, and T. Goldstein, "Certified defenses for adversarial patches," in *8th International Conference on Learning Representations (ICLR)*, 2020.

[48] C. Xiang, A. N. Bhagoji, V. Sehwag, and P. Mittal, "Patchguard: A provably robust defense against adversarial patches via small receptive fields and masking," in *30th USENIX Security Symposium (USENIX Security 21)*, USENIX Association, Aug. 2021. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/xiang.

[49] A. Levine and S. Feizi, "(De)randomized smoothing for certifiable defense against patch attacks," in *Conference on Neural Information Processing Systems, (NeurIPS)*, 2020.

[50] Z. Zhang, B. Yuan, M. McCoyd, and D. Wagner, "Clipped bagnet: Defending against sticker attacks with clipped bag-of-features," in *3rd Deep Learning and Security Workshop (DLS)*, 2020.

[51] J. H. Metzen and M. Yatsura, "Efficient certified defenses against patch attacks on image classifiers," in *9th International Conference on Learning Representations (ICLR)*, 2021. [Online]. Available: https://openreview.net/forum?id=hr-3PMvDpil.

[52] H. Salman, S. Jain, E. Wong, and A. Mądry, "Certified patch robustness via smoothed vision transformers," *arXiv preprint arXiv:2110.07719*, 2021.

[53] M. McCoyd *et al.*, "Minority reports defense: Defending against adversarial patches," in *Applied Cryptography and Network Security Workshops (ACNS Workshops)*, vol. 12418, Springer, 2020, pp. 564–582.

[54] C. Xiang and P. Mittal, "Patchguard++: Efficient provable attack detection against adversarial patches," *arXiv preprint arXiv:2104.12609*, 2021.

[55] H. Han *et al.*, "Scalecert: Scalable certified defense against adversarial patches with sparse superficial layers," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[56] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *6th International Conference on Learning Representations (ICLR)*, 2018.

[57] A. Raghunathan, S. M. Xie, F. Yang, J. C. Duchi, and P. Liang, "Adversarial training can hurt generalization," *arXiv preprint arXiv:1906.06032*, 2019.

[58] A. Raghunathan, S. M. Xie, F. Yang, J. Duchi, and P. Liang, "Understanding and mitigating the tradeoff between robustness and accuracy," *arXiv preprint arXiv:2002.10716*, 2020.

[59] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[60] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, 2017, pp. 506–519.

[61] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (S&P)*, 2017, pp. 39–57.

[62] Y. Sharma and P.-Y. Chen, "Bypassing feature squeezing by increasing adversary strength," *arXiv preprint arXiv:1803.09868*, 2018.

[63] L. Ma *et al.*, "Deepct: Tomographic combinatorial testing for deep learning systems," in *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, IEEE, 2019, pp. 614–618.

[64] L. Ma *et al.*, "Deepmutation: Mutation testing of deep learning systems," in *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*, IEEE, 2018, pp. 100–111.

[65] X. Xie *et al.*, "Deephunter: A coverage-guided fuzz testing framework for deep neural networks," in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2019, pp. 146–157.

[66] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th international conference on software engineering*, 2018, pp. 303–314.

[67] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems," in *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, IEEE, 2018, pp. 132–142.

[68] S. Wang and Z. Su, "Metamorphic object insertion for testing object detection systems," in *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, IEEE, 2020, pp. 1053–1065.

[69] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," in *proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 1–18.

[70] L. Ma *et al.*, "Deepgauge: Multi-granularity testing criteria for deep learning systems," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 120–131.

[71] Z. Gong, W. Wang, and W.-S. Ku, "Adversarial and clean data are not twins," *arXiv preprint arXiv:1704.04960*, 2017.

[72] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," in *5th International Conference on Learning Representations (ICLR)*, 2017.

[73] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel, "On the (statistical) detection of adversarial examples," *arXiv preprint arXiv:1702.06280*, 2017.

[74] F. Carrara, R. Becarelli, R. Caldelli, F. Falchi, and G. Amato, "Adversarial examples detection in features distance spaces," in *Proceedings of the European conference on computer vision (ECCV) workshops*, 2018, pp. 0–0.

[75] J. Wang *et al.*, "Smsnet: A new deep convolutional neural network model for adversarial example detection," *IEEE Transactions on Multimedia*, vol. 24, pp. 230–244, 2021.

[76] A. Abusnaina *et al.*, “Adversarial example detection using latent neighborhood graph,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 7687–7696.

[77] N. Liu, H. Yang, and X. Hu, “Adversarial detection with model interpretation,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1803–1811.

[78] P. Yang, J. Chen, C.-J. Hsieh, J.-L. Wang, and M. Jordan, “Ml-loo: Detecting adversarial examples with feature attribution,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 6639–6647.

[79] D. Meng and H. Chen, “Magnet: A two-pronged defense against adversarial examples,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017, pp. 135–147.

[80] P. Samangouei, M. Kabkab, and R. Chellappa, “Defense-gan: Protecting classifiers against adversarial attacks using generative models,” *arXiv preprint arXiv:1805.06605*, 2018.

[81] W. Xu, D. Evans, and Y. Qi, “Feature squeezing: Detecting adversarial examples in deep neural networks,” in *25th Annual Network and Distributed System Security Symposium (NDSS)*, 2018.

[82] N. Carlini and D. A. Wagner, “Adversarial examples are not easily detected: Bypassing ten detection methods,” in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security (AISec@CCS)*, 2017, pp. 3–14.

[83] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, “Reluplex: An efficient smt solver for verifying deep neural networks,” in *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*, Springer, 2017, pp. 97–117.

[84] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev, “Ai2: Safety and robustness certification of neural networks with abstract interpretation,” in *2018 IEEE symposium on security and privacy (SP)*, IEEE, 2018, pp. 3–18.

[85] A. Krizhevsky, V. Nair, and G. Hinton, “Cifar-10 (canadian institute for advanced research),” [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html.

[86] M. Lécuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, “Certified robustness to adversarial examples with differential privacy,” in *2019 IEEE Symposium on Security and Privacy (S&P)*, 2019, pp. 656–672.

[87] J. M. Cohen, E. Rosenfeld, and J. Z. Kolter, “Certified adversarial robustness via randomized smoothing,” in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019, pp. 1310–1320.

[88] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li, “ImageNet: A large-scale hierarchical image database,” in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255.

[89] W. Brendel, *Pretrained bag-of-local-features neural networks*, https://github.com/wielandbrendel/bag-of-local-features-models, 2020.

[90] E. Wong and Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *International Conference on Machine Learning*, PMLR, 2018, pp. 5286–5295.

[91] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *International Conference on Learning Representations*, 2018.

[92] Z. Zhang, C. Xie, J. Wang, L. Xie, and A. L. Yuille, "Deepvoting: A robust and explainable deep network for semantic part detection under partial occlusion," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1372–1380.

[93] A. Acsintoae *et al.*, "Ubnormal: New benchmark for supervised open-set video anomaly detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 20 143–20 153.

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

[95] W. Brendel and M. Bethge, "Approximating CNNs with bag-of-local-features models works surprisingly well on ImageNet," in *7th International Conference on Learning Representations (ICLR)*, 2019.

[96] Y. Geifman and R. El-Yaniv, "Selectivenet: A deep neural network with an integrated reject option," in *International Conference on Machine Learning*, PMLR, 2019, pp. 2151–2159.

[97] P. Cousot and R. Cousot, "Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints," in *Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, 1977, pp. 238–252.

[98] D. J. Richardson and L. A. Clarke, "Partition analysis: A method combining testing and verification," *IEEE Transactions on Software Engineering*, no. 12, pp. 1477–1490, 1985.

[99] M. Castro, B. Liskov, *et al.*, "Practical byzantine fault tolerance," in *OsDI*, vol. 99, 1999, pp. 173–186.

[100] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 1999.

[101] K. Kawaguchi, L. P. Kaelbling, and Y. Bengio, "Generalization in deep learning," *arXiv preprint arXiv:1710.05468*, 2017.

[102] A. Krizhevsky, *Learning multiple layers of features from tiny images*, https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf, 2009.

[103] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2012, pp. 1106–1114.

[104] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The german traffic sign recognition benchmark: A multi-class classification competition," in *The 2011 international joint conference on neural networks*, IEEE, 2011, pp. 1453–1460.

[105] L. Bossard, M. Guillaumin, and L. V. Gool, "Food-101–mining discriminative components with random forests," in *European conference on computer vision*, Springer, 2014, pp. 446–461.

[106] R. Wightman, *Pytorch image models*, https://github.com/rwightman/pytorch-image-models, 2019. DOI: 10.5281/zenodo.4414861.

[107] H. Su, J. Deng, and L. Fei-Fei, "Crowdsourcing annotations for visual object detection," in *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[108] J. He *et al.*, "Partimagenet: A large, high-quality dataset of parts," *arXiv preprint arXiv:2112.00933*, 2021.

[109] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness," *arXiv preprint arXiv:1811.12231*, 2018.

[110] J. Gu, V. Tresp, and Y. Qin, "Evaluating model robustness to patch perturbations," in *ICML 2022 Shift Happens Workshop*, 2022.