PatchDEMUX: A Certifiably Robust Framework for Multi-label Classifiers Against Adversarial Patches

Dennis Jacob
UC Berkeley
Berkeley, CA
djacob18@berkeley.edu

Chong Xiang
Princeton University
Princeton, NJ
cxiang@princeton.edu

Prateek Mittal
Princeton University
Princeton, NJ
pmittal@princeton.edu

Abstract

Deep learning techniques have enabled vast improvements in computer vision technologies. Nevertheless, these models are vulnerable to adversarial patch attacks which catastrophically impair performance. The physically realizable nature of these attacks calls for certifiable defenses, which feature provable guarantees on robustness. While certifiable defenses have been successfully applied to single-label classification, limited work has been done for multi-label classification. In this work, we present PatchDEMUX, a certifiably robust framework for multi-label classifiers against adversarial patches. Our approach is a generalizable method which can provably extend any existing certifiable defense for single-label classification; this is done by considering the multi-label classification task as a series of isolated binary classification problems. In addition, because one patch can only be placed at one location, we further develop a novel certification procedure that provides a tighter robustness certification bound. Using the current state-of-the-art (SOTA) single-label certifiable defense PatchCleanser as a backbone, we find that PatchDEMUX can achieve non-trivial robustness on the MSCOCO 2014 validation dataset while maintaining high clean performance.

1. Introduction

Deep learning-based computer vision systems have helped transform modern society, contributing to the development of technologies such as self-driving cars, facial recognition, and more [12]. Unfortunately these performance boosts have come at a security cost [22]. *Adversarial patches*, which maliciously perturb patch-shaped regions in a target image, are capable of fooling deep learning models despite visual similarity to humans [2]. The patch threat model presents a unique challenge for the security community due to its physically-realizable nature; for instance, adversarial patches can be printed out on billboards or as stickers to breach computer vision systems in the wild [2, 8, 18]. This is particularly alarming for safety critical systems, such as autonomous vehicles.

The importance of adversarial patches has made the design of effective defenses a key research goal. Defense strategies typically fall into one of two categories: *empirical defenses* and *certifiable defenses*. The former leverages clever observations and heuristics to prevent attacks [9, 17]. However, these defenses can be vulnerable to *adaptive attacks*, which exploit fundamental weaknesses in design to bypass the defense [3]. As a result, certifiable defenses against patch attacks have become popular for tasks such as single-label classification and object detection [4, 11, 16, 21, 23–27, 29]; these methods feature provable guarantees on robustness that account for adaptive attacks within the patch threat model. Despite these successes, progress on certifiable defenses against patch attacks has been limited for multi-label classification. Multi-label classification is a key area of interest, as it provides the ability to identify an arbitrary number of classes in an image without the additional overhead of localization.

To address this challenge we propose PatchDEMUX, a certifiably robust framework against patch attacks in the multi-label classification domain. Our design objective is to extend any existing certifiable defense for single-label classification to the multi-label classification domain. For this purpose, we leverage the key insight that any multi-label classifier can be separated into individual binary classification tasks. In addition, by carefully designing the *inference procedure* and *certification procedure* for PatchDEMUX we can bootstrap notions of certified robustness based on precision and recall; these are lower bounds on performance which are guaranteed *across all attack strategies in the patch threat model*. Finally, we leverage the fact that one adversarial patch can only be placed at a single location and develop a location-aware certification procedure with tighter robustness bounds. We demonstrate that PatchDEMUX achieves non-trivial robustness on the MSCOCO 2014 validation dataset while maintaining high performance on clean data. Specifically, when using the current SOTA single-label certifiable defense PatchCleanser as a backbone, PatchDEMUX attains 85.275% average precision on clean data and 44.899% certified robust average precision. For reference, an undefended

model achieves 91.142% average precision on clean data. We hope that future work will integrate with our framework and lead to increased robustness for multi-label classifiers against the patch threat model.

2. Problem Formulation

In this section, we first provide a primer on the multi-label classification task along with standard metrics for evaluation. Next, we outline the adversarial patch threat model. Finally, we discuss the concept of *certifiable defenses* and how they have been used so far to protect single-label classifiers against the patch attack.

2.1. Multi-label classification

In multi-label classification, images $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^{w \times h \times \gamma}$ with width w, height h, and number of channels γ can contain several objects *simultaneously*, with each object corresponding to one of c classes [30]. A classifier is then tasked with recovering each of objects present in an image. Note that this contrasts single-label classification, where *exactly one* object is recovered from an image.

More rigorously, each input datapoint is a pair $(\mathbf{x},\mathbf{y})$ where $\mathbf{x} \in \mathcal{X}$ corresponds to an image and $\mathbf{y} \in \mathcal{Y}$ is the associated image label. Each label $\mathbf{y} \in \mathcal{Y} \subseteq \{0,1\}^c$ is a bitstring where $\mathbf{y}[i] = 1$ means class i is present and $\mathbf{y}[i] = 0$ means class i is absent; note that this implies the set of labels is 2^c in size (i.e., exponential). A *multi-label classifier* $\mathbb{F} : \mathcal{X} \to \mathcal{Y}$ is then trained with a loss function such that the predicted label $\hat{\mathbf{y}} := \mathbb{F}(\mathbf{x})$ is equivalent to $\mathbf{y}$. One popular loss function used for training is asymmetric loss (ASL) [1].

To evaluate the performance of a multi-label classifier, it is common to compute the number of *true positives* (i.e., classes i where $\mathbf{y}[i] = \hat{\mathbf{y}}[i] = 1$), the number of *false positives* (i.e., classes i where $\mathbf{y}[i] = 0$ and $\hat{\mathbf{y}}[i] = 1$), and the number of *false negatives* (i.e., classes i where $\mathbf{y}[i] = 1$ and $\hat{\mathbf{y}}[i] = 0$). These can be summarized by the *precision* and *recall* metrics [30]:

$$precision = \frac{TP}{TP+FP} \qquad recall = \frac{TP}{TP+FN} \tag{1}$$

The values TP, FP, and FN represent the number of true positives, false positives, and false negatives respectively.

2.2. The patch threat model

Theoretical formulation. The patch threat model describes attackers which are limited to arbitrarily adjusting pixel values within a restricted region placed anywhere on a target image $\mathbf{x} \in \mathcal{X}$; the size of this region can be tuned to alter the strength of the attack [2]. As discussed in Sec. 1, defending against this threat model is critical due to its physically realizable nature [2, 8, 18].

We can formally specify patch attacks for an image $\mathbf{x} \in \mathcal{X}$ as follows. Define $\mathcal{R} \subseteq \{0,1\}^{w \times h}$ as the set of binary matrices which represent restricted regions, where elements inside the region are 0 and those outside the region are 1 [25]. Then, the associated patch attacks are:

$$S_{\mathbf{x},\mathcal{R}} := \{\mathbf{r} \circ \mathbf{x} + (\mathbf{1} - \mathbf{r}) \circ \mathbf{x}' | \mathbf{x}' \in \mathcal{X}, \mathbf{r} \in \mathcal{R}\} \tag{2}$$

The $\circ$ operator refers to element-wise multiplication with broadcasting to ensure shape compatibility. Note that this formulation demonstrates how the patch attack can be considered a special case of the ℓ_0-norm threat model [11]. In this work, we primarily focus on defending a single adversarial patch as it is a popular setting in the single-label classifier task [4, 11, 16, 23–25, 27].

Adversarial patches in the multi-label setting. Patch attacks in multi-label classification aim to induce the maximum number of class mismatches between a ground-truth label $\mathbf{y} \in \mathcal{Y}$ and prediction $\hat{\mathbf{y}} \in \mathcal{Y}$. Unlike single-label classification, different types of mismatches are possible in this setting. For instance, patches can increase the number of false negatives and/or the number of false positives predicted by the classifier $\mathbb{F}$. In general, adversarial patches are found by representing the desired objective as an optimization problem and then applying an iterative technique such as projected gradient descent (PGD) over $S_{\mathbf{x},\mathcal{R}}$ [15].

2.3. Certifiable defenses against patch attacks

At a high-level, certifiable defenses against patch attacks (CDPA) provide provable guarantees on performance for deep learning-based computer vision systems $\mathbb{F} : \mathcal{X} \to \mathcal{Y}$ against all possible attacks in the patch threat model [4, 11, 16, 21, 23–27, 29]. This ensures that the defense robustness will not be compromised by future adaptive attacks.

We formulate a CDPA as having an inference procedure and a certification procedure; additional security parameters manage the trade-off between robust performance and inference time [25], denoted by σ. The inference procedure $INFER_{[\mathbb{F},\sigma]} : \mathcal{X} \to \mathcal{Y}$ takes an image $\mathbf{x} \in \mathcal{X}$ as input and outputs a prediction $\hat{\mathbf{y}} \in \mathcal{Y}$. The quality of prediction $\hat{\mathbf{y}}$ with respect to the ground-truth label $\mathbf{y}$ can be evaluated using a performance metric (e.g., precision, recall), which we denote by $\rho : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. In addition to the inference procedure, the certification procedure $CERT_{[\mathbb{F},\sigma]} : \mathcal{X} \times \mathcal{Y} \times \mathbb{P}(\mathcal{R}) \to \mathbb{R}$ ($\mathbb{P}()$ denotes power set) takes image $\mathbf{x}$, ground-truth label $\mathbf{y}$, and the threat model represented by the set of allowable patch regions $\mathcal{R}$ to determine the worst possible performance of $INFER$ on image $\mathbf{x}$. The certification procedure is only used for evaluation. Formally, we have

$$\rho(INFER_{[\mathbb{F},\sigma]}(\mathbf{x}'), \mathbf{y}) \geq \tau, \forall \mathbf{x}' \in S_{\mathbf{x},\mathcal{R}} \tag{3}$$

Here, $\tau := CERT(\mathbf{x},\mathbf{y},\mathcal{R})$ is the lower bound of model prediction quality against an adversary who can use any patch region $\mathbf{r} \in \mathcal{R}$ and introduce arbitrary patch content. Datapoints with a non-trivial lower bound are considered *certifiable*.

We can summarize these concepts as follows.

Definition 1 (CDPA)**.** *A* certifiable defense against patch attacks (CDPA) *for model* $\mathbb{F}: \mathcal{X} \to \mathcal{Y}$ *is a tuple of procedures* $DEF := (INFER_{[\mathbb{F},\sigma]} : \mathcal{X} \to \mathcal{Y}, CERT_{[\mathbb{F},\sigma]} : \mathcal{X} \times \mathcal{Y} \times \mathbb{P}(\mathcal{R}) \to \mathbb{R})$ *where the former is the inference procedure, the latter is the certification procedure, and* $\sigma \subseteq \{0,1\}^*$ *are security parameters. Certifiable datapoints satisfy Eq.* (3) *for a performance metric* $\rho: \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$.

We note that we have different ρ for different tasks. For instance, CDPAs for single-label classifiers ensure that the output label is preserved for certifiable datapoints[1].

Definition 2 (Single-label CDPA)**.** *A* single-label CDPA *is a CDPA for single-label classifiers* $\mathbb{F}_s : \mathcal{X} \to \{1,2,...,c\}$. *The performance metric is* $\rho(y_1, y_2) := [y_1 = y_2]$. *The certification procedure* $CERT$ *evaluates to* 1 *for certifiable datapoints and* 0 *otherwise.*

For multi-label classification, we consider the interpretation where the performance metric is $\rho(\mathbf{y_1}, \mathbf{y_2}) := \Sigma_{i=1}^{c}[\mathbf{y_1}[i] = 1 \cap \mathbf{y_2}[i] = 1]$ and $CERT$ lower bounds the number of true positives. This helps bootstrap robust metrics like certified precision and recall (see Sec. 3.3).

2.4. Certifiable defenses for single-label classifiers against patch attacks

A variety of CDPA have been developed for single-label classifiers [4, 11, 16, 21, 23–25, 27]. Current techniques roughly fall into one of two categories: *small receptive field* defenses and *masking* defenses. With regards to the former, the general idea involves limiting the set of image features exposed to the undefended model and then robustly accumulating results across several evaluation calls. Some examples of this approach include De-randomized Smoothing [11], BagCert [16], and PatchGuard [23, 24]. On the other hand, masking defenses curate a set of masks to provably occlude an adversarial patch regardless of location. PatchCleanser, the current SOTA certifiable defense, uses such a method [25]. Our proposed framework PatchDEMUX is theoretically compatible with any of these techniques.

3. PatchDEMUX Design

In this section we propose *PatchDEMUX*, a certifiably robust framework for multi-label classifiers against patch attacks. We first outline the key property that any multi-label classification problem can be separated into constituent binary classification tasks. Next, we use this observation to construct a generalizable framework which can theoretically integrate any existing single-label CDPA. We then describe the inference and certification procedures in more detail along with robust evaluation metrics. Finally, we propose a novel location-aware certification method which provides tighter robustness bounds.

[1]We use Iverson bracket notation for convenience

3.1. An overview of the defense framework

Isolating binary classifiers in multi-label classification. As discussed in Sec. 2.1, labels $\mathbf{y} \in \{0,1\}^c$ in multi-label classification are bitstrings where $\mathbf{y}[i] \in \{0,1\}$ corresponds to the presence/absence of class $i \in \{1,2,...,c\}$. Note that predictions for each class $\mathbf{y}[i]$ are independent of each other; therefore, the multi-label classification task *can be represented as a series of isolated binary classification problems corresponding to each class*. This motivates a defense formulation for multi-label classifiers in terms of "isolated" binary classifiers, where each class is individually protected by a single-label CDPA. Given a multi-label classifier $\mathbb{F}: \mathcal{X} \to \mathcal{Y}$[2], we use the notation $\mathbb{F}[i]: \mathcal{X} \to \{0,1\}$ to refer to the isolated classifier for class i.

In practice, defining the isolated classifier is complicated as some single-label CDPA designs require architectural restrictions [16, 23, 24]. Nevertheless, a workaround is possible; specifically, we can initialize the multi-label classifier $\mathbb{F}: \mathcal{X} \to \mathcal{Y}$ as an ensemble of c binary classifiers which each satisfy the required architecture. Then, for each class $i \in \{1,2,...,c\}$ we can define the isolated classifier $\mathbb{F}[i]$ as the associated ensemble model. Other defenses are architecture-agnostic [25]. In these cases we can use any off-the-shelf multi-label classifier $\mathbb{F}: \mathcal{X} \to \mathcal{Y}$ and for each class $i \in \{1,2,...,c\}$ define the isolated classifier $\mathbb{F}[i]$ as having outputs $\mathbb{F}[i](\mathbf{x}) := \mathbb{F}(\mathbf{x})[i]$ for all $\mathbf{x} \in \mathcal{X}$.

Our framework. At a high-level, the PatchDEMUX defense framework takes advantage of the isolation principle to extend any existing single-label CDPA to the multi-label classification task. The *PatchDEMUX inference procedure* consists of three stages (see Fig. 1). In the input stage, it preprocesses the input image $\mathbf{x} \in \mathcal{X}$. In the demultiplexing stage it isolates binary classifiers for each class $i \in \{1,2,...,c\}$ and applies the underlying single-label CDPA inference procedure. Finally, in the aggregation stage we return the final prediction vector by pooling results from the individual classes. The *PatchDEMUX certification procedure* works similarly; it separately applies the underlying single-label CDPA certification procedure to each isolated classifier and then robustly accumulates the results.

3.2. PatchDEMUX inference procedure

The PatchDEMUX inference procedure is described in more detail in Algorithm 1. We first procure the inference procedure $SL\text{-}INFER$ from a single-label CDPA and prepare it with security parameters σ. Then, we initialize a $preds \in \{0,1\}^c$ array on line 2 to keep track of individual class predictions. Finally, in line 4 we run $SL\text{-}INFER$ with isolated binary classifier $\mathbb{F}[i]$ on the input image $\mathbf{x}$ and update $preds$ for class i.

Remark. If the time complexity for $SL\text{-}INFER$ is $\mathcal{O}(f(n))$, the time complexity for Algorithm 1 will be $\mathcal{O}(c \cdot f(n))$. However, in practice the isolation principle allows us to run $SL\text{-}INFER$ for each class concurrently; this can reduce the time complexity to $\mathcal{O}(f(n))$.

[2]From here on, $\mathcal{Y}$ will denote a multi-label label set with c classes

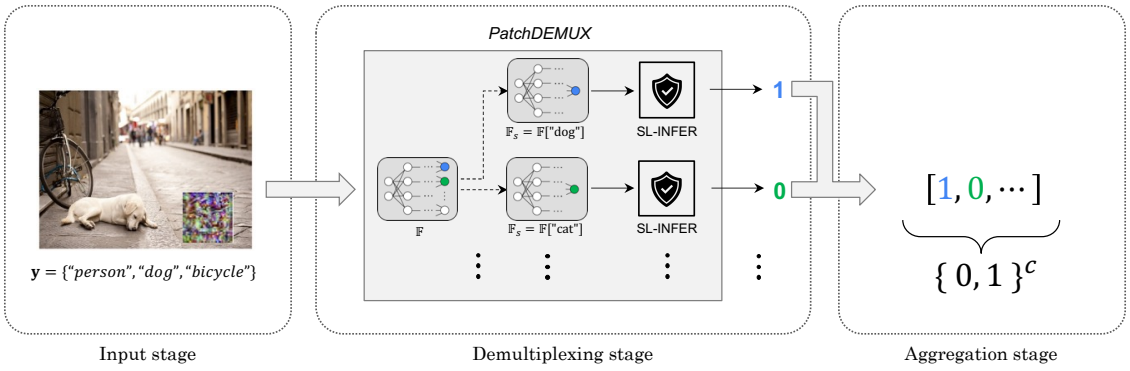


Figure 1. *A diagram which illustrates the demux defense framework from PatchDEMUX. In the* input stage*, the (potentially attacked) image is preprocessed. In the* demultiplexing stage*, the $SL\text{-}INFER$ inference procedure from a single-label CDPA is applied to each individual class in the multi-classification task. This is done by considering the multi-label classifier $\mathbb{F}$ as a series of isolated binary classifiers $\mathbb{F}[i]$ for $i \in \{1,2,...,c\}$. Finally, in the* aggregation stage *the individual outputs are returned as a single label.*

Algorithm 1 *The inference procedure associated with PatchDEMUX*

Input: Image $\mathbf{x} \in \mathcal{X}$, multi-label classifier $\mathbb{F} : \mathcal{X} \to \mathcal{Y}$, inference procedure $SL\text{-}INFER$ and security parameters σ from a single-label CDPA, number of classes c
Output: Prediction $preds \in \{0,1\}^c$

procedure DEMUXINFER($\mathbf{x}$,$\mathbb{F}$,$SL\text{-}INFER$,σ,c)
  $preds \leftarrow \{0\}^c$ ▷ Set predictions to zero vector
  for $i \leftarrow 1$ to c **do** ▷ Consider classes individually
    $preds[i] \leftarrow$SL-INFER$_{[\mathbb{F}[i],\sigma]}(\mathbf{x})$
  end for
  return $preds$
end procedure

3.3. PatchDEMUX certification procedure

The PatchDEMUX certification procedure is outlined in Algorithm 2. We first set up the certification procedure $SL\text{-}CERT$ from a single-label CDPA with security parameters σ. Then we initialize the $\boldsymbol{\kappa}$ array on line 2 to store a list of certifiable classes. Next, in line 5 we run $SL\text{-}CERT$ with isolated binary classifier $\mathbb{F}[i]$ on the datapoint $(\mathbf{x},\mathbf{y}[i])$ and return the result in $\boldsymbol{\kappa}[i]$; recall from Definition 2 that $SL\text{-}CERT$ returns 1 for protected datapoints and 0 otherwise. Finally, on lines $7-10$ we count a *successful true positive* for classes with $\mathbf{y}[i]=1$ that also have $\boldsymbol{\kappa}[i]=1$. Otherwise, we assign a false negative or false positive as we cannot guarantee the accuracy of these classes. We now establish the correctness of these bounds.

Theorem 1 (Algorithm 2 Correctness)**.** *Suppose we have an image data point $(\mathbf{x},\mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, a single-label CDPA $SL\text{-}DEF$, and a multi-label classification model $\mathbb{F} : \mathcal{X} \to \mathcal{Y}$. Then, under the patch threat model $S_{\mathbf{x},\mathcal{R}}$ the bounds returned by Algorithm 2 are correct.*

Proof. See *Supplementary Material*, Appendix A. □

Algorithm 2 *The certification procedure associated with PatchDEMUX*

Input: Image $\mathbf{x} \in \mathcal{X}$, ground-truth $\mathbf{y} \in \mathcal{Y}$, multi-label classifier $\mathbb{F} : \mathcal{X} \to \mathcal{Y}$, certification procedure $SL\text{-}CERT$ and security parameters σ from a single-label CDPA, patch locations $\mathcal{R}$
Output: Certified number of true positives TP_{lower}, false positives upper bound FP_{upper}, false negatives upper bound FN_{upper}, class certification list $\boldsymbol{\kappa}$

1: **procedure** DEMUXCERT($\mathbf{x}$,$\mathbf{y}$,$\mathbb{F}$,$SL\text{-}CERT$,σ,$\mathcal{R}$)
2: $c \leftarrow$ len($\mathbf{y}$)
3: $\boldsymbol{\kappa} \leftarrow \{0\}^c$
4: **for** $i \leftarrow 1$ to c **do** ▷ Certify each class separately
5: $\boldsymbol{\kappa}[i] \leftarrow$SL-CERT$_{[\mathbb{F}[i],\sigma]}(\mathbf{x},\mathbf{y}[i],\mathcal{R})$
6: **end for**
7: ▷ Compute robust metrics
8: $TP_{lower}, FP_{upper}, FN_{upper} \leftarrow 0,0,0$
9: $TP_{lower} \leftarrow \Sigma_{i=1}^{c}[\boldsymbol{\kappa}[i]=1 \cap \mathbf{y}[i]=1]$
10: $FP_{upper} \leftarrow \Sigma_{i=1}^{c}[\boldsymbol{\kappa}[i]=0 \cap \mathbf{y}[i]=0]$
11: $FN_{upper} \leftarrow \Sigma_{i=1}^{c}[\boldsymbol{\kappa}[i]=0 \cap \mathbf{y}[i]=1]$
12: **return** $TP_{lower}, FP_{upper}, FN_{upper}, \boldsymbol{\kappa}$
13: **end procedure**

Thus, using Algorithm 2 we have established a *lower bound* for true positives (TP_{lower}) and an *upper bound* for both false positives (FP_{upper}) and false negatives (FN_{upper}) regardless of an attempted patch attack. This allows us to bootstrap notions of *certified precision* and *certified* recall by referencing Eq. (1):

$$certifiedprecision = \frac{TP_{lower}}{TP_{lower}+FP_{upper}} \tag{4}$$

$$certifiedrecall = \frac{TP_{lower}}{TP_{lower}+FN_{upper}} \tag{5}$$

Note that both metrics will serve as the lower bounds for

precision and recall on a datapoint $(\mathbf{x}, \mathbf{y})$ *irrespective of an attempted patch attack*. Additionally, by *micro-averaging* these metrics across datapoints we can determine the lower bounds on precision and recall for an entire dataset [30].

3.4. Location-aware certification

We now discuss an improved method which extends Algorithm 2 called *location-aware certification*. The general intuition is that if we track vulnerable patch locations for each class, we can use the constraint that an adversarial patch can only be placed at one location to extract stronger robustness guarantees. For instance, suppose we have an image with a dog, a bicycle, and people (see Fig. 2). If we directly apply Algorithm 2, it is possible that each of these classes would individually fail to be certified (i.e., certified recall is 0). However, this method does not account for the fact that different classes may be vulnerable at different locations; for example, the "dog" and "bicycle" classes might be at risk in the bottom left corner of the image, while the "people" class is at risk near the top. Because the patch cannot exist in two places simultaneously, at least one class must be robust and the actual certified recall will be $1/3$.

3.4.1. Tracking vulnerable patch locations

In this section, we give a formal treatment of our core ideas. Suppose we have a single-label CDPA $SL\text{-}DEF$. For many existing single-label defenses, it is possible to relate the associated certification procedure $SL\text{-}CERT$ to the complete list of patch locations $\mathcal{R}$ from Eq. (2) [4, 11, 16, 21, 23–25, 27]. In these cases, we extend Definition 2 and allow $SL\text{-}CERT$ to additionally return a vulnerability status array, which we denote by $\boldsymbol{\lambda} \in \{0,1\}^{|\mathcal{R}|}$. A value of 1 implies the image $\mathbf{x} \in \mathcal{X}$ is protected from attacks in the corresponding patch location $\mathbf{r} \in \mathcal{R}$, while 0 means it is not.

This provides a convenient formulation with which to express our improved method. Consider a multi-label classifier $\mathbb{F}: \mathcal{X} \to \mathcal{Y}$. We first obtain vulnerability status arrays $\boldsymbol{\lambda}$ for each class in Algorithm 2 that could not be certified; this is done by isolating the associated binary classifiers. We then note that given k classes of a common failure mode (i.e., FN or FP), the sum of the inverted arrays $\mathbf{1} - \boldsymbol{\lambda}$ will represent the frequency of the failure type at each patch location. The key insight is that the maximum value, v_{opt}, from the combined array will represent the patch location $\mathbf{r_{opt}} \in \mathcal{R}$ of the image most vulnerable to a patch attack; an attacker must place an adversarial patch at this location to maximize malicious effects. Note however that it is possible $v_{opt} < k$. Then, as per the construction of each $\boldsymbol{\lambda}$ these $k - v_{opt} > 0$ classes will be *guaranteed robustness under the optimal patch location.*

3.4.2. Proposing our novel algorithm

These insights are encapsulated by Algorithm 3, the location-aware certification method for false negatives.[3] It works by first computing robustness bounds for data point $(\mathbf{x}, \mathbf{y})$ via Algorithm 2. On line 5 we determine the false negative classes that failed certification in Algorithm 2. During the *for* loop on lines 8–12, we extract the vulnerability status array $\boldsymbol{\lambda}$ for each false negative by isolating the associated binary classification task. Finally, we sum the inverted arrays $\mathbf{1} - \boldsymbol{\lambda}$ on line 13 and pick the patch location with the largest value; this is the max number of false negatives an attacker can induce at test time. We then alter the lower bound for true positives on line 16.

Algorithm 3 *Location-aware certification for FN*

Input: Image $\mathbf{x} \in \mathcal{X}$, ground-truth $\mathbf{y} \in \mathcal{Y}$, multi-label classifier $\mathbb{F}: \mathcal{X} \to \mathcal{Y}$, certification procedure $SL\text{-}CERT$ and security parameters σ from a single-label CDPA, patch locations $\mathcal{R}$
Output: Certified number of true positives TP_{new}, false negatives upper bound FN_{new}

1: **procedure** LocCert($\mathbf{x}$,$\mathbf{y}$,$\mathbb{F}$,$SL\text{-}CERT$,σ,$\mathcal{R}$)
2: ▷ Pass all args to DemuxCert(...)
3: TP,FP,FN,$\boldsymbol{\kappa} \leftarrow$ DemuxCert(...)
4: ▷ Initialize array with list of FN indices
5: $c \leftarrow \text{len}(\mathbf{y})$
6: $fnIdx \leftarrow \text{list}(\{1 \le i \le c : \boldsymbol{\kappa}[i] = 0 \cap \mathbf{y}[i] = 1\})$
7: $fnCertFails \leftarrow [0]^{FN \times |\mathcal{R}|}$
8: **for** $k \leftarrow 1$ to FN **do** ▷ Isolate each FN classifier
9: $\mathbb{F}_s \leftarrow \mathbb{F}[fnIdx[k]]$
10: $\boldsymbol{\lambda} \leftarrow$ SL-CERT$_{[\mathbb{F}_s,\sigma]}(\mathbf{x},\mathbf{y}[i],\mathcal{R})$
11: $fnCertFails[k] = \mathbf{1} - \boldsymbol{\lambda}$
12: **end for**
13: $fnTotal \leftarrow \text{sum}(fnCertFails, \text{dim} = 0)$
14: $FN_{new} = \max(fnTotal)$ ▷ Pick worst location
15: $TP_{new} = TP + (FN - FN_{new})$
16: **return** TP_{new}, FN_{new}
17: **end procedure**

We now demonstrate the superiority of the bounds from Algorithm 3 in comparison to Algorithm 2.

Theorem 2 (Algorithm 3 Correctness)**.** *Suppose we have an image data point $(\mathbf{x},\mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, a single-label CDPA $SL\text{-}DEF$, and a multi-label classification model $\mathbb{F}: \mathcal{X} \to \mathcal{Y}$. If $SL\text{-}CERT$ returns the vulnerability status array $\boldsymbol{\lambda}$ associated with each $\mathbf{r} \in \mathcal{R}$, then under the patch threat model $S_{\mathbf{x},\mathcal{R}}$ the bounds from Algorithm 3 are correct and stronger than Algorithm 2.*

Proof. See *Supplementary Material*, Appendix A. □

We note that an analogue for Theorem 2 also exists for FP bounds, and can be proved the same way albeit referencing a correspondingly modified version of Algorithm 3.

[3] Obtaining FP_{new} is similar, with line 5 changed to track FP indices

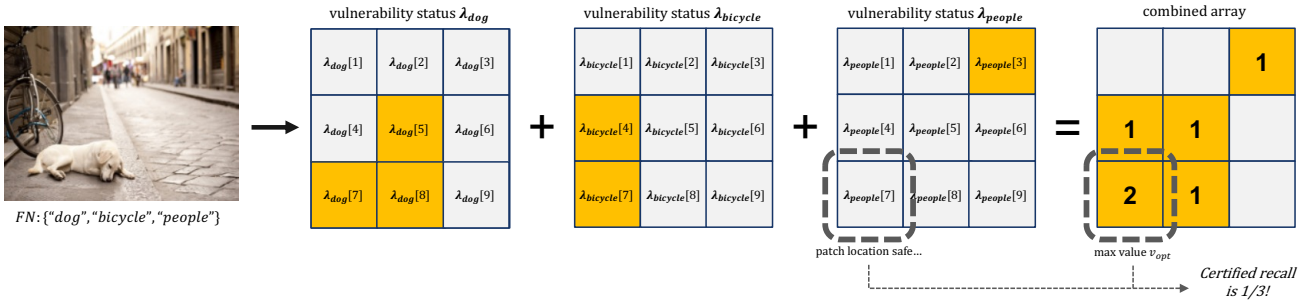


Figure 2. *A diagram which illustrates the key intuition for the location-aware approach. In the sample image we assume all three objects (i.e., "dog", "bicycle", "people") are false negatives. Thus, for each FN we extract the vulnerability status over all patch locations (orange means vulnerable) and accumulate them to find the most vulnerable patch location; this happens to be in the bottom left corner of the image. However, the "people" class by itself is not vulnerable to this location; thus, we can claim stronger robustness bounds than initially suggested by Algorithm 2.*

4. Main Results

4.1. Setup

Backbone initialization and parameters. Recall as discussed in Sec. 3.1 that PatchDEMUX requires an underlying single-label CDPA to operate. For our experiments we choose PatchCleanser, as it is the current SOTA single-label CDPA and it is architecture-agnostic (i.e., it is compatible with any off-the-shelf multi-label classifier) [25]. PatchCleanser works by using a novel double-masking algorithm along with a specially generated mask set to provably remove an arbitrary adversarial patch [25]. The mask generation process has two security parameters. The first is the number of masks in each axis of the image $k_1 \times k_2$; this serves as a "computational budget" where more masks requires longer inference but can result in stronger robustness [25]. The second is the estimated size of the patch p in pixels. Our experiments with PatchDEMUX leverage 6×6 masks and assume the patch is $\sim 2\%$ of the overall image size, which are the default settings in Xiang et al. [25]; we vary these parameters in Sec. 5. For more details on how PatchCleanser fits into the PatchDEMUX framework see *Supplementary Material*, Appendix B.

Dataset and model architectures. Our experiments are done on the 2014 MSCOCO validation dataset, a collection of $\sim 41,000$ images containing several "common objects in context"[13]. This is a popular benchmark for multi-label classification [1, 14, 19, 28], and as such we adopt it for this work due to its accessibility and challenging nature. For the multi-label classification model, we consider two architectures. The first is a ResNet-based architecture from Ben-Baruch et al. [1], which leverages convolution kernels to extract useful features and perform inference. The second is a vision transformer-based (ViT) architecture from Liu et al. [14], which uses the self-attention mechanism for inference [7, 14, 28]. Both models are chosen as they are readily accessible and perform well on MSCOCO. Finally, we also apply different pre-training methods (i.e., Random Cutout [6], Greedy Cutout [20]) to obtain stronger performance.

Evaluation settings and metrics. Due to its certifiable nature, PatchDEMUX will have several evaluation settings.

1. *Undefended clean:* This setting represents evaluation on clean data without the PatchDEMUX defense.
2. *Defended clean:* This setting refers to evaluation on clean data with the PatchDEMUX defense.
3. *Certified robust:* This setting represents lower bounds on performance determined using Algorithm 2.
4. *Location-aware robust:* This setting represents tighter bounds from Algorithm 3. We consider the worst-case attacker (for more details see *Supplementary Material*, Appendix E).

The first two are *clean settings*, where precision and recall metrics are empirically computed for each datapoint. The latter two are *certified robust settings*, where certified precision and certified recall metrics are computed using Algorithm 2 and Algorithm 3.

In all four evaluation settings we micro-average metrics over the entire dataset [30]. In addition, we sweep model outputs across a range of threshold values to create a set of *precision-recall plots*. The associated area-under-curve (AUC) values quantitatively aggregate performance and effectively serve as measures of *average precision* (AP); more details on this computation are in *Supplementary Material*, Appendix C.

4.2. PatchDEMUX overall performance

In this section we report our main results for PatchDEMUX. We summarize the precision values associated with key recall levels in Tab. 1. Fig. 3 features precision-recall plots, while AP values are present in Tab. 1. Because the ViT architecture demonstrates strictly better performance compared to the Resnet architecture, we focus on the ViT model here (see *Supplementary Material*, Appendix D for Resnet performance).

High clean performance. As shown in Tab. 1a and Fig. 3a, the inference procedure from PatchDEMUX features excellent clean performance. In fact, the AP associated with the defended clean setting is able to capture $\sim 94\%$ of the performance

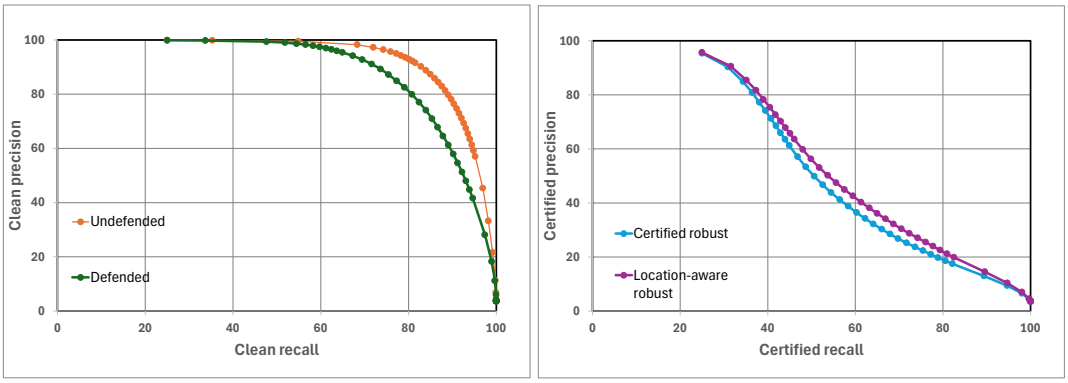
Table 1. *PatchDEMUX performance with ViT architecture on the MSCOCO* 2014 *validation dataset. Precision values are evaluated at key recall levels along with the approximated average precision. The threat model assumes the patch attack is at most* 2% *of the image area.*

(a) *Clean setting precision values*

Architecture	ViT			
Clean recall	25%	50%	75%	*AP*
Undefended	99.930	99.704	96.141	91.142
Defended	99.894	99.223	87.764	85.275

(b) *Certified robust setting precision values*

Architecture	ViT			
Certified recall	25%	50%	75%	*AP*
Certified robust	95.369	50.950	22.662	41.759
Location-aware	95.670	56.038	26.375	44.899



(a) *Clean setting precision-recall curves* (b) *Certified robust setting precision-recall curves*

Figure 3. *PatchDEMUX precision-recall curves with ViT architecture over the MSCOCO 2014 validation dataset. We consider the clean and certified robust evaluation settings. The threat model assumes that the adversarial patch is at most* 2% *of the image area.*

achieved by the undefended model. These results demonstrate that PatchDEMUX can be deployed at test time with minimal loss in performance utility.

Non-trivial robustness. The results from Tab. 1b and Fig. 3b demonstrate that PatchDEMUX achieves non-trivial certifiable robustness on the MSCOCO 2014 validation dataset. For instance, when fixed at 50% certified recall PatchDEMUX features 56.038% certified precision. This performance remains stable across a variety of thresholds, as evidenced by the 44.899% certified AP value. Note that boosts from location-aware certification play a key role in these results. For instance, the location-aware robust setting improves certified AP by almost 3 points compared to the certified robust setting. Improvements are most notable in the *mid recall-mid precision* region of the certified robust precision-recall plot (Fig. 3b), where a benefit of $\sim 5-6$ points to precision is present.

Interestingly, the defended clean precision-recall plot (Fig. 3a) is concave in shape while the certified robust plots (Fig. 3b) are slightly convex. This gap in performance is likely due to the underlying certification procedure of PatchCleanser, which is sensitive to object occlusion by the generated mask set. This limitation is compounded by the fact that MSCOCO images often contain several objects which are small relative to image size [13, 25].

4.3. Ablation studies

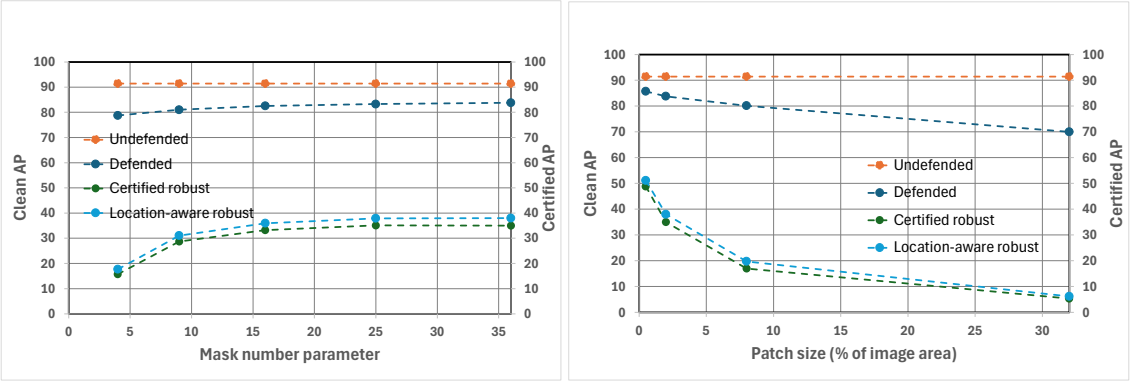
We next perform a set of ablation studies for PatchDEMUX. In one investigation, we empirically compare different attackers in the location-aware robust setting and find that attacks targeting false positives are relatively "weaker" (see *Supplementary Material*, Appendix E). We also investigate different pre-training routines, and find that variants of cutout pre-training (i.e., Random Cutout [6], Greedy Cutout [20]) can boost model performance compared to the vanilla PatchCleanser method (see *Supplementary Material*, Appendix F); the strongest results for the defended clean setting are featured in the previous section.

5. Security Parameter Experiments

Recall from Sec. 4.1 that the PatchCleanser backbone leverages two security parameters: the number of masks desired in each axis $k_1 \times k_2$ (i.e., the "computational budget") and the estimated size of the patch p in pixels [25]. In this section, we study how changing these parameters impacts the performance of PatchDEMUX. We use ViT checkpoints without pre-training as our baseline in order to isolate performance changes from security parameter variation.

5.1. Impact of varying mask number

We present results associated with varying the mask number parameter in Fig. 4a (the associated table is in *Supplementary*



(a) *Clean AP and certified AP as a function of mask number.* (b) *Clean AP and certified AP as a function of patch size.*

Figure 4. *The impact of varying PatchCleanser security parameters on PatchDEMUX performance. We compute clean AP for the clean setting evaluations, and certified AP for the certified robust setting evaluations.*

Material, Appendix G). We assume the number of masks in each axis is the same $k := k_1 = k_2$ and evaluate with respect to k^2. We keep the patch size at the default value of $\sim 2\%$.

Limited tradeoff between computational budget and robustness. We find that PatchDEMUX provides consistent defended clean and certified robust performance even after greatly reducing the number of masks. For instance, when decreasing the number of masks from 36 to 16 we find that the AP for all settings decrease at most 2 points. When analyzing the extreme of $k^2 = 4$ masks we notice more substantial performance drops. This is expected, as the underlying mask generation method from PatchCleanser creates larger masks to compensate for reduced mask number; this causes increased occlusion and fewer certification successes [25].

5.2. Impact of varying patch size

We present results associated with varying the patch size estimate in Fig. 4b (the associated table is in *Supplementary Material*, Appendix G). We keep the mask number at the default value of 6×6 masks.

Persistent clean performance over different patch sizes. In general, PatchDEMUX provides relatively strong robustness on smaller patches (i.e., $\leq 2\%$) and performance degrades for larger patches (i.e., $\geq 8\%$); certified AP drops close to 0% once a patch of 32% is considered. This aligns with intuition; a larger patch size requires PatchCleanser to create larger masks, which leads to increased occlusion and certification failures [25]. On the other hand, the defended clean setting of PatchDEMUX demonstrates more resilience to increasing patch size; indeed, the clean AP for this setting only drops from 85.731 against the smallest patch to 69.952 against the largest. This shows that even for unlikely scenarios (i.e., a patch size of $\geq 32\%$ would be easily detectable by hand) PatchDEMUX maintains strong inference performance. These trends align with some of the experiments done by Xiang et al. [25] in the single-label classification domain.

5.3. Overall takeaways

Overall, we find that the performance tradeoffs with PatchDEMUX agree at a high-level with findings from Xiang et al. [25]. This illustrates a key feature of our defense framework: PatchDEMUX successfully adapts the strengths of underlying single-label CDPAs to the multi-label classification setting.

6. Related Work

Certifiable defenses against patch attacks. CDPAs have been designed for various computer vision applications. In single-label classification, defense strategies include bound propagation methods [4], small receptive field methods [11, 16, 23, 24], and masking methods [25]. CDPAs have also been proposed for object detection [26] and semantic segmentation [29], although notions of certifiable robustness are more difficult to define in these domains.

Certifiable defenses in multi-label classification. Jia et al. [10] proposed the first certifiably robust defense for multi-label classifiers by generalizing the randomized smoothing technique [5]. However, it is designed to protect against ℓ_2-norm attacks and does not address the threat of adversarial patches.

7. Conclusion

The threat of adversarial patch attacks has compromised countless computer vision systems, including multi-label classifiers. To this end we introduced PatchDEMUX, a certifiably robust framework for multi-label classifiers against adversarial patches which provably extends any existing single-label CDPA. PatchDEMUX is highly adaptable and configurable, as evidenced by its ability to interface with the current SOTA single-label CDPA PatchCleanser and achieve both strong clean and robust performance on the MSCOCO dataset. We hope that future work will integrate with the PatchDEMUX framework and contribute to increased robustness for multi-label classifiers.

References

[1] Emanuel Ben-Baruch, Tal Ridnik, Nadav Zamir, Asaf Noy, Itamar Friedman, Matan Protter, and Lihi Zelnik-Manor. Asymmetric Loss For Multi-Label Classification. In *ICCV 2021*. arXiv, 2021. arXiv:2009.14119 [cs]. 2, 6, 5, 7

[2] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial Patch. In *NeurIPS 2017 Workshops*. arXiv, 2018. arXiv:1712.09665 [cs]. 1, 2

[3] Nicholas Carlini and David Wagner. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. In *CCS 2017 Workshop on Artificial Intelligence and Security (AISec 2017)*, pages 3–14, Dallas Texas USA, 2017. ACM. 1

[4] Ping-Yeh Chiang, Renkun Ni, Ahmed Abdelkader, Chen Zhu, Christoph Studer, and Tom Goldstein. Certified Defenses for Adversarial Patches. In *ICLR 2020*. arXiv, 2020. arXiv:2003.06693 [cs, stat]. 1, 2, 3, 5, 8

[5] Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified Adversarial Robustness via Randomized Smoothing. In *ICML 2019*. arXiv, 2019. arXiv:1902.02918 [cs, stat]. 8

[6] Terrance DeVries and Graham W. Taylor. Improved Regularization of Convolutional Neural Networks with Cutout, 2017. arXiv:1708.04552 [cs]. 6, 7

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR 2021*. arXiv, 2021. arXiv:2010.11929 [cs]. 6, 5

[8] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust Physical-World Attacks on Deep Learning Visual Classification. In *CVPR 2018*, pages 1625–1634, Salt Lake City, UT, USA, 2018. IEEE. 1, 2

[9] Jamie Hayes. On Visible Adversarial Perturbations & Digital Watermarking. In *CVPR 2018 Workshops (CVPRW 2018)*, pages 1678–16787, Salt Lake City, UT, USA, 2018. IEEE. 1

[10] Jinyuan Jia, Wenjie Qu, and Neil Zhenqiang Gong. MultiGuard: Provably Robust Multi-label Classification against Adversarial Examples. In *NeurIPS 2022*. arXiv, 2022. arXiv:2210.01111 [cs]. 8

[11] Alexander Levine and Soheil Feizi. (De)Randomized Smoothing for Certifiable Defense against Patch Attacks. In *NeurIPS 2020*. arXiv, 2021. arXiv:2002.10733 [cs, stat]. 1, 2, 3, 5, 8

[12] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):6999–7019, 2022. 1

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context. In *ECCV 2014*. arXiv, 2015. arXiv:1405.0312 [cs]. 6, 7

[14] Shilong Liu, Lei Zhang, Xiao Yang, Hang Su, and Jun Zhu. Query2Label: A Simple Transformer Way to Multi-Label Classification, 2021. arXiv:2107.10834 [cs]. 6, 7

[15] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *ICLR 2018*. arXiv, 2019. arXiv:1706.06083 [cs, stat]. 2

[16] Jan Hendrik Metzen and Maksym Yatsura. Efficient Certified Defenses Against Patch Attacks on Image Classifiers. In *ICLR 2021*. arXiv, 2021. arXiv:2102.04154 [cs, stat]. 1, 2, 3, 5, 8

[17] Muzammal Naseer, Salman H. Khan, and Fatih Porikli. Local Gradients Smoothing: Defense against localized adversarial attacks. In *WACV 2019*. arXiv, 2018. arXiv:1807.01216 [cs]. 1

[18] Federico Nesti, Giulio Rossolini, Saasha Nair, Alessandro Biondi, and Giorgio Buttazzo. Evaluating the Robustness of Semantic Segmentation for Autonomous Driving against Real-World Adversarial Patch Attacks. In *WACV 2022*. arXiv, 2021. arXiv:2108.06179 [cs]. 1, 2

[19] Tal Ridnik, Gilad Sharir, Avi Ben-Cohen, Emanuel Ben-Baruch, and Asaf Noy. ML-Decoder: Scalable and Versatile Classification Head. In *WACV 2023*, pages 32–41, Waikoloa, HI, USA, 2023. IEEE. 6

[20] Aniruddha Saha, Shuhua Yu, Mohammad Sadegh Norouzzadeh, Wan-Yi Lin, and Chaithanya Kumar Mummadi. Revisiting Image Classifier Training for Improved Certified Robust Defense against Adversarial Patches. *Transactions on Machine Learning Research*, 2023. 6, 7, 8

[21] Hadi Salman, Saachi Jain, Eric Wong, and Aleksander Madry. Certified Patch Robustness via Smoothed Vision Transformers. In *CVPR 2022*, pages 15116–15126, New Orleans, LA, USA, 2022. IEEE. 1, 2, 3, 5

[22] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014. arXiv:1312.6199 [cs]. 1

[23] Chong Xiang and Prateek Mittal. PatchGuard++: Efficient Provable Attack Detection against Adversarial Patches. In *ICLR 2021 Workshop on Security and Safety in Machine Learning Systems*. arXiv, 2021. arXiv:2104.12609 [cs]. 1, 2, 3, 5, 8

[24] Chong Xiang, Arjun Nitin Bhagoji, Vikash Sehwag, and Prateek Mittal. PatchGuard: A Provably Robust Defense against Adversarial Patches via Small Receptive Fields and Masking. In *USENIX Security 2021*. arXiv, 2021. arXiv:2005.10884 [cs, stat]. 3, 8

[25] Chong Xiang, Saeed Mahloujifar, and Prateek Mittal. PatchCleanser: Certifiably Robust Defense against Adversarial Patches for Any Image Classifier. In *USENIX Security 2022*. arXiv, 2022. arXiv:2108.09135 [cs]. 2, 3, 5, 6, 7, 8

[26] Chong Xiang, Alexander Valtchanov, Saeed Mahloujifar, and Prateek Mittal. ObjectSeeker: Certifiably Robust Object Detection against Patch Hiding Attacks via Patch-agnostic Masking. In *IEEE Symposium on Security and Privacy 2023*. arXiv, 2022. arXiv:2202.01811 [cs]. 8

[27] Chong Xiang, Tong Wu, Sihui Dai, Jonathan Petit, Suman Jana, and Prateek Mittal. PatchCURE: Improving Certifiable Robustness, Model Utility, and Computation Efficiency of Adversarial Patch Defenses. In *USENIX Security 2024*. arXiv, 2024. arXiv:2310.13076 [cs]. 1, 2, 3, 5

[28] Shichao Xu, Yikang Li, Jenhao Hsiao, Chiuman Ho, and Zhu Qi. Open Vocabulary Multi-Label Classification with Dual-Modal Decoder on Aligned Visual-Textual Features, 2023. arXiv:2208.09562 [cs]. 6

[29] Maksym Yatsura, Kaspar Sakmann, N. Grace Hua, Matthias Hein, and Jan Hendrik Metzen. Certified Defences Against Adversarial Patch Attacks on Semantic Segmentation. In *ICLR 2023*. arXiv, 2023. arXiv:2209.05980 [cs]. 1, 2, 8

[30] Min-Ling Zhang and Zhi-Hua Zhou. A Review on Multi-Label Learning Algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837, 2014. 2, 5, 6

PatchDEMUX: A Certifiably Robust Framework for Multi-label Classifiers Against Adversarial Patches

Supplementary Material

A. Certification Robustness Proofs

A.1. Baseline certification correctness

In this section, we provably demonstrate robustness for our baseline certification procedure. Specifically, we prove Theorem 1, which ensures correctness of the bounds returned by Algorithm 2. For convenience, we re-state the theorem.

Theorem 1 (Algorithm 2 Correctness)**.** *Suppose we have an image data point* $(\mathbf{x},\mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$*, a single-label CDPA* $SL\text{-}DEF$*, and a multi-label classification model* $\mathbb{F}: \mathcal{X} \to \mathcal{Y}$*. Then, under the patch threat model* $S_{\mathbf{x},\mathcal{R}}$ *the bounds returned by Algorithm 2 are correct.*

Proof. We first demonstrate that classes included in TP_{lower} will be guaranteed correctness. Consider an arbitrary class $i^* \in \{1, 2, \dots, c\}$ with label $\mathbf{y}[i^*] = 1$. If this class is included in TP_{lower}, then we must have $\boldsymbol{\kappa}[i^*] = 1$ (i.e., line 9 in Algorithm 2). This implies that on line 5 we must have $SL\text{-}CERT_{[\mathbb{F}[i^*],\sigma]}(\mathbf{x},\mathbf{y}[i^*],\mathcal{R}) = 1$. Now consider when Algorithm 1 reaches index $i^* \in \{1,2,...,c\}$ in the *for* loop on line 3. Because the datapoint $(\mathbf{x},\mathbf{y}[i^*])$ was certifiable, by Definition 2 we will have

$$SL\text{-}INFER_{[\mathbb{F}[i^*],\sigma]}(\mathbf{x}') = 1 \quad \forall \mathbf{x}' \in S_{\mathbf{x},\mathcal{R}}$$

This implies that every class accounted for in TP_{lower} will be successfully recovered by Algorithm 1 regardless of the attempted patch attack.

Next, we demonstrate that classes included in FN_{upper} will not be guaranteed correctness. Consider an arbitrary class $i^* \in \{1, 2, \dots, c\}$ with label $\mathbf{y}[i^*] = 1$. In this case we will have $\boldsymbol{\kappa}[i^*] = 0$, and thus classes included in FN_{upper} will have $SL\text{-}CERT_{[\mathbb{F}[i^*],\sigma]}(\mathbf{x},\mathbf{y}[i^*],\mathcal{R}) = 0$. Now consider when Algorithm 1 reaches index $i^* \in \{1,2,...,c\}$ in the *for* loop on line 3. By Definition 2 it is possible that

$$\exists \mathbf{x}' \in S_{\mathbf{x},\mathcal{R}} \quad | \quad SL\text{-}INFER_{[\mathbb{F}[i^*],\sigma]}(\mathbf{x}') = 0$$

Essentially, in the worst-case scenario these classes might be mispredicted and be false negatives. Thus, none of the classes included in FN_{upper} can be guaranteed correctness. Because every class with $\mathbf{y}[i^*] = 1$ will be accounted for by either TP_{lower} or FN_{upper} (mutually exclusive), we conclude that TP_{lower} will be the correct lower bound for objects recovered and FN_{upper} will be the correct upper bound for objects missed.

The correctness of the FP_{upper} bound can be shown in a similar fashion, albeit by considering classes with $\mathbf{y}[i^*] = 0$. □

A.2. Location-aware certification correctness

In this section we demonstrate the correctness of our novel location-based certification method. To do so, it is helpful to use the following lemma.

Lemma 1 (Algorithm 3 Tightness)**.** *Given that we have derived a bound on* FN *using the technique from Algorithm 2, Algorithm 3 will return a new bound* $FN_{new} \leq FN$*.*

Proof. We will show that FN_{new} provides a tighter bound (i.e., the inequality $FN_{new} \leq FN$ is true). To see this, we note as per lines 13 and 14 of Algorithm 3 that the worst-case sum will occur if some patch location is vulnerable for every false negative. Because summation is done over the set of false negatives, this implies the worst-case sum is FN. □

We also provide a formal definition for the concept of a *vulnerability status array*; recall that this array extends the certification procedure for a single-label CDPA (Sec. 3.4.1). We leverage similar notation as Eq. (2).

Definition 3 (Vulnerability status array)**.** *Suppose we have datapoint* $(\mathbf{x},y)$*, a single-label classifier* $\mathbb{F}_s: \mathcal{X} \to \{1,2,...,c\}$*, a certification procedure* $SL\text{-}CERT$ *with security parameters* σ *from a single-label CDPA, and patch locations* $\mathcal{R}$*. Then we define the* vulnerability status array $\boldsymbol{\lambda} := SL\text{-}CERT_{[\mathbb{F}_s,\sigma]}(\mathbf{x},y,\mathcal{R}) \in \{0,1\}^{|\mathcal{R}|}$ *such that if* $\boldsymbol{\lambda}[\mathbf{r}] = 1$ *for a patch location* $\mathbf{r} \in \mathcal{R}$ *then*[4]

$$SL\text{-}INFER_{[\mathbb{F}_s,\sigma]}(\mathbf{r} \circ \mathbf{x} + (\mathbf{1} - \mathbf{r}) \circ \mathbf{x}') = y \quad \forall \mathbf{x}' \in \mathcal{X}$$

Essentially, the vulnerability status array $\boldsymbol{\lambda}$ denotes the certification status of individual patch locations $\mathbf{r} \in \mathcal{R}$.

We can now prove Theorem 2. For convenience we re-state the theorem.

Theorem 2 (Algorithm 3 Correctness)**.** *Suppose we have an image data point* $(\mathbf{x},\mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$*, a single-label CDPA* $SL\text{-}DEF$*, and a multi-label classification model* $\mathbb{F}: \mathcal{X} \to \mathcal{Y}$*. If* $SL\text{-}CERT$ *returns the vulnerability status array* $\boldsymbol{\lambda}$ *associated with each* $\mathbf{r} \in \mathcal{R}$*, then under the patch threat model* $S_{\mathbf{x},\mathcal{R}}$ *the bounds from Algorithm 3 are correct and stronger than Algorithm 2.*

Proof. We will demonstrate the correctness and tightness of the new bound FN_{new} proposed in Algorithm 3. We first note as per Lemma 1 that $FN_{new} \leq FN$; this ensures that the new bound will be stronger than Algorithm 2. In the case with equality $FN_{new} = FN$, correctness is guaranteed by Theorem 1. We thus focus on the case with strict inequality $FN_{new} < FN$.

[4] For the term $\boldsymbol{\lambda}[\mathbf{r}] = 1$ we slightly abuse notation and use $\mathbf{r}$ to refer to the countable index associated with the patch location

Define $\mathbf{r_{opt}} \in \mathcal{R}$ as the patch location which induces the maximum number of false negatives on line 14 of Algorithm 3. By assumption, a total of $FN - FN_{new} > 0$ false negatives will have contributed a value of 0 to the sum $fnTotal[\mathbf{r_{opt}}]$ on line 13. Consider an arbitrary such class $i^* \in \{1,2,...,c\}$. Because the $fnCertFails$ value for this class at patch location $\mathbf{r_{opt}}$ is 0, on line 10 we must have for $\boldsymbol{\lambda} := SL\text{-}CERT_{[\mathbb{F}[i^*],\sigma]}(\mathbf{x},\mathbf{y}[i^*],\mathcal{R})$

$$\boldsymbol{\lambda}[\mathbf{r_{opt}}] = 1$$

As per Definition 3, this means that we will have

$$SL\text{-}INFER_{[\mathbb{F}[i^*],\sigma]}(\mathbf{r_{opt}} \circ \mathbf{x} + (\mathbf{1} - \mathbf{r_{opt}}) \circ \mathbf{x}') = 1 \quad \forall \mathbf{x}' \in \mathcal{X}$$

In other words, $SL\text{-}INFER$ will be robust against any patch attack contained in location $\mathbf{r_{opt}} \in \mathcal{R}$. Because the patch must be placed at the optimal location $\mathbf{r_{opt}}$, this implies that Algorithm 1 will return the correct prediction for class i^* as desired. Overall, each of the $FN - FN_{new}$ classes will now be certified true positives instead of false negatives, and thus the new bounds from Algorithm 3 will be correct. □

B. Double-masking Algorithm from PatchCleanser

In this section, we provide a brief outline of the double-masking algorithm from the PatchCleanser defense and how it integrates into the PatchDEMUX framework; recall from Sec. 4.1 that PatchCleanser is the current SOTA single-label CDPA. For more details, we direct the reader to the original reference by Xiang et al. [25].

B.1. Double-masking overview

At a glance, the double-masking algorithm works by curating a specialized set of masks, $\mathcal{M} \subseteq \{0,1\}^{w \times h}$, to recover the output label $y \in \{1,2,...,c\}$ for certifiable input images $\mathbf{x} \in \mathcal{X}$ regardless of the content and/or location of an adversarial patch [25]. More specifically, these masks satisfy the following $\mathcal{R}$-*covering* property from Xiang et al. [25].

Definition 4 ($\mathcal{R}$-covering)**.** *A mask set $\mathcal{M}$ is $\mathcal{R}$-covering if, for any patch in the patch region set $\mathcal{R}$, at least one mask from the mask set $\mathcal{M}$ can cover the entire patch, i.e.,*

$$\forall \mathbf{r} \in \mathcal{R}, \exists \mathbf{m} \in \mathcal{M} \quad s.t. \quad \mathbf{m}[i,j] \leq \mathbf{r}[i,j], \forall (i,j)$$

Here $\mathcal{R}$ refers to the set of patch locations from Eq. (2), and $\mathcal{M}$ represents binary matrices where elements inside the mask are 0 and elements outside the mask are 1 [25]. Given an input image size $n_1 \times n_2$, an upper estimate on patch size[5] p, and number of desired masks $k_1 \times k_2$, a procedure from Xiang et al. [25] can readily create a mask set $\mathcal{M}$ with stride length $s_1 \times s_2$ and mask size $m_1 \times m_2$ which is $\mathcal{R}$-covering. The patch size p and mask number $k_1 \times k_2$ serve as key security parameters, where the former corresponds to the threat level of $\mathcal{R}$ (i.e., larger patches will necessitate larger masks) and the latter represents a computational budget (i.e., more masks will require more checks to be performed) [25].

Once the $\mathcal{R}$-covering mask set $\mathcal{M}$ is generated, the double-masking inference procedure removes the patch by selectively occluding the image $\mathbf{x} \in \mathcal{X}$ with mask pairs $\mathbf{m_0}, \mathbf{m_1} \in \mathcal{M} \times \mathcal{M}$. Correctness is verified through the associated certification procedure, which checks if predictions on $\mathbf{x}$ are preserved across all possible mask pairs [25].

B.2. Double-masking inference procedure

Algorithm 4 *The double-masking inference procedure from PatchCleanser [25]*

Input: Image $\mathbf{x} \in \mathcal{X}$, single-label classifier $\mathbb{F}_s : \mathcal{X} \to \{1,2,...,c\}$, $\mathcal{R}$-covering mask set $\mathcal{M}$
Output: Prediction $\hat{y} \in \{1,2,...,c\}$

procedure DOUBLEMASKINGINFER($\mathbf{x}, \mathbb{F}_s, \mathcal{M}$)
  $\hat{y}_{maj}, \mathcal{P}_{dis} \leftarrow$ MASKPRED($\mathbf{x}, \mathbb{F}_s, \mathcal{M}$) ▷ First-round
  if $\mathcal{P}_{dis} = \emptyset$ **then**
    return $\hat{y}_{maj}$ ▷ Case I: agreed prediction
  end if
  for each $(\mathbf{m_{dis}}, \hat{y}_{dis}) \in \mathcal{P}_{dis}$ **do** ▷ Second round
    $\hat{y}', \mathcal{P}' \leftarrow$ MASKPRED($\mathbf{x} \circ \mathbf{m_{dis}}, \mathbb{F}_s, \mathcal{M}$)
    if $\mathcal{P}' = \emptyset$ **then**
      return $\hat{y}_{dis}$ ▷ Case II: disagreer pred.
    end if
  end for
  return $\hat{y}_{maj}$ ▷ Case III: majority prediction
end procedure

Input: Image $\mathbf{x} \in \mathcal{X}$, single-label classifier $\mathbb{F}_s : \mathcal{X} \to \{1,2,...,c\}$, $\mathcal{R}$-covering mask set $\mathcal{M}$
Output: Majority prediction $\hat{y}_{maj} \in \{1,2,...,c\}$, disagreer masks $\mathcal{P}_{dis}$

procedure MASKPRED($\mathbf{x}, \mathbb{F}_s, \mathcal{M}$)
  $\mathcal{P} \leftarrow \emptyset$ ▷ A set for mask-prediction pairs
  for $\mathbf{m} \in \mathcal{M}$ **do** ▷ Enumerate every mask $\mathbf{m}$
    $\hat{y} \leftarrow \mathbb{F}_s(\mathbf{x} \circ \mathbf{m})$ ▷ Evaluate masked prediction
    $\mathcal{P} \leftarrow \mathcal{P} \bigcup \{(\mathbf{m}, \hat{y})\}$ ▷ Update set $\mathcal{P}$
  end for
  $\hat{y}_{maj} \leftarrow \text{argmax}_{y^*} |\{(\mathbf{m}, \hat{y}) \in \mathcal{P} | \hat{y} = y^*\}|$ ▷ Majority
  $\mathcal{P}_{dis} \leftarrow \{(\mathbf{m}, \hat{y}) \in \mathcal{P} | \hat{y} \neq \hat{y}_{maj}\}$ ▷ Disagreers
  return $\hat{y}_{maj}, \mathcal{P}_{dis}$
end procedure

The double-masking inference procedure from Xiang et al. [25] is outlined in Algorithm 4. It works by running up to two rounds of masking on the input image $\mathbf{x} \in \mathcal{X}$. In each round, the single-label classifier $\mathbb{F}_s : \mathcal{X} \to \{1,2,...,c\}$ is queried on copies

[5] PatchCleanser provides an option to specify the patch size for each axis; we simplify the notation here for convenience

of $\mathbf{x}$ which have been augmented by masks $\mathbf{m} \in \mathcal{M}$ [25].

- *First-round masking:* The classifier runs $\mathbb{F}_s(\mathbf{m} \circ \mathbf{x})$ for every mask $\mathbf{m} \in \mathcal{M}$ (line 2). If there is consensus, this is returned as the overall prediction (line 4); the intuition is that a clean image with no patch will be predicted correctly regardless of the mask present [25]. Otherwise, the minority/"disagreer" predictions trigger a second-round of masking (line 6). This is done to determine whether to trust the majority prediction $\hat{y}_{maj}$ or one of the disagreers [25].
- *Second-round masking:* For each disagreer mask $\mathbf{m_{dis}}$, the classifier runs $\mathbb{F}_s(\mathbf{x} \circ \mathbf{m_{dis}} \circ \mathbf{m})$ for every mask $\mathbf{m} \in \mathcal{M}$ to form *double-mask predictions* [25]. If there is consensus, the disagreer label $\hat{y}_{dis}$ associated with $\mathbf{m_{dis}}$ is returned as the overall prediction (lines $6-10$). The intuition is that consensus is likely to occur if $\mathbf{m_{dis}}$ successfully covered the patch [25]. Otherwise, $\mathbf{m_{dis}}$ is ignored and the next available disagreer mask is considered; the assumption here is that $\mathbf{m_{dis}}$ failed to cover the patch [25]. Finally, if none of the disagreer masks feature consensus the majority label $\hat{y}_{maj}$ from the first-round is returned instead (line 12).

A key property of this method is that it is architecture agnostic and can be integrated with any single-label classifier [25].

B.3. Double-masking certification procedure

Algorithm 5 *The double-masking certification procedure from PatchCleanser [25]*

Input: Image $\mathbf{x} \in \mathcal{X}$, ground-truth $y \in \{1,2,...,c\}$, single-label classifier $\mathbb{F}_s : \mathcal{X} \to \{1,2,...,c\}$, patch locations $\mathcal{R}$, $\mathcal{R}$-covering mask set $\mathcal{M}$
Output: Overall certification status of $(\mathbf{x},y)$, vulnerability status array $\boldsymbol{\lambda} \in \{0,1\}^{|\mathcal{M}|}$

1: **procedure** DOUBLEMASKINGCERT($\mathbf{x},y,\mathbb{F}_s,\mathcal{R},\mathcal{M}$)
2: $certVal \leftarrow 1$
3: $\boldsymbol{\lambda} \leftarrow [1]^{|\mathcal{M}|}$
4: **if** $\mathcal{M}$ is not $\mathcal{R}$-covering **then** ▷ Insecure mask set
5: **return** 0, $[0]^{|\mathcal{M}|}$
6: **end if**
7: **for** every $(\mathbf{m_0},\mathbf{m_1}) \in \mathcal{M} \times \mathcal{M}$ **do**
8: $\hat{y}' \leftarrow \mathbb{F}_s(\mathbf{x} \circ \mathbf{m_0} \circ \mathbf{m_1})$ ▷ Two-mask prediction
9: **if** $\hat{y}' \neq y$ **then**
10: $certVal \leftarrow 0$ ▷ Input possibly vulnerable
11: $\boldsymbol{\lambda}[\mathbf{m_0}],\boldsymbol{\lambda}[\mathbf{m_1}] \leftarrow 0,0$ ▷ Vulnerable masks
12: **end if**
13: **end for**
14: **return** $certVal,\boldsymbol{\lambda}$
15: **end procedure**

The double-masking certification procedure from Xiang et al. [25] is outlined in Algorithm 5; we extend the original version to additionally return a vulnerability status array $\boldsymbol{\lambda}$. It works by first ensuring that the mask set $\mathcal{M}$ is $\mathcal{R}$-covering (line 4); otherwise, no guarantees on robustness can be made. Then, during the *for* loop on lines $7-13$ the procedure computes $\mathbb{F}_s(\mathbf{x} \circ \mathbf{m_0} \circ \mathbf{m_1})$ for every possible mask pair $\mathbf{m_0},\mathbf{m_1} \in \mathcal{M} \times \mathcal{M}$ [25]. If all of the predictions are the label y, then $(\mathbf{x},y)$ is certifiable and $certVal$ is set to 1; recall from Definition 2 that this implies that the inference procedure Algorithm 4 will be correct regardless of an attempted patch attack $\mathbf{x}' \in S_{\mathbf{x},\mathcal{R}}$. Otherwise, $certVal$ is set to 0 and the $\boldsymbol{\lambda}$ array is updated to reflect vulnerable points.

The correctness of $certVal$ is guaranteed by the following theorem. Essentially, if predictions across all possible mask pairs are correct, it ensures that each of the three cases in Algorithm 4 will work as intended [25].

Theorem 3. *Suppose we have an image data point $(\mathbf{x},y)$, a single-label classification model $\mathbb{F}_s : \mathcal{X} \to \{1,2,...,c\}$, a patch threat model $S_{\mathbf{x},\mathcal{R}}$, and a $\mathcal{R}$-covering mask set $\mathcal{M}$. If $\mathbb{F}_s(\mathbf{x} \circ \mathbf{m_0} \circ \mathbf{m_1}) = y$ for all $\mathbf{m_0},\mathbf{m_1} \in \mathcal{M} \times \mathcal{M}$, then Algorithm 4 will always return a correct label.*

Proof. This theorem is proved in Xiang et al. [25]. □

We next consider the vulnerability status array $\boldsymbol{\lambda} \in \{0,1\}^{|\mathcal{M}|}$ returned by Algorithm 5. Notice that the length of the array is $|\mathcal{M}|$ rather than $|\mathcal{R}|$; this is a helpful consequence of the $\mathcal{R}$-covering property of the mask set $\mathcal{M}$, which ensures that every patch location $\mathbf{r} \in \mathcal{R}$ will be contained in at least one of the masks $\mathbf{m} \in \mathcal{M}$. As such, an implementation-level abstraction is possible for PatchCleanser, where each element $\boldsymbol{\lambda}[\mathbf{m}]$ summarizes the vulnerability status *for all patch locations contained within the mask* $\mathbf{m} \in \mathcal{M}$. The correctness of this construction can be demonstrated through the following lemma.

Lemma 2. *Suppose we have an image data point $(\mathbf{x},y)$, a single-label classification model $\mathbb{F}_s : \mathcal{X} \to \{1,2,...,c\}$, a patch threat model $S_{\mathbf{x},\mathcal{R}}$, and a $\mathcal{R}$-covering mask set $\mathcal{M}$. Then the array $\boldsymbol{\lambda} \in \{0,1\}^{|\mathcal{M}|}$ returned by Algorithm 5 will be a valid vulnerability status array that satisfies Definition 3.*

Proof. Define $\mathcal{R}^* \subseteq \mathcal{R}$ as the set of patch locations contained in an arbitrary mask $\mathbf{m}^* \in \mathcal{M}$. To demonstrate the validity of $\boldsymbol{\lambda}$, we need to show that $\boldsymbol{\lambda}[\mathbf{m}^*] = 1$ implies Algorithm 4 will be protected from all attacks located in $\mathcal{R}^*$. To do so, we first note that we will only have $\boldsymbol{\lambda}[\mathbf{m}^*] = 1$ in Algorithm 5 if $\mathbb{F}_s(\mathbf{x} \circ \mathbf{m}^* \circ \mathbf{m}) = y$ for all $\mathbf{m} \in \mathcal{M}$; otherwise, $\boldsymbol{\lambda}[\mathbf{m}^*]$ would have been marked with 0 at some point.

We can use this robustness property to guarantee correctness in Algorithm 4. Suppose we have an arbitrary patch attack with a location in $\mathcal{R}^*$ and that $\boldsymbol{\lambda}[\mathbf{m}^*] = 1$. In the first-round masking stage the attack will be completely covered by the mask $\mathbf{m}^*$ (due to the $\mathcal{R}$-covering property) and form the masked image $\mathbf{x} \circ \mathbf{m}^* \in \mathcal{X}$. Note that this is the same as the image $\mathbf{x} \circ \mathbf{m}^* \circ \mathbf{m}^* \in \mathcal{X}$; therefore, the robustness property from above will guarantee that $\mathbb{F}_s(\mathbf{x} \circ \mathbf{m}^*) = y$. We have thus shown that the correct prediction will be represented at least once in the first-round, leaving three possible scenarios.

- *Scenario #1 (consensus):* In this scenario, the classifier returns the correct prediction y for every first-round mask. Then, line 4 of Algorithm 4 will ensure that y is correctly returned as the overall prediction.
- *Scenario #2 (majority of masks are correct):* In this scenario, the classifier returns the correct prediction y for the majority of first-round masks. The set of disagreer masks, $\mathcal{M}_{dis} \subseteq \mathcal{M}$, will thus run a second round of masking. This eventually requires computing $\mathbb{F}_s(\mathbf{x} \circ \mathbf{m_{dis}} \circ \mathbf{m}^*)$ for each $\mathbf{m_{dis}} \in \mathcal{M}_{dis}$. By leveraging symmetry and the robustness property from earlier, these are all guaranteed to return the correct prediction y. Therefore, none of the disagreer masks will have consensus in the second-round, and line 12 of Algorithm 4 will ensure that y is correctly returned as the overall prediction.
- *Scenario #3 (minority of masks are correct):* In this scenario, the classifier returns the correct prediction y for a minority of first-round masks. This implies that $\mathbf{m}^*$ will be a disagreer mask. During the second round of masking, the robustness property from earlier will ensure that $\mathbb{F}_s(\mathbf{x} \circ \mathbf{m}^* \circ \mathbf{m}) = y$ for each $\mathbf{m} \in \mathcal{M}$. Therefore, we will have consensus in the second round of $\mathbf{m}^*$. Because disagreers with incorrect predictions will fail to have consensus (i.e., using the logic from *Scenario #2*), line 9 of Algorithm 4 will ensure that y is correctly returned as the overall prediction.

Overall, we conclude that Algorithm 4 will return the correct prediction y. We have thus shown that $\boldsymbol{\lambda}[\mathbf{m}^*] = 1$ implies Algorithm 4 will be protected from any arbitrary patch attack located in $\mathcal{R}^*$, as desired. □

B.4. Integration with PatchDEMUX

To integrate PatchCleanser into the PatchDEMUX framework, we first generate a $\mathcal{R}$-covering set of masks $\mathcal{M}$; the mask set $\mathcal{M}$ essentially serves as a holistic representation of the security parameters σ. We then incorporate Algorithm 4 into the PatchDEMUX inference procedure (Algorithm 1) and Algorithm 5 into the PatchDEMUX certification procedure (Algorithm 2). Finally, we use the location-aware certification method (Algorithm 3) with the vulnerability status arrays expressed in terms of masks.

C. Further Details on Evaluation Metrics

In this section, we discuss the evaluation metrics from Sec. 4 and Sec. 5 in more detail.

C.1. Threshold analysis

When evaluating a multi-label classifier, we compute precision and recall metrics over a wide variety of thresholds. This results in a large set of evaluation data from which to build precision-recall plots. More specifically, we first evaluate a set of *standard thresholds*:

$$T_{standard} := \{0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9\}$$

We then evaluate a set of *high-value thresholds*. These queries help to fill out the *low recall-high precision* region of a precision-recall curve:

$$T_{high} := \{0.91,0.92,0.93,0.94,0.95,0.96,0.97,0.98,0.99\}$$

We next evaluate a set of *very high-value thresholds*. These evaluations provide points at which recall is close to 0%:

$$T_{veryhigh} := \{0.999,0.9999,0.99999\}$$

Finally, we evaluate a set of mid-value thresholds. These help to smoothen out a precision-recall curve:

$$T_{mid} := T_{mid1} \cup T_{mid2} \cup T_{mid3} \cup T_{mid4}$$

where

$$T_{mid1} := \{0.5+0.02 \cdot t : t \in \{1,2,3,4\}\}$$
$$T_{mid2} := \{0.6+0.02 \cdot t : t \in \{1,2,3,4\}\}$$
$$T_{mid3} := \{0.7+0.02 \cdot t : t \in \{1,2,3,4\}\}$$
$$T_{mid4} := \{0.8+0.02 \cdot t : t \in \{1,2,3,4\}\}$$

For ViT-based models specifically, we found that the *low precision-high recall* region of a precision-recall curve does not readily appear if we limit evaluation to the thresholds outlined above. We thus further evaluate the following set of *low-value* thresholds for ViT-based models:

$$T_{low} := \{5 \cdot 10^{-5}, 10^{-4}, 5 \cdot 10^{-4}, 10^{-3}, 5 \cdot 10^{-3}, 0.01, 0.05\}$$

In order to obtain precision values at key recall levels (i.e., 25%, 50%, 75%), we can perform linear interpolation between relevant recall bounds. However, recall values computed using the thresholds above are often not close enough to these target values. To this end, we use an iterative bisection scheme to find overestimated and underestimated bounds within 0.5 points of the target recalls. The precision values are then calculated by linearly interpolating between these tighter bounds.

C.2. Computing average precision

In order to compute an approximation for average precision, we leverage the area-under-the-curve (AUC) of the associated precision-recall curves. However, in practice the threshold analysis from Appendix C.1 can result in different leftmost endpoints for the precision-recall curves. In order to enforce consistency, we fix the leftmost endpoints for each precision-recall plot at exactly 25% recall. Then, the AUC is computed using the trapezoid sum technique and normalized by a factor of 0.75 (i.e., the ideal precision-recall curve). Note that we pick 25% recall because some evaluations lower than this value demonstrate floating-point precision errors (i.e., the required threshold is extremely high).

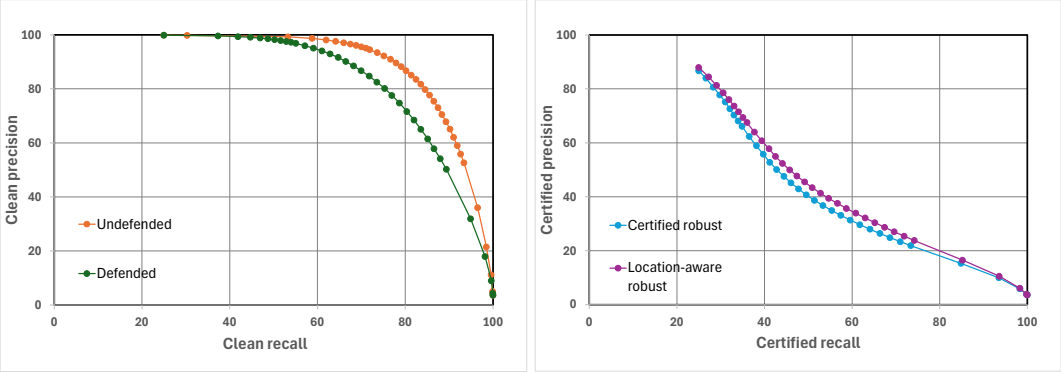
Table 2. *PatchDEMUX performance with Resnet architecture on the MSCOCO* 2014 *validation dataset. Precision values are evaluated at key recall levels along with the approximated average precision. The threat model assumes the patch attack is at most* 2% *of the image area.*

(a) *Clean setting precision values*

Architecture	Resnet			
Clean recall	25%	50%	75%	*AP*
Undefended	99.832	99.425	92.341	87.608
Defended	99.835	98.257	80.612	81.031

(b) *Certified robust setting precision values*

Architecture	Resnet			
Certified recall	25%	50%	75%	*AP*
Certified robust	86.696	40.190	20.959	34.859
Location-aware	87.950	44.373	23.202	37.544



(a) *Clean setting precision-recall curves* (b) *Certified robust setting precision-recall curves*

Figure 5. *PatchDEMUX precision-recall curves with Resnet architecture over the MSCOCO 2014 validation dataset. We consider the clean and certified robust evaluation settings. The threat model assumes that the adversarial patch is at most* 2% *of the image area.*

D. Resnet Architecture Analysis

In this section, we report results for PatchDEMUX while using the Resnet architecture [1]; we leverage the same pre-training routine as Sec. 4.2 to achieve stronger performance. The precision values associated with key recall levels are in Tab. 2. Fig. 5 features precision-recall plots, while AP values are present in Tab. 2.

We find that the Resnet architecture exhibits performance trends similar to those of the ViT architecture in Sec. 4.2. For instance, the defended clean setting is able to keep pace with the performance of the undefended model across a variety of thresholds (see Fig. 5a and Tab. 2a). The Resnet-based variant of PatchDEMUX also achieves non-trivial robustness, as evidenced by the 37.544% certified average precision value in Tab. 2b. In general, the precision-recall curves for the two architectures are similar in shape for all four evaluation settings.

Despite these similarities, the ViT model achieves consistently superior performance compared to the Resnet model. For instance, the ViT-based variant of PatchDEMUX provides a $\sim$4 point increase in clean AP for the defended clean setting and a $\sim$7 point increase in certified AP for the two certified robust settings (see Tab. 1). One possible reason for this improvement is the underlying training process of vision transformers, which involves a masking process similar in principle to the double-masking approach implemented by PatchCleanser [7, 25].

E. Location-aware Certification Analysis

Table 3. *PatchDEMUX performance with ViT architecture on the MSCOCO* 2014 *validation dataset for different location-aware attackers. Precision values are evaluated at key recall levels along with the approximated average precision. The threat model assumes the patch attack is at most* 2% *of the image area.*

Architecture	ViT			
Certified recall	25%	50%	75%	*AP*
FP attacker	95.724	62.132	33.112	49.474
FN attacker	95.971	58.158	27.199	45.951
Location-aware robust	95.670	56.038	26.375	44.899
Certified robust	95.369	50.950	22.662	41.759

In this section we investigate the location-aware certification approach from Sec. 3.4 in more detail.

E.1. Attack vectors

We note that based on Sec. 3.4.2, there are a couple different ways to evaluate the robustness provided by the location-aware method.

- *FN attacker:* Here, we only track vulnerability status arrays $\boldsymbol{\lambda}$ for false negatives. Intuitively, this corresponds to the optimal attacker from Sec. 3.4.2 constructing a patch with the

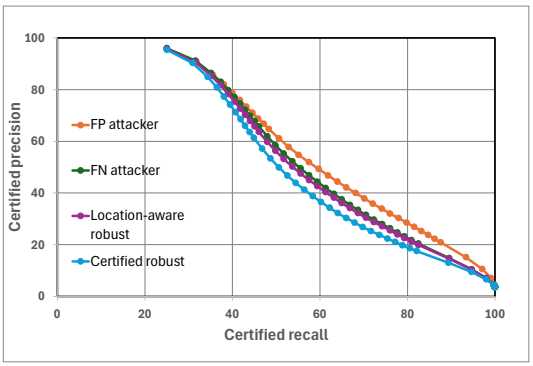


Figure 6. *PatchDEMUX precision-recall curves with ViT architecture over the MSCOCO 2014 validation dataset for different location-aware attackers. The standard certified robust evaluation setting is included as a baseline. The threat model assumes that the adversarial patch is at most* 2% *of the image area.*

sole intent of increasing false negatives (i.e., a FN attack). In Algorithm 3, tie-breakers are decided by picking the location which induces more false positives.

- FP *attacker:* In this evaluation method we only only track vulnerability status arrays $\boldsymbol{\lambda}$ for false positives. This corresponds to the optimal attacker from Sec. 3.4.2 constructing a patch with the sole intent of increasing false positives (i.e., a FP attack). In the FP version of Algorithm 3, tie-breakers are decided by picking the location which induces more false negatives.

We also consider "worst case" performance where we simultaneously determine the worst patch location for both false negatives and false positives in an image. Note that these two locations do not have to be identical, and as a result this "worst case" performance is not necessarily realizable. However, we consider this evaluation setting as it represents the theoretical lower bound on robustness for Algorithm 3 given an arbitrarily motivated attacker.

E.2. Experiment results

We now empirically compare the different attack vectors possible under location-aware certification. We consider the ViT architecture alone as it provides better performance compared to Resnet. In addition, we leverage the same pre-trained model checkpoints used in Sec. 4.2 for consistency. Precision values corresponding to different attackers are present in Tab. 3, while precision-recall plots are in Fig. 6.

Provable robustness improvements. Regardless of the attack strategy employed, location-aware certification provides improved robustness compared to the certified robust baseline setting; this is expected due to Theorem 2. Improvement is most notable in both the *mid recall-mid precision* and *high recall-low precision* sections of the precision-recall curve. Overall, the most favorable evaluation approach provides an $\sim$8 point increase in certified AP compared to the baseline. Despite these improvements, location-aware certification does not fundamentally change the shape of the robust precision-recall curve under any of the three attack settings.

Asymmetric attack performance. Interestingly, location-aware certification provides the strongest robustness guarantees under the FP attack strategy. This is likely due to the asymmetric dependence of precision and recall metrics on false positives. Specifically, both metrics depend on false negatives[6], but the recall metric does not depend on false positives. This makes FP attacks "weaker" relative to other methods.

F. Model Pre-training for PatchDEMUX

Single-label CDPAs often leverage pre-training routines to improve the robustness of underlying single-label classifiers; these work by fine-tuning the model on specially augmented training data [6, 20, 25]. In this section, we investigate whether some of these pre-training routines can extend to multi-label classifiers and improve PatchDEMUX performance. We specifically consider pre-training strategies used by PatchCleanser, as PatchCleanser is the certifiable backbone for PatchDEMUX in this work.

F.1. Pre-training techniques for PatchCleanser

Two different pre-training techniques have been used so far to improve the performance of PatchCleanser: *Random Cutout* [6] and *Greedy Cutout* [20]. The former works by placing two square masks at random locations on training images, with each mask covering at most 25% of the image area [6, 25]. When designing PatchCleanser, Xiang et al. [25] found that Random Cutout pre-training provides significant boosts in robustness; intuitively, integrating cutout masks in pre-training helps the underlying model become more tolerant to occlusion effects from the double-masking procedures. Later, Saha et al. [20] proposed the Greedy Cutout pre-training procedure and demonstrated superior performance to Random Cutout for PatchCleanser. This approach works by placing the pair of certification masks that greedily induce the highest loss for each training image.

F.2. Model pre-training methodology

In our experiments we compare the following three pre-training scenarios, which are representative of the settings used in prior work [6, 20, 25].

- Random Cutout pre-training with two square 25% masks
- Greedy Cutout pre-training with 6×6 certification masks
- Greedy Cutout pre-training with 3×3 certification masks.

For Greedy Cutout, we compute the loss for candidate masks while models are in evaluation mode; this approach helps avoid consistency issues associated with batch normalization. We do not consider the more complex multi-size greedy cutout approach from Saha et al. [20] due to difficulties with mask decompositions.

[6]Precision indirectly depends on false negatives via the true positive count

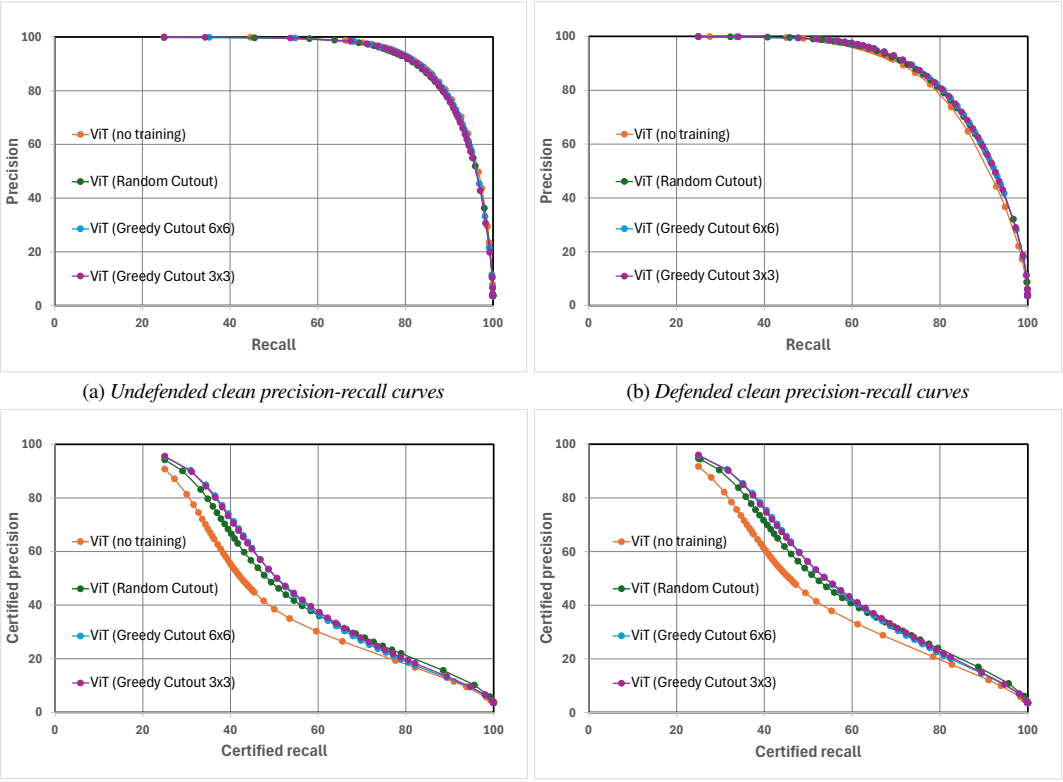
Table 4. *PatchDEMUX performance with ViT architecture on the MSCOCO* 2014 *validation dataset when using different pre-training techniques. Precision values are evaluated at key recall levels along with the approximated average precision. The threat model assumes that the adversarial patch is at most* 2% *of the image area.*

(a) *Clean setting precision values*

Architecture	ViT (vanilla/no pretraining)				ViT (Random Cutout)				ViT (Greedy Cutout 6×6)				ViT (Greedy Cutout 3×3)			
Recall	25%	50%	75%	*AP*	25%	50%	75%	*AP*	25%	50%	75%	*AP*	25%	50%	75%	*AP*
Undefended	99.940	99.749	96.265	91.449	99.770	99.642	95.951	90.900	99.930	99.704	96.141	91.142	99.930	99.736	95.973	90.903
Defended	99.930	99.138	85.757	83.776	99.858	99.224	87.273	85.028	99.894	99.223	87.764	85.275	99.900	99.230	87.741	85.271

(b) *Certified robust setting precision values*

Architecture	ViT (vanilla/no pretraining)				ViT (Random Cutout)				ViT (Greedy Cutout 6×6)				ViT (Greedy Cutout 3×3)			
Certified recall	25%	50%	75%	*AP*	25%	50%	75%	*AP*	25%	50%	75%	*AP*	25%	50%	75%	*AP*
Certified robust	90.767	38.490	20.846	35.003	94.192	47.548	24.603	40.975	95.369	51.580	22.662	41.759	95.574	51.095	23.454	42.077
Location-aware robust	91.665	43.736	23.163	38.001	94.642	52.491	27.526	43.908	95.670	56.038	26.375	44.899	95.959	55.958	27.105	45.122



(a) *Undefended clean precision-recall curves*

(b) *Defended clean precision-recall curves*

(c) *Certified robust precision-recall curves*

(d) *Location-aware robust precision-recall curves*

Figure 7. *PatchDEMUX precision-recall curves with ViT architecture over the MSCOCO 2014 validation dataset when using different pre-training techniques. We consider each of the four evaluation settings in separate plots. The threat model assumes that the adversarial patch is at most* 2% *of the image area.*

To perform model fine-tuning, we first obtain existing checkpoints for the MSCOCO 2014 classification task [1, 14]. We then follow the fine-tuning methodology for multi-label classifiers outlined by Ben-Baruch et al. [1]. Specifically, we use asymmetric loss (ASL) as the loss function, a 1cycle learning rate policy with max learning rate $\alpha_{max} = 5.0 \cdot 10^{-5}$, automatic mixed precision (AMP) for faster training, and exponential moving average (EMA) of model checkpoints for improved inference [1]. Models are fine-tuned on copies of the MSCOCO 2014 training dataset augmented by Random Cutout and Greedy Cutout. We use the Adam optimizer for 5 epochs, and best

checkpoints are picked according to the loss on held out data[7]. A cluster of NVIDIA A100 40GB GPUs are used to perform the associated computations.

F.3. Experiment results

Results for the different pre-training routines are in Tab. 4. In addition, precision-recall plots comparing the pre-training routines for each of the four PatchDEMUX evaluation settings are present in Fig. 7. We consider the ViT architecture alone as it provides better performance compared to Resnet.

Model pre-training boosts performance. In general, we find that using a pre-training routine of any kind leads to performance boosts for PatchDEMUX. For instance, pre-training helps the two certified robust evaluation settings achieve a $6-7$ point improvement in certified AP compared to the vanilla checkpoints, while the defended clean setting demonstrates a $\sim$2 point improvement in clean AP compared to the baseline. Greedy Cutout also provides additional robustness boosts compared to Random Cutout, with certified AP metrics being almost a full point higher; this corroborates with findings from Saha et al. [20]. Note that in general pre-training strategies provide stronger improvements in the certified robust evaluation settings compared to the clean settings; this is likely because the clean settings already demonstrate (relatively) strong performance, and thus potential gains from pre-training are more marginal. Nevertheless, we prioritize the defended clean setting overall as it is most representative of typical performance. The Greedy Cutout 6×6 pre-training strategy, which achieves the highest defended clean AP value, is therefore featured in Sec. 4.2.

Location-aware certification provides consistent improvements. An interesting observation from Tab. 4 is that the location-aware robust setting provides a consistent 3 point boost to certified AP regardless of the presence/absence of pre-training. This suggests that our location-aware certification technique has general utility across a variety of model pre-training scenarios and that it "stacks" with other sources of robustness improvements.

G. Tables for Security Parameter Experiments

In this section we provide the tables associated with the security parameter experiments in Sec. 5. In Tab. 5 we list clean and certified metrics associated with the *mask number* experiments from Sec. 5.1. In Tab. 6 we list clean and certified metrics associated with the *patch size* experiments from Sec. 5.2.

[7]We find that fine-tuning for longer leads to overfitting.

Table 5. *PatchDEMUX performance with ViT architecture on the MSCOCO* 2014 *validation dataset. We vary the mask number security parameter associated with the underlying single-label CDPA PatchCleanser and fix the estimated patch size at* 2% *of the image area. We list even mask number values for brevity. Precision values are evaluated at key recall levels along with the approximated average precision.*

(a) *Clean setting precision values*

Architecture	ViT (2×2 masks)				ViT (4×4 masks)				ViT (6×6 masks)			
Recall	25%	50%	75%	*AP*	25%	50%	75%	*AP*	25%	50%	75%	*AP*
Undefended	99.940	99.749	96.265	91.449	99.940	99.749	96.265	91.449	99.940	99.749	96.265	91.449
Defended	99.910	96.999	75.393	78.727	99.930	98.845	83.388	82.529	99.930	99.138	85.757	83.776

(b) *Certified robust setting precision values*

Architecture	ViT (2×2 masks)				ViT (4×4 masks)				ViT (6×6 masks)			
Certified recall	25%	50%	75%	*AP*	25%	50%	75%	*AP*	25%	50%	75%	*AP*
Certified robust	41.577	17.924	9.909	15.735	87.976	37.163	19.798	33.231	90.767	38.490	20.846	35.003
Location-aware robust	46.553	20.624	10.798	17.690	89.259	41.490	21.763	35.953	91.665	43.736	23.163	38.001

Table 6. *PatchDEMUX performance with ViT architecture on the MSCOCO* 2014 *validation dataset. We vary the patch size security parameter associated with the underlying single-label CDPA PatchCleanser and fix the mask number parameter at* 6×6*. Precision values are evaluated at key recall levels along with the approximated average precision.*

(a) *Clean setting precision values*

Architecture	ViT (0.5% patch)				ViT (2% patch)				ViT (8% patch)				ViT (32% patch)			
Recall	25%	50%	75%	*AP*	25%	50%	75%	*AP*	25%	50%	75%	*AP*	25%	50%	75%	*AP*
Undefended	99.940	99.749	96.265	91.449	99.940	99.749	96.265	91.449	99.940	99.749	96.265	91.449	99.940	99.749	96.265	91.449
Defended	99.947	99.470	89.150	85.731	99.930	99.138	85.757	83.776	99.907	97.798	78.712	80.093	99.529	89.813	60.543	69.952

(b) *Certified robust setting precision values*

Architecture	ViT (0.5% patch)				ViT (2% patch)				ViT (8% patch)				ViT (32% patch)			
Certified recall	25%	50%	75%	*AP*	25%	50%	75%	*AP*	25%	50%	75%	*AP*	25%	50%	75%	*AP*
Certified robust	97.670	61.867	30.239	48.820	90.767	38.490	20.846	35.003	44.666	19.249	11.832	16.961	6.933	5.827	4.854	5.297
Location-aware robust	97.769	66.350	32.850	51.158	91.665	43.736	23.163	38.001	50.263	22.965	13.363	19.713	9.169	6.997	5.307	6.195