

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

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

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

Prateek Mittal
Princeton University
Princeton, NJ, USA
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 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 to provably guarantee robustness. Furthermore, in the scenario where an attacker is limited to a single patch we propose an additional certification procedure that can provide tighter robustness bounds. 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 MS-COCO and PASCAL VOC datasets while maintaining high clean performance¹.

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 [18]. Unfortunately, these performance boosts have come at a security cost; attackers can use *adversarial patches* to perturb patch-shaped regions in images and fool deep learning systems [4, 30]. The patch threat model presents a unique challenge for the security community due to its physically-realizable nature; for instance, even a single well-designed patch that is printed out can induce failure in the wild [4, 12, 26].

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, but can be vulnerable to *adaptive attacks* which bypass the defense through fundamental weaknesses in design [5, 14, 25]. As a result, certifiable defenses against patch attacks have become popular for computer vision tasks such as single-label classification and object detection [6, 17, 24, 29, 31–35, 37]; these methods feature provable guarantees on robustness under any arbitrary patch attack.

Despite these successes, progress on certifiable defenses against patch attacks has been limited for multi-label classification. Multi-label classifiers provide important capabilities for simultaneously tracking multiple objects while maintaining scalability. Many safety-critical systems depend on the visual sensing capabilities of multi-label classifiers, such as traffic pattern recognition in autonomous vehicles [16], video surveillance [10], and product identification for retail checkout [13]. Some of these applications have become mainstream in industry (i.e., Waymo robotaxis, Just Walk Out checkout, etc.).

To address this challenge we propose PatchDEMUX, a certifiably robust framework against patch attacks for 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. To do so, we leverage the key insight that any multi-label classifier can be separated into individual binary classification tasks. This approach allows us to 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*. We also consider the scenario where an attacker is restricted to a single patch and propose a novel certification procedure that achieves stronger robustness bounds by using constraints in vulnerable patch locations.

We find that PatchDEMUX achieves non-trivial robustness on the MS-COCO and PASCAL VOC datasets while maintaining high performance on clean data. Specifically, when using the current SOTA single-label certifiable defense PatchCleanser as a backbone, PatchDEMUX attains 85.276% average precision on clean MS-COCO images and 44.902% certified robust average precision. On the PASCAL VOC dataset PatchDEMUX achieves 92.593% clean average precision and 56.030%

¹Our source code is available at <https://github.com/inspire-group/PatchDEMUX>

certified robust average precision. For reference, an undefended model achieves 91.146% average precision on clean MSCOCO images and 96.140% average precision on clean PASCAL VOC images. Overall, the key contributions of our work can be summarized as follows:

- We address the challenge of patch attacks in the multi-label domain via a general framework that can interface with any existing/future single-label defense. To the best of our knowledge, our approach is the first of its kind.
- Our framework provably guarantees lower bounds on performance irrespective of the chosen patch attack (i.e., the patch can contain an optimized attack, random noise, etc.).
- We instantiate a version of our defense framework with the current SOTA single-label defense and achieve strong robust performance on popular benchmarks.

We hope that future work will integrate with the PatchDEMUX framework and further strengthen the robustness of multi-label classifiers to adversarial patches.

2. Problem Formulation

In this section, we provide a primer on the multi-label classification task along with standard metrics for evaluation. We next outline the adversarial patch threat model and its relevance in the multi-label setting. 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

Multi-label classification is a computer vision task where images $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^{w \times h \times \gamma}$ with width w , height h , and number of channels γ contain multiple objects simultaneously, with each object belonging to one of c classes [38]. 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; this implies that the set of labels \mathcal{Y} is 2^c in size, i.e., exponential. A *multi-label classifier* $\mathbb{F}: \mathcal{X} \rightarrow \mathcal{Y}$ 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) [3].

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 [38]:

$$\text{precision} = \frac{TP}{TP+FP} \quad \text{recall} = \frac{TP}{TP+FN} \quad (1)$$

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. In the patch threat model, attackers possess the ability to arbitrarily adjust pixel values within a restricted region located 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 [4]. As discussed in Sec. 1, defending against this threat model is critical due to its physically realizable nature [4, 12, 26]. In this paper, we primarily focus on defending against a single adversarial patch as it is a popular setting in prior work [6, 17, 24, 31–33, 35]. However, our baseline certification methods can also handle multiple patches, provided the underlying single-label defense strategy already has this capability [33].

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 [33]. Then, the associated patch attacks are:

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

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 [17].

Adversarial patches in the multi-label setting. Patch attacks in multi-label classification aim to induce 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 generated by representing the desired objective as an optimization problem and then applying an iterative technique such as projected gradient descent (PGD) over $S_{\mathbf{x}, \mathcal{R}}$ [21].

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 $\mathbb{F}: \mathcal{X} \rightarrow \mathcal{Y}$ against all possible attacks in the patch threat model [6, 17, 24, 29, 31–35, 37]. This ensures that 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, denoted by σ , manage the trade-off between robust performance and inference time [33]. The inference procedure $INFER_{[\mathbb{F}, \sigma]}: \mathcal{X} \rightarrow \mathcal{Y}$ 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 $\rho: \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$. In addition to the inference procedure, the certification procedure $CERT_{[\mathbb{F}, \sigma]}: \mathcal{X} \times \mathcal{Y} \times \mathbb{P}(\mathcal{R}) \rightarrow \mathbb{R}$ ($\mathbb{P}(\cdot)$ 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, for a performance metric ρ and a patch threat model $S_{\mathbf{x}, \mathcal{R}}$ we will have

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

Here, $\tau := \text{CERT}_{[\mathbb{F}, \sigma]}(\mathbf{x}, \mathbf{y}, \mathcal{R})$ 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 $\mathbb{F}: \mathcal{X} \rightarrow \mathcal{Y}$ is a tuple of procedures $\text{DEF} := (\text{INFER}_{[\mathbb{F}, \sigma]}: \mathcal{X} \rightarrow \mathcal{Y}, \text{CERT}_{[\mathbb{F}, \sigma]}: \mathcal{X} \times \mathcal{Y} \times \mathbb{P}(\mathcal{R}) \rightarrow \mathbb{R})$ 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 $\rho: \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$.

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 datapoints².

Definition 2 (Single-label CDPA). A single-label CDPA is a CDPA for single-label classifiers $\mathbb{F}_s: \mathcal{X} \rightarrow \{1, 2, \dots, c\}$. 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 $\rho(\mathbf{y}_1, \mathbf{y}_2) := \sum_{i=1}^c [\mathbf{y}_1[i] = 1 \cap \mathbf{y}_2[i] = 1]$ and CERT lower bounds the number of true positives. This helps bootstrap robust metrics such as 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 [6, 17, 24, 29, 31–33, 35]. Current techniques roughly fall into one of two categories: *small receptive field* defenses and *masking* defenses. With regards to the former, the general principle 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 [17], BagCert [24], and PatchGuard [31, 32]. 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 [33]. 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.

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 $i \in \{1, 2, \dots, c\}$. 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 classifier³ $\mathbb{F}: \mathcal{X} \rightarrow \mathcal{Y}$, we use the notation $\mathbb{F}[i]: \mathcal{X} \rightarrow \{0, 1\}$ 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 [24, 31, 32]. Nevertheless, a workaround is possible; specifically, we can initialize the multi-label classifier $\mathbb{F}: \mathcal{X} \rightarrow \mathcal{Y}$ as an ensemble of c binary classifiers which each satisfy the required architecture. Then, for each class $i \in \{1, 2, \dots, c\}$ we can define the isolated classifier $\mathbb{F}[i]$ as the associated ensemble model. Other defenses are architecture-agnostic [33]. In these cases we can use any off-the-shelf multi-label classifier $\mathbb{F}: \mathcal{X} \rightarrow \mathcal{Y}$ and for each class $i \in \{1, 2, \dots, c\}$ 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 $i \in \{1, 2, \dots, c\}$ 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 creates a lower bound for true positives by accumulating the results.

3.2. PatchDEMUX inference procedure

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

²We use Iverson bracket notation for convenience

³From here on, \mathcal{Y} will denote a multi-label label set with c classes

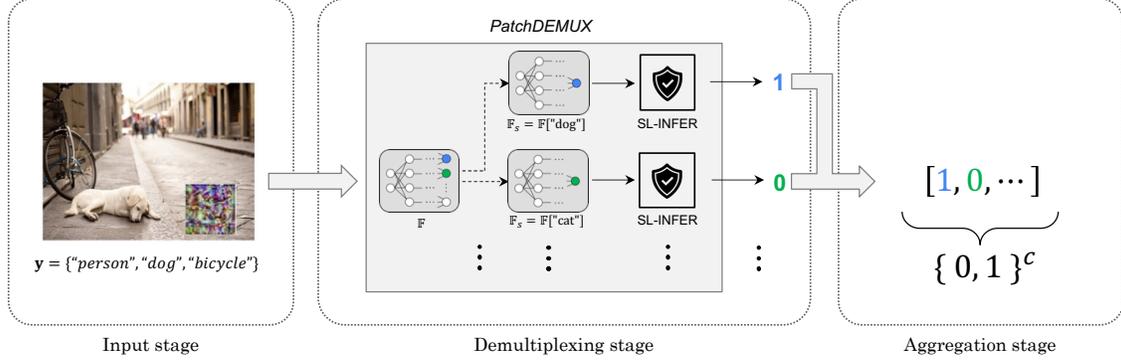


Figure 1. A diagram which illustrates the defense framework from PatchDEMUX. In the input stage, the (potentially attacked) image is preprocessed. In the demultiplexing stage, the *SL-INFER* inference procedure from a single-label CDPA is applied to each individual class in the multi-classification task. This is done by considering the multi-label classifier \mathbb{F} as a series of isolated binary classifiers $\mathbb{F}[i]$ for $i \in \{1, 2, \dots, c\}$. 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 $\mathbb{F} : \mathcal{X} \rightarrow \mathcal{Y}$, inference procedure *SL-INFER* and security parameters σ from a single-label CDPA, number of classes c
Output: Prediction $\text{preds} \in \{0, 1\}^c$

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

image \mathbf{x} and update preds for class i .

Remark. If the time complexity for *SL-INFER* is $\mathcal{O}(f(n))$, the time complexity for Algorithm 1 will be $\mathcal{O}(c \cdot f(n))$. However, in practice it is possible to take advantage of relatively negligible defense post-processing and effectively reduce the time complexity to $\mathcal{O}(f(n))$. See *Supplementary Material*, Appendix G.

3.3. PatchDEMUX certification procedure

The PatchDEMUX certification procedure is outlined in Algorithm 2. We first initialize the certification procedure *SL-CERT* from a single-label CDPA with security parameters σ . On line 2, we create the κ array to store certifiable classes. On line 5, we run *SL-CERT* with the isolated binary classifier $\mathbb{F}[i]$ on datapoint $(\mathbf{x}, \mathbf{y}[i])$ and place the result in $\kappa[i]$; recall from Definition 2 that *SL-CERT* returns 1 for protected datapoints and 0 otherwise. Finally, on lines 7–10 we count a successful true positive for classes with $\mathbf{y}[i] = 1$ and $\kappa[i] = 1$. 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.

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 $\mathbb{F} : \mathcal{X} \rightarrow \mathcal{Y}$, certification procedure *SL-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 κ

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

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-DEF*, and a multi-label classification model $\mathbb{F} : \mathcal{X} \rightarrow \mathcal{Y}$. 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. □

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

$$\text{certifiedprecision} = \frac{TP_{\text{lower}}}{TP_{\text{lower}} + FP_{\text{upper}}} \quad (4)$$

$$\text{certifiedrecall} = \frac{TP_{\text{lower}}}{TP_{\text{lower}} + FN_{\text{upper}}} \quad (5)$$

Note by construction that both metrics provide lower bounds for precision and recall on a datapoint (\mathbf{x}, \mathbf{y}) *irrespective of any attempted patch attack*; the real-world performance of our defense will always be higher. Therefore, an empirical evaluation of existing multi-label attack vectors is not necessary [1, 2, 22, 23]. Furthermore, micro-averaging these metrics across datapoints provides lower bounds on precision and recall for an entire dataset [38].

3.4. Location-aware certification

We now discuss an improved method called *location-aware certification* which extends Algorithm 2. This method works in the scenario where an attacker is restricted to a single patch. 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. 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

We now give a formal treatment of our core idea. Suppose we have a single-label CDPA $SL\text{-}DEF$. For many existing single-label defenses, it is possible to relate the certification procedure $SL\text{-}CERT$ to the complete list of patch locations \mathcal{R} from Eq. (2) [6, 17, 24, 29, 31–33, 35]. In these cases, we extend Definition 2 and allow $SL\text{-}CERT$ to return a *vulnerability status array*, which we denote by $\lambda \in \{0, 1\}^{|\mathcal{R}|}$. A value of 1 implies the image $\mathbf{x} \in \mathcal{X}$ is protected from attacks located in $\mathbf{r} \in \mathcal{R}$, while 0 means it is not.

This provides a convenient formulation with which to express our improved method. Consider a multi-label classifier $\mathbb{F}: \mathcal{X} \rightarrow \mathcal{Y}$. We first obtain vulnerability status arrays λ 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 $\mathbf{1} - \lambda$ 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 $\mathbf{r}_{\text{opt}} \in \mathcal{R}$ 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 $v_{\text{opt}} < k$. Then, as per the construction of each λ these

$k - v_{\text{opt}} > 0$ classes will be *guaranteed robustness under the optimal patch location*.

3.4.2. Proposing our novel algorithm

Algorithm 3 Location-aware certification for FN

Input: Image $\mathbf{x} \in \mathcal{X}$, ground-truth $\mathbf{y} \in \mathcal{Y}$, multi-label classifier $\mathbb{F}: \mathcal{X} \rightarrow \mathcal{Y}$, 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($\mathbf{x}, \mathbf{y}, \mathbb{F}, SL\text{-}CERT, \sigma, \mathcal{R}$)
- 2: ▷ Pass all args to DEMUXCERT(...)
- 3: $TP, FP, FN, \kappa \leftarrow$ DEMUXCERT(...)
- 4: ▷ Initialize array with list of FN indices
- 5: $c \leftarrow \text{len}(\mathbf{y})$
- 6: $fnIdx \leftarrow \text{list}(\{1 \leq i \leq c: \kappa[i] = 0 \cap \mathbf{y}[i] = 1\})$
- 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: $\lambda \leftarrow SL\text{-}CERT_{[\mathbb{F}_s, \sigma]}(\mathbf{x}, \mathbf{y}[fnIdx[k]], \mathcal{R})$
- 11: $fnCertFails[k] = \mathbf{1} - \lambda$
- 12: **end for**
- 13: $fnTotal \leftarrow \text{sum}(fnCertFails, \text{dim} = 0)$
- 14: $FN_{\text{new}} = \max(fnTotal)$ ▷ Pick worst location
- 15: $TP_{\text{new}} = TP + (FN - FN_{\text{new}})$
- 16: **return** $TP_{\text{new}}, FN_{\text{new}}$
- 17: **end procedure**

These insights are encapsulated by Algorithm 3, the location-aware certification method for false negatives.⁴ 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 λ for each false negative by isolating the associated binary classification task. Finally, we sum the inverted arrays $\mathbf{1} - \lambda$ on line 13 and pick the patch location with the largest value; this is the maximum number of false negatives an attacker can induce at test time. We then alter the lower bound for true positives on line 16.

We now demonstrate that Algorithm 3 provides superior bounds 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 $\mathbb{F}: \mathcal{X} \rightarrow \mathcal{Y}$. If $SL\text{-}CERT$ returns the vulnerability status array λ 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. □

An analogue to Theorem 2 also exists for FP bounds, and can be proved using a modified version of Algorithm 3 that tracks FP indices.

⁴Obtaining FP_{new} is similar, with line 5 changed to track FP indices

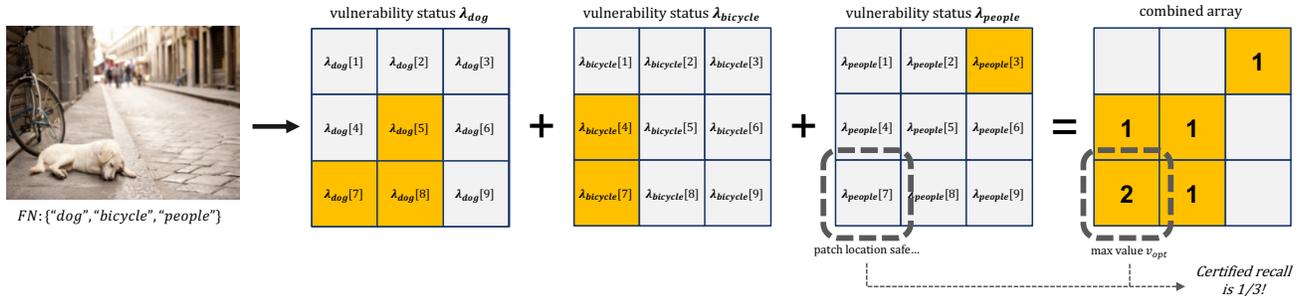


Figure 2. A diagram which illustrates the key intuition for the location-aware approach. In the sample image we assume all three objects (i.e., “dog”, “bicycle”, “people”) are false negatives. Thus, for each FN we extract the vulnerability status over all patch locations (orange means vulnerable) and 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

In this section, we discuss our evaluation setup. The associated source code is available at <https://github.com/inspire-group/PatchDEMUX>.

Backbone initialization and parameters. Recall from 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 is architecture-agnostic (i.e., it is compatible with any off-the-shelf multi-label classifier) [33]. PatchCleanser works by using a novel double-masking algorithm along with a specially generated certification mask set to provably remove adversarial patches [33]. The mask generation process has two security parameters. The first is the number of masks for each image dimension $k_1 \times k_2$; using more masks leads to longer inference time but results in stronger robustness, effectively serving as a “computational budget” [33]. The second is the estimated size of the patch p in pixels. Our experiments with PatchDEMUX use 6×6 masks and assume the patch is $\sim 2\%$ of the overall image size, which are the default settings in Xiang et al. [33]; we vary these parameters in Sec. 5. For more details on how PatchCleanser fits into the PatchDEMUX framework see *Supplementary Material*, Appendix B.

We note that PatchCleanser can also provide protection against multiple patches [33]. Because our baseline certification method provably extends single-label guarantees to multi-label setting, it will also feature resistance against multiple patches. In our experiments, we focus on the single patch setting for simplicity.

Dataset and model architectures. We evaluate our defense on two datasets: MS-COCO [19] and PASCAL VOC [11]. The former is a challenging collection of images that feature “common objects in context” [19], while the latter focuses on “realistic scenes” [11]. For our experiments we test on the MS-COCO 2014 validation split, which contains $\sim 41,000$ images and 80 classes, and the PASCAL VOC 2007 test split, which has $\sim 5,000$ images and 20 classes. Both of these splits are commonly used in the multi-label classification community

[3, 20, 27, 36].

For the multi-label classifier architecture, we evaluate two options. The first is a ResNet-based architecture from Ben-Baruch et al. [3] that uses convolution kernels and has an input size of 448×448 . The second is a vision transformer-based (ViT) architecture from Liu et al. [20] that uses the self-attention mechanism and has an input size of 384×384 [9, 20, 36]. These models are chosen as they perform well on the multi-label classification task and have publicly available checkpoints. We resize images to fit on each model and apply different defense fine-tuning methods (i.e., Random Cutout [8], Greedy Cutout [28]) to achieve stronger robustness guarantees.

Evaluation settings and metrics. Our results feature several evaluation settings.

1. *Un defended 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 activated.
3. *Certified robust*: This setting represents lower bounds on performance determined using Algorithm 2.
4. *Location-aware robust*: This setting represents the tighter certification bounds from Algorithm 3. We report performance corresponding to the worst-case attacker (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 [38]. In addition, we sweep model outputs across a range of threshold values to create *precision-recall plots*. The associated area-under-curve values aggregate performance and are used to approximate *average precision* (AP); more details are in *Supplementary Material*, Appendix C.

4.2. PatchDEMUX overall performance

In this section we report our main results for PatchDEMUX on the MS-COCO 2014 validation dataset. We summarize the precision values associated with key recall levels in Tab. 1. Fig. 3

Table 1. PatchDEMUX performance with ViT architecture on the MS-COCO 2014 validation dataset. Precision values are evaluated at key recall levels along with the approximated average precision. We assume the patch attack is at most 2% of the image area and use a computational budget of 6×6 masks

(a) Clean setting precision values					(b) Certified robust setting precision values				
Architecture	ViT				Architecture	ViT			
Clean recall	25%	50%	75%	AP	Certified recall	25%	50%	75%	AP
Undefended	99.930	99.704	96.141	91.146	Certified robust	95.369	50.950	22.662	41.763
Defended	99.894	99.223	87.764	85.276	Location-aware	95.670	56.038	26.375	44.902

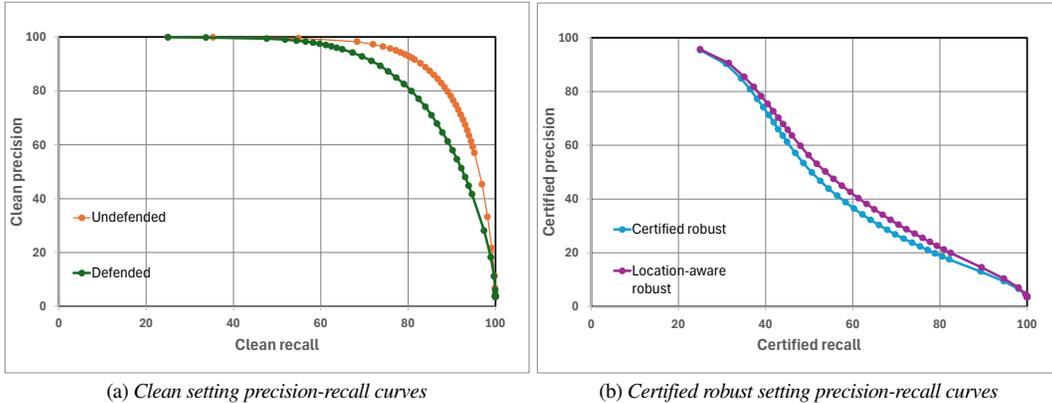


Figure 3. PatchDEMUX precision-recall curves with ViT architecture over the MS-COCO 2014 validation dataset. We consider the clean and certified robust evaluation settings. We assume the patch attack is at most 2% of the image area and use a computational budget of 6×6 masks.

features precision-recall plots, while AP values are present in Tab. 1. Because the ViT architecture outperforms the Resnet architecture (see *Supplementary Material, Appendix D*) we focus on the ViT model here. Performance of the ViT architecture on the PASCAL VOC 2007 test dataset is in *Supplementary Material, Appendix H*.

High clean performance. As shown in Tab. 1a and Fig. 3a, the PatchDEMUX inference procedure features excellent performance on clean data. Specifically, the defended clean setting achieves $\sim 94\%$ of the undefended model’s AP. These results demonstrate that PatchDEMUX can be deployed at test time with minimal loss in performance utility.

Non-trivial robustness. Tab. 1b and Fig. 3b also show that PatchDEMUX attains non-trivial certifiable robustness on the MS-COCO 2014 validation dataset. For instance, when fixed at 50% certified recall PatchDEMUX achieves 56.038% certified precision. This performance remains stable across a variety of thresholds, as evidenced by the 44.902% certified AP value. Location-aware certification is a key factor in these results, improving 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).

Interestingly, the defended clean precision-recall plot (Fig. 3a) is concave in shape while the certified robust plots (Fig. 3b) are slightly convex. This performance gap is likely due to the sensitivity of PatchCleanser’s certification procedure to ob-

ject occlusion from the generated mask set. This limitation is compounded by the fact that many MS-COCO images contain objects that are small relative to the overall image size [19, 33].

4.3. Ablation studies

We also perform a series of ablation studies for PatchDEMUX using the MS-COCO 2014 validation dataset. We first 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 then investigate the impact of different defense fine-tuning routines, and find that variants of cutout fine-tuning (i.e., Random Cutout [8], Greedy Cutout [28]) can boost model robustness (see *Supplementary Material, Appendix F*); the strongest results for the defended clean setting are featured in the previous section.

5. Security Parameter Experiments

As discussed in Sec. 4.1, the PatchCleanser backbone has two security parameters: the number of masks desired in each dimension $k_1 \times k_2$ (i.e., the “computational budget”) and the estimated size of the patch p in pixels [33]. In this section, we study the impact of these parameters on PatchDEMUX performance. To isolate the effects of security parameter variation, we use ViT checkpoints without defense fine-tuning. Experiments are done on the MS-COCO 2014 validation dataset.

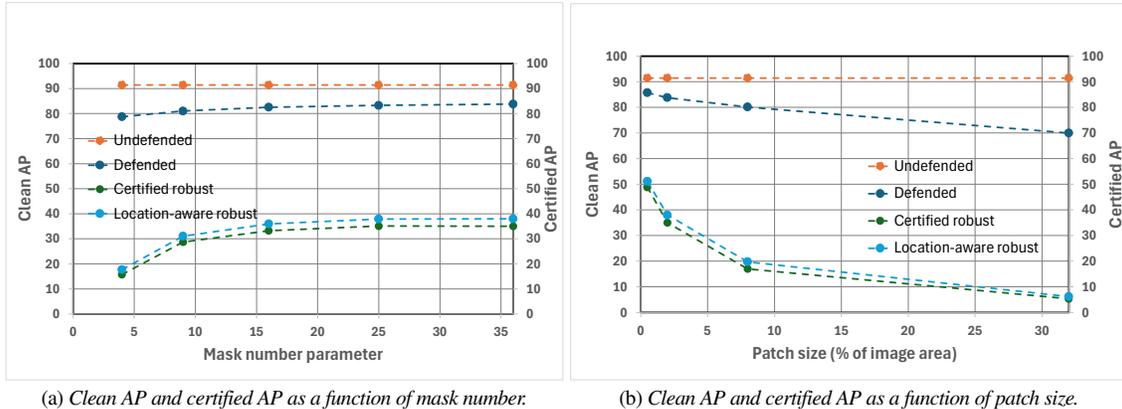


Figure 4. The impact of varying PatchCleanser security parameters on PatchDEMUX performance. Experiments performed on MS-COCO 2014 validation dataset. We compute clean AP for the clean setting evaluations, and certified AP for the certified robust setting evaluations.

5.1. Impact of varying mask number

We present results when varying the mask number parameter in Fig. 4a (the associated table is in *Supplementary Material, Appendix I*). We assume the number of masks in each dimension is the same (i.e., $k := k_1 = k_2$) and evaluate with respect to k^2 . We keep the patch size parameter its 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, decreasing the number of masks from 36 to 16 results in a maximum AP drop of 2 points across all evaluation settings. At the extreme of $k^2 = 4$ masks more substantial performance drops are noticeable. This is expected, as the mask generation method from PatchCleanser will create larger masks to compensate for reduced mask number; this leads to increased occlusion and fewer certification successes [33].

5.2. Impact of varying patch size

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

Strong clean performance over different patch sizes. We find that the defended clean performance of PatchDEMUX is resilient to increasing patch size; indeed, clean AP only drops from 85.731 in the smallest patch setting to 69.952 in the largest. Thus, even in unlikely scenarios (i.e., a patch size of $\geq 32\%$ would be easily detectable by hand) PatchDEMUX maintains strong inference performance. For the certified robust settings, PatchDEMUX provides relatively strong robustness guarantees on smaller patches (i.e., $\leq 2\%$) and performance degrades for larger patches (i.e., $\geq 8\%$); certified AP drops close to 0% when a patch size of 32% is considered. These trends align with experiments done by Xiang et al. [33] in the single-label classification domain; the general intuition is that larger patch sizes require PatchCleanser to generate larger masks, making

certification failures more likely [33].

5.3. Overall takeaways

Overall, we find that PatchDEMUX performance tradeoffs corroborate with findings from Xiang et al. [33]. 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 [6], small receptive field methods [17, 24, 31, 32], and masking methods [33]. CDPAs have also been proposed for object detection [34] and semantic segmentation [37], although notions of certifiable robustness are more difficult to define in these domains.

Certifiable defenses in multi-label classification. Jia et al. [15] proposed MultiGuard, a certifiably robust defense for multi-label classifiers that generalizes randomized smoothing [7]. However, MultiGuard is designed to protect against ℓ_2 -norm attacks and does not address adversarial patches.

7. Conclusion

The threat of adversarial patch attacks has compromised real-world computer vision systems, including those that depend on multi-label classifiers. To this end we introduced PatchDEMUX, a certifiably robust framework for multi-label classifiers against adversarial patches. PatchDEMUX can extend any existing single-label CDPAs, including the current SOTA single-label CDPAs PatchCleanser, and demonstrates strong performance on the MS-COCO and PASCAL VOC datasets. We hope that future work will take advantage of our modular framework to significantly mitigate the impact of adversarial patches.

8. Acknowledgements

We would like to thank the anonymous CVPR reviewers for their helpful feedback. This work was supported by National Science Foundation grants IIS-2229876 (the ACTION center) and CNS-2154873. Prateek Mittal acknowledges the support of NSF grant CNS-2131938, Princeton SEAS Innovation award, and OpenAI & FarAI superalignment grants.

References

- [1] Abhishek Aich, Calvin-Khang Ta, Akash Gupta, Chengyu Song, Srikanth V. Krishnamurthy, M. Salman Asif, and Amit K. Roy-Chowdhury. GAMA: Generative Adversarial Multi-Object Scene Attacks. In *NeurIPS 2022*. arXiv, 2022. 5
- [2] Abhishek Aich, Shasha Li, Chengyu Song, M. Salman Asif, Srikanth V. Krishnamurthy, and Amit K. Roy-Chowdhury. Leveraging Local Patch Differences in Multi-Object Scenes for Generative Adversarial Attacks. In *WACV 2023*, pages 1308–1318, Waikoloa, HI, USA, 2023. IEEE. 5
- [3] 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, 4, 8
- [4] 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
- [5] 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
- [6] 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
- [7] 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
- [8] Terrance DeVries and Graham W. Taylor. Improved Regularization of Convolutional Neural Networks with Cutout, 2017. arXiv:1708.04552 [cs]. 6, 7
- [9] 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
- [10] Mohamed Elhoseiny, Amr Bakry, and Ahmed Elgammal. MultiClass Object Classification in Video Surveillance Systems - Experimental Study. In *CVPR 2013 Workshops (CVPRW 2013)*, pages 788–793, OR, USA, 2013. IEEE. 1
- [11] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 6
- [12] Kevin Eykholt, Ivan Evtimov, Earlene 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
- [13] Marian George and Christian Floerckemeier. Recognizing Products: A Per-exemplar Multi-label Image Classification Approach. In *ECCV 2014*, pages 440–455. Springer International Publishing, 2014. 1
- [14] 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
- [15] 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
- [16] Chi-Hsi Kung, Shu-Wei Lu, Yi-Hsuan Tsai, and Yi-Ting Chen. Action-Slot: Visual Action-Centric Representations for Multi-Label Atomic Activity Recognition in Traffic Scenes. In *CVPR 2024*, pages 18451–18461, Seattle, WA, USA, 2024. IEEE. 1
- [17] 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
- [18] 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
- [19] 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
- [20] 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
- [21] 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
- [22] Hassan Mahmood and Ehsan Elhamifar. Semantic-Aware Multi-Label Adversarial Attacks. In *CVPR 2024*, pages 24251–24262, Seattle, WA, USA, 2024. IEEE. 5
- [23] Stefano Melacci, Gabriele Ciravegna, Angelo Sotgiu, Ambra Demontis, Battista Biggio, Marco Gori, and Fabio Roli. Domain Knowledge Alleviates Adversarial Attacks in Multi-Label Classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):9944–9959, 2022. 5
- [24] 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
- [25] 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
- [26] 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
- [27] 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
- [28] 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
- [29] 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
- [30] 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
- [31] 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
- [32] Chong Xiang, Arjun Nitin Bhagoji, Vikash Sehwal, 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
- [33] Chong Xiang, Saeed Mahloujifar, and Prateek Mittal. Patch-Cleanser: 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
- [34] 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
- [35] 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
- [36] 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
- [37] 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
- [38] 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 $\mathbb{F}: \mathcal{X} \rightarrow \mathcal{Y}$. 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 $\kappa[i^*] = 1$ (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 $i^* \in \{1, 2, \dots, c\}$ 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 $\kappa[i^*] = 0$, 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 $i^* \in \{1, 2, \dots, c\}$ 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 $\mathbf{y}[i^*] = 0$. \square

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 . \square

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}, \mathbf{y}) , a single-label classifier $\mathbb{F}_s: \mathcal{X} \rightarrow \{1, 2, \dots, c\}$, 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 $\lambda := SL\text{-}CERT_{[\mathbb{F}_s, \sigma]}(\mathbf{x}, \mathbf{y}, \mathcal{R}) \in \{0, 1\}^{|\mathcal{R}|}$ such that if $\lambda[\mathbf{r}] = 1$ for a patch location $\mathbf{r} \in \mathcal{R}$ then⁵*

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

Essentially, the vulnerability status array λ 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 $\mathbb{F}: \mathcal{X} \rightarrow \mathcal{Y}$. If $SL\text{-}CERT$ returns the vulnerability status array λ 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 $FN_{new} < FN$.

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.

⁵For the term $\lambda[\mathbf{r}] = 1$ we slightly abuse notation and use \mathbf{r} to refer to the index associated with the patch location

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 $i^* \in \{1, 2, \dots, c\}$. Because the $fnCertFails$ value for this class at patch location \mathbf{r}_{opt} is 0, on line 10 we must have for $\lambda := SL-CERT_{[\mathbb{F}[i^*], \sigma]}(\mathbf{x}, \mathcal{V}[i^*], \mathcal{R})$

$$\lambda[\mathbf{r}_{opt}] = 1$$

As per Definition 3, this means that we will have

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

In other words, $SL-INFER$ 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. \square

B. Double-masking Algorithm from Patch-Cleanser

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. [33].

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 $y \in \{1, 2, \dots, c\}$ for certifiable input images $\mathbf{x} \in \mathcal{X}$ [33]. More specifically, these masks satisfy the following \mathcal{R} -covering property from Xiang et al. [33].

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 [33]. Given an input image size $n_1 \times n_2$, an upper estimate on patch size⁶ p , and number of desired masks $k_1 \times k_2$, a procedure from Xiang et al. [33] 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 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) [33].

⁶PatchCleanser provides an option to specify the patch size for each axis; we simplify the notation here for convenience

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 [33].

B.2. Double-masking inference procedure

Algorithm 4 *The double-masking inference procedure from PatchCleanser [33]*

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

- 1: **procedure** DOUBLEMASKINGINFERENCE($\mathbf{x}, \mathbb{F}_s, \mathcal{M}$)
- 2: $\hat{y}_{maj}, \mathcal{P}_{dis} \leftarrow \text{MASKPRED}(\mathbf{x}, \mathbb{F}_s, \mathcal{M})$ \triangleright First-round
- 3: **if** $\mathcal{P}_{dis} = \emptyset$ **then**
- 4: **return** \hat{y}_{maj} \triangleright Case I: agreed prediction
- 5: **end if**
- 6: **for each** $(\mathbf{m}_{dis}, \hat{y}_{dis}) \in \mathcal{P}_{dis}$ **do** \triangleright Second round
- 7: $\hat{y}', \mathcal{P}' \leftarrow \text{MASKPRED}(\mathbf{x} \circ \mathbf{m}_{dis}, \mathbb{F}_s, \mathcal{M})$
- 8: **if** $\mathcal{P}' = \emptyset$ **then**
- 9: **return** \hat{y}_{dis} \triangleright Case II: disagree pred.
- 10: **end if**
- 11: **end for**
- 12: **return** \hat{y}_{maj} \triangleright Case III: majority prediction
- 13: **end procedure**

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

- 14: **procedure** MASKPRED($\mathbf{x}, \mathbb{F}_s, \mathcal{M}$)
- 15: $\mathcal{P} \leftarrow \emptyset$ \triangleright A set for mask-prediction pairs
- 16: **for** $\mathbf{m} \in \mathcal{M}$ **do** \triangleright Enumerate every mask \mathbf{m}
- 17: $\hat{y} \leftarrow \mathbb{F}_s(\mathbf{x} \circ \mathbf{m})$ \triangleright Evaluate masked prediction
- 18: $\mathcal{P} \leftarrow \mathcal{P} \cup \{(\mathbf{m}, \hat{y})\}$ \triangleright Update set \mathcal{P}
- 19: **end for**
- 20: $\hat{y}_{maj} \leftarrow \text{argmax}_{y^*} |\{(\mathbf{m}, \hat{y}) \in \mathcal{P} | \hat{y} = y^*\}|$ \triangleright Majority
- 21: $\mathcal{P}_{dis} \leftarrow \{(\mathbf{m}, \hat{y}) \in \mathcal{P} | \hat{y} \neq \hat{y}_{maj}\}$ \triangleright Disagreeers
- 22: **return** $\hat{y}_{maj}, \mathcal{P}_{dis}$
- 23: **end procedure**

The double-masking inference procedure from Xiang et al. [33] 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 $\mathbb{F}_s : \mathcal{X} \rightarrow \{1, 2, \dots, c\}$ is queried on copies of \mathbf{x} which have been augmented by masks $\mathbf{m} \in \mathcal{M}$ [33].

- *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 [33]. Otherwise, the minority/“disagreeer” 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 disagreeers [33].

- *Second-round masking*: For each disagreeer 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* [33]. If there is consensus, the disagreeer 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 [33]. Otherwise, \mathbf{m}_{dis} is ignored and the next available disagreeer mask is considered; the assumption here is that \mathbf{m}_{dis} failed to cover the patch [33]. Finally, if none of the disagreeer 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 [33].

B.3. Double-masking certification procedure

Algorithm 5 *The double-masking certification procedure from PatchCleanser [33]*

Input: Image $\mathbf{x} \in \mathcal{X}$, ground-truth $y \in \{1, 2, \dots, c\}$, single-label classifier $\mathbb{F}_s : \mathcal{X} \rightarrow \{1, 2, \dots, c\}$, patch locations \mathcal{R} , \mathcal{R} -covering mask set \mathcal{M}

Output: Overall certification status of (\mathbf{x}, y) , vulnerability status array $\lambda \in \{0, 1\}^{|\mathcal{M}|}$

```

1: procedure DOUBLEMASKINGCERT( $\mathbf{x}, y, \mathbb{F}_s, \mathcal{R}, \mathcal{M}$ )
2:    $certVal \leftarrow 1$ 
3:    $\lambda \leftarrow [1]^{|\mathcal{M}|}$ 
4:   if  $\mathcal{M}$  is not  $\mathcal{R}$ -covering then  $\triangleright$  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:      $\hat{y}' \leftarrow \mathbb{F}_s(\mathbf{x} \circ \mathbf{m}_0 \circ \mathbf{m}_1)$   $\triangleright$  Two-mask prediction
9:     if  $\hat{y}' \neq y$  then
10:        $certVal \leftarrow 0$   $\triangleright$  Input possibly vulnerable
11:        $\lambda[\mathbf{m}_0], \lambda[\mathbf{m}_1] \leftarrow 0, 0$   $\triangleright$  Vulnerable masks
12:     end if
13:   end for
14:   return  $certVal, \lambda$ 
15: end procedure

```

The double-masking certification procedure from Xiang et al. [33] is outlined in Algorithm 5; we extend the original version to additionally return a vulnerability status array λ . 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 $\mathbf{m}_0, \mathbf{m}_1 \in \mathcal{M} \times \mathcal{M}$ [33]. 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. Otherwise, $certVal$ is set to 0 and the λ 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 [33].

Theorem 3. *Suppose we have an image data point (\mathbf{x}, y) , a single-label classification model $\mathbb{F}_s : \mathcal{X} \rightarrow \{1, 2, \dots, c\}$, 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. [33]. \square

We next consider the vulnerability status array $\lambda \in \{0, 1\}^{|\mathcal{M}|}$ 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 $\lambda[\mathbf{m}]$ 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 $\mathbb{F}_s : \mathcal{X} \rightarrow \{1, 2, \dots, c\}$, a patch threat model $S_{\mathbf{x}, \mathcal{R}}$, and a \mathcal{R} -covering mask set \mathcal{M} . Then the array $\lambda \in \{0, 1\}^{|\mathcal{M}|}$ 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 λ , we need to show that $\lambda[\mathbf{m}^*] = 1$ implies Algorithm 4 will be protected from all attacks located in \mathcal{R}^* . To do so, we first note that we will only have $\lambda[\mathbf{m}^*] = 1$ in Algorithm 5 if $\mathbb{F}_s(\mathbf{x} \circ \mathbf{m}^* \circ \mathbf{m}) = y$ for all $\mathbf{m} \in \mathcal{M}$; otherwise, $\lambda[\mathbf{m}^*]$ 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 $\lambda[\mathbf{m}^*] = 1$. 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 disagreeer 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 disagreeer 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 disagreeer 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 disagreeers 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 $\lambda[\mathbf{m}^*] = 1$ implies Algorithm 4 will be protected from any arbitrary patch attack located in \mathcal{R}^* , as desired. \square

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

We evaluate multi-label classifiers by computing precision and recall metrics over a variety of different thresholds; classes with output higher than the threshold are predicted 1, otherwise 0. This helps establish a large set of evaluation data from which to build precision-recall plots. We start by evaluating a set of *standard thresholds*:

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

We then evaluate a set of *high-value thresholds*. This helps fill out the *low recall-high precision* region of a precision-recall curve:

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

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

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

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

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

where

$$T_{\text{mid1}} := \{0.5 + 0.02 \cdot t : t \in \{1, 2, 3, 4\}\}$$

$$T_{\text{mid2}} := \{0.6 + 0.02 \cdot t : t \in \{1, 2, 3, 4\}\}$$

$$T_{\text{mid3}} := \{0.7 + 0.02 \cdot t : t \in \{1, 2, 3, 4\}\}$$

$$T_{\text{mid4}} := \{0.8 + 0.02 \cdot t : t \in \{1, 2, 3, 4\}\}$$

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:

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

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 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 points for the precision-recall curves. In order to enforce consistency, we fix the leftmost points 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 a few evaluations under this value demonstrate floating-point precision errors (i.e., the required threshold is too high).

D. Resnet Architecture Analysis

In this section, we report results for PatchDEMUX while using the Resnet architecture [3]; we leverage the same defense fine-tuning routine as Sec. 4.2 to achieve stronger performance. Experiments are done on the MS-COCO 2014 validation dataset. 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 and ViT architectures show similar qualitative trends. For instance, defended clean performance is close to undefended performance across a variety of thresholds (see Fig. 5a and Tab. 2a). The Resnet-based variant of PatchDEMUX also achieves non-trivial robustness, with a certified average precision of 37.544%. In general, the precision-recall curves for the two architectures are similar in shape across all four evaluation settings.

Despite these similarities, the ViT model consistently outperforms the Resnet model. More specifically, the ViT-based variant of PatchDEMUX provides a ~ 4 point boost to clean

Table 2. PatchDEMUX performance with Resnet architecture on the MS-COCO 2014 validation dataset. Precision values are evaluated at key recall levels along with the approximated average precision. We assume the patch attack is at most 2% of the image area and use a computational budget of 6×6 masks.

(a) Clean setting precision values					(b) Certified robust setting precision values				
Architecture	Resnet				Architecture	Resnet			
Clean recall	25%	50%	75%	AP	Certified recall	25%	50%	75%	AP
Undefended	99.832	99.425	92.341	87.608	Certified robust	86.696	40.190	20.959	34.859
Defended	99.835	98.257	80.612	81.031	Location-aware	87.950	44.373	23.202	37.544

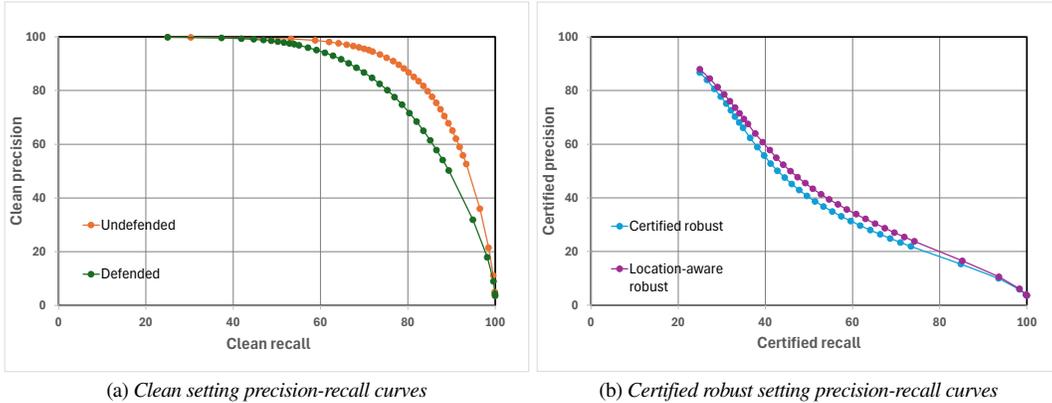


Figure 5. PatchDEMUX precision-recall curves with Resnet architecture over the MS-COCO 2014 validation dataset. We consider the clean and certified robust evaluation settings. We assume the patch attack is at most 2% of the image area and use a computational budget of 6×6 masks.

AP in the defended clean setting and a ~ 7 point boost to certified AP in the two certified robust settings (see Tab. 1). This improvement might be attributable to the training procedure of vision transformers, which involves a masking process that is similar in concept to PatchCleanser’s double-masking algorithm [9, 33].

E. Location-aware Certification Analysis

Table 3. ViT-based PatchDEMUX performance with different location-aware attackers. Experiments performed on the MS-COCO 2014 validation dataset. Precision values are evaluated at key recall levels along with the approximated average precision. We assume the patch attack is at most 2% of the image area and use a computational budget of 6×6 masks.

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	44.902
Certified robust	95.369	50.950	22.662	41.763

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

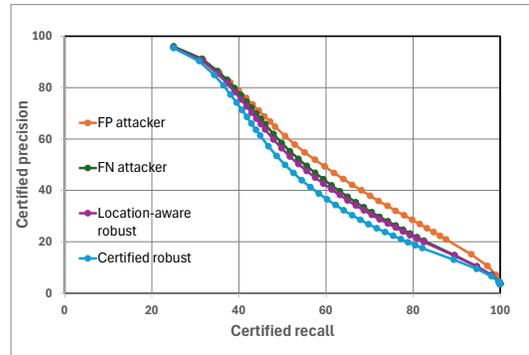


Figure 6. ViT-based PatchDEMUX precision-recall curves with different location-aware attackers. Experiments performed on the MS-COCO 2014 validation dataset. The baseline certified robust evaluation setting is included for comparison. We assume the patch attack is at most 2% of the image area and use a computational budget of 6×6 masks.

E.1. Attack vectors

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 λ for false negatives. Intuitively, this corresponds to the optimal attacker from Sec. 3.4.2 constructing a patch with the 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 scenario we only track vulnerability status arrays λ 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. 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 evaluate this approach because 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. Experiments are done on the MS-COCO 2014 validation dataset. 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 baseline certified robust 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 ~ 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 negatives⁷, but the recall metric does not depend on false positives. This makes *FP* attacks “weaker” relative to other methods.

F. Defense Fine-tuning for PatchDEMUX

Single-label CDPAs often leverage defense fine-tuning routines to improve the robustness of underlying single-label classifiers; these work by training the model on specially augmented data [8, 28, 33]. In this section, we investigate whether some of these routines can extend to multi-label classifiers and improve PatchDEMUX performance. We specifically consider fine-tuning strategies used by PatchCleanser, as PatchCleanser is the certifiable backbone for PatchDEMUX in this work.

⁷Precision indirectly depends on false negatives via the true positive count

F.1. Defense fine-tuning techniques for PatchCleanser

Two different defense fine-tuning techniques have been used to improve the performance of PatchCleanser: *Random Cutout* [8] and *Greedy Cutout* [28]. 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 [8, 33]. Xiang et al. [33] found that Random Cutout fine-tuning provides significant boosts to the robustness of PatchCleanser; intuitively, using cutout masks for defense fine-tuning helps the underlying model become more tolerant to occlusion effects from double-masking procedures. Later, Saha et al. [28] proposed the Greedy Cutout fine-tuning procedure and demonstrated superior performance to Random Cutout for PatchCleanser. This approach works by augmenting each training image with the pair of certification masks that greedily induce the highest loss.

F.2. Defense fine-tuning methodology

In our experiments we compare the following three defense fine-tuning methods, which are representative of settings used in prior work [8, 28, 33].

- Random Cutout fine-tuning with two square 25% masks
- Greedy Cutout fine-tuning with 6×6 certification masks
- Greedy Cutout fine-tuning with 3×3 certification masks

For Greedy Cutout, we compute the loss for 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. [28] due to difficulties with mask decompositions.

To train the model with these methods, we first obtain existing checkpoints for the MS-COCO 2014 classification task [3, 20]. We then follow the training methodology for multi-label classifiers outlined by Ben-Baruch et al. [3]. 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 [3]. Models are fine-tuned on copies of the MS-COCO 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 data⁸. A cluster of NVIDIA A100 40GB GPUs are used to perform the fine-tuning.

F.3. Experiment results

Results for the different defense fine-tuning routines are in Tab. 4. In addition, precision-recall plots comparing the defense fine-tuning 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. Experiments are done on the MS-COCO 2014 validation dataset.

Defense fine-tuning boosts performance. In general, we find that using a defense fine-tuning routine of any kind leads

⁸We find that fine-tuning for longer leads to overfitting.

Table 4. PatchDEMUX performance with ViT architecture on the MS-COCO 2014 validation dataset when using different defense fine-tuning techniques. Precision values are evaluated at key recall levels along with the approximated average precision. We assume the patch attack is at most 2% of the image area and use a computational budget of 6×6 masks.

(a) Clean setting precision values

Architecture	ViT (vanilla/no fine-tuning)				ViT (Random Cutout)				ViT (Greedy Cutout 6×6)				ViT (Greedy Cutout 3×3)			
	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	91.146	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	85.276	99.900	99.230	87.741	85.271

(b) Certified robust setting precision values

Architecture	ViT (vanilla/no fine-tuning)				ViT (Random Cutout)				ViT (Greedy Cutout 6×6)				ViT (Greedy Cutout 3×3)			
	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	41.763	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	44.902	95.959	55.958	27.105	45.122

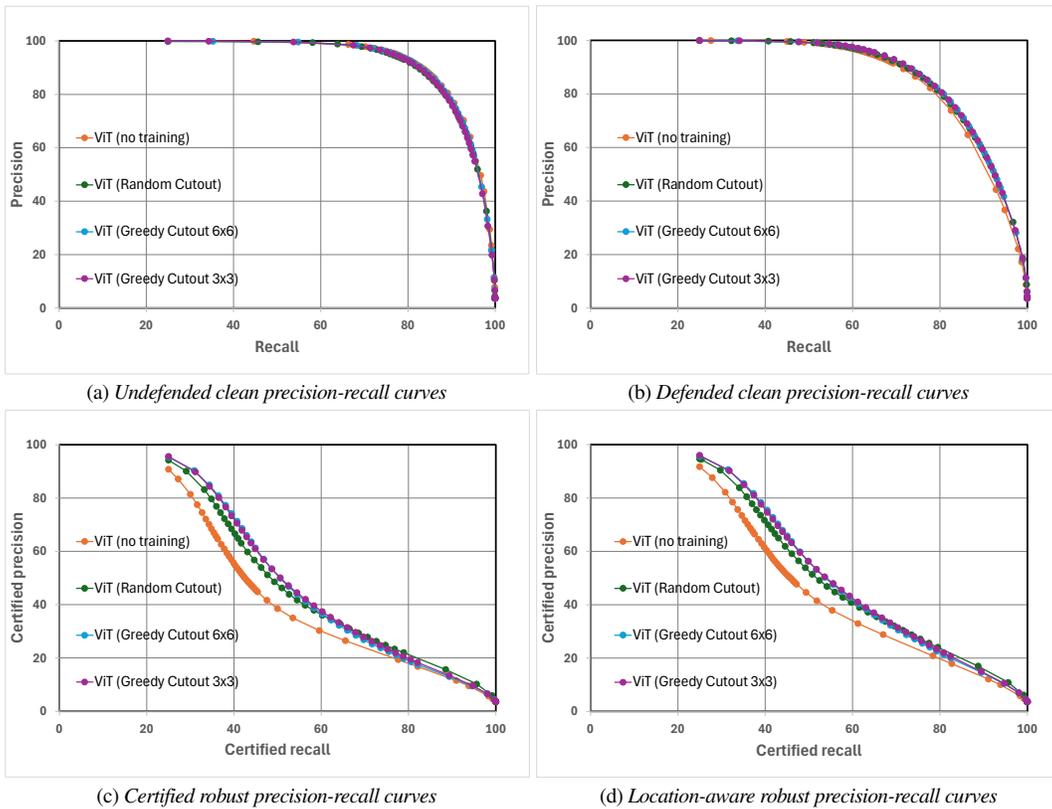


Figure 7. PatchDEMUX precision-recall curves with ViT architecture over the MS-COCO 2014 validation dataset when using different defense fine-tuning techniques. We consider each of the four evaluation settings in separate plots. We assume the patch attack is at most 2% of the image area and use a computational budget of 6×6 masks.

to performance boosts for PatchDEMUX. For instance, fine-tuning 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 ~ 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. [28]. Note that in general defense fine-tuning strategies are less effective in the clean settings. This is likely because the clean settings already demonstrate (relatively) strong performance, and thus potential gains from fine-tuning are more marginal. Nevertheless, we prioritize the defended clean setting overall as it is most representative of typical performance.

The Greedy Cutout 6×6 fine-tuning 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 defense fine-tuning. This suggests that our location-aware certification technique has general utility across a variety of scenarios and that it “stacks” with other sources of robustness improvements.

G. Runtime Analysis of PatchDEMUX

Table 5. Runtime experiments on PatchDEMUX. We report median per-sample inference time (in milliseconds) across a random sample of 2000 datapoints from the MS-COCO 2014 validation dataset. We assume the patch attack is at most 2% of the image area.

Architecture	ViT (2×2 masks)	ViT (4×4 masks)	ViT (6×6 masks)
Undefended	31.130	31.130	31.130
Defended (single-label)	200.61	674.98	1451.1
Defended (multi-label)	317.30	1892.3	5668.7

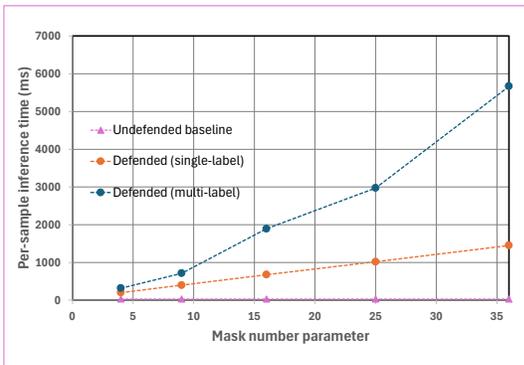


Figure 8. Plot of PatchDEMUX runtime as a function of mask number. We report median per-sample inference time (in milliseconds) across a random sample of 2000 datapoints from the MS-COCO 2014 validation dataset. We assume the patch attack is at most 2% of the image area.

In this section, we analyze the runtime of the PatchDEMUX inference procedure. To determine the impact of class number, we create a restricted version of our inference procedure that operates only on the first class (i.e., it ignores the remainder of the label $\mathbf{y} \in \{0, 1\}^c$); this is essentially an instance of PatchCleanser isolated to a single class. We then track the runtime for 2000 random datapoints from the MS-COCO 2014 validation dataset. We use the ViT checkpoints from Sec. 4.2 and use a batch size of 1 to directly obtain per-sample inference time. The median per-sample inference times for different mask numbers are present in Tab. 5 and Fig. 8.

We note that for most mask numbers the full multi-label inference procedure takes roughly $3 \times$ longer than the single-label implementation. Given that MS-COCO has $c=80$ classes, this is significantly faster than the expected runtime for the naive

method from Algorithm 1. The reason for this improvement is an implementation-level optimization that takes advantage of relatively negligible defense post-processing. Specifically, the primary bottleneck for single-label inference procedures is often model query time; the associated defense post-processing is negligible in comparison. This means that for each feedforward through the multi-label classifier we can apply single-label defense post-processing to every class and re-use individual class outputs as needed for multi-label inference. As an example, with PatchCleanser this is done by saving intermediate outputs that correspond to double-masked images. Overall, this technique helps prevent computation cost from increasing drastically with the number of classes.

Despite this optimization, we note that the multi-label inference implementation is still not as fast as the single-label inference implementation. This is because many single-label inference procedures have worst-case scenarios which take significantly longer than typical cases. Increasing the number of classes increases the possibility that at least one class will trigger a worst-case scenario, leading to longer overall runtime.

H. Performance on PASCAL VOC

In this section we report evaluation results for PatchDEMUX on PASCAL VOC. Because model checkpoints for PASCAL VOC are not readily available, we first create a multi-label classifier for the PASCAL VOC task. To do so, we use model checkpoints pre-trained on the MS-COCO dataset and fine-tune it for the PASCAL VOC dataset. We use asymmetric loss (ASL) as the loss function, a 1cycle learning rate policy with max learning rate $\alpha_{max} = 2.0 \cdot 10^{-3}$, automatic mixed precision (AMP) for faster training, and exponential moving average (EMA) of model checkpoints for improved inference [3]. Models are fine-tuned on the PASCAL VOC 2007 training split. We use the Adam optimizer for 15 epochs and select the best checkpoint according to average loss on the PASCAL VOC 2007 validation split. A cluster of NVIDIA A100 40GB GPUs are used to perform the fine-tuning. We omit additional security fine-tuning to focus on baseline performance.

We evaluate the fine-tuned model on the PASCAL VOC 2007 test dataset. We summarize the precision values associated with key recall levels in Tab. 6. Fig. 9 features precision-recall plots, while AP values are present in Tab. 6. We consider the ViT architecture alone as it provides better performance compared to Resnet.

Strong all-around performance. As shown in Tab. 6 and Fig. 9, PatchDEMUX achieves strong performance in all evaluation settings. In fact, PatchDEMUX’s performance on PASCAL VOC is significantly higher than its performance on MS-COCO, with a ~ 7 point increase in defended clean performance and ~ 12 point increase in certified robustness metrics (see Sec. 4.2). Overall, these stronger results are expected given that the PASCAL VOC benchmark has fewer classes than MS-COCO, making it an easier benchmark for classifiers to predict.

Concave robustness curves. An interesting observation is that both of the PASCAL VOC robustness curves are concave

Table 6. PatchDEMUX performance with ViT architecture on the PASCAL VOC 2007 validation dataset. Precision values are evaluated at key recall levels along with the approximated average precision. We assume the patch attack is at most 2% of the image area and use a computational budget of 6×6 masks.

(a) Clean setting precision values					(b) Certified robust setting precision values				
Architecture	ViT				Architecture	ViT			
Clean recall	25%	50%	75%	AP	Certified recall	25%	50%	75%	AP
Undefended	99.790	99.710	98.506	96.140	Certified robust	90.520	74.675	38.100	54.904
Defended	99.894	99.870	98.167	92.593	Location-aware robust	90.591	75.672	40.320	56.030

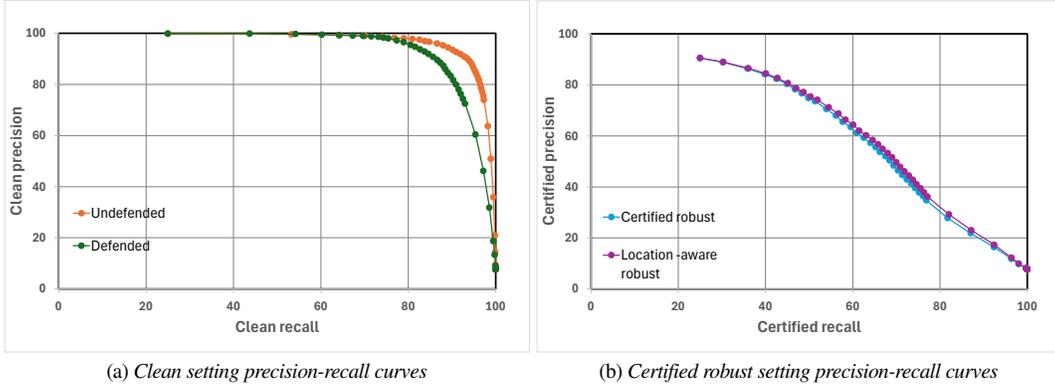


Figure 9. PatchDEMUX precision-recall curves with ViT architecture over the PASCAL VOC 2007 test dataset. We consider the clean and certified robust evaluation settings. We assume the patch attack is at most 2% of the image area and use a computational budget of 6×6 masks.

in Fig. 9b. This is in contrast to MS-COCO experiments, where even after applying security fine-tuning methods the robustness curves remained convex (see Fig. 7). An important takeaway from this is that PatchDEMUX performance is dataset dependent, and robustness bounds will ultimately depend on the nature of image datapoints and/or labels. Additionally, we note that location-aware certification only provides a ~ 1 point boost to certified AP; this suggests that location-aware certification is most beneficial when baseline robustness bounds are weak.

I. Tables for Security Parameter Experiments

In this section we provide the tables associated with the security parameter experiments in Sec. 5. In Tab. 7 we list metrics associated with the mask number experiments from Sec. 5.1. In Tab. 8 we list metrics associated with the patch size experiments from Sec. 5.2.

Table 7. PatchDEMUX performance with ViT architecture on the MS-COCO 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)			
	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)			
	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 8. PatchDEMUX performance with ViT architecture on the MS-COCO 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)			
	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)			
	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