WIP: Towards a Certifiably Robust Defense for Multi-label Classifiers Against Adversarial Patches

Dennis G. Jacob
Princeton University
djacob@princeton.edu

Chong Xiang
Princeton University
cxiang@princeton.edu

Prateek Mittal
Princeton University
pmittal@princeton.edu

***Abstract*—The advent of deep learning has brought about vast improvements to computer vision systems and facilitated the development of self-driving vehicles. Nevertheless, these models have been found to be susceptible to adversarial attacks. Of particular importance to the research community are patch attacks, which have been found to be realizable in the physical world. While certifiable defenses against patch attacks have been developed for tasks such as single-label classification, there does not exist a defense for multi-label classification. In this work, we propose such a defense called Multi-Label PatchCleanser, an extension of the current state-of-the-art (SOTA) method for single-label classification. We find that our approach can achieve non-trivial robustness on the MSCOCO 2014 validation dataset while maintaining high clean performance. Additionally, we leverage a key constraint between patch and object locations to develop a novel procedure and improve upon baseline robust performance.**

I. 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 [4]. However, the vast improvements in performance comes at a security cost; indeed, many of these models are vulnerable to *adversarial attacks* [8]. Specifically, by perturbing regular input images in a systematic manner it is possible to craft *adversarial samples* which fool the deep learning model despite visual similarity to humans.

The types of images allowed in a given attack are characterized by an adversarial *threat model*. One commonly analyzed framework is the patch threat model, in which the attacker is limited to adjusting values within a specified patch-shaped region of the target image [9]. The patch threat model presents a unique challenge for the security community due to its physically-realizable nature. For instance, [6] were able to print out life-sized billboards with adversarial samples and successfully fool semantic segmentation models. This is particularly alarming for safety critical systems, such as autonomous vehicles. It is thus of utmost importance that *certifiable defenses*, which are invulnerable to arbitrarily designed patch attacks, are proposed for such systems.

The recently proposed PatchCleanser certifiable defense against patch attacks is able to achieve state-of-the-art (SOTA) robustness in the single-label classification domain [9]. This method works by leveraging a set of double-masks (i.e., pairs of masks which occlude sections of the input image) to provably recover the model prediction. Despite this success, a certifiable defense against patch attacks does not exist for certain other computer vision tasks, such as multi-label classification. To this end, we propose a certifiable defense against patch attacks in the multi-label classification domain called Multi-Label PatchCleanser by implementing an extension of the PatchCleanser algorithm. Specifically, we design our defense to apply the double-masking algorithm from PatchCleanser on each class individually. We demonstrate that Multi-Label PatchCleanser achieves non-trivial robustness with 47.36% certified precision and 38.55% certified recall on the MSCOCO 2014 validation dataset. It also maintains high performance on clean data with 92.67% precision and 61.75% recall. Finally, we identify a key constraint between patch and object locations in the multi-label classification domain that allows us to develop a better certification methodology. This approach results in tighter bounds on robustness (i.e., $\sim 4\%$ improvement on certified precision and recall metrics).

II. Background

A. The patch threat model

As discussed in Section I, a *threat model* defines the set of allowable adversarial attacks. Suppose we have an input vector $x \in \mathbb{R}^d$, where d corresponds to the input dimension of the defending model. Then, we can define the patch threat model as equivalent to the ℓ_0 threat model constrained to square geometric regions. Specifically, we will have the following set of allowable attacks:

$$S_x = \{v \circ x + (1-v) \circ x' | x, x' \in \mathbb{R}^d, v \in \mathcal{V}\} \tag{1}$$

Here, x' refers to another arbitrary vector in $\mathbb{R}^d$ and $v \in \{0,1\}^d$ represents a binary vector. The $\circ$ operator refers to element-wise multiplication. Finally, $\mathcal{V} \subseteq \{0,1\}^d$ refers to the set of vectors which represent square-shaped restricted regions; elements inside the region are 0 while those outside the region are 1 [9].

B. Certifiable defenses against adversarial attacks

Certifiable defenses are beneficial in that they can provide provable guarantees on robustness. Within the context of the patch threat model, certifiable robustness implies that for some robustness criterion $\mathcal{C} : \mathbb{R}^d \to \{0,1\}^*$ and *certifiable* image-label pairs (x, y) we can guarantee:

$$\mathcal{C}(x) = \mathcal{C}(x') \quad \forall\, x' \in S_x \tag{2}$$

Workshop on AI Systems with Confidential Computing (AISCC) 2024
26 February 2024, San Diego, CA, USA
ISBN 979-8-9894372-4-5
https://dx.doi.org/10.14722/aiscc.2024.23008
www.ndss-symposium.org

In other words, a certifiable defense ensures that for certain (x, y) the robustness property described by the function $\mathcal{C}$ is preserved for every possible attack in the threat model.

One example of a certifiable defense against patch attacks is the recently proposed PatchCleanser framework, which achieves SOTA performance in the single-label classification domain [9]. It leverages a special set of double-masks in order to recover the label y given an arbitrary patch anywhere on a certifiable input image x [9]. Thus, the associated robustness criterion for PatchCleanser follows:

$$\mathcal{C}(x') = y \quad \forall\, x' \in S_x \tag{3}$$

One important feature of this work is that it is architecture agnostic, and works in essence as a "pre-processing" routine on top of the machine learning model itself.

C. Multi-label classification

We next give an overview of the multi-label classification domain [1]. Unlike the single-label classification domain, here each input image x can potentially contain several objects *simultaneously* with each object corresponding to one of c classes. A classifier is then tasked with recovering the labels associated with each of objects present. As such, a label in the multi-label classification setting will be a bitstring $\ell \in \{0,1\}^c$ where $\ell[i] = 1$ implies class i is present and $\ell[i] = 0$ implies class i is absent.

More rigorously, consider an input datapoint (x, y) where $x \in \mathbb{R}^d$ corresponds to an input image and $y \in \{0,1\}^c$ corresponds to the image label. Then the multi-label classification task aims to train a model $F_{\theta^*} : \mathbb{R}^d \to \{0,1\}^c$ with optimal weights θ^* such that the predicted label $\ell = F(x)$ is equal to y. Note that in comparison to single-label classification, the set of labels for multi-label classification is 2^c in size (i.e., exponential) instead of just c (i.e., linear).

To evaluate the performance of a multi-label classifier, we will use the *precision* and *recall* metrics defined as follows:

$$precision = \frac{TP}{TP + FP} \qquad recall = \frac{TP}{TP + FN} \tag{4}$$

Here, for a given ground-truth label $y \in \{0,1\}^c$ the values TP, FP, FN represent the number of true positives, false positives, and false negatives predicted by F_θ respectively. Successful patch attacks in the multi-label setting introduce incorrect predictions in order to lower the associated precision and/or recall.

III. Methodology

In this section, we propose *Multi-Label PatchCleanser*. This is an extension to the original PatchCleanser in [9] and is designed to provide certifiable robustness against the patch attack in the multi-label classification domain. We first describe our approach of applying the double-masking method from PatchCleanser to each class individually. This is followed by proofs of correctness for the inference algorithm, along with an outline of the associated certification procedure which determines which images can be certified. Finally, an additional certification method is proposed based upon failure points in the original defense framework; this is able to achieve tighter bounds on robustness.

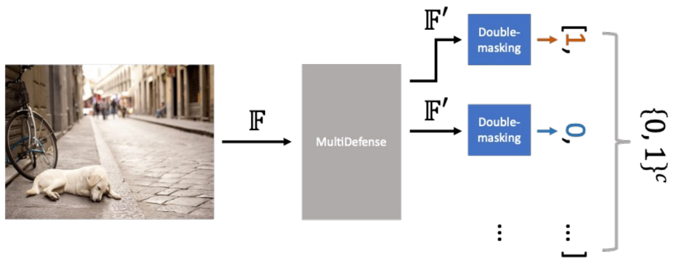


Fig. 1: *A diagram which illustrates the structure of our inference algorithm. Specifically, the double-masking algorithm from PatchCleanser is applied to each class in a label individually. This is done by considering the multi-label classifier $\mathbb{F}$ as a set of individual binary classification tasks $\mathbb{F}'$.*

A. The Multi-Label PatchCleanser defense

In *Multi-Label PatchCleanser*, we propose a defense which applies double-masking separately to each class and then aggregates the results (see Figure 1). Specifically, for a given datapoint (x, y) we will consider each class in the ground-truth label y. Then we will apply the double-masking algorithm from PatchCleanser to this individual class, as if they were an isolated binary classification task. We continue for all classes $1 \le i \le c$ and return the final prediction vector $\ell_{pred} \in \{0,1\}^c$ by pooling the results associated with each of the individual classes.

Algorithm 1 *The inference algorithm associated with Multi-Label PatchCleanser*

```
    Input: Image $x \in \mathbb{R}^d$, multi-label classifier $\mathbb{F} : \mathbb{R}^d \to \{0,1\}^c$, mask set $\mathcal{M}$, number of classes $c$
    Output: Prediction $\hat{y} \in \{0,1\}^c$
1: procedure MultiDefense($x, \mathbb{F}, \mathcal{M}, c$)
2:     $preds \leftarrow \{0\}^c$    ▷ Set class predictions to zero vector
3:     for $i \leftarrow 1$ to $c$ do    ▷ Consider each class individually
4:         $\mathbb{F}' \leftarrow \mathbb{F}[i]$    ▷ Define binary classifier for class $i$
5:         $preds[i] \leftarrow$ DoubleMasking($x, \mathbb{F}', \mathcal{M}$)
6:     end for
7:     return $preds$
8: end procedure
```

1) Inference algorithm description: The Multi-Label PatchCleanser inference algorithm is outlined in Algorithm 1. It works by first generating a set of masks $\mathcal{M}$ which satisfy the following $\mathcal{R}$-covering property from [9]:

Definition 1 ($\mathcal{R}$-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.,*

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

We then initialize a $preds \in \{0,1\}^c$ variable on line 2 which will end up containing the individual predictions for each of the classes. Next, on line 4 we define the binary classifier $\mathbb{F}'$ which represents the outputs for a given class i from the multi-label classifier $\mathbb{F}$. Finally, on line 5 we run up to two rounds of masking on the input image via the $DOUBLEMASKING(x, \mathbb{F}', \mathcal{M})$ procedure from [9] (more details on this subroutine are present in Algorithm 3 of Appendix A) and update the $preds$ variable for class i. Note that in practice it is not necessary to run model inference c times for different $\mathbb{F}'$; instead, by construction we can get predictions for all $\mathbb{F}'$ with one call to $\mathbb{F}$.

2) Inference algorithm proofs of robustness: We now provably demonstrate the robustness associated with the approach outlined in Algorithm 1. To this end, we first define the concept of *class two-mask correctness*. This is similar to the definition for *two-mask correctness* from [9], but generalized for the multi-label setting.

Definition 2 (class two-mask correctness)**.** *Suppose we have the set of class labels $\mathcal{L} = \{1, 2, \ldots, c\}$ and a clean image data point (x, y) with $x \in \mathbb{R}^d$ and $y \in \{0,1\}^c$. A multi-label classification model $\mathbb{F} : \mathbb{R}^d \to \{0,1\}^c$ has class two-mask correctness for a class $i \in \mathcal{L}$ and a mask set $\mathcal{M}$ and a clean image data point (x, y), if model predictions for class i on all possible two-masked images are correct:*

$$\mathbb{F}(x \circ m_0 \circ m_1)[i] = y[i], \forall m_0 \in \mathcal{M}, \forall m_1 \in \mathcal{M}$$

We now have the following theorem, which demonstrates that the proposed inference algorithm will be provably correct for an arbitrary class i provided that we have a $\mathcal{R}$-covering mask set $\mathcal{M}$ and also class two-mask correctness on class i.

Theorem 1 (Multi-Label PatchCleanser correctness on one class)**.** *Suppose we have the set of class labels $\mathcal{L} = \{1, 2, \ldots, c\}$ and a clean image data point (x, y) with $x \in \mathbb{R}^d$ and $y \in \{0,1\}^c$. Given a multi-label classification model $\mathbb{F} : \mathbb{R}^d \to \{0,1\}^c$, a mask set $\mathcal{M}$, and the threat model S_x, if $\mathcal{M}$ is $\mathcal{R}$-covering and $\mathbb{F}$ has class two-mask correctness on class $i \in \mathcal{L}$ for $\mathcal{M}$ and (x, y), then Algorithm 1 will always return the correct class label:*

$$MULTIDEFENSE(x', \mathbb{F}, \mathcal{M}, c)[i] = y[i], \forall x' \in S_x$$

Proof: Assume that the conditions of Theorem 1 hold (i.e., $\mathcal{R}$-covering and class two-mask correctness on class $i \in \mathcal{L}$). Now consider when Algorithm 1 reaches index $i \in \mathcal{L}$ in the for loop on line 3. By assumption, the binary classifier $\mathbb{F}'$ corresponding to class i defined on line 5 will have two-mask correctness (i.e., Definition 2 from [9]) on the associated datapoint $(x, y[i])$. Therefore, we can apply Theorem 1 from [9] and guarantee that the correct label for class $i \in \mathcal{L}$ is returned. ■

3) Multi-Label PatchCleanser certification algorithm: Assuming that we have a $\mathcal{R}$-covering mask set $\mathcal{M}$, then via Theorem 1 we can determine if a datapoint (x, y) is certifiable for class i by checking if Definition 2 is satisfied. This is encapsulated by Algorithm 2, the certification procedure for Multi-Label PatchCleanser. Specifically, as shown on lines $7 - 13$ we can run $\hat{y} = \mathbb{F}(x \circ m_0 \circ m_1)$ for all $m_0, m_1 \in \mathcal{M}$; if we have $\hat{y}[i] \neq y[i]$ for any two-mask combination, then certification of class i fails and we mark this on the $certifyStatus$ array. The loop on line 9 accounts for the fact that we eventually perform this class-level certification for every class.

B. Robust metrics

In this subsection, we discuss how to compute robustness metrics such as *certified precision* and *certified recall* for multi-label classifiers. To do so, we first consider a datapoint (x, y)

Algorithm 2 *The certification procedure associated with Multi-Label PatchCleanser*

```
    Input: Image $x \in \mathbb{R}^d$, ground-truth $y \in \{0,1\}^c$, multi-label classifier $\mathbb{F} : \mathbb{R}^d \to \{0,1\}^c$, mask set $\mathcal{M}$, number of classes $c$
    Output: Boolean list of classes certified $certifyStatus$
 1: procedure MULTCERTIFICATION($x, y, \mathbb{F}, \mathcal{M}, c$)
 2:     if $\mathcal{M}$ is not $\mathcal{R}$-covering then
 3:         return $\{False\}^c$
 4:     end if
 5:     $certifyStatus \leftarrow \{True\}^c$
 6:     for every $(m_0, m_1) \in \mathcal{M} \times \mathcal{M}$ do
 7:         $\hat{y} \leftarrow \mathbb{F}(x \circ m_0 \circ m_1)$                ▷ Two-mask pred.
 8:         for $i \leftarrow 1$ to $c$ do   ▷ Certify each class separately
 9:             if $\hat{y}[i] \neq y[i]$ then
10:                 $certifyStatus[i] \leftarrow False$
11:             end if
12:         end for
13:     end for
14:     return $cerifyStatus$
15: end procedure
```

and define the following robustness criterion for the multi-label classification domain:

$$\mathcal{C}(x) = \{TP_{lower}(x), FP_{upper}(x), FN_{upper}(x)\} \quad (5)$$

Here, $TP_{lower}(x)$ represents the lower bound on true positives for x, $FP_{upper}(x)$ is the upper bound on false positives for x, and $FN_{upper}(x)$ is the upper bound on false negatives for x. These values can be computed by running Algorithm 4 from Appendix B, which leverages the $certifystatus$ array returned by the certification procedure Algorithm 2.

We now leverage these values to craft notions of certified robustness for the multi-label classification domain. Specifically, we leverage Equation 4 in order to now define:

$$certified\, precision = \frac{TP_{lower}(x)}{TP_{lower}(x) + FP_{upper}(x)} \quad (6)$$

$$certified\, recall = \frac{TP_{lower}(x)}{TP_{lower}(x) + FN_{upper}(x)} \quad (7)$$

Note that by monotonicity of the numerators, both of these metrics will serve as the lower bounds for the precision and recall metrics associated with the datapoint (x, y). Additionally, by accumulating the tracked certification values for each datapoint (x, y) over the entirety of the dataset we can compute the lower bounds on precision and recall corresponding to the dataset as a whole. We leverage this in Section IV.

C. Proposing a new certification method

We now discuss an improved certification algorithm which extends Algorithm 2. The general intuition is that the certification masks which induce failure can be different across classes, and ultimately the adversarial patch can only be placed at one location. By leveraging this constraint we can recover residual robustness by considering the most frequent mask failure across classes.

Figure 2 uses a sample image from the 2014 MSCOCO validation dataset to help demonstrate the general strategy. The

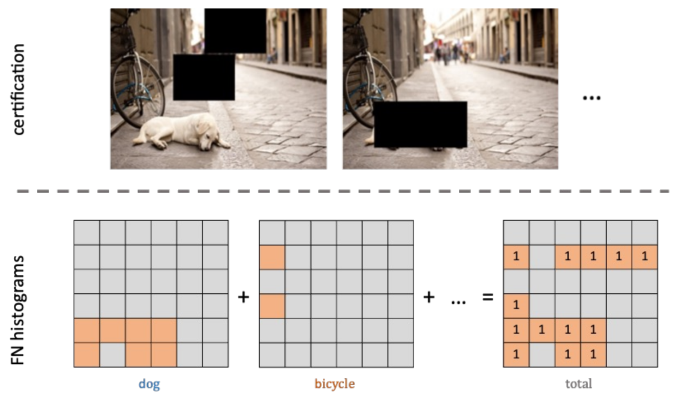


Fig. 2: *A diagram which demonstrates the certification histogram generation process on a sample MSCOCO image; here, we only consider false negatives FN. Each histogram is represented as a 6×6 grid of masks, the default configuration in PatchCleanser. Orange tiles correspond to masks which caused certification failure, and numeric values correspond to the number of classes failed.*

image contains three objects: a dog, a person, and a bicycle. As shown in the figure, during certification certain double-mask pairs are able to occlude the dog in the bottom-left hand corner of the image. In fact, each of the three objects fails certification due to this occlusion effect; if we were to use Algorithm 2 on this image we would only certify 0% precision and recall.

To this end, we define the concept of a *certification histogram*. For a given class i that failed certification and $\mathcal{R}$-covering mask set $\mathcal{M}$ we define the histogram $\mathcal{H}_i \in \{0,1\}^{|\mathcal{M}|}$:

$$\mathcal{H}_i(m) = \begin{cases} 1 & \exists (m, m') \in \mathcal{M} \times \mathcal{M} \; s.t. \; \mathcal{A} = \text{True} \\ 0 & otherwise \end{cases} \tag{8}$$

Here we leverage notation from Definition 2 and define the Boolean expression $\mathcal{A} := \mathbb{F}(x \circ m \circ m')[i] \neq y[i]$ for convenience. Essentially, a mask in $\mathcal{H}_i$ is marked if it contributes to the certification failure of class i. The bottom half of Figure 2 displays the certification histograms associated with the dog and the bicycle; a tile is colored orange if the corresponding mask contributed to certification failure.

The key insight is that we can add individual class histograms together to form a total histogram; here, each tile value represents the number of classes failed by the corresponding mask. In the worst case, one mask can lead to certification failure in all classes. Note however that in Figure 2 no mask contributes to more than one class certification failure. This implies that at most one class can fail at inference time, regardless of where the attacker places the adversarial patch (a proof for this intuition is presented in Appendix C); this is a tighter bound on robustness compared to Algorithm 2!

Our novel histogram-based certification method generalizes this procedure for an arbitrary image in Algorithm 5 of Appendix C. Strengths of Algorithm 5 include the fact that it does not require adjustments to the current inference method Algorithm 1. Additionally, any robustness discovered through this method is residual in nature and can only improve upon the bounds found via the current cetification strategy Algorithm 2.

TABLE I: *Performance on overall dataset, threshold of* 0.8

	Class prec.	Class rec.	Overall prec.	Overall rec.
Undefended clean	87.41%	76.38%	88.07%	79.23%
Defended clean	90.25%	58.00%	92.67%	61.75%
Certified robust	39.39%	33.74%	47.36%	38.55%

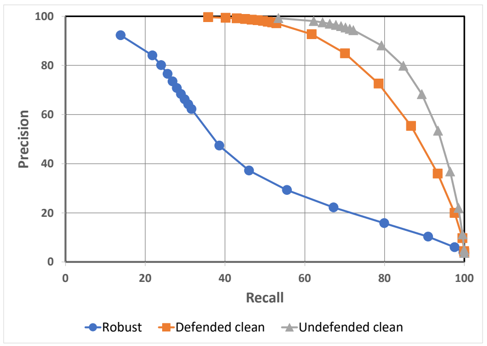


Fig. 3: *Precision-recall plots of the three model settings for the overall dataset. Threshold ranges from 0.0 to 0.99. Ideally, as threshold increases precision will increase at the expense of recall in a concave fashion.*

TABLE II: *Normalized area-under-curve (AUC) associated with precision-recall plots for overall dataset*

	Overall
Undefended clean	0.803
Defended clean	0.765
Certified robust	0.370

Weaknesses of this procedure include the fact that it removes the ability to analyze metrics at the individual class level.

IV. Results

A. Setup

We first outline the computational setup for the work. Experiments are done on the 2014 MSCOCO validation dataset [1], [5], [7]. For the multi-label classification model, we select an existing ResNet-based classifier created by [1]; this model has good precision and recall performance on the MSCOCO benchmark and is also attractive due to its well-maintained code repository online. The design of the architecture means that each MSCOCO image is resized to 448×448 pixels before being input into the model. We additionally use the default threshold value of 0.8 for inference unless stated otherwise.

With regards to the inference procedure Algorithm 1, note that the $\mathcal{R}$-covering mask set generation method discussed in [9] has two security parameters. The first is the number of masks desired in each axis $k_1 \times k_2$ and the second is the estimated size of the patch p in pixels [9]. For our experiments, we leverage 6×6 masks and a patch size of 64 (i.e., corresponds to about $\sim 2\%$ of pixels in the image overall); this is the default setting used by [9].

B. Multi-Label PatchCleanser performance

In this section we evaluate the robustness and clean performance of Multi-Label PatchCleanser. We consider the following three model settings:

- *Undefended clean:* This model setting represents performance of the multi-label classifier from [1] when no patch attack is present and Multi-Label PatchCleanser is not utilized.
- *Defended clean:* This model setting represents performance when no patch attack is present and the Multi-Label PatchCleanser defense is activated.
- *Certified robust:* This model setting considers certified precision and recall values determined according to Algorithm 2 (i.e., the original certification method).

In both clean settings, all precision and recall metrics are computed empirically. For the certified robust model setting, *overall precision* and *overall recall* metrics are computed according to the methodology discussed at the end of Section III-B; *class precision* and *class recall* metrics are computed by accumulating certified metrics separately across each of the classes and then taking the average.

We outline results with the default threshold value of 0.8 in Table I. Additionally, we perform the classification task across a range of threshold values from 0.0 to 0.99 to check how both precision and recall metrics change. Figure 3 visualizes this via a precision-recall plot while Table II presents the associated area-under-curve (AUC) values. AUC provides a single quantitative metric to aggregate performance across a wide variety of thresholds. Each AUC value is computed using the trapezoid sum technique and is normalized with respect to the ideal precision-recall curve.

Non-trivial robustness. The results from Figure 3 and Table II demonstrate that Multi-Label Patchcleanser is able to successfully achieve non-trivial certifiable robustness across a variety of threshold values. However, robust performance still lags behind both the undefended clean and defended clean settings. Indeed, the AUC associated with the certified robust setting is only able to capture $\sim 46\%$ of the performance achieved by the undefended clean setting for the overall dataset. Note that as shown in Table I the class precision and class recall metrics are appreciably lower than their overall counterparts in the certified robust setting. Thus, one possibility is that certain classes are responsible for hampering overall robustness; this is investigated further in Appendix D.

High clean performance. As seen in Table II, the AUC associated with the defended clean setting is able to capture $\sim 95\%$ of the performance achieved by the undefended clean setting for the overall dataset. This implies that the inference procedure of Multi-Label PatchCleanser can demonstrate high clean performance across a wide range of different threshold values. Further inspection of single threshold results in Table I reveals that the defended clean setting demonstrates a tangible ($\sim$ 4%) increase in precision metrics and a simultaneous large decrease ($\sim$ 15%) in recall metrics compared to the undefended clean setting. A hypothesis for this tradeoff is that in the undefended clean setting small unrelated objects might be confused with each other leading to false positives (i.e., lower precision). On the other hand, in the defended clean setting masks may occlude these objects and cause false negatives instead (i.e., lower recall). More experimentation is needed to test the validity of this claim.

C. *New certification method results*

TABLE III: *Comparison of different certification procedures*

	Overall precision	Overall recall
Old certification procedure	47.36%	38.55%
New certification procedure (worst case)	49.85%	40.02%
New certification procedure (FN attack)	51.54%	40.02%
New certification procedure (FP attack)	51.38%	42.56%

We now present results associated with the new certification procedure from Section III-C in Table III. For the new procedure, two attackers are analyzed. The FN attacker picks an optimal mask according to a certification histogram of only false negatives, while the FP attacker only leverages a histogram of false positives. We also consider "worst case" performance where two optimal masks are chosen; one for FN and one for FP. This represents the lower bound on the performance of Algorithm 5 for an arbitrary attacker.

In general, we note that the new certification procedure across all attackers is able to provide better guarantees on robustness compared to the old certification method. However, the improvement in robustness is limited to only about $\sim 4\%$ for both precision and recall. This suggests that updating the inference algorithm might be required to get stronger robustness guarantees.

V. Related Work

A variety of certifiable defenses against the patch threat model have been proposed for single-label classifiers. These include bound propagation methods [2], small receptive field methods [10], and masking methods [9]. For other computer vision domains, a masking method has been proposed for the object detection task [11]. Additionally, [3] proposed a certifiably robust defense for multi-label classifiers in the ℓ_2-norm global perturbation threat model. However, none of these works address certifiable defenses against patch attacks in the multi-label classification domain.

VI. Conclusions and Future Work

In this work, we investigated certifiable defenses against the patch attack for computer vision systems and recognized the lack of an existing defense for the multi-label classification domain. We thus proposed Multi-Label PatchCleanser, an extension of the SOTA PatchCleanser defense for single-label classification [9]. Our technique is able to achieve non-trivial robustness while also maintaining relatively high clean performance on the 2014 MSCOCO validation dataset. Additionally, we develop a novel certification procedure based upon a key constraint from the multi-label classification domain in order to improve bounds on robustness.

Nevertheless, there are possibilities for improvement. We first note that the novel certification method proposed in Section III-C has room for improvement. Future work will continue to leverage the observations from Section III-C and possibly develop a new inference algorithm. Another opportunity involves the use of vision transformers (ViT). Currently, we use a ResNet-based approach for all experiments; however, as shown in the original PatchCleanser paper using ViTs instead can lead to a performance boost [9].

References

[1] E. Ben-Baruch, T. Ridnik, N. Zamir, A. Noy, I. Friedman, M. Protter, and L. Zelnik-Manor, "Asymmetric Loss For Multi-Label Classification." arXiv, Jul. 2021, arXiv:2009.14119 [cs]. [Online]. Available: http://arxiv.org/abs/2009.14119

[2] P.-Y. Chiang, R. Ni, A. Abdelkader, C. Zhu, C. Studer, and T. Goldstein, "Certified Defenses for Adversarial Patches." arXiv, Sep. 2020, arXiv:2003.06693 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2003.06693

[3] J. Jia, W. Qu, and N. Z. Gong, "MultiGuard: Provably Robust Multi-label Classification against Adversarial Examples." arXiv, Oct. 2022, arXiv:2210.01111 [cs]. [Online]. Available: http://arxiv.org/abs/2210.01111

[4] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999–7019, Dec. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9451544/

[5] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft COCO: Common Objects in Context." arXiv, Feb. 2015, arXiv:1405.0312 [cs]. [Online]. Available: http://arxiv.org/abs/1405.0312

[6] F. Nesti, G. Rossolini, S. Nair, A. Biondi, and G. Buttazzo, "Evaluating the Robustness of Semantic Segmentation for Autonomous Driving against Real-World Adversarial Patch Attacks," Aug. 2021, arXiv:2108.06179 [cs]. [Online]. Available: http://arxiv.org/abs/2108.06179

[7] T. Ridnik, G. Sharir, A. Ben-Cohen, E. Ben-Baruch, and A. Noy, "ML-Decoder: Scalable and Versatile Classification Head," in *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Waikoloa, HI, USA: IEEE, Jan. 2023, pp. 32–41. [Online]. Available: https://ieeexplore.ieee.org/document/10030822/

[8] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," Feb. 2014, arXiv:1312.6199 [cs]. [Online]. Available: http://arxiv.org/abs/1312.6199

[9] C. Xiang, S. Mahloujifar, and P. Mittal, "PatchCleanser: Certifiably Robust Defense against Adversarial Patches for Any Image Classifier." arXiv, Apr. 2022, arXiv:2108.09135 [cs]. [Online]. Available: http://arxiv.org/abs/2108.09135

[10] C. Xiang and P. Mittal, "PatchGuard++: Efficient Provable Attack Detection against Adversarial Patches." arXiv, Apr. 2021, arXiv:2104.12609 [cs]. [Online]. Available: http://arxiv.org/abs/2104.12609

[11] C. Xiang, A. Valtchanov, S. Mahloujifar, and P. Mittal, "ObjectSeeker: Certifiably Robust Object Detection against Patch Hiding Attacks via Patch-agnostic Masking." arXiv, Dec. 2022, arXiv:2202.01811 [cs]. [Online]. Available: http://arxiv.org/abs/2202.01811

Appendix A
PatchCleanser double-masking method

Algorithm 3 *The inference algorithm associated with PatchCleanser from [9]*

```
    Input: Image $x \in \mathbb{R}^d$, single-label classifier $\mathbb{F}_s : \mathbb{R}^d \to \{1, 2, \ldots, c\}$, mask set $\mathcal{M}$
    Output: Prediction $\hat{y} \in \{1, 2, \ldots, c\}$
 1: procedure DoubleMasking($x, \mathbb{F}_s, \mathcal{M}$)
 2:     $\hat{y}_{maj}, \mathcal{P}_{dis} \leftarrow$ MaskPred($x, \mathbb{F}_s, \mathcal{M}$)        ▷ First-round
 3:     if $\mathcal{P}_{dis} = \emptyset$ then
 4:         return $\hat{y}_{maj}$                 ▷ Case I: agreed prediction
 5:     end if
 6:     for each $(m_{dis}, \hat{y}_{dis}) \in \mathcal{P}_{dis}$ do        ▷ Second round
 7:         $\hat{y}', \mathcal{P}' \leftarrow$ MaskPred($x \circ m_{dis}, \mathbb{F}_s, \mathcal{M}$)
 8:         if $\mathcal{P}' = \emptyset$ then
 9:             return $\hat{y}_{dis}$             ▷ Case II: disagreer pred.
10:         end if
11:     end for
12:     return $\hat{y}_{maj}$                 ▷ Case III: majority prediction
13: end procedure

    Input: Image $x \in \mathbb{R}^d$, single-label classifier $\mathbb{F}_s : \mathbb{R}^d \to \{1, 2, \ldots, c\}$, mask set $\mathcal{M}$
    Output: Majority prediction $\hat{y}_{maj} \in \{1, 2, \ldots, c\}$, disagreer masks $\mathcal{P}_{dis}$
14: procedure MaskPred($x, \mathbb{F}_s, \mathcal{M}$)
15:     $\mathcal{P} \leftarrow \emptyset$                  ▷ A set for mask-prediction pairs
16:     for $m \in \mathcal{M}$ do                ▷ Enumerate every mask $m$
17:         $\hat{y} \leftarrow \mathbb{F}_s(x \circ m)$         ▷ Evaluate masked prediction
18:         $\mathcal{P} \leftarrow \mathcal{P} \bigcup \{(m, \hat{y})\}$              ▷ Update set $\mathcal{P}$
19:     end for
20:     $\hat{y}_{maj} \leftarrow \mathrm{argmax}_{y^*} |\{(m, \hat{y}) \in \mathcal{P} | \hat{y} = y^*\}|$   ▷ Majority
21:     $\mathcal{P}_{dis} \leftarrow \{(m, \hat{y}) \in \mathcal{P} | \hat{y} \neq \hat{y}_{maj}\}$         ▷ Disagreers
22:     return $\hat{y}_{maj}, \mathcal{P}_{dis}$
23: end procedure
```

We present the double-masking method from PatchCleanser in Algorithm 3, which is able to achieve certified robustness for single-label classifiers [9]. It runs up to two rounds of masking on the input image x. In each round, masks m from the $\mathcal{R}$-covering set $\mathcal{M}$ are augmented onto x and then the single-label classifier $\mathbb{F}_s$ is queried for each of these augmentations [9].

- *First-round masking:* Here, every mask $m \in \mathcal{M}$ is applied sequentially to the input image x. We then run $\mathbb{F}_s(m \circ x)$ for each of these single masks on line 2 [9]. If there is consensus in the predictions, we return this value as the overall prediction on line 4; this aligns with the intuition that a clean image with no patch will be predicted correctly regardless of the mask present. Otherwise, the minority/"disagreer" predictions trigger a second-round of masking on line 6. This is done to determine whether to trust the majority prediction $\hat{y}_{maj}$ or one of the disagreers [9].
- *Second-round masking:* For each disagreer mask m_{dis}, we once again apply every mask $m \in \mathcal{M}$ sequentially to the input image x. This time however, we query $\mathbb{F}_s(m \circ m_{dis} \circ x)$ for each of the masks m to form double-mask predictions [9]. If all of these predictions are in consensus, then we return the disagreer label $\hat{y}_{dis}$ associated with m_{dis} as the overall prediction on line 9. The intuition is that if m_{dis} covered the patch, all double-mask predictions in the second round would be done on effectively a "clean" image [9]. If there is lack of consensus, then we disregard m_{dis} and apply second-round masking to the next available disagreer mask; in this case, we assumed that m_{dis} did not cover the patch [9]. Finally, if all disagreer masks fail second-round masking we return the majority label $\hat{y}_{maj}$ from the first-round.

Assuming that the mask set $\mathcal{M}$ is $\mathcal{R}$-covering and that two-mask correctness (i.e., Definition 2 from [9]) holds for the datapoint (x, y), then the correctness of Algorithm 3 is guaranteed by Theorem 1 in [9].

Appendix B
Computation of robust metrics

Algorithm 4 *Computation of robust metrics associated with an image datapoint (x, y)*

```
    Input: Ground-truth $y \in \{0, 1\}^c$, certification class list $\ell \in \{True, False\}^c$, number of classes $c$
    Output: Number of classes certified $k$, certified number of true positives $TP$, worst case number of false positives $FP$, worst case number of false negatives $FN$
 1: procedure MetricComputation($y, \ell, c$)
 2:     $k, TP, FP, FN \leftarrow 0, 0, 0, 0$   ▷ Initialize return values
 3:     for $i \leftarrow 1$ to $c$ do              ▷ Compute return values
 4:         if $\ell[i] = True$ then
 5:             $k \leftarrow k + 1$
 6:             if $y[i] = 1$ then
 7:                 $TP \leftarrow TP + 1$
 8:             end if
 9:         else      ▷ If class not certified, assume worst case
10:             if $y[i] = 1$ then
11:                 $FN \leftarrow FN + 1$
12:             else
13:                 $FP \leftarrow FP + 1$
14:             end if
15:         end if
16:     end for
17:     return $k, TP, FP, FN$
18: end procedure
```

Algorithm 4 uses the result of the certification procedure Algorithm 2 in order to compute the lower bound for true positives and the upper bound for both false positives and false negatives. To see this, note that as shown on line 4 we only count a successful TP if we were able to certify the associated class. Otherwise, as shown in line 9 we assume the worst case and assign a FN or FP to the class.

Appendix C
Histogram-based certification

In this section, we discuss the histogram-based certification method introduced in Section III-C in more detail. We first provide a formal definition for a *certification histogram* as follows.

Algorithm 5 *The novel certification procedure for FN*

```
    Input: Image $x \in \mathbb{R}^d$, ground-truth $y \in \{0,1\}^c$, certification class list $\ell \in \{True, False\}^c$, multi-label classifier $\mathbb{F} : \mathbb{R}^d \rightarrow \{0,1\}^c$, mask set $\mathcal{M}$, number of classes $c$, certified number of true positives $TP$, worst case number of false negatives $FN$
    Output: New certified number of true positives $TP_{new}$, new worst case number of false negatives $FN_{new}$
 1: procedure ResRobust($x, y, \ell, \mathbb{F}, \mathcal{M}, c, TP, FN$)
 2:     $fnIndices \leftarrow \{0\}^{FN}$            ▷ Initialize $FN$ index array
 3:     $i \leftarrow 1$
 4:     for $j \leftarrow 1$ to $c$ do                 ▷ Find $FN$ indices
 5:         if $\ell[j] = False$ and $y[j] = 1$ then
 6:             $fnIndices[i] \leftarrow j$
 7:             $i \leftarrow i + 1$
 8:         end if
 9:     end for
10:     $fnHistograms \leftarrow [0]^{FN \times |\mathcal{M}|}$       ▷ Track each $FN$
11:     for every $(m_0, m_1) \in \mathcal{M} \times \mathcal{M}$ do
12:         $out \leftarrow \mathbb{F}(x \circ m_0 \circ m_1)$      ▷ Check double-masks
13:         for $k \leftarrow 1$ to $FN$ do
14:             $idx \leftarrow fnIndices[k]$
15:             if $out[idx] = 0$ then           ▷ Mark both masks
16:                 $fnHistograms[k][m_0] = 1$
17:                 $fnHistograms[k][m_1] = 1$
18:             end if
19:         end for
20:     end for
21:     $fnTotal \leftarrow \text{sum}(fnHistograms, \text{dim} = 0)$
22:     $FN_{new} = \max(fnTotal)$ ▷ Pick worst possible mask
23:     $TP_{new} = TP + (FN - FN_{new})$
24:     return $TP_{new}, FN_{new}$
25: end procedure
```

Definition 3 (Certification histogram)**.** *Suppose we have the set of class labels $\mathcal{L} = \{1, 2, \dots, c\}$ and a clean image data point (x, y) with $x \in \mathbb{R}^d$ and $y \in \{0,1\}^c$. Additionally assume that we have a multi-label classification model $\mathbb{F} : \mathbb{R}^d \rightarrow \{0,1\}^c$ and a $\mathcal{R}$-covering mask set $\mathcal{M}$. Then, for a class $i \in \mathcal{L}$ we define the certification histogram $\mathcal{H}_i \in \{0,1\}^{|\mathcal{M}|}$ as follows:*

$$\mathcal{H}_i(m) = \begin{cases} 1 & \exists (m, m') \in \mathcal{M} \times \mathcal{M} \; s.t. \; \mathcal{A} = \textit{True} \\ 0 & otherwise \end{cases} \tag{8}$$

Here, $\mathcal{A} := \mathbb{F}(x \circ m \circ m')[i] \neq y[i]$ is a Boolean expression.

Note that in order to correctly update certified performance metrics we must create separate histograms for the false negatives and the false positives in an image. Without loss of generality, we consider false negatives only in Algorithm 5.

Algorithm 5 works by leveraging the certification class list from Algorithm 2 and the robust metrics from Algorithm 4. Specifically, it first uses the certification class list to determine which true positive classes were not certified via Algorithm 2 on line 6. Then, it constructs a certification histogram for each of these false negatives on line 10. During the for loop on line 11, we consider every possible double-mask pair. If a pair $(m_0, m_1) \in \mathcal{M} \times \mathcal{M}$ causes mis-classification of a class i, then we note as per Equation 8 that both masks should be marked in the corresponding histogram; this is done in lines 16 and 17. Finally, we compute the total histogram on line 21 and pick the mask with the largest value; this will correspond to maximum number of false negatives an attacker can induce at inference time. We correspondingly alter the lower bound for true positives on line 23.

Note that constructing histograms for false positives can be done using effectively the same procedure, albeit with the for loop on line 4 changed to track FP indices and the condition on line 15 changed to $out[idx] = 1$ instead.

We now demonstrate the superiority of the bounds from Algorithm 5 in comparison to Algorithm 4.

Lemma 1 (Histogram-based certification bound tightness)**.** *Given that we have derived a bound on FN using the technique from Algorithm 4, we note that Algorithm 5 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 21 and 22 of Algorithm 5 that the worst-case sum will occur if every certification histogram is marked for a given mask. Because the summation is taken over the set of false negatives, this implies that the worst-case sum is FN. ■

We next prove the correctness of the bounds from Algorithm 5.

Theorem 2 (Histogram-based certification bound correctness)**.** *Suppose we have the set of class labels $\mathcal{L} = \{1, 2, \dots, c\}$ and a clean image data point (x, y) with $x \in \mathbb{R}^d$ and $y \in \{0,1\}^c$. Additionally assume that we have a multi-label classification model $\mathbb{F} : \mathbb{R}^d \rightarrow \{0,1\}^c$ and a $\mathcal{R}$-covering mask set $\mathcal{M}$. Then, given that we have derived a bound on FN using the technique from Algorithm 4 we note that the bound FN_{new} returned by Algorithm 5 will be correct.*

Proof: We will show correctness of the new bound FN_{new} defined on line 22 of Algorithm 5. First, we define $m_{opt} := argmax(fnTotal) \in \mathcal{M}$ as the corresponding mask location where the adversarial patch will placed in order to induce the maximum number of false negatives. Next, as per Lemma 1 we will have $FN_{new} \leq FN$. Note that with equality $FN_{new} = FN$, correctness is guaranteed by Algorithm 4. We thus focus on the case with strict inequality $FN_{new} < FN$; this implies the existence of false negatives with unmarked certification histograms. Consider an arbitrary false negative class $i \in \mathcal{L}$ such that $\mathcal{H}_i(m_{opt}) = 0$. We will show that this class i will be correctly predicted if the attacker places a patch at the m_{opt} location.

During inference (i.e., Algorithm 1), we note that the mask $m_{opt} \in \mathcal{M}$ will cover the adversarial patch during the first-round of double-masking. This forms the masked image $x \circ m_{opt} \circ m_{opt}$; by definition of a certification histogram $\mathcal{H}_i$, the evaluation $\mathbb{F}(x \circ m_{opt} \circ m_{opt}) = y[i]$ will be correct. We now show that the overall output of the double-masking procedure will be correct by considering three cases:

- *Case #1 (consensus):* In this case, every first-round mask results in the correct prediction. Specifically, we have $\mathbb{F}(x \circ m) = \mathbb{F}(x \circ m_{opt} \circ m_{opt}) = y[i]$ for all

$m \in \mathcal{M}$. Then, as per line 4 in Algorithm 3 we will return the correct prediction for class i.

- *Case #2 (first round majority):* In this case, a minority subset of masks $\mathcal{M}_{minority} \subseteq \mathcal{M}$ will have an incorrect first-round prediction. Specifically, we will have $\mathbb{F}(x \circ m') \neq y[i]$ for all $m' \in \mathcal{M}_{minority}$. Note that during the second-round of masking, we will correctly evaluate $\mathbb{F}(x \circ m' \circ m_{opt}) = y[i]$ for each disagreer mask $m' \in \mathcal{M}_{minority}$; this is due to the definition of a certification histogram $\mathcal{H}_i$. Therefore, each of the disagreer masks will fail to have consensus in the second-round and as per line 12 in Algorithm 3 we will return the correct prediction for class i.
- *Case #3 (first round minority):* In this case, a majority subset of masks $\mathcal{M}_{majority} \subseteq \mathcal{M}$ will have an incorrect first-round prediction. Specifically, we will have $\mathbb{F}(x \circ m') \neq y[i]$ for all $m' \in \mathcal{M}_{majority}$. Therefore, the mask m_{opt} will be a disagreer. During the second-round, we will correctly predict $\mathbb{F}(x \circ m_{opt} \circ m_{second\,round}) = y[i]$ for every second-round mask $m_{second\,round} \in \mathcal{M}$; this is due to the definition of a certification histogram $\mathcal{H}_i$. Thus we will have consensus in the second-round and as per line 9 in Algorithm 3 we will return the correct prediction for class i.

The correctness of the underlying double-masking method implies that Algorithm 1 will return the correct prediction for class i as desired. Thus, class i must now be counted as a true positive instead of a false negative. ■

Analogues for Lemma 1 and Theorem 2 also exist for bounds on FP, and can be proved the same way albeit referencing a modified version of Algorithm 5.

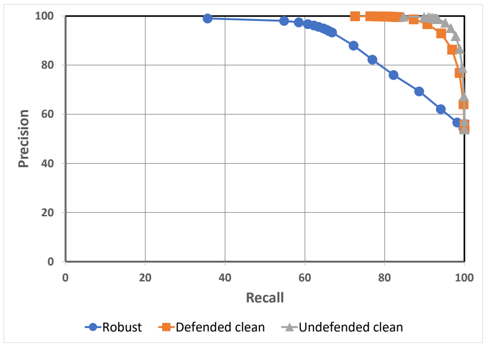
APPENDIX D
SELECTED CLASS PRECISION-RECALL CURVES

TABLE IV: *Normalized AUC for class precision-recall plots*

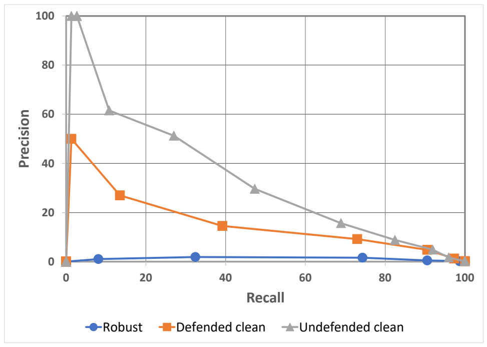
	Class 0/“person”	Class 70/“toaster”
Undefended clean	0.961	0.333
Defended clean	0.952	0.158
Certified robust	0.858	0.013

In this section we consider the precision-recall plots associated two sample classes from the 2014 MSCOCO validation dataset. Figure 4a plots precision-recall curves limited to Class 0/“person”. Figure 4b plots precision-recall curves for Class 70/“toaster”. Additionally, Table IV lists area-under-curve (AUC) values associated with each of the individual plots in Figure 4.

Some general trends are apparent through Table IV. For instance, the AUC associated with the defended clean setting tracks closely to the undefended clean setting for both of the classes; additionally, the AUC for certified robustness is lower in both cases. Nevertheless, as shown in Figure 4b the Class 70/“toaster” demonstrates an extreme example in which almost none of the undefended clean performance is able to be certified regardless of the threshold. This implies that Algorithm 2 is a bottleneck for certification performance in certain classes, contributing to the overall reduced robustness in Table II.



(a)



(b)

Fig. 4: *Precision-recall plots of the three model settings for* a) *Class* 0/*"person", and* b) *Class* 70/*"toaster". Threshold ranges from 0.0 to 0.99. Ideally, as threshold increases precision will increase at the expense of recall in a concave fashion.*