Understanding and Defending Patched-based Adversarial Attacks for Vision Transformer

Liang Liu [1] **Yanan Guo** [1] **Youtao Zhang** [2] **Jun Yang** [1]

Abstract

Vision Transformer (ViT) is an attention-based model architecture that has demonstrated superior performance on many computer vision tasks. However, its security properties, in particular, the robustness against adversarial attacks, are yet to be thoroughly studied. Recent works have shown that ViT is vulnerable to attention-based adversarial patch attacks, which covers 1∼3% area of the input image using adversarial patches and degrades the model accuracy to 0%.

This work generally studies the attention-based patch attack. First, we experimentally observe that adversarial patches only activate in a few layers and become lazy during attention updating. According to experiments, we study how a small adversarial patch perturbates the whole model. Based on understanding adversarial patch attacks, we propose a simple but efficient defense that correctly detects more than 95%.

1. Introduction

The recent research discovers that the attention-based transformer (Vaswani et al., 2017) achieves a remarkable outcome on computer vision, known as Vision Transformers (Dosovitskiy et al., 2020) and its variants, e.g., DeiT (Touvron et al., 2021b), etc. However, ViTs do not present strong robustness and are vulnerable to security attacks such as adversarial attacks (Szegedy et al., 2013). Adversarial attacks exploit the model gradient and construct an imperceptible background noise onto the input images that can fool the models into malfunctioning.

Early studies show that in defending background-noised-based adversarial attacks, ViTs are more robust than classic CNN networks with a comparable model size(Paul & Chen, 2022; Naseer et al., 2021; Gu et al., 2022). The later research further showed that by adopting adversarial enhancements such as Image Augmentation(Mao et al., 2022) and Adversarial Training(Herrmann et al., 2022), ViTs achieve even higher robustness than CNN. However, recent studies (Gu et al., 2022; Fu et al., 2022; Lovisotto et al., 2022) found an exception in that ViT can be crushed by attention-based adversarial patch attacks. Attackers inject the adversarial token to alter the attention of ViT, which perturbs only 1∼3% area of the input and degrades the model accuracy drastically, e.g., 0% (Lovisotto et al., 2022). In contrast, a 20 ∼ 30% of input area should be patched for the state-of-the-art attack in CNN to achieve a similar degradation (Brown et al., 2017; Wang et al., 2021).

The main strategy of existing defenses toward such attention-based attacks is adopting Derandomized Smoothing to ViT (Salman et al., 2022; Chen et al., 2022), which infers randomly sampled images 1,000 times and statistically votes the model outputs. Such defenses introduce 1,000 times computational overhead but only achieve 40 ∼ 50% adversarial robustness. Another defense deploys a mask in the attention block, filtering the largest element (Mu & Wagner, 2021) as they are likely related to adversarial inputs. Attention mask achieves ∼ 60% robustness but compromises the benign accuracy from ∼ 85% to ∼ 75%.

In this work, we first design two experiments to deeply understand why such a small-size patch can crash the entire ViT model. In the experiment, we remove the connection of attention blocks. We observe that about 40% of adversarial patches are neutralized by removing the attention block in only one layer, where there are a total of 12 layers. Moreover, we observe that layer-wise attention updates of adversarial tokens are about 50% smaller than benign tokens. Through observations, we uncover that the adversarial token only activates for a few layers. Further, we analyze the attention via the Key/Query product. We discover that the adversarial patch fabricates its key close to queries of noise features, elevates the column of its score matrix, and propagates its adversarial pattern to the benign tokens. According to the behavior of the adversarial token in different layers, we divide this process into three stages: the inactivate stage; the activate stage, where the score of

[1]Department of ECE, University of Pittsburgh [2]Department of CS, University of Pittsburghy. Correspondence to: Liang Liu <lil125@pitt.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

the adversarial token becomes the most salient; the polluted stage, where vast noise keys and queries gather closely and dominate the attention.

We propose AbnoRmality-Masking RObust (ARMRO) that precisely detects the position of adversarial patches in the activate stage and masks them from the input. We correctly detect 95% of adversarial patches and maintain above 80% robustness against one-patch attacks with only 1% clean accuracy drop. We achieve 20% $\sim$ 30% robustness gain compared to the prior defense. Further, we extend our detecting algorithm and enable the detection against multi-patch adversarial attacks, and we achieve around 85% correct detection >70% model robustness.

2. Background

2.1. Attention-Based Transformer

Vision Transformer, by exploiting the self-attention mechanism from the NLP models, e.g., BERT (Devlin et al., 2018), can outperform traditional convolutional neural networks (CNNs) for many image-processing tasks. Given an input image (also the input at the first layer), $x^{(0)}$, that has the $H \times W$ resolution and C channels, ViT divides the input image into a sequence of patches, each of which has the $P \times P$ resolution, and then flattens all patches into vectors. That is, the input image can be denoted in patch form: $x^{(0)} \in \mathbb{R}^{P \times P, N}$, where $N = \frac{H \times W \times C}{P^2}$ is the number of patches.

The first layer of ViT embeds the input patches into tokens, $x^{(1)}$, with a higher dimension d ($d > P^2$). ViT models often introduce additional tokens for better performance, such as class tokens or distillation tokens. A ViT model has L layers where the later layers keep the same dimension. We denote the tokens matrix as $x^{(l)} \in \mathbb{R}^{n \times d}, l \in \{1, 2, \cdots L\}$, where $n = N + 1$ is the number of tokens; d is the length of each token; and l is the layer number.

ViT projects the tokens into *query*: $Q^{(l)} = x^{(l)} W_Q^{(l)}$, *key*: $K^{(l)} = x^{(l)} W_K^{(l)}$, and *value*: $V^{(l)} = x^{(l)} W_V^{(l)}$, where $Q^{(l)}, K^{(l)}, V^{(l)} \in \mathbb{R}^{n,d}$. In practice, the projected matrices are further divided into H heads, which allows the model to gather the information from lower dimension subspace and benefit the model accuracy (Vaswani et al., 2017). These matrices are then denoted as, $Q^{(l,h)}, K^{(l,h)}, V^{(l,h)} \in \mathbb{R}^{n,d/h}$. Next, ViT computes the self-attention scores from the dot-product as follows.

$$S^{(l,h)} = \text{softmax}(\frac{Q^{(l,h)} K^{(l,h)T}}{\sqrt{d/h}}) \ , \ A^{(l,h)} = S^{(l,h)} V^{(l,h)}$$

The single-head scores, $S^{(l,h)} \in \mathbb{R}^{n,n}$, measure the correlation between every two tokens; The attention $A^{(l,h)} \in \mathbb{R}^{n,d/h}$ multiple the corresponding value to the scores. The tokens of the main features demonstrate a higher correlating score with other tokens, and their value contributes more to the output, whereas the noise features have a smaller score and contribute less. The single-head attention scores are then concatenated to form the multi-head attention, $A^{(l)} \in \mathbb{R}^{n,d} = \text{Concat}(A^{(l,1)}, A^{(l,2)}, \cdots, A^{(l,H)})$. Further, ViT adopts the skip connection between layers, which are $\hat{x}^{(l)} = x^{(l)} + A^{(l)}$ and $x^{(l+1)} = \hat{x}^{(l)} + \text{MLP}(\hat{x}^{(l)})$.

2.2. Adversarial Patch Attacks in ViT

Szegedy (Szegedy et al., 2013) pioneered the development of adversarial attack, where imperceptible perturbations to the inputs can significantly alter the model output. Many adversarial attacks and defenses were later studied in the literature. Among them, the adversarial patch attack (Brown et al., 2017), attaches an adversarial patch to the input image to comprise the model accuracy. While the patch only changes the pixels in a confined region, it may be placed freely in the image to yield the strongest attack. Recent studies showed that ViT is robust against background-noise-based adversarial attacks (Aldahdooh et al., 2021; Shao et al., 2021) but more vulnerable to adversarial patch attacks, compared with CNN. For example, Token-Attack (Joshi et al., 2021), and Patch-Perturbation (Gu et al., 2021) are patch-based attacks specially developed on ViT.

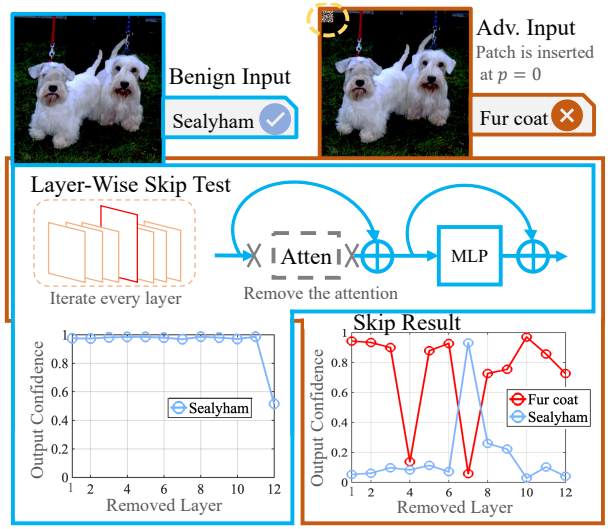
A strong adversarial patch consists of two critical attributes: the patch position and the patch pattern. The patch pattern can be computed by Projected Gradient Descent (PGD)(Madry et al., 2017), To maximize the loss of the model and alter the output, PGD repeats the gradient descent by multiple times to train the adversarial patch. We denote the adversarial patch as $x_p^{(0)}$, where subscript $p \in \{1, \cdots n\}$ denotes the position of the target patch. For the intermediate layers ($l > 0$), the adversarial token, $x_p^{(l)} \in \mathbb{R}^d$, is the p-th token over n tokens. Instead of identifying the position that yields the strongest attack, existing works exploit heuristic approaches to obtain good positions.

The state-of-the-art, Patch-Fool (Fu et al., 2022) and Give-Me-Your-Attention (GMYA) (Lovisotto et al., 2022) enhance patch attacks by redesigning the loss functions. Both schemes integrate the scores into the loss and maximize the sum of the scores between every two tokens. Both successfully degrade all ViT/DeiT models to 0% within five 16$\times$ 16-pixel patches.

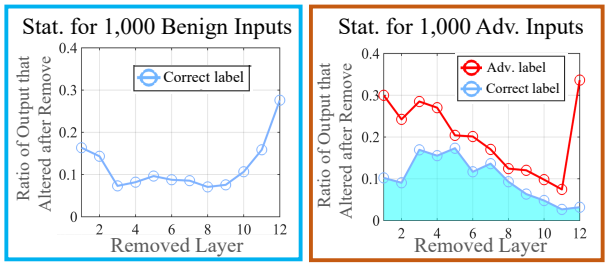
Details of how to generate the adversarial patch are in Appendix A.

3. Prior Art

To defend against patch-based adversarial attacks on ViT, there exist two types of strategies. One is to adopt a classic CNN-based adversarial patch defense, Derandomized-



(a) Attention-Block-Removing Test for The Sealyham Example



(b) Attention-Block-Removing Test for 1,000 Images

Figure 1: **Attention-Block Removing Test.** The results of benign inputs are plotted in blue boxes, and those for adversarial inputs are in red boxes. In (a), the curve charts depict the output of ViT-B/32 after removing the attention block in each layer. In (b), the curve charts depict the ratio that model output is altered after the corresponding attention block is removed.

Smoothing (DS) (Levine & Feizi, 2020), into ViT, which is certifiably proven effective in CNN. DS first generates a large amount of structurally-ablated images by randomly removing 60% to 70% of the content from the original image. Then, it inferences all the ablated images to vote for the majority to get the classification result. Certifiable-Patch-Defense (Chen et al., 2022) and Smooth-ViT(Salman et al., 2022) achieved 30% to 40% robustness with a trade-off that it introduces around 1,000 times of computational overhead. Details are in Appendix B.

The other strategy is attention-Mask (Mu & Wagner, 2021), which integrates an average mask at each attention block for each layer. They assume that the adversarial tokens perform suspiciously in the attention block of each layer. Attention-Mask cannot distinguish whether the adversarial exists, but it indiscriminately selects N_d (commonly $N_d = 5$) tokens from each layer to mask, and it expects all adversarial patches, if exists, to be removed. It first monitors the value vector for all layers and records the position of the tokens that are the top-N_d largest. Second, it masks the attention scores and value of all those tokens with the average. The authors evaluate their schemes at both CIFAR-10 and a down-sampled ImageNet, ImageNet-100, which shows that the robustness is improved from 45% to 84%.

Limitations of existing defenses: The main limitation of the Randomized-Smoothing-based defenses is the heavy computation, where they randomly sampled one input image and repeatably inferred the Network model about 1,000 times. Moreover, ViT is structurally different from CNN. The smoothing-based neutralizes the weak connections generated by adversarial patches, but in ViT, such connections are stronger. Adopting such protection only achieves about 40% robustness (Salman et al., 2022). The second defense, attention mask, remove all salient tokens without knowing which one is adversarial, and the robustness is around 65%. Moreover, the attention mask would falsely mask the benign tokens, which leads to about 5% to 10% accuracy drops (cf. Figure 5 and Appendix J).

4. Observations

None of the existing defenses correctly detect the position of the adversarial tokens. To precisely locate the adversarial tokens, this work starts with in-depth studies of the difference between adversarial and benign tokens.

In contrast to applying the defense on all layers non-discriminatorily, as in the Attention-Mask, we consider that adversarial tokens behave differently in different layers. In this work, we derive a layer-wise study of the behavior of adversarial tokens. We design two experiments: 1. bypassing the attention block for each layer and observing the change of the model output; 2. studying the update of adversarial tokens in the attention block and comparing it with benign tokens.

4.1. One-Attention-Block-Removing Test

The prior work(Raghu et al., 2021) uncovered an intriguing factor of ViT that removing the connection of one attention block only degrades the accuracy by 4%. It implies that most information is robustly distributed in multiple layers, and the attention block for non-removed layers contains enough information to maintain the correct output. In our work, we conduct this experiment for the adversarial inputs and verify their stability and robustness.

In Figure 1 (a), we capture a benign image, labeled as "Sealyham", from ImageNet. An adversarial patch placed at $p = 0$ (the upper left corner) successfully fools the model into producing incorrect output as "Fur coat". For these two images, we repeatedly infer the model 12 times for every 12 layers. Each time, we removed the connection of one

attention block for each layer and leave the connection of the other 11 blocks unchanged. The output of the benign image after removing always produces the correct label, Sealyham (blue curve), where the previous 11 layers do not significantly alter after removing, and the worst layer, the 12th layer, still maintains 50% confidence. However, for the adversarial input, the most output after removing still produces the incorrect label, fur coat (red curve). When removing the 4th and 7th layers, the output of fur-coat drops to nearly zero, and also we observe that removing the 7th layer can recover the correct label, Sealyham.

We continue this study for the other 1,000 images and record the altering rate of the 1-layer-removing model, and the results are plotted in Figure 1 (b). For benign inputs, only around 10% of model output is altered by removing 1-layer attention, except the last two layers. However, the removing outputs of adversarial inputs are less stable, and the altering rate of the front and middle layers increases to more than 0.2. Moreover, we observe that removing 1-layer attention can correctly recover the benign classification from the adversarially-patched inputs, shown by the blue curve at the right of Figure 1 (b). The recovering rate reaches nearly 0.2 at the middle layer.

4.2. The Update of Adversarial Patch

Through the removing test, we observe that adversarial patches are commonly activated at a few (or even one) attention blocks at the middle layers. The correct output can be recovered by removing the connection in these layers. Since the skip-connection of the adversarial token is formulated as, $\hat{x}_p^{(l)} = x_p^{(l)} + A_p^{(l)}$, if the attention update, $A_p^{(l)}$, is small, the adversarial tokens are not active in these layers.

We collect 1,000 pairs of adversarial and benign images and study the attention update of the adversarial tokens in each layer. At the top of Figure 2 (More results are in Appendix F), we first study the magnitude of the attention update by computing the ratio between the l_2 norm of $A_p^{(l)}$ and $x_p^{(l)}$. As is shown, the update ratio of the adversarial patch for each layer is around 0.1, and it is 44% smaller than that of the benign tokens. Such a ratio is close to zero (<0.01) for the first few layers. Moreover, to demonstrate the directional change, we further measure the cosine similarity between tokens in different layers and plot them at the bottom of Figure 2. We can also observe that the similarity of adversarial tokens is 47% higher than that of the benign tokens.

5. Understanding Adversarial Patches

The existing literature (Allen-Zhu & Li, 2022; Carlini et al., 2019) well explains the principle of adversarial attacks in CNN, which is manipulating the noise pattern of weights in Conv or FC layers and forging them into the inputs that

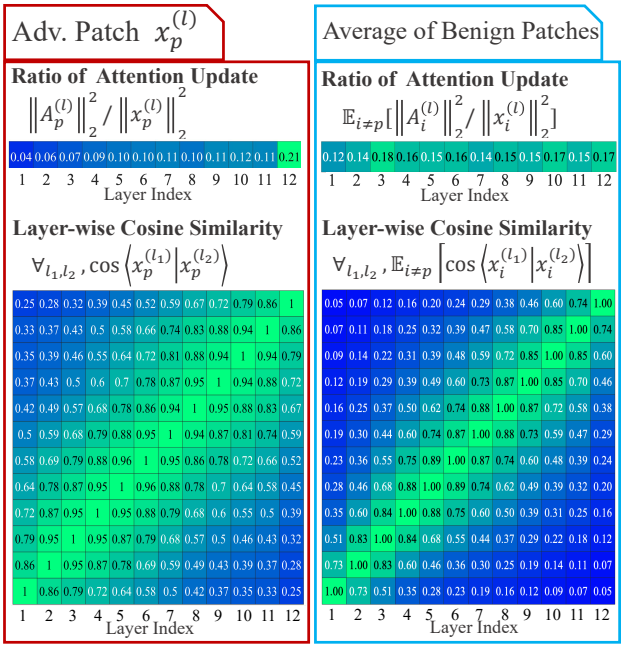


Figure 2: **Layer-Wise Update of Adversarial Token.** The upper two heatmaps depict the distant change of the tokens. The bottom two heatmaps depict the directional change of the same token between every two layers. Data is captured from the average of 1,000 images in ViT-B/32. Results for other models are in Appendix F.

dominate the model output. Similarly, in ViT, the previous work(Lovisotto et al., 2022; Fu et al., 2022) shows that adversarial patches or examples manipulate attention scores, but they did not provide a comprehensive analysis. Instead, they implement attacks by simply maximizing the entire attention score matrix. In this section, we provide a theoretical study of how adversarial patches affect the attention score. We show that a successful adversarial patch maximally amplifies the attention score of the noise tokens and lowers the attention score of main-feature tokens. (Main-feature tokens represent the major part of the target object, e.g., the body of Sealyham in Figure1.)

5.1. The Propagation of The Adversarial Pattern

The key part of understanding the adversarial patch is to convert the matrix multiplication into the following linear combination format:

$$A_i^{(l,h)} = \sum_{j=1}^{n} S_{i,j}^{(l,h)} \cdot V_j^{(l,h)}$$

where $S_{i,j}^{(l,h)}$ is the element in the i-th row and j-th column of the score matrix, and $V_j^{(l)}$ and $A_j^{(l)}$ are the jth row of the value matrix and attention matrix. We use $S_{i,:}^{(l,h)}$ to denote the i-th row in the score matrix and $S_{:,i}^{(l,h)}$ to denote the i-th column. During the attention update, the p-th col-

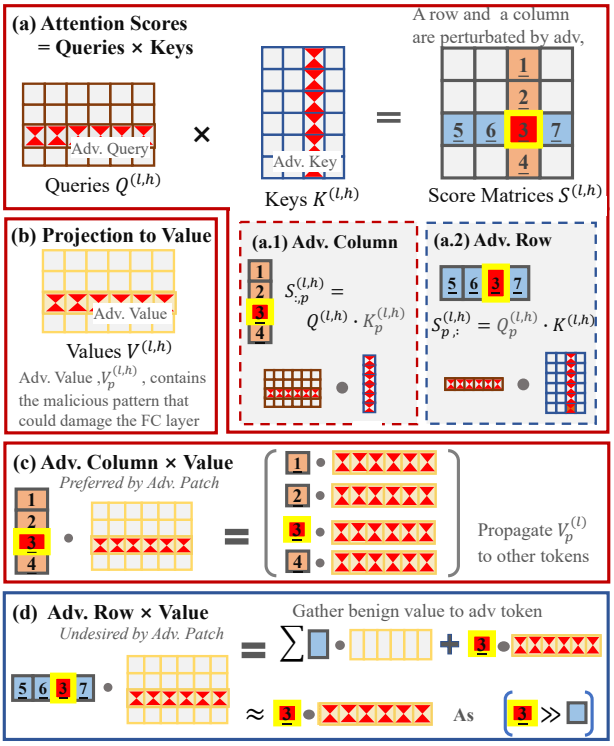


Figure 3: **The propagation of adversarial patterns.** This figure depicts all the computations in the attention block with the highlights of adversarially perturbed elements.

umn, $S_{:,p}^{(l,h)}$, and the p-th row, $S_{p,:}^{(l,h)}$, are perturbed by the adversarial token.

5.2. How to maximally perturb the attention?

The column score of the adversarial token plays an essential role in perturbing attention. The attention output of a benign token i is affected by the adversarial patch by a scalar product, $A_i^{(l)} = S_{i,p}^{(l)} \cdot V_p^{(l)}$. As shown in Figure 3 (c), if the adversarial token successfully elevates the scores in block 1, 2, and 4, it can propagate the adversarial pattern from its value vector to other vectors replace the benign patterns with the adversarial pattern. To maximize this propagation, the adversarial token needs to elevate more scores at p-th column or a with higher value. The column score is computed by the matrix-vector product of the query matrix and key vector. Thus, moving the adversarial keys close to the center of the query matrix, e.g., $K_p^{(l,h)} \leftarrow \mathbb{E}_i[Q_i^{(l,h)}]$, is preferred by adversarial patch. Further, the propagated adversarial pattern is negative to the Fully-Connected (FC) layers, so the more tokens are polluted, the higher possibility that the model being crashed. An optimized adversarial token, $x_p^{*,(l)}$, should reach a balance of both maximizing the score column and containing the most adversarial value vector.

5.3. The Origin of The Adversarial Patch

To perturb the score matrix in a different layer, the local optimum, $x_p^{*,(l)}$, for each layer, are also different since the projection weights, $W_{Q,K,V}^{(l)}$, are different. During the early step of PGD training, the adversarial patches are trapped by a local optimum in a particular layer, i.e., $x_p^{*,(l_*)}$ (Details can be found in Appendix C). As the training continues, the gradient from these layers significantly increases since the adversarial patch approaches the optimum of these layers. During the late steps, the adversarial patch is over-fitted into the local maximum of a few layers without entering the global maximum of all layers. After the adversarial token is trapped in one layer, it will lower its row score in other layers. Figure 3 (d) plots the multiplication between the adversarial row and value matrix. The product, $A_p^{(l)}$, is the attention output of the adversarial token, and it gathers the benign values based on the row score. As benign patterns are added to the adversarial token, the adversarial token will be neutralized by benign patterns if the row score is too large. Therefore, the PGD method moves the query vector of the adversarial token away from other keys and then minimizes the row score. Hence, the update of the adversarial token during each layer will be small, $x_p^{(l+1)} \rightarrow x_p^{(l)}$, which explains the observation in Figure 2.

5.4. Three Stages to Crash The Model

With the analysis above, we can illustrate how adversarial patches change inference results layer by layer, which is shown in Figure 4. We use PCA to project the high-dimensional queries and keys into a 2D plane. We divide the inference with an adversarial input into three stages. (1) Inactive stage, where the adversarial patch does not fall into the local optimum of these layers, and the adversarial patch is inactivated. (2) Activated stage, where the adversarial token reaches the local optimum of these layers and becomes the most salient one. For example, in Layer 7, the key of the adversarial token is shifted into the region of queries. In the activated stage, the score column of the adversarial token will be highly elevated. The sorted column score is in the activate stage of Figure 4. The red curve is the adversarial column, which is significantly larger than other columns. (3) Polluted stage, where more and more noise tokens are triggered and acquire the adversarial pattern, and the adversarial pattern is further broadcast which makes the entire attention graph adversarial. In this stage, the score matrices become chaotic, and the most salient column might be from either the adversarial tokens or the polluted tokens.

6. Abnormality-Masking Robust Defense

We propose an abnormality-masking robust (ARMOR) defense, a detect and mask scheme that effectively removes

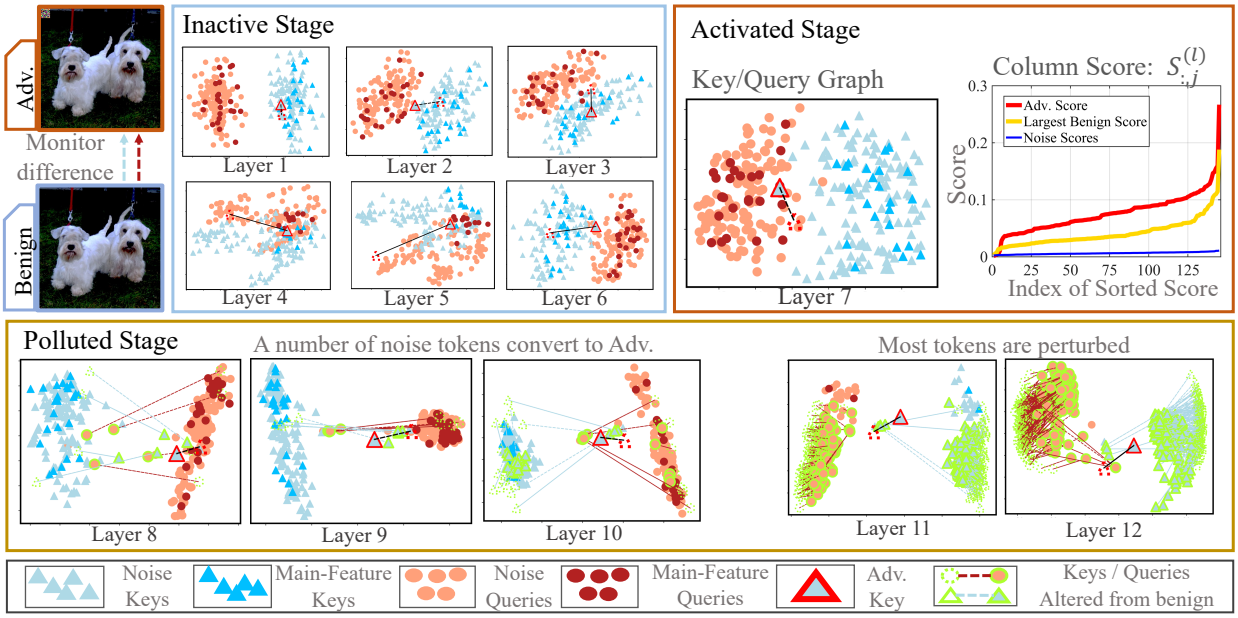


Figure 4: **Three Stages of How Adversarial Patch Crashes the Model.** Data is captured from both benign and adv. of the Sealyham image in the ViT-B/32 model. Each scatter graph plots 145 keys (blue triangles) and 145 queries (red circles). We use green lines to circle the adv. keys and queries whose positions are 10% different from their benign examples and dotted lines to link them. We highlight 23 **main-feature** keys and queries from the patches that contain the main body Sealyham, and the rest patches, e.g., the background, are noted as **noise** keys and queries. We capture three columns in the score matrix from the adv. column, the largest benign column, and a random noise column, each containing 125 scores. We sort and plot them in the curve chart. More key/query graphs are in Appendix K.

Algorithm 1 ARMOR

Input: $N_d \leftarrow$ The number of tokens to detect
Input: $\tau \leftarrow$ Threshold to identify whether adversarial
Input: $x^{(0)}$, $f(\cdot) \leftarrow$ The input and the model
Input: $p_{\text{adv}} \leftarrow \emptyset$
for $l = 1$ **to** L **do**
  $\overline{S}_i^{(l)} \leftarrow 1/n \cdot \sum_j \sum_h S_{i,j}^{(l,h)}$ ▷ Get the mean column score
  $\{s_1, \cdots, s_{N_d}\}, \{p_1, \cdots, p_{N_d}\} \leftarrow top\text{-}N_d(\overline{S}_i^{(l)})$
  ▷ Record the positions of the top N_d largest scores
  for $n = 1$ **to** $N_d - 1$ **do**
    if $s_n > \tau \cdot s_{n+1}$ **then**
      $\mathbf{p}_{\text{adv}} \leftarrow \mathbf{p}_{\text{adv}} \cup p_n$ ▷ Mark the suspicious token
    end if
    $x_{\text{clean}}^{(0)} \leftarrow \{x^{(0)} \setminus x_{\mathbf{p}_{\text{adv}}}^{(0)}\} \cup \{x_{\mathbf{p}_{\text{adv}}}^{(0)} \leftarrow \mathbb{E}[x^{(0)}]\}$
    ▷ Averagely mask suspicious patches
  end for
end for
Output: $y_{\text{clean}} = f(x_{\text{clean}}^{(0)})$ =0

the effect of adversarial patches. This scheme is based on the two major weaknesses we found in adversarial patches. First, in the first few layers, the adversarial tokens are in the inactive stage (cf. Figure 4). Second, when adversarial tokens enter the activated stage, their column scores, $S_{:,p}^{(l)}$, are inevitably elevated.

In detecting multi-patch adversarial attacks, the first challenge is distinguishing between main-feature tokens and adversarial tokens. For main-feature tokens, typically, tens of them have a similar attention pattern, so their column scores should be consistent. For adversarial tokens, since the number of adversarial patches is smaller, they must elevate their column score to a higher level to draw enough attention and inhibit the attention from benign tokens. Therefore, if only a small amount of tokens obtain much higher column scores than others, we can consider them adversarial. Another challenge is distinguishing polluted tokens from adversarial tokens. In the polluted stage, a number of noise tokens are polluted, and the column scores of polluted tokens approach or even exceed the adversarial tokens. They are much more polluted tokens than adversarial tokens, e.g., Layer 9 and Layer 10 in Figure 4, and there exists no clear gap between each polluted token. Although the column scores of polluted tokens are higher than that of main-feature tokens, we should not consider them as adversarial and remove them since doing so will damage the clean accuracy.

Since adversarial tokens are only activated in a few layers, we conduct layer-wise scanning that efficiently catches the adversarial tokens in the activated stage. If adversarial tokens are detected, a mask using the average of the image can efficiently remove the threat without damaging the model, which has been verified in prior literature (Naseer et al., 2021; Paul & Chen, 2022).

We adopt the mask in the images and perform the inference

Table 1: Robustness Result of ARMRO Against Various Attacks on Different Models

Models	AR-MRO	Clean Acc.	Token Attack			Patch Fool			GMYA		
			1-patch	2-patch	5-patch	1-patch	2-patch	5-patch	1-patch	2-patch	5-patch
ViT-B/16	w/o	85.0%	47.1%	12.2%	0%	33.5%	8.4%	0%	30.2%	2.5%	0%
	with	84.1%	83.2%	80.0%	72.5%	82.8%	80.2%	74.0%	82.4%	81.7%	75.0%
ViT-B/16 (224)	w/o	84.4%	31.6%	0.7%	0%	17.9%	0%	0%	12.2%	0%	0%
	with	83.0%	82.4%	79.7%	70.1%	82.3%	78.8%	72.3%	81.5%	79.8%	67.5%
ViT-B/32	w/o	81.6%	12.7%	0%	0%	7.1%	0%	0%	9.9%	0%	0%
	with	80.7%	79.6%	76.6%	67.2%	79.3%	77.1%	69.1%	78.5%	77.4%	70.7 %
ViT-L/32	w/o	83.4%	28.1%	5.8%	0%	12.7%	0.6%	0%	19.4%	6.3%	0%
	with	81.9%	80.8%	79.4%	73.1%	80.3%	76.5%	71.9%	81.0%	78.4%	70.9%
DeiT-B/16	w/o	85.2%	57.7%	28.4%	3.5%	41.7%	17.3%	0%	43.0%	14.6%	0%
	with	84.5%	84.1%	82.4%	75.3%	83.5%	80.6%	73.6%	83.6%	79.3%	74.3%

again, which makes all layers clean. Algorithm 1 depicts the detail of ARMOR. First, we scan the mean value of each column in score matrices for all layers. Second, we look for the top-N_d largest mean column scores, where N_d is a preset coefficient stating the number of tokens needed to detect. Empirically, we set $N_d = 5$, because we consider that the number of adversarial patches should be clearly smaller than the main-feature patches, and setting a larger detecting number, e.g., $N_d = 10$, increases the possibility of false positives. If encountering more than N_d adversarial patches, we can perform multiple rounds of ARMOR. Third, we detect the abnormally large elements within the top-N_d scores. We set a threshold, τ; if one or a few elements are τ times larger than others, we mark it (or them) as suspicious. For each model, we use 100 images to find an optimal τ, and usually, setting τ is set between 1.5 to 2.5 for most models (see Section D). Finally, we record all the suspicious patches and adopt a mask with the average of the image at the image input (Layer 0), $\mathbb{E}[x^{(0)}]$, and perform the inference again to acquire the clean output.

Note that we assume the area (equivalent to the number) of adversarial patches is limited, as pointed out in prior studies (Brown et al., 2017; Liu et al., 2018), or it would be easily observed by human eyes.

7. Evaluation

7.1. Experiment Setting

To evaluate the variety of our defense, we experiment with five models: ViT-B/16, ViT-B/16-224, ViT-B/32, ViT-L/32, and DeiT-T/16. The testing models contain different patch sizes (16 and 32), different numbers of layers (Base: 12, and Large: 24), different input sizes of tokens (224 and 384), and different model architectures (DeiT and ViT). All models are pre-trained from the open-source database, where three ViT models are from `PyTorch-Pretrained-ViT` (Wightman, 2019), and the DeiT model is from `Facebook-Research-DeiT` (Touvron et al., 2021a). We also verify the robustness of our proposal, ARMRO, via three adversarial patch attacks, Token-Attack, PatchFool, and GMYA. PatchFool attack is open-sourced (Fu et al., 2022). Since we failed to access the fine-tuning detail of GMYA, our implementation of GMYA is weaker than they reported. For comparison with the related work, we implement Attention-Mask. We choose $384 \times 384 \times 3$ images randomly sampled from ImageNet 2012 (Deng et al., 2009). All codes are written in Python and PyTorch (Paszke et al., 2019) Platform.

7.2. Defense Robustness

In Table 1, we test three attacks to verify the robustness of ARMRO. In the experiment, we use 100 images to learn the detection threshold, τ, for each model. We set the number of detection as $N_d = 5$. We show the difference between inference with defense and without defense for each model. Initially, we evaluate the clean accuracy by inferring the clean dataset. ARMRO compromises around 1% of clean accuracy, and the degradation on ViT-L is more extensive for its larger model size or patch size. More layer in the model enlarges the chances of incorrectly detecting the benign patch as adversarial, and a larger patch size amplifies the impact of masking the benign.

We evaluate the defense robustness toward three different patch attacks, and we test all 1-patch, 2-patch, and 5-patch attacks. We set the perturbation radius of the adversarial patch to 0.8, which means the value of each pixel can be altered to $204/255$. The size of adversarial patches is equal to the patch size of the model, e.g., it is $32\times$ 32 pixels for ViT/32 and 16×16 pixels for ViT/16. A 32×32 patch covers 0.69% area of 384×384 images, and five patches cover 3.4% of the area.

Our proposal achieves promising robustness, where the robustness is measured by inferring the images with adversarial patches. For the 1-patch attack, we can correctly detect

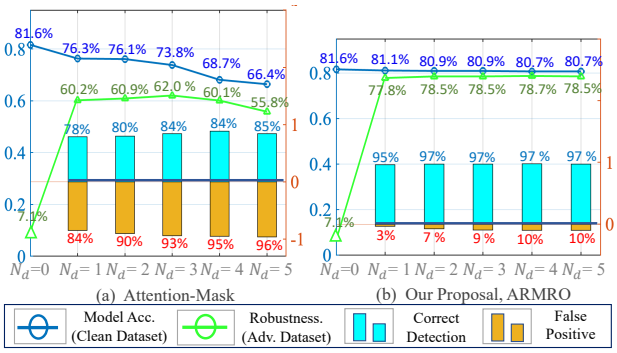


Figure 5: **Comparison to Attention-Mask.** Both schemes are tested by 1-patch Patch-Fool attacks on ViT-B/32. N_d states the number of detection, and $N_d = 0$ means no defense. Results for other models are in Appendix J.

around 97% of adversarial patches, and the robustness is 2∼3% near the accuracy. For 2-patch attacks, the robustness is about 5% less than the accuracy. The 5-patch attack is challenging since missing 1 or 2 of the patches leads to the failure of this defense. Our defense can achieve around 85% of the success detecting rate and maintain the robustness to around 70%.

7.3. Comparison to Related Work

Comparison VS. Attention-Mask: In Figure 5, we compare our work with the attention mask. We implement the attention-mask defense scheme and evaluate it with our proposed defense result, and we choose 1-patch Patch-Fool as the targeted attack. We test 5 different detecting numbers,$N_d = \{1, \cdots 5\}$, for both defense. The result for $N_d = 0$ is tested on the w/o defense model. The blue bars plot the adversarial patch detection rate, the percentage of successfully detected patches out of the total adversarial samples. The orange bar plots the false positive ratios of detecting benign tokens. The false positive of the attention-mask is greater than 90% for two reasons: first, the attention mask monitors the value matrix, but commonly the value of adversarial tokens is not the highest one; second, the attention mask spends equal effort on both inactivate stage and activate, but the adversarial token is less salient than the benign patch in the inactivate stage, which vastly increases the false positive. Attention mask produces low clean accuracy on ViT-B/32, where it is dropped from 81.6% to 66.4% when $N_d = 5$. Masking too many benign tokens becomes unacceptable as the area of tokens increases.

Our proposed defense achieves higher correctness, above 90% when $N_d = 1$, and is 97% $\sim$ 98% when $N_d = 5$. The overall degradation in the clean accuracy is negligible (less than 3%). Moreover, our proposal achieves 20% higher robustness than the attention mask and reaches above 80%.

Comparison VS. CNN-Based Patch Defense:

Table 2: Comparison to Patch Defenses from CNN

Defense	Model	1-Patch	2-Patch	5-Patch
Salience Map	ViT-B/16	55%	45%	33%
	ViT-B/32	37%	30%	24%
	ViT-L/32	40%	30%	21%
Patch Guard	ViT-B/16	64%	60%	51%
	ViT-B/32	62%	57%	49%
	ViT-L/32	59%	46%	42%
Ours	ViT-B/16	83%	80%	72%
	ViT-B/32	79%	76%	67%
	ViT-L/32	80%	79%	73%

At present, there is a dearth of patch-detecting defenses developed specifically for ViT. To facilitate a more comprehensive comparison, we have adapted two established patch defenses, originally designed for Convolutional Neural Networks (CNN), to work with ViT. (a) `Saliency-Map` (Hayes, 2018; Smilkov et al., 2017) generates a saliency map by computing saliency from the gradient of the input and masks the most salient areas. It has been observed that adversarial patches tend to be more salient than benign ones. In our implementation, we calculate the saliency map using $|\nabla_x \ell_{\text{CE}}(f(x^{(0)}), y)|$ and mask the top 5% of the largest area. (b) `Patch-Guard` (Xiang et al., 2021) is a method that creates a robust mask in the feature map of CNN models. For our comparison with ViT, we adapted this method to apply the same robust mask within the score matrix.

In Table 2, we present the accuracy of both the Saliency Map and Patch Guard defenses after 1-patch, 2-patch, and 5-patch iterations of the Patch Fool Attack. It's evident that both defenses only provide basic protection. Patch Guard achieves an accuracy ranging from 50% to 60%, whereas the Saliency Map ranges from 40% to 50%. Notably, these results are 30% to 40% less effective than the performance of our defense.

Comparison Vs. Certified defense:

We present comparative results in Appendix H. Our defense method consistently outperforms others across all measured metrics, including clean accuracy, inference time, and robustness. However, it's crucial to note that comparing empirical defenses, like our proposed one, with provable defenses, such as Smooth-ViT, may not be entirely appropriate. These two kinds of defenses serve different application scenarios. Our defense is rapid, and highly robust, boasting an adversarial detection rate of over 97%, making it particularly advantageous in computationally intensive domains like self-driving technology.

7.4. Detecting Adaptive Attacks

As all existing attention-based patch attacks create a strong anomaly, our defense is able to efficiently identify and mask

them at an early stage. However, we also consider whether attackers could lessen the strength of the adversarial patch in an attempt to bypass our detection and harm the model. In this section, we explore three types of adaptive attacks that could potentially threaten our defense. During these experiments, we mimic a real-world scenario in which our defense is unaware of any reduction in the strength of the attack.

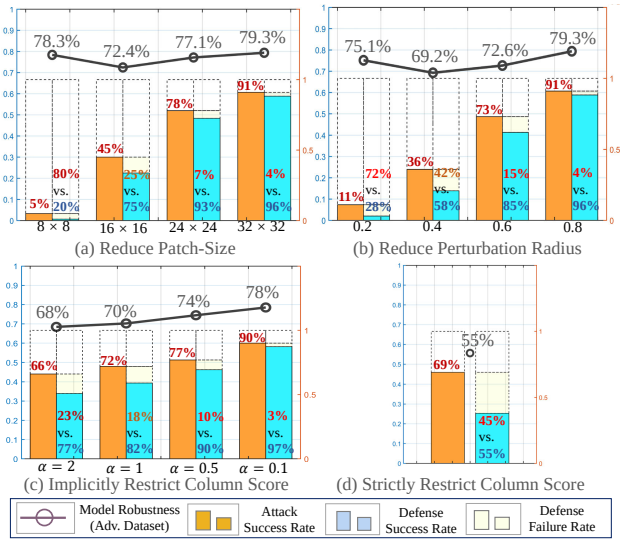


Figure 6: **Defense Against Adaptive Attack.** Tested on ViT-B/32 against Patch-Fool attack. More Results for other models are in Appendix I.

Reduced-Strength Attacks: In Figure 6 (a) and (b), we first try reducing the patch size from 32 pixels to 24, 16, and 8 pixels. Second, we experiment with the perturbation radius from 0.8 to 0.6, 0.4, and 0.2. The model is ViT-B/32 (results for other models are in Appendix I). The robustness measures the average accuracy over 1,000 adversarially patched images. The attack success rate is the percentage of attacks that successfully alter the output of the benign model. The defense success rate is the percentage that the ARMOR defends those successful attacks.

The results show that ARMOR achieves adequate robustness in defending reduced-strength attacks. For 24-pixel and 0.6-radius attacks, the attack success rate is above 70%, and our defense maintains above 75% success defending rate. For the 16-pixel size or 0.4-radius attacks, our defense becomes weaker (to 58% 75%) since the adversarial tokens are less salient and harder to detect, but the attack success rate decreases to less than 50%, and the overall robustness drop is within 10%.

Column-Score-Restricted Patch: In order for an adaptive attack to succeed, the column score of the adversarial patch must not surpass τ times the score of other benign patches. This constraint must hold true across all 12 layers. To further probe the potential of this approach, we devise two strategies for fabricating patches: one with weak restrictions, and the other with strict restrictions. The loss function used for the weak restriction strategy is as follows:

$$\ell_1 = \ell_{CE}(f(x), y) - \alpha \cdot \Sigma_l ||\bar{S}_p^{(l)}||_2^2$$

This loss function aims to maximize the distance between the model's output and the target label, while minimizing its column scores. The regulation term's coefficient, α, is small (ranging from 0.5 to 2 in our experiments). Although this can prevent the column score from increasing indefinitely, it does not guarantee that the patches will always remain within the detection boundary. For scenarios that require a strict restriction, we employ the following loss function:

$$\ell_2 = \ell_{CE}(f(x), y) - \beta \cdot \Sigma_{l \in L} ||\bar{S}_p^{(l)} - \tau \cdot \max(\bar{S}_{\text{benign}}^{(l)})||_2^2$$
$$\text{where} \quad L = \{l | \bar{S}_p^{(l)} > \tau \cdot \max(\bar{S}_{\text{benign}}^{(l)})\}$$

This loss function imposes a strict regulation, ensuring the column score remains below τ times the largest benign token, denoted as $\max(\bar{S}_{\text{benign}}^{(l)})$. The regulatory term accumulates only for the layers, $l \in L$, that surpass the detection threshold. Here, we employ a substantial coefficient, either $\beta = 50$ or $\beta = 100$, which exerts the maximum restriction to keep the adversarial patch outside the detection range. Further details can be found in Appendix E.

In Figure 6 (c) and (d), we test the performance of our defense against patches trained with both weak and strict restrictions. We evaluate four coefficients, $\alpha = 0.1, 0.5, 1, 2$, for the weak restriction, and our defense maintains a robustness level of over 68%. The strategy with strict restrictions proves to be the most effective, achieving a high attack success rate of 69%. However, only 55% of these attacks are detected by our defense, resulting in an overall degradation of robustness to 55%.

Multi-Patch Adaptive Attack: We also conducted experiments on the adaptive attack using two or five adversarial patches. Under weak restriction, the adaptive attack achieves the highest fool rate, resulting in a reduction of our defense's robustness to 35% in ViT-L/32. Despite this, our defense still outperforms Attention-Mask, maintaining an accuracy that's 13% higher. Detailed results can be found in Appendix G.

8. Conclusion

This work comprehensively studies the attention-based adversarial patch attack. Through two experiments, we find the adversarial patches are only activated in a few layers. Further, we uncover that adversarial patches elevate the score columns and propagate their pattern. Finally, we propose ARMOR to generate robust networks.

References

Aldahdooh, A., Hamidouche, W., and Deforges, O. Reveal of vision transformers robustness against adversarial attacks. *arXiv preprint arXiv:2106.03734*, 2021.

Allen-Zhu, Z. and Li, Y. Feature purification: How adversarial training performs robust deep learning. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 977–988. IEEE, 2022.

Brown, T. B., Mané, D., Roy, A., Abadi, M., and Gilmer, J. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.

Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I., Madry, A., and Kurakin, A. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9650–9660, 2021.

Chen, Z., Li, B., Xu, J., Wu, S., Ding, S., and Zhang, W. Towards practical certifiable patch defense with vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15148–15158, 2022.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Fu, Y., Zhang, S., Wu, S., Wan, C., and Lin, Y. Patch-fool: Are vision transformers always robust against adversarial perturbations? *arXiv preprint arXiv:2203.08392*, 2022.

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

Gu, J., Tresp, V., and Qin, Y. Are vision transformers robust to patch perturbations? In *European Conference on Computer Vision*, pp. 404–421. Springer, 2022.

Hayes, J. On visible adversarial perturbations & digital watermarking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1597–1604, 2018.

Herrmann, C., Sargent, K., Jiang, L., Zabih, R., Chang, H., Liu, C., Krishnan, D., and Sun, D. Pyramid adversarial training improves vit performance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13419–13429, 2022.

Joshi, A., Jagatap, G., and Hegde, C. Adversarial token attacks on vision transformers. *arXiv preprint arXiv:2110.04337*, 2021.

Levine, A. and Feizi, S. (de) randomized smoothing for certifiable defense against patch attacks. *Advances in Neural Information Processing Systems*, 33:6465–6475, 2020.

Liu, X., Yang, H., Liu, Z., Song, L., Li, H., and Chen, Y. Dpatch: An adversarial patch attack on object detectors. *arXiv preprint arXiv:1806.02299*, 2018.

Lovisotto, G., Finnie, N., Munoz, M., Mummadi, C. K., and Metzen, J. H. Give me your attention: Dot-product attention considered harmful for adversarial patch robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15234–15243, 2022.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Mao, X., Qi, G., Chen, Y., Li, X., Duan, R., Ye, S., He, Y., and Xue, H. Towards robust vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12042–12051, 2022.

Mu, N. and Wagner, D. Defending against adversarial patches with robust self-attention. In *ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning*, 2021.

Naseer, M. M., Ranasinghe, K., Khan, S. H., Hayat, M., Shahbaz Khan, F., and Yang, M.-H. Intriguing properties of vision transformers. *Advances in Neural Information Processing Systems*, 34:23296–23308, 2021.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Paul, S. and Chen, P.-Y. Vision transformers are robust learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 2071–2081, 2022.

Raghu, M., Unterthiner, T., Kornblith, S., Zhang, C., and Dosovitskiy, A. Do vision transformers see like convolutional neural networks? *Advances in Neural Information Processing Systems*, 34:12116–12128, 2021.

Salman, H., Jain, S., Wong, E., and Madry, A. Certified patch robustness via smoothed vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15137–15147, 2022.

Shao, R., Shi, Z., Yi, J., Chen, P.-Y., and Hsieh, C.-J. On the adversarial robustness of vision transformers. *arXiv preprint arXiv:2103.15670*, 2021.

Smilkov, D., Thorat, N., Kim, B., Viégas, F., and Wattenberg, M. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jegou, H. Training data-efficient image transformers amp; distillation through attention. In *International Conference on Machine Learning*, volume 139, pp. 10347–10357, July 2021a.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pp. 10347–10357. PMLR, 2021b.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, J., Liu, A., Yin, Z., Liu, S., Tang, S., and Liu, X. Dual attention suppression attack: Generate adversarial camouflage in physical world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8565–8574, 2021.

Wightman, R. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019.

Xiang, C., Bhagoji, A. N., Sehwag, V., and Mittal, P. Patchguard: A provably robust defense against adversarial patches via small receptive fields and masking. In *USENIX Security Symposium*, pp. 2237–2254, 2021.

A. Details of Generating Adversarial Patches

The position of the adversarial patch, e.g., p in x_p, significantly affects the attack success rate. The existing adversarial patch attacks have developed some heuristics to find an optimal position. Token-Attack (Joshi et al., 2021) chooses the patch that has the highest backward gradient on its input:

$$\text{Token-Attack}: \ p = \underset{1 \leq p \leq n}{\text{argmax}} ||\nabla_{x_p^{(0)}} \mathcal{L}_0||_2^2 \tag{1}$$

Patch-Fool (Fu et al., 2022) sums the attention scores between the query of the adversarial token, $x_p^{(l)} W_Q$, and all keys, $x^{(l)} W_K$, for all L layers, and select the position where the patch, p, which gets the highest summation score:

$$\text{Patch-Fool}: \ p = \underset{1 \leq p \leq n}{\text{argmax}} \sum_{l=1}^{L} ||x_p^{(l)} W_Q (x^{(l)} W_K)^T||_2^2 \tag{2}$$

The patch position found by Token-Attack has the largest gradient, where altering the patch in such position will maximumly shift model output. However, since the model output is a high-dimension vector, e.g., 1,000 classes, it is difficult to control the output shift occurring in the target class. Patch-Fool selects the patch with the strongest connection with other patches. We experiment with these two position-finding algorithms and uncover that the patch-fool achieves around 5% to 10% higher attack success rate.

After selecting the patch, the following work is to construct the patch pattern. The Projected Gradient Descend (PGD) can be formulated as follows:

$$x_p^{t,(0)} = x_p^{t-1,(0)} + \sigma \cdot \text{sign}(\nabla_{x^{(0)}} \mathcal{L}_0) \ \ , \ \ t \in \{1, 2, \cdots T\} \tag{3}$$

PGD repeats the gradient descent by T times to train the adversarial patch, $x_p^{(0)}$, which maximizes the loss of model, $\mathcal{L}_0 = \ell(f(x_p^{t,(0)}), y)$.

Patch-Fool (Fu et al., 2022) and Give-Me-Your-Attention (GMYA) (Lovisotto et al., 2022) enhance patch attacks by redesigning the loss functions. when conducting the PGD:

$$\begin{aligned} \text{Patch-Fool}: \mathcal{L}_0 + \sum_{l=1}^{l_1} \sum_{h} S^{(l,h)} \\ \text{GMYA}: \mathcal{L}_0 + \sum_{l=1}^{l_2} \sum_{h} Q^{(l,h)} K^{(l,h)T} \end{aligned} \tag{4}$$

Both schemes integrate the scores into the loss and maximize the sum of the scores between every two tokens. In addition, based on the observation that the tokens in the rear layers are less affected by the adversarial patch, these schemes only sum the scores for the first l_1 and l_2 layers (l_1,$l_2 < L$), respectively. The difference between them is that Patch-Fool uses the post-softmax scores while GMYA uses the pre-softmax ones. Both successfully degrade all ViT/DeiT models to 0% within five patches.

B. Details of De-Randomized Smoothing

After the adversarial attack was uncovered and received considerable attention, a statistic-based defense was proposed, which is Randomized Smoothing (RS). RS considers the adversarial perturbation a special outcome under a Gaussian Distribution. It generates a larger number of random noises under the same distribution and in the same space, which can be formulated as:

$$f_{RS}(x) = \mathbb{E}_{\epsilon \in \mathcal{N}(0, \sigma^2 I)}[f(x^{(0)} + \epsilon)] \tag{5}$$

where ϵ is Gaussian Noise under $\mathcal{N}(0, \sigma^2 I)$, and σ is the perturbation radius equal to the adversarial. RS randomly samples a large amount, e.g., 1,000, of the same image inputs and selects the majority as the final class.

De-Randomized Smoothing (DS) (Levine & Feizi, 2020) is a variant of Randomized Smoothing specifically targeting adversarial patch attacks. DS assumes that adversarial patches only appear in a small area of the image. DS randomly

removes parts of the image, and the parts-ablated images have a large change to exclude the adversarial patches. Thus, the statistical outputs are expected to produce the correct classification results. DS is formulated as:

$$f_{DS}(x) = \mathbb{E}_{\mathcal{M}}[f(\mathcal{M} \odot x^{(0)})] \tag{6}$$

where $\mathcal{M}$ is the randomly-sampled parts-removing function, and it varies from different implementations.

For defending the adversarial attack toward ViT models, Certifiable-Patch-Defense (Chen et al., 2022) applies DS schemes and adopts the line segment parts-removing function, which is:

$$\mathcal{M}_p \odot x^{(0)} = x^{(0)}_{p:p+D,\,:} \tag{7}$$

where p is the randomly-sampled position, and D is the length of the line segment. Certifiable-Patch-Defense crops the image into column-wise or row-wise line segments and uses it in model inference.

They conducted experiments on the classic DS schemes in ViT. They improved both efficiency and robustness by proposing adaptations. which are progressive-smoothed image modeling and isolated band unit self-attention. They generated 1,024 ablated images from the original image and achieved 30% to 40% of robustness for ViT models, but the clean accuracy drops from 85% to 66%. Smooth-ViT(Salman et al., 2022) achieved similar robustness (30% to 40%) and batch size (1,024 ablated images), but they improved the clean accuracy to 69%.

C. Layer-Wise Updating of The Adversarial Token During Training

Through our theoretical study, we discovered that the layer-wise local optimum aims to optimally position the adversarial keys at the center of the query tokens. Achieving this optimum maximizes the column scores. Consequently, we experimented with monitoring the changes in the mean column score of the adversarial token, S , during the training process to obtain the adversarial patch. We randomly selected an image from ImageNet and charted the changes in its column score across various epochs and layers, as shown in the table below.

D. The Detecting Threshold

In our defense, we choose a coefficient τ as the threshold to identify the abnormality of whether a token is adversarial. Before deploying ViT models, we use 100 samples to compute the optimal setting of τ. In Table 3, we present the statistics of the abnormality ratio of both benign tokens and adversarial tokens. If the abnormality ratio is greater than τ, it will be considered adversarial. The abnormality ratio is calculated by the largest mean column score, $\max_i(\bar{S}_i)$, divided by the second-largest mean column score, second-$\max_i(\bar{S}_i)$, where the mean column score use the average of all layers and heads. μ_{benign} and μ_{adv} represent the mean values of the abnormality ratio of benign tokens and adversarial tokens, respectively. σ_{benign} and σ_{adv} denote the standard deviations.

As is shown in Table 3, we observe there is a clear gap between the distribution of benign tokens and adversarial tokens, so there is a wide range in choosing the threshold, τ. In our defense, we adopt the three-sigma rule, $\tau = \mu_{\text{benign}} + 3 \cdot \sigma_{\text{benign}}$, where $> 99.7\%$ of benign samples are within the three-sigma range, so our detection can avoid faulty detection and precisely capture the adversarial patches.

Table 3: Statistic of The Abnormality Ratio

Model	μ_{benign}	σ_{benign}	μ_{benign}	σ_{benign}	τ
ViT-B/32	1.12	0.11	2.86	1.87	1.61
ViT-B/16	1.09	0.11	2.84	1.55	1.43
ViT-L/32	1.07	0.09	2.66	1.49	1.35
DeiT-B/16	1.10	0.10	2.92	1.73	1.42

E. Adaptive Attack: Strict Column-Score-Restricted Attack

We have experimented with a variety of formulations to create the adversarial patch. The adaptive attack under strict restriction is the most successful among all our attempts. Under the strict restriction, adversarial patches learns the adversarial pattern to enlarge CE loss of the output, and their column scores for every layer are bounded to a undetectable range.

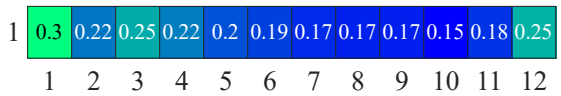
Table 4: Example of an Adversarial Patch Trained by Adaptive Attack (Strict Restriction).

Epoch	Abnormality Ratio for Each Layer $\lvert\bar{S}_p^{(l)}\rvert / \max \lvert\bar{S}_{\text{benign}}^{(l)}\rvert$												CE Loss
	1	2	3	4	5	6	7	8	9	10	11	12	
1	0.53	0.27	0.77	0.79	1.05	0.59	0.47	0.50	0.17	0.32	0.31	0.29	0.20
10	0.64	0.34	0.50	0.67	0.78	0.89	0.82	1.18	0.34	0.62	0.71	1.07	1.90
50	0.41	0.32	0.45	0.87	0.79	0.98	1.00	1.11	0.41	0.80	0.86	0.94	2.45
100	0.39	0.29	0.49	0.90	0.85	1.20	1.25	1.08	0.45	1.03	0.89	1.49	3.65
150	0.37	0.28	0.40	0.86	0.87	1.47	1.49	1.10	0.47	1.05	0.73	1.33	4.35
200	0.37	0.28	0.36	0.82	0.85	1.40	1.47	1.12	0.52	1.17	0.88	1.39	4.58

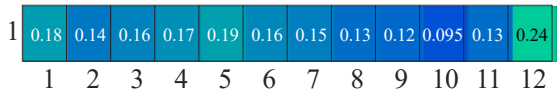
In Table 4, we capture an successful adversarial patch trained by such adaptive attack. We monitor the abnormality ratios, $|\bar{S}_p^{(l)}| / \max |\bar{S}_{\text{benign}}^{(l)}|$, for all 12 layers during the training. The abnormality ratio is calculated by the largest mean column score, $\max_i(\bar{S}_i)$, divided by the second-largest mean column score, second-$\max_i(\bar{S}_i)$, where the mean column score use the average of all layers and heads. The detection threshold, τ, is 1.5 where the adversarial patch will be detected if the abnormality ratios ratio exceeds 1.5. As the adversarial pattern is learned, the CE loss incrementally rises, and the model successfully produces the wrong output.

During the training, whenever the abnormality ratios of the column scores surpass the detection threshold (1.5), the strict regulation term will produce a significant gradient term to drag abnormality ratios to the undetectable boundary. For example, at the 100th epoch, the abnormality ratios in the 12th reach 1.49, which is close to the threshold, and it drops to 1.33 at the 150th epoch. As is shown, none of the abnormality ratios exceed the threshold, and such patch is undetectable to our defense.

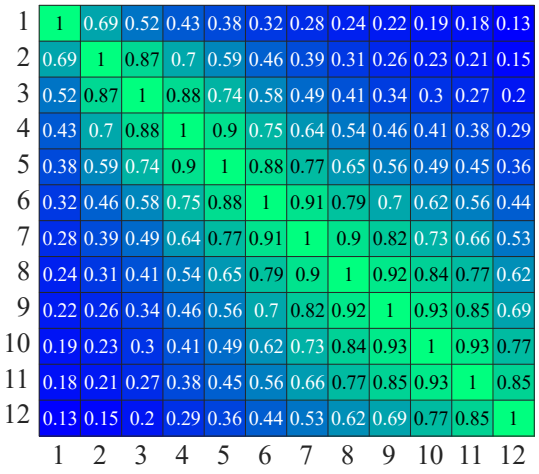
F. Token Similarity Test on Various Models



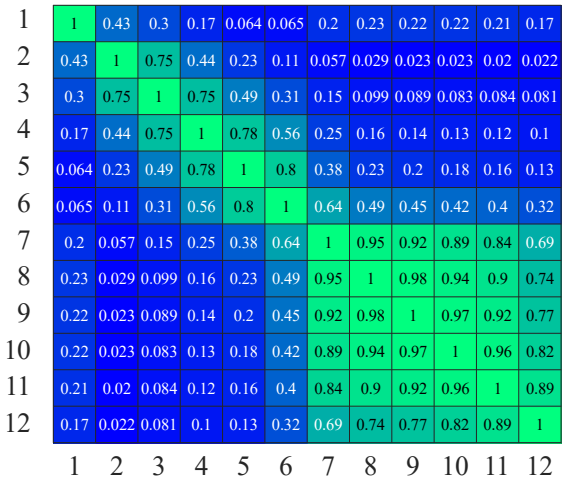
(a) Layer-Wise Update Ratio of Adversarial Token



(b) Layer-Wise Update Ratio of The Average of All Tokens



(c) Layer-Wise Cosine Similarity of Adversarial Token



(d) Layer-Wise Cosine Similarity of The Average of All Tokens

Figure 7: **ViT-B/16:** Comparison of Token Similarity between Adversarial Patches and Benign Patches.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0.1	0.091	0.13	0.15	0.16	0.12	0.14	0.16	0.15	0.12	0.13	0.12	0.075	0.069	0.045	0.051	0.051	0.054	0.054	0.048	0.058	0.065	0.068	0.07

(a) Layer-Wise Update Ratio of Adversarial Token

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0.15	0.27	0.36	0.33	0.3	0.28	0.28	0.25	0.25	0.2	0.23	0.18	0.16	0.12	0.11	0.12	0.13	0.11	0.11	0.11	0.12	0.14	0.16	0.22

(b) Layer-Wise Update Ratio of The Average of All Tokens

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	1	0.74	0.67	0.6	0.55	0.52	0.48	0.44	0.42	0.38	0.37	0.35	0.33	0.31	0.29	0.28	0.26	0.25	0.25	0.24	0.23	0.23	0.23	0.22
2	0.74	1	0.96	0.87	0.8	0.73	0.65	0.59	0.53	0.48	0.46	0.43	0.41	0.38	0.36	0.35	0.33	0.32	0.31	0.31	0.29	0.29	0.29	0.29
3	0.67	0.96	1	0.94	0.88	0.8	0.72	0.65	0.58	0.53	0.51	0.47	0.44	0.42	0.39	0.38	0.37	0.36	0.34	0.33	0.32	0.32	0.31	0.31
4	0.6	0.87	0.94	1	0.97	0.91	0.84	0.76	0.69	0.62	0.6	0.56	0.52	0.49	0.47	0.45	0.44	0.42	0.41	0.4	0.39	0.38	0.38	0.37
5	0.55	0.8	0.88	0.97	1	0.97	0.91	0.85	0.78	0.71	0.69	0.64	0.6	0.56	0.54	0.52	0.5	0.48	0.47	0.46	0.45	0.44	0.44	0.43
6	0.52	0.73	0.8	0.91	0.97	1	0.97	0.92	0.86	0.8	0.77	0.71	0.67	0.63	0.6	0.58	0.56	0.54	0.53	0.52	0.51	0.5	0.5	0.49
7	0.48	0.65	0.72	0.84	0.91	0.97	1	0.98	0.93	0.87	0.84	0.79	0.74	0.71	0.68	0.66	0.63	0.61	0.6	0.59	0.58	0.57	0.57	0.56
8	0.44	0.59	0.65	0.76	0.85	0.92	0.98	1	0.96	0.92	0.89	0.85	0.8	0.77	0.74	0.71	0.69	0.67	0.66	0.65	0.64	0.63	0.62	0.62
9	0.42	0.53	0.58	0.69	0.78	0.86	0.93	0.96	1	0.97	0.94	0.9	0.86	0.83	0.79	0.77	0.75	0.73	0.71	0.71	0.69	0.69	0.68	0.68
10	0.38	0.48	0.53	0.62	0.71	0.8	0.87	0.92	0.97	1	0.98	0.95	0.91	0.88	0.85	0.82	0.8	0.79	0.77	0.76	0.75	0.74	0.74	0.73
11	0.37	0.46	0.51	0.6	0.69	0.77	0.84	0.89	0.94	0.98	1	0.98	0.94	0.92	0.88	0.86	0.84	0.82	0.81	0.8	0.79	0.78	0.77	0.77
12	0.35	0.43	0.47	0.56	0.64	0.71	0.79	0.85	0.9	0.95	0.98	1	0.97	0.95	0.92	0.9	0.88	0.87	0.85	0.84	0.83	0.82	0.81	0.81
13	0.33	0.41	0.44	0.52	0.6	0.67	0.74	0.8	0.86	0.91	0.94	0.97	1	0.98	0.96	0.94	0.92	0.9	0.89	0.87	0.86	0.85	0.85	0.84
14	0.31	0.38	0.42	0.49	0.56	0.63	0.71	0.77	0.83	0.88	0.92	0.95	0.98	1	0.98	0.96	0.95	0.93	0.91	0.9	0.89	0.88	0.88	0.87
15	0.29	0.36	0.39	0.47	0.54	0.6	0.68	0.74	0.79	0.85	0.88	0.92	0.96	0.98	1	0.99	0.97	0.96	0.94	0.93	0.92	0.91	0.91	0.9
16	0.28	0.35	0.38	0.45	0.52	0.58	0.66	0.71	0.77	0.82	0.86	0.9	0.94	0.96	0.99	1	0.99	0.97	0.96	0.94	0.93	0.93	0.92	0.91
17	0.26	0.33	0.37	0.44	0.5	0.56	0.63	0.69	0.75	0.8	0.84	0.88	0.92	0.95	0.97	0.99	1	0.99	0.98	0.96	0.95	0.94	0.94	0.93
18	0.25	0.32	0.36	0.42	0.48	0.54	0.61	0.67	0.73	0.79	0.82	0.87	0.9	0.93	0.96	0.97	0.99	1	0.99	0.98	0.97	0.96	0.95	0.94
19	0.25	0.31	0.34	0.41	0.47	0.53	0.6	0.66	0.71	0.77	0.81	0.85	0.89	0.91	0.94	0.96	0.98	0.99	1	0.99	0.98	0.98	0.97	0.95
20	0.24	0.31	0.33	0.4	0.46	0.52	0.59	0.65	0.71	0.76	0.8	0.84	0.87	0.9	0.93	0.94	0.96	0.98	0.99	1	0.99	0.99	0.98	0.96
21	0.23	0.29	0.32	0.39	0.45	0.51	0.58	0.64	0.69	0.75	0.79	0.83	0.86	0.89	0.92	0.93	0.95	0.97	0.98	0.99	1	0.99	0.99	0.97
22	0.23	0.29	0.32	0.38	0.44	0.5	0.57	0.63	0.69	0.74	0.78	0.82	0.85	0.88	0.91	0.93	0.94	0.96	0.98	0.99	0.99	1	1	0.98
23	0.23	0.29	0.31	0.38	0.44	0.5	0.57	0.62	0.68	0.74	0.77	0.81	0.85	0.88	0.91	0.92	0.94	0.95	0.97	0.98	0.99	1	1	0.99
24	0.22	0.29	0.31	0.37	0.43	0.49	0.56	0.62	0.68	0.73	0.77	0.81	0.84	0.87	0.9	0.91	0.93	0.94	0.95	0.96	0.97	0.98	0.99	1

(c) Layer-Wise Cosine Similarity of Adversarial Token

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	1	0.83	0.62	0.38	0.29	0.17	0.1	0.072	0.047	0.064	0.062	0.071	0.055	0.061	0.07	0.072	0.07	0.06	0.051	0.069	0.065	0.074	0.079	0.082
2	0.83	1	0.84	0.57	0.44	0.27	0.18	0.11	0.084	0.1	0.09	0.089	0.072	0.07	0.07	0.072	0.067	0.051	0.041	0.054	0.045	0.043	0.043	0.04
3	0.62	0.84	1	0.83	0.67	0.5	0.35	0.2	0.16	0.16	0.14	0.13	0.11	0.095	0.088	0.089	0.078	0.053	0.043	0.054	0.039	0.027	0.023	0.013
4	0.38	0.57	0.83	1	0.89	0.74	0.58	0.38	0.32	0.3	0.26	0.22	0.2	0.17	0.15	0.14	0.13	0.098	0.087	0.097	0.079	0.069	0.061	0.048
5	0.29	0.44	0.67	0.89	1	0.9	0.75	0.52	0.46	0.41	0.37	0.32	0.29	0.24	0.23	0.21	0.19	0.16	0.15	0.16	0.13	0.12	0.12	0.1
6	0.17	0.27	0.5	0.74	0.9	1	0.88	0.63	0.56	0.51	0.47	0.42	0.38	0.33	0.3	0.28	0.27	0.23	0.21	0.22	0.19	0.18	0.17	0.15
7	0.1	0.18	0.35	0.58	0.75	0.88	1	0.82	0.75	0.68	0.63	0.56	0.5	0.44	0.41	0.39	0.37	0.33	0.31	0.31	0.29	0.28	0.27	0.23
8	0.072	0.11	0.2	0.38	0.52	0.63	0.82	1	0.92	0.86	0.81	0.73	0.65	0.59	0.56	0.53	0.51	0.49	0.47	0.47	0.45	0.45	0.44	0.39
9	0.047	0.084	0.16	0.32	0.46	0.56	0.75	0.92	1	0.94	0.87	0.78	0.7	0.62	0.58	0.55	0.53	0.5	0.49	0.48	0.47	0.46	0.45	0.4
10	0.064	0.1	0.16	0.3	0.41	0.51	0.68	0.86	0.94	1	0.94	0.86	0.78	0.7	0.66	0.63	0.61	0.58	0.57	0.56	0.54	0.53	0.51	0.46
11	0.062	0.09	0.14	0.26	0.37	0.47	0.63	0.81	0.87	0.94	1	0.93	0.86	0.78	0.75	0.72	0.69	0.66	0.64	0.64	0.61	0.59	0.56	0.5
12	0.071	0.089	0.13	0.22	0.32	0.42	0.56	0.73	0.78	0.86	0.93	1	0.95	0.89	0.85	0.82	0.79	0.76	0.74	0.73	0.69	0.67	0.63	0.56
13	0.055	0.072	0.11	0.2	0.29	0.38	0.5	0.65	0.7	0.78	0.86	0.95	1	0.95	0.91	0.88	0.85	0.82	0.8	0.78	0.74	0.71	0.65	0.58
14	0.061	0.07	0.095	0.17	0.24	0.33	0.44	0.59	0.62	0.7	0.78	0.89	0.95	1	0.98	0.96	0.93	0.89	0.87	0.85	0.81	0.78	0.71	0.63
15	0.07	0.07	0.088	0.15	0.23	0.3	0.41	0.56	0.58	0.66	0.75	0.85	0.91	0.98	1	0.98	0.96	0.92	0.9	0.87	0.84	0.8	0.73	0.65
16	0.072	0.072	0.089	0.14	0.21	0.28	0.39	0.53	0.55	0.63	0.72	0.82	0.88	0.96	0.98	1	0.98	0.95	0.92	0.9	0.87	0.83	0.76	0.69
17	0.07	0.067	0.078	0.13	0.19	0.27	0.37	0.51	0.53	0.61	0.69	0.79	0.85	0.93	0.96	0.98	1	0.97	0.95	0.93	0.9	0.86	0.79	0.71
18	0.06	0.051	0.053	0.098	0.16	0.23	0.33	0.49	0.5	0.58	0.66	0.76	0.82	0.89	0.92	0.95	0.97	1	0.98	0.96	0.93	0.89	0.82	0.74
19	0.051	0.041	0.043	0.087	0.15	0.21	0.31	0.47	0.49	0.57	0.64	0.74	0.8	0.87	0.9	0.92	0.95	0.98	1	0.98	0.95	0.91	0.84	0.76
20	0.069	0.054	0.054	0.097	0.16	0.22	0.31	0.47	0.48	0.56	0.64	0.73	0.78	0.85	0.87	0.9	0.93	0.96	0.98	1	0.98	0.94	0.87	0.79
21	0.065	0.045	0.039	0.079	0.13	0.19	0.29	0.45	0.47	0.54	0.61	0.69	0.74	0.81	0.84	0.87	0.9	0.93	0.95	0.98	1	0.97	0.91	0.84
22	0.074	0.043	0.027	0.069	0.12	0.18	0.28	0.45	0.46	0.53	0.59	0.67	0.71	0.78	0.8	0.83	0.86	0.89	0.91	0.94	0.97	1	0.96	0.9
23	0.079	0.043	0.023	0.061	0.12	0.17	0.27	0.44	0.45	0.51	0.56	0.63	0.65	0.71	0.73	0.76	0.79	0.82	0.84	0.87	0.91	0.96	1	0.96
24	0.082	0.04	0.013	0.048	0.1	0.15	0.23	0.39	0.4	0.46	0.5	0.56	0.58	0.63	0.65	0.69	0.71	0.74	0.76	0.79	0.84	0.9	0.96	1

(d) Layer-Wise Cosine Similarity of The Average of All Tokens

Figure 8: **ViT-L/16:** Comparison of Token Similarity between Adversarial Patches and Benign Patches.

0.069	0.08	0.093	0.13	0.1	0.098	0.097	0.088	0.068	0.047	0.027	0.0098	0.0089	0.0083	0.0079	0.0073	0.0073	0.0072	0.0071	0.007	0.0083	0.0083	0.011	0.018
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

(a) Layer-Wise Update Ratio of Adversarial Token

0.1	0.13	0.13	0.18	0.15	0.13	0.13	0.13	0.12	0.11	0.12	0.1	0.1	0.1	0.097	0.085	0.088	0.088	0.083	0.098	0.091	0.095	0.13	0.22
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

(b) Layer-Wise Update Ratio of The Average of All Tokens

1	1	0.83	0.63	0.41	0.3	0.22	0.19	0.16	0.1	0.05	0.0056	-0.0072	-0.0091	-0.0092	-0.009	-0.0091	-0.0092	-0.0094	-0.0094	-0.009	-0.0089	-0.0096	-0.0095	-0.0076
2	0.83	1	0.89	0.67	0.49	0.4	0.35	0.28	0.17	0.066	-0.02	-0.053	-0.058	-0.061	-0.062	-0.063	-0.063	-0.063	-0.063	-0.063	-0.062	-0.063	-0.063	-0.066
3	0.63	0.89	1	0.86	0.68	0.58	0.52	0.42	0.3	0.17	0.058	0.0096	0.00091	-0.0032	-0.0049	-0.006	-0.0068	-0.0074	-0.0075	-0.0077	-0.0071	-0.0074	-0.0081	-0.013
4	0.41	0.67	0.86	1	0.87	0.78	0.69	0.57	0.42	0.24	0.082	0.019	0.0078	0.0019	-0.00038	-0.0024	-0.0036	-0.005	-0.0052	-0.0059	-0.0058	-0.006	-0.0072	-0.012
5	0.3	0.49	0.68	0.87	1	0.94	0.86	0.75	0.58	0.36	0.17	0.093	0.079	0.072	0.07	0.068	0.066	0.064	0.064	0.063	0.064	0.063	0.061	0.055
6	0.22	0.4	0.58	0.78	0.94	1	0.93	0.84	0.66	0.4	0.17	0.084	0.067	0.059	0.056	0.053	0.051	0.049	0.048	0.047	0.047	0.046	0.044	0.038
7	0.19	0.35	0.52	0.69	0.86	0.93	1	0.93	0.77	0.49	0.24	0.13	0.11	0.1	0.097	0.094	0.091	0.089	0.087	0.086	0.085	0.084	0.081	0.075
8	0.16	0.28	0.42	0.57	0.75	0.84	0.93	1	0.9	0.64	0.37	0.24	0.21	0.2	0.2	0.19	0.19	0.18	0.18	0.18	0.18	0.18	0.17	0.16
9	0.1	0.17	0.3	0.42	0.58	0.66	0.77	0.9	1	0.88	0.66	0.52	0.5	0.48	0.48	0.47	0.47	0.46	0.46	0.46	0.46	0.45	0.45	0.43
10	0.05	0.066	0.17	0.24	0.36	0.4	0.49	0.64	0.88	1	0.91	0.81	0.79	0.78	0.77	0.77	0.76	0.76	0.76	0.75	0.75	0.75	0.74	0.73
11	0.0056	-0.02	0.058	0.082	0.17	0.17	0.24	0.37	0.66	0.91	1	0.98	0.97	0.96	0.96	0.95	0.95	0.95	0.95	0.95	0.95	0.94	0.94	0.93
12	-0.0072	-0.053	0.0096	0.019	0.093	0.084	0.13	0.24	0.52	0.81	0.98	1	1	1	1	1	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.98
13	-0.0091	-0.058	0.00091	0.0078	0.079	0.067	0.11	0.21	0.5	0.79	0.97	1	1	1	1	1	1	1	1	1	0.99	0.99	0.99	0.98
14	-0.0092	-0.061	-0.0032	0.0019	0.072	0.059	0.1	0.2	0.48	0.78	0.96	1	1	1	1	1	1	1	1	1	1	1	0.99	0.98
15	-0.009	-0.062	-0.0049	-0.00038	0.07	0.056	0.097	0.2	0.48	0.77	0.96	1	1	1	1	1	1	1	1	1	1	1	1	0.99
16	-0.0091	-0.063	-0.006	-0.0024	0.068	0.053	0.094	0.19	0.47	0.77	0.95	1	1	1	1	1	1	1	1	1	1	1	1	0.99
17	-0.0092	-0.063	-0.0068	-0.0036	0.066	0.051	0.091	0.19	0.47	0.76	0.95	0.99	1	1	1	1	1	1	1	1	1	1	1	0.99
18	-0.0094	-0.063	-0.0074	-0.005	0.064	0.049	0.089	0.18	0.46	0.76	0.95	0.99	1	1	1	1	1	1	1	1	1	1	1	0.99
19	-0.0094	-0.063	-0.0075	-0.0052	0.064	0.048	0.087	0.18	0.46	0.76	0.95	0.99	1	1	1	1	1	1	1	1	1	1	1	0.99
20	-0.009	-0.063	-0.0077	-0.0059	0.063	0.047	0.086	0.18	0.46	0.75	0.95	0.99	1	1	1	1	1	1	1	1	1	1	1	0.99
21	-0.0089	-0.062	-0.0071	-0.0058	0.064	0.047	0.085	0.18	0.46	0.75	0.95	0.99	0.99	1	1	1	1	1	1	1	1	1	1	0.99
22	0.088	-0.041	0.016	0.07	0.2	0.25	0.32	0.36	0.38	0.47	0.6	0.78	0.84	0.88	0.9	0.91	0.93	0.95	0.96	0.97	0.99	1	0.98	0.86
23	-0.0095	-0.063	-0.0081	-0.0072	0.061	0.044	0.081	0.17	0.45	0.74	0.94	0.99	0.99	0.99	1	1	1	1	1	1	1	1	1	0.99
24	-0.0076	-0.066	-0.013	-0.012	0.055	0.038	0.075	0.16	0.43	0.73	0.93	0.98	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	1
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

(c) Layer-Wise Cosine Similarity of Adversarial Token

1	1	0.78	0.59	0.37	0.24	0.13	0.1	0.078	0.053	0.043	0.044	0.052	0.05	0.062	0.064	0.065	0.066	0.073	0.072	0.074	0.083	0.088	0.083	0.093
2	0.78	1	0.88	0.64	0.38	0.25	0.18	0.13	0.085	0.047	0.014	-0.028	-0.044	-0.048	-0.054	-0.058	-0.055	-0.055	-0.054	-0.053	-0.045	-0.041	-0.037	-0.013
3	0.59	0.88	1	0.86	0.6	0.47	0.38	0.3	0.23	0.18	0.13	0.076	0.051	0.038	0.024	0.016	0.014	0.011	0.0094	0.0097	0.014	0.016	0.02	0.033
4	0.37	0.64	0.86	1	0.84	0.72	0.63	0.54	0.45	0.37	0.29	0.2	0.16	0.13	0.11	0.098	0.091	0.079	0.076	0.072	0.073	0.07	0.068	0.072
5	0.24	0.38	0.6	0.84	1	0.93	0.86	0.78	0.69	0.61	0.52	0.41	0.35	0.31	0.28	0.26	0.25	0.23	0.23	0.22	0.21	0.2	0.19	0.17
6	0.13	0.25	0.47	0.72	0.93	1	0.95	0.89	0.81	0.73	0.64	0.51	0.43	0.38	0.35	0.32	0.31	0.29	0.28	0.27	0.26	0.25	0.24	0.21
7	0.1	0.18	0.38	0.63	0.86	0.95	1	0.96	0.89	0.82	0.73	0.61	0.53	0.47	0.43	0.41	0.39	0.37	0.36	0.35	0.34	0.32	0.31	0.27
8	0.078	0.13	0.3	0.54	0.78	0.89	0.96	1	0.95	0.89	0.81	0.68	0.59	0.53	0.48	0.45	0.43	0.41	0.39	0.38	0.37	0.36	0.34	0.3
9	0.053	0.085	0.23	0.45	0.69	0.81	0.89	0.95	1	0.96	0.88	0.74	0.64	0.58	0.53	0.5	0.47	0.45	0.43	0.41	0.4	0.38	0.36	0.32
10	0.043	0.047	0.18	0.37	0.61	0.73	0.82	0.89	0.96	1	0.95	0.83	0.74	0.67	0.62	0.59	0.57	0.54	0.52	0.5	0.49	0.47	0.44	0.39
11	0.044	0.014	0.13	0.29	0.52	0.64	0.73	0.81	0.88	0.95	1	0.93	0.85	0.8	0.75	0.73	0.7	0.68	0.66	0.64	0.62	0.6	0.57	0.49
12	0.052	-0.028	0.076	0.2	0.41	0.51	0.61	0.68	0.74	0.83	0.93	1	0.97	0.94	0.92	0.9	0.88	0.86	0.84	0.82	0.81	0.78	0.75	0.63
13	0.05	-0.044	0.051	0.16	0.35	0.43	0.53	0.59	0.64	0.74	0.85	0.97	1	0.98	0.96	0.95	0.93	0.92	0.9	0.88	0.87	0.84	0.81	0.68
14	0.062	-0.048	0.038	0.13	0.31	0.38	0.47	0.53	0.58	0.67	0.8	0.94	0.98	1	0.99	0.98	0.96	0.95	0.93	0.92	0.9	0.88	0.84	0.71
15	0.064	-0.054	0.024	0.11	0.28	0.35	0.43	0.48	0.53	0.62	0.75	0.92	0.96	0.99	1	0.99	0.98	0.97	0.95	0.94	0.92	0.9	0.86	0.72
16	0.065	-0.058	0.016	0.098	0.26	0.32	0.41	0.45	0.5	0.59	0.73	0.9	0.95	0.98	0.99	1	0.99	0.98	0.96	0.95	0.93	0.91	0.88	0.74
17	0.066	-0.055	0.014	0.091	0.25	0.31	0.39	0.43	0.47	0.57	0.7	0.88	0.93	0.96	0.98	0.99	1	0.99	0.98	0.97	0.95	0.93	0.89	0.75
18	0.073	-0.055	0.011	0.079	0.23	0.29	0.37	0.41	0.45	0.54	0.68	0.86	0.92	0.95	0.97	0.98	0.99	1	0.99	0.98	0.97	0.95	0.91	0.77
19	0.072	-0.054	0.0094	0.076	0.23	0.28	0.36	0.39	0.43	0.52	0.66	0.84	0.9	0.93	0.95	0.96	0.98	0.99	1	0.99	0.98	0.96	0.93	0.79
20	0.074	-0.053	0.0097	0.072	0.22	0.27	0.35	0.38	0.41	0.5	0.64	0.82	0.88	0.92	0.94	0.95	0.97	0.98	0.99	1	0.99	0.97	0.94	0.81
21	0.083	-0.045	0.014	0.073	0.21	0.26	0.34	0.37	0.4	0.49	0.62	0.81	0.87	0.9	0.92	0.93	0.95	0.97	0.98	0.99	1	0.99	0.96	0.83
22	0.088	-0.041	0.016	0.07	0.2	0.25	0.32	0.36	0.38	0.47	0.6	0.78	0.84	0.88	0.9	0.91	0.93	0.95	0.96	0.97	0.99	1	0.98	0.86
23	0.083	-0.037	0.02	0.068	0.19	0.24	0.31	0.34	0.36	0.44	0.57	0.75	0.81	0.84	0.86	0.88	0.89	0.91	0.93	0.94	0.96	0.98	1	0.9
24	0.093	-0.013	0.033	0.072	0.17	0.21	0.27	0.3	0.32	0.39	0.49	0.63	0.68	0.71	0.72	0.74	0.75	0.77	0.79	0.81	0.83	0.86	0.9	1
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

(d) Layer-Wise Cosine Similarity of The Average of All Tokens

Figure 9: **ViT-L/32:** Comparison of Token Similarity between Adversarial Patches and Benign Patches.

G. Defense Against The Multi-Patch Adaptive Attack

In this work, we continue the study of adaptive attacks and revise the aforementioned training strategy from single-adversarial patches to multiple-adversarial patches. This experiment is conducted under both weak and strict restriction.

Adaptive Attack under Weak Restriction: We employ a loss function with weak regulation by setting the coefficient. The loss function is expressed as follows:

$$\ell_3 = \ell_{CE}(f(x^{(0)} + \sum_{i}^{N} x_{p_i}^{(0)}), y) - \sum_{i,l}^{N,L} \alpha \cdot ||\bar{S}_{p_i}^{(l)-\tau\cdot\bar{S}_{\text{benign}}^{(l)}}||_2^2$$

We introduce N as the number of patches, and the regulation is the sum of the column score of all patches. The loss function employs the methodology previously discussed, using a weak restriction coefficient, $\alpha = 2$, to minimize the distance to the largest benign token. The results are presented in Table 5.

Table 5: Defense against Multi-Patch Adaptive Attack under Weak Restriction

# of Patches	Model	Attack Suc. Rate	Defense Suc. Rate	# of Patches being Detected	Robustness (Ours)	Robustness (Attention-Mask)
2	ViT-B/32	89%	80%	1.4	70%	45%
	ViT-B/16	16%	43%	1.1	74%	66%
	ViT-L/32	77%	75%	1.3	65%	45%
5	ViT-B/32	100%	45%	2.7	37%	21%
	ViT-B/16	69%	27%	2.4	43%	35%
	ViT-L/32	100%	42%	3.1	35%	22%

By infusing each patch with minimal adversarial features, our detection method faces greater difficulty in identifying abnormalities. During the experiment, we observe an interesting phenomenon. In defending against these stealthy attacks, such as the 5-patch attack, it is unnecessary to identify all five patches accurately. In some cases, detecting and masking two or three of these adversarial patches is sufficient to thwart the entire attack. We document the average number of adversarial patches detected in the **Number of Detected Patch column**.

Adaptive Attack under Strict Restriction: Similarly, we revise the loss function of the strictly restricted patch.

$$\ell_4 = \ell_{CE}(f(x^{(0)} + \sum_{i}^{N} x_{p_i}^{(0)}), y) - \beta \cdot \sum_{l,i\in C} ||\bar{S}_p^{(l)} - 0.95\tau \cdot \max(\bar{S}_{\text{benign}}^{(l)})||_2^2$$

$$\text{where} \quad C := \{i \in \{1, 2, \cdots, N\}\ ,\ l \in \{1, 2, \cdots, L\} \ \mid \ \bar{S}_{p_i}^{(l)} > \tau \cdot \max(\bar{S}_{\text{benign}}^{(l)})\ \}$$

Here, we adopt a strict coefficient, $\beta = 100$, to the regulation term. The regulation is the sum of the distance from the adversarial token to the largest beinign token, for all N patches and L layers. Moreover, in case the column score of the adversarial token bounces across the detection boundary, i.e., $\bar{S}_{p_i}^{(l)} \to \tau \cdot \max(\bar{S}_{\text{benign}}^{(l)})$. We introduce another coefficient, 0.95, to further limit the column scores, which removes the chance that the adversarial patch is too close to the detection boundary.

Table 6: Defense against Multi-Patch Adaptive Attack under Weak Restriction

# of Patches	Model	Attack Suc. Rate	Defense Suc. Rate	# of Patches being Detected	Robustness (Ours)	Robustness (Attention-Mask)
2	ViT-B/32	75%	67%	1.8	63%	47%
	ViT-B/16	35%	55%	1.6	71%	50%
	ViT-L/32	77%	75%	1.3	68%	55%
5	ViT-B/32	100%	69%	3.9	58%	25%
	ViT-B/16	86%	76%	3.7	67%	41%
	ViT-L/32	100%	80%	4.1	68%	28%

In Table 6, we show the result of the multi-patch adaptive attack under strict restriction. Formulating a perfect version of such an attack is a complex problem. The current results show that this attack has a worse performance compared to the weak restricted one. The strictly restricted attack can be easily detected by the multi-patch detection method. Since the loss function is too aggressive, most of the adversarial patches trained by such loss function tend to be overfitted.

Discussion: We maintain that the experimental results presented are consistent with our theoretical analysis. Adversarial patches must elevate their column scores to propagate their adversarial patterns to the noise tokens. Incorporating regulation during adversarial training can help distribute this propagation more evenly across layers. However, such stealthy attacks are inherently less powerful. As we increase the number of adversarial patches and reduce the strength of each patch, the trained adversarial patches will increasingly resemble benign patches, posing a significant challenge for all adversarial defenses. Our defense method remains the best empirical defense among all the baselines.

H. Comparison to The Certified Defense

In Section 3, the prior art section, we discussed three baselines, two of which are certified defenses. These certified defenses offer protection against adversarial patches with various shapes, sizes, and magnitudes. However, they demand substantial computational resources, and their robustness remains limited. We extracted the results from their papers, scaled our defense to equivalent settings, and generated the comparison in Table 7 and Table 8.

Table 7: Comparison to Cert-Patch

Defense	Model	Clean Accuracy	Inference time of 512 images	1-Patch	2-Patch	5-Patch
Cert-Patch	ViT-S/16 (224)	63.88%	9.66 second	35.6%	30.0%	25.9%
	ViT-B/16(224)	66.92%	16.63 second	47.3%	41.7%	37.3%
Ours	ViT-S/16 (224)	77.2%	0.334 second	74.3%	71.2%	62.5%
	ViT-B/16(224)	83.0%	0.737 second	79.4%	74.1%	67.0%

Table 8: Comparison to Smooth-ViT

Defense	Model	Clean Accuracy	Inference time of 512 images	1-Patch	2-Patch	5-Patch
Cert-Patch	ViT-S/16 (224)	67.1%	20.5 second	36.8%	31.6%	28.2%
	ViT-B/16(224)	73.2%	58.7 second	44.0%	38.2%	34.1%
Ours	ViT-S/16 (224)	77.2%	0.334 second	74.3%	71.2%	62.5%
	ViT-B/16(224)	83.0%	1.11 second	79.4%	74.1%	67.0%

Our code runs on a 4090 GPU, while Certified-Patch and Smooth-ViT use a V100 GPU. Therefore, we roughly scale our inference time by a factor of 3.1x based on the number of CUDA cores (5,120 vs. 16,380).

We convert the pixel area (1% or 2%) equivalently to the number of 16x16 or 32x32 patches: 1. For models using a 16x16 patch size: 1% pixel area corresponds to two 16x16 patches; 2% pixel area corresponds to four patches; 3% pixel area corresponds to six patches. 2. For models using a 32x32 patch size: 1% pixel area corresponds to one patch; 2% pixel area corresponds to one patch; 3% pixel area corresponds to two patches.

For the comparison, we implement our defense in the ViT-S/16 model and download the pre-trained ViT-S model from DINO-pretrained (Caron et al., 2021). We will update more attack/defense results for ViT-S/16 in the revised manuscript.

As demonstrated, our defense achieves superior performance across all metrics, including clean accuracy, inference time, and robustness. However, comparing empirical defenses (e.g., our proposed defense) with provable defenses (e.g., Smooth-ViT) might be inappropriate. We believe these two types of defenses have distinct application scenarios. Our defense is fast, highly robust (with an adversarial detection rate exceeding 97%), and can benefit computation-intensive domains such as self-driving.

I. Defense Performance against Reduced-Strength Attack

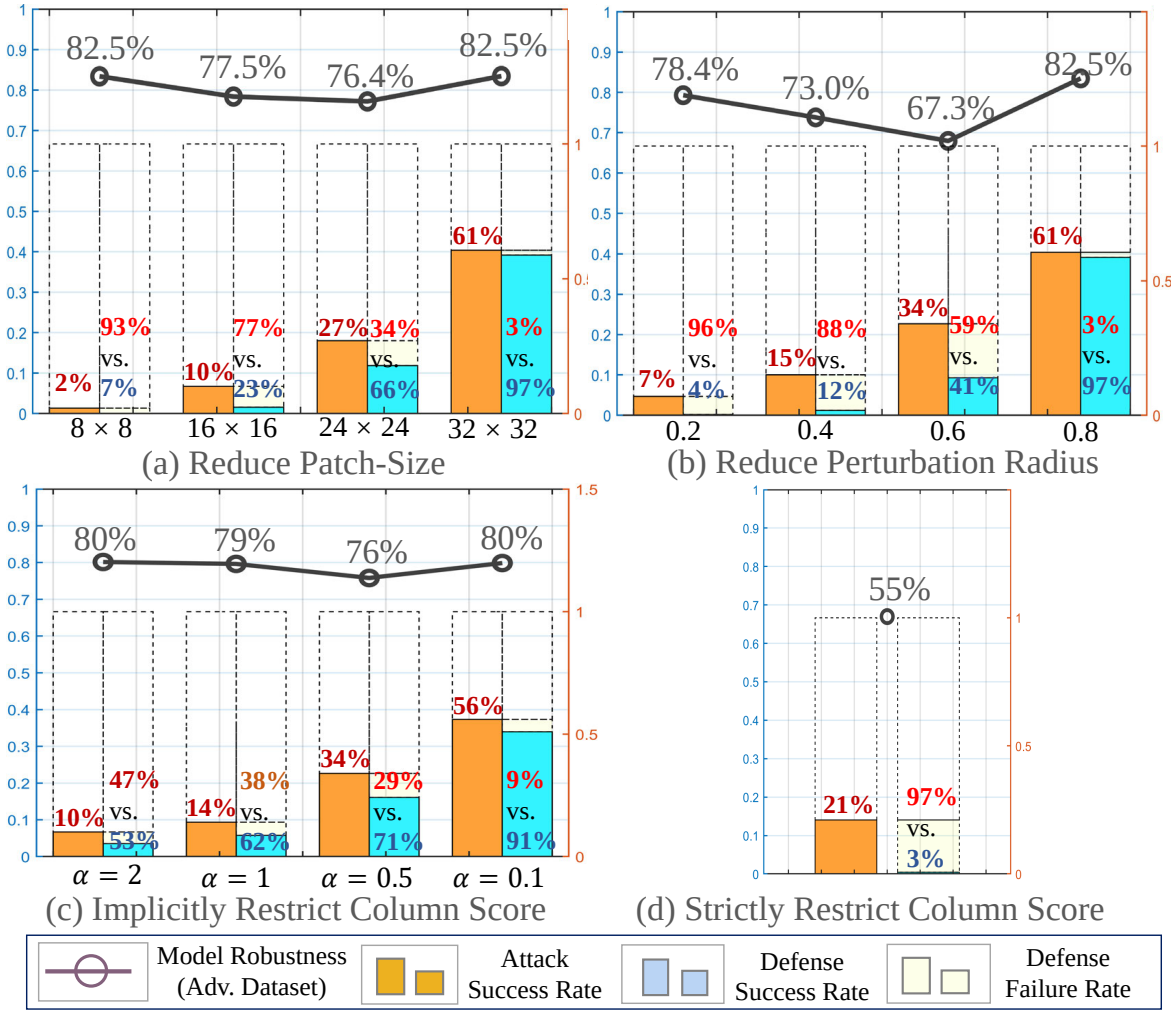


Figure 10: **ViT-B/16:** Defense Performance against Reduced-Strength Attack.

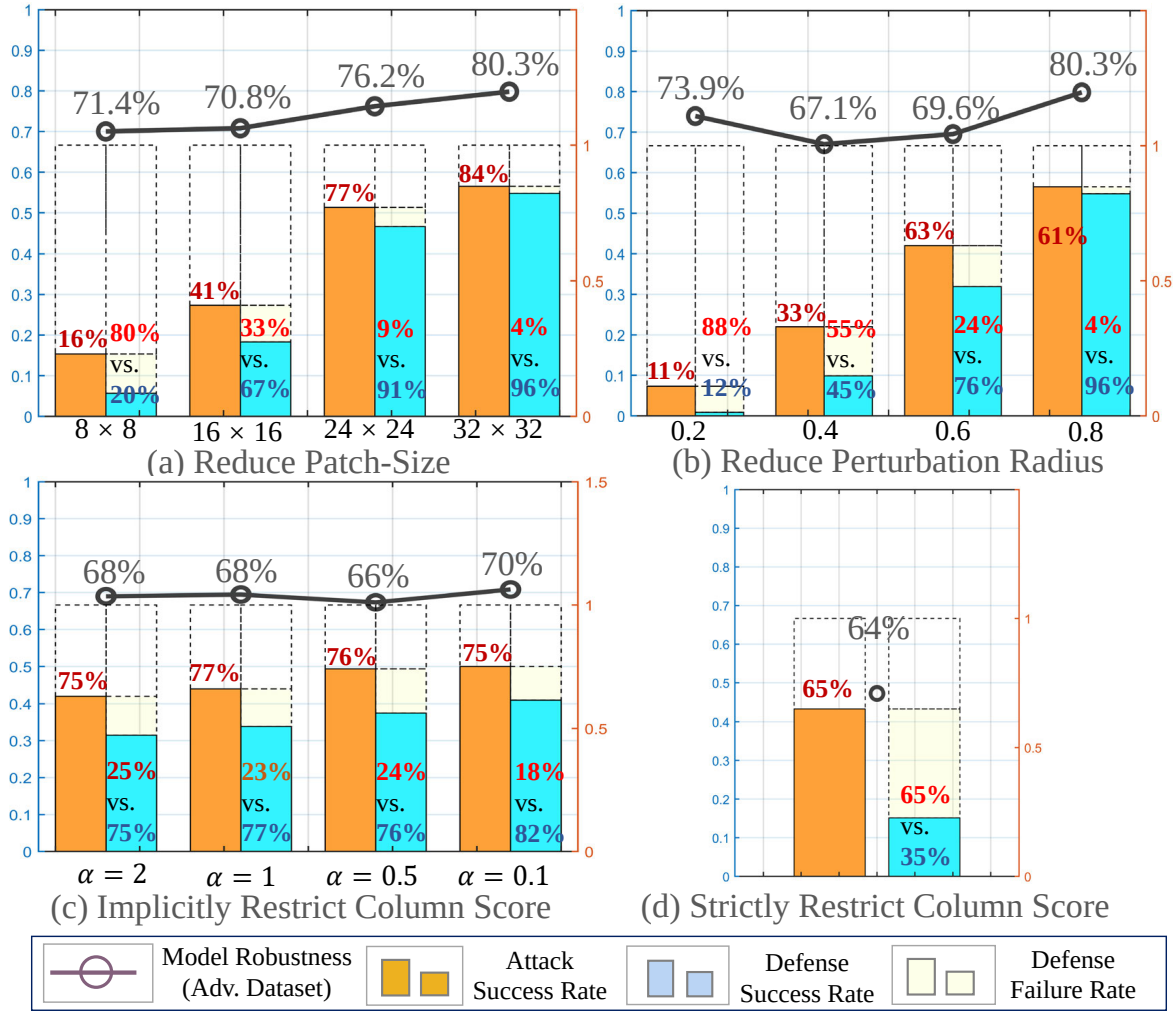


Figure 11: **ViT-L/32:** Defense Performance against Reduced-Strength Attack.

J. Defense Performance Compared to Attention-Mask

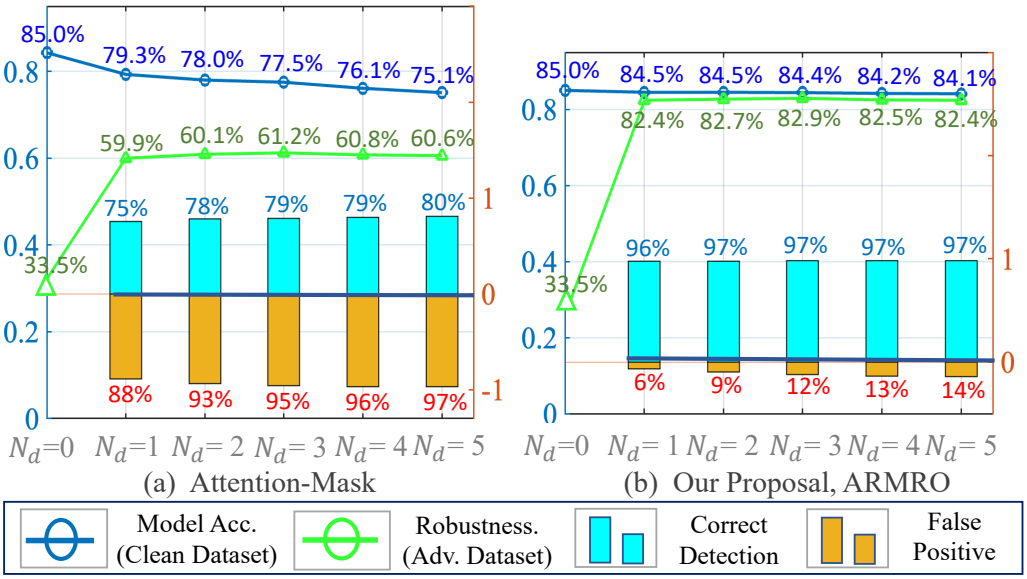


Figure 12: **ViT-L/32:** Comparison between Attention-Mask and our proposed ARMOR via different numbers of detection.

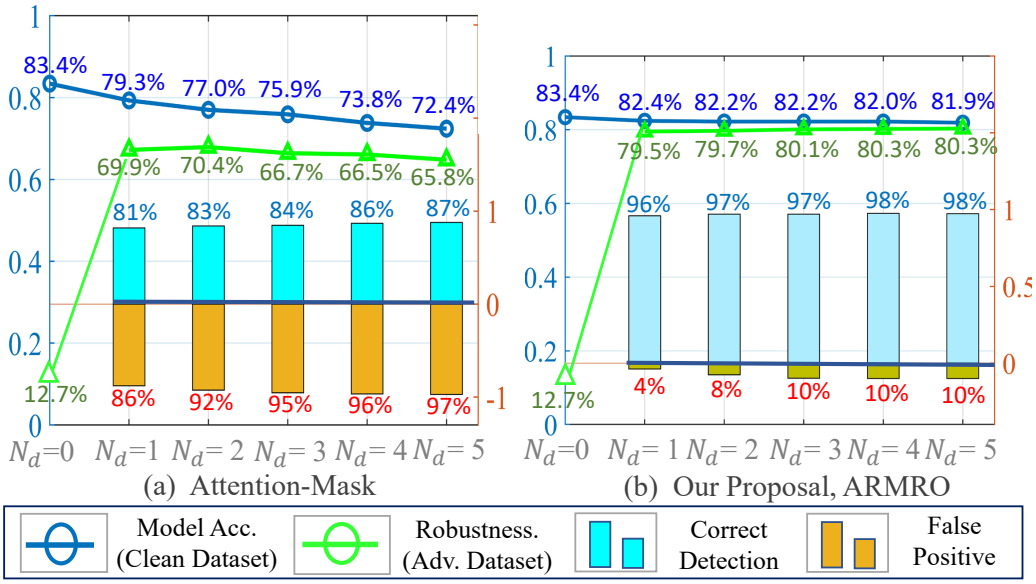


Figure 13: **ViT-B/16:** Comparison between Attention-Mask and our proposed ARMOR via different numbers of detection.

K. Query/Key Figures

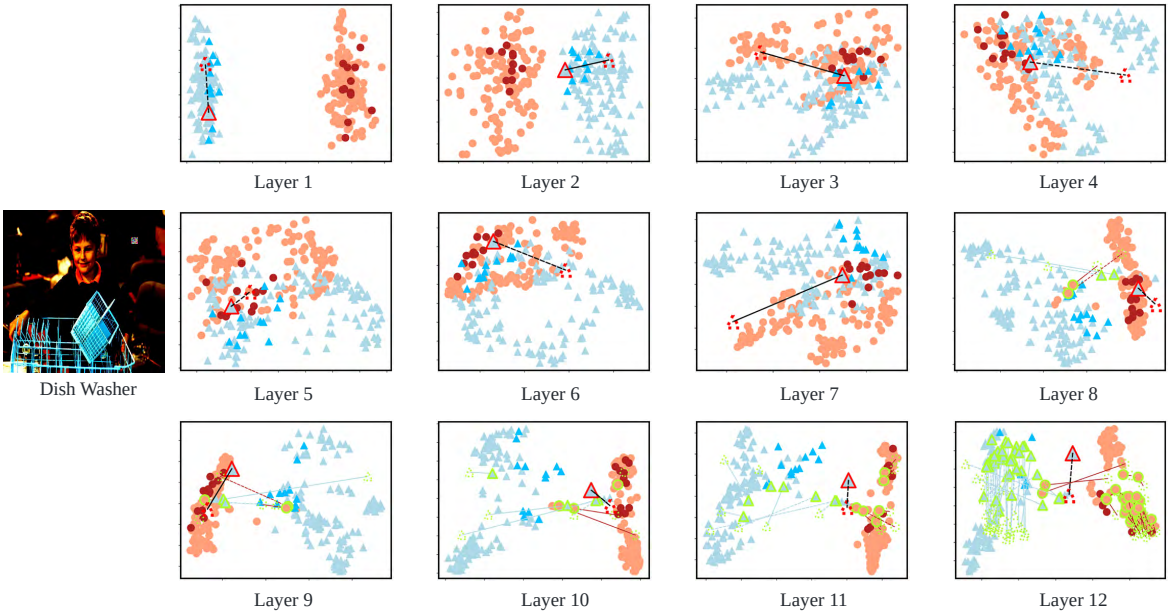


Figure 14: Query/Key Figures of Dish Washer Example

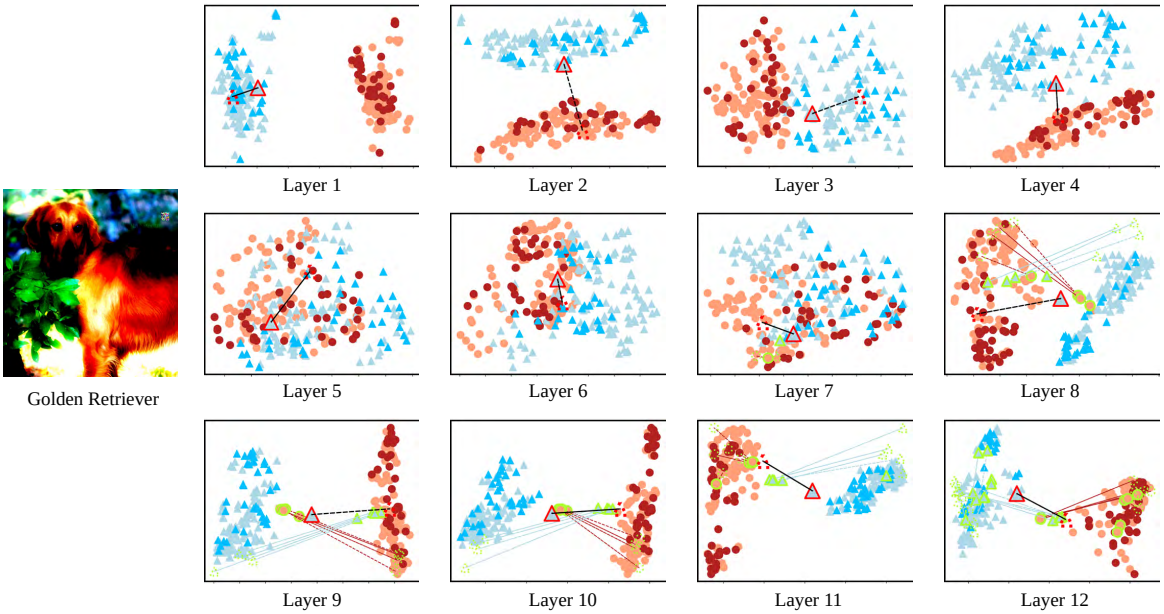


Figure 15: Query/Key Figures of Golden Retriever Example

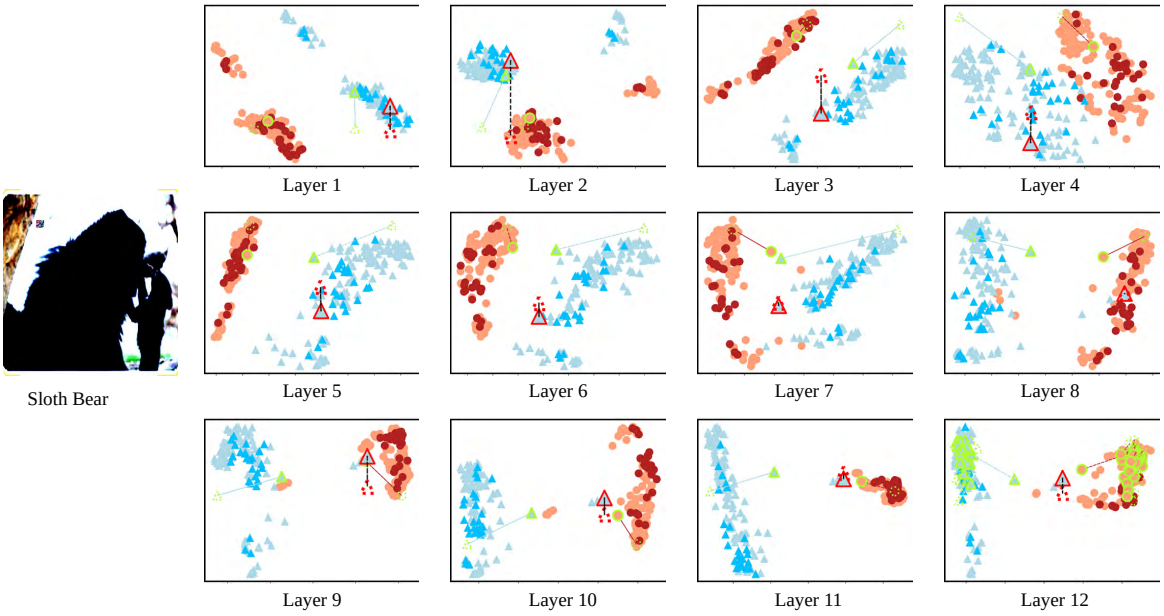


Figure 16: Query/Key Figures of Sloth Bear Example

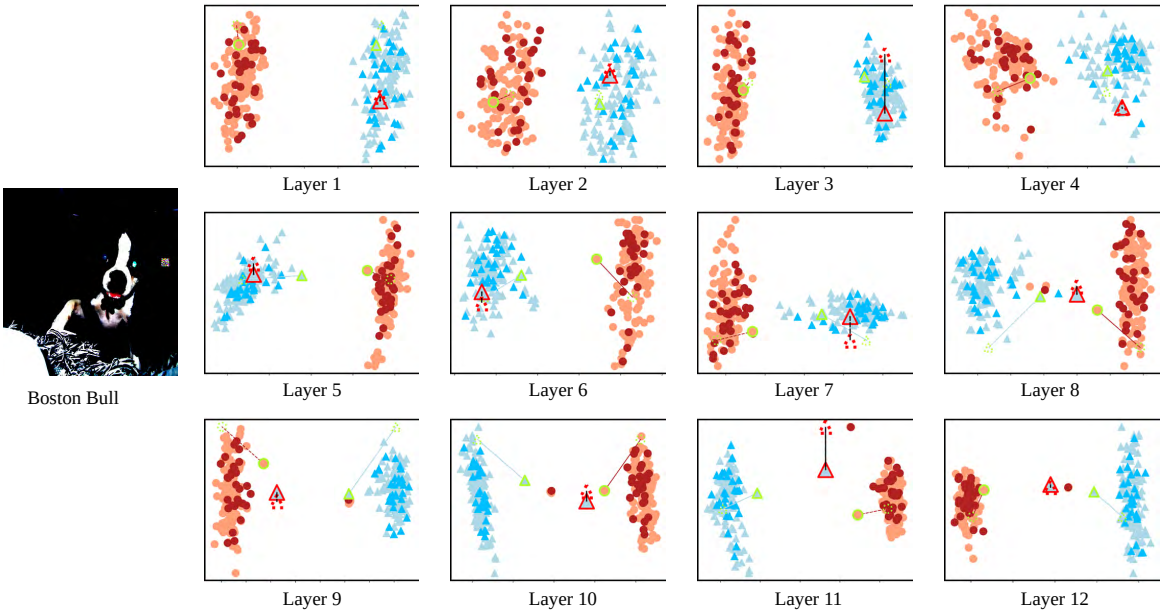


Figure 17: Query/Key Figures of Boston Bull Example

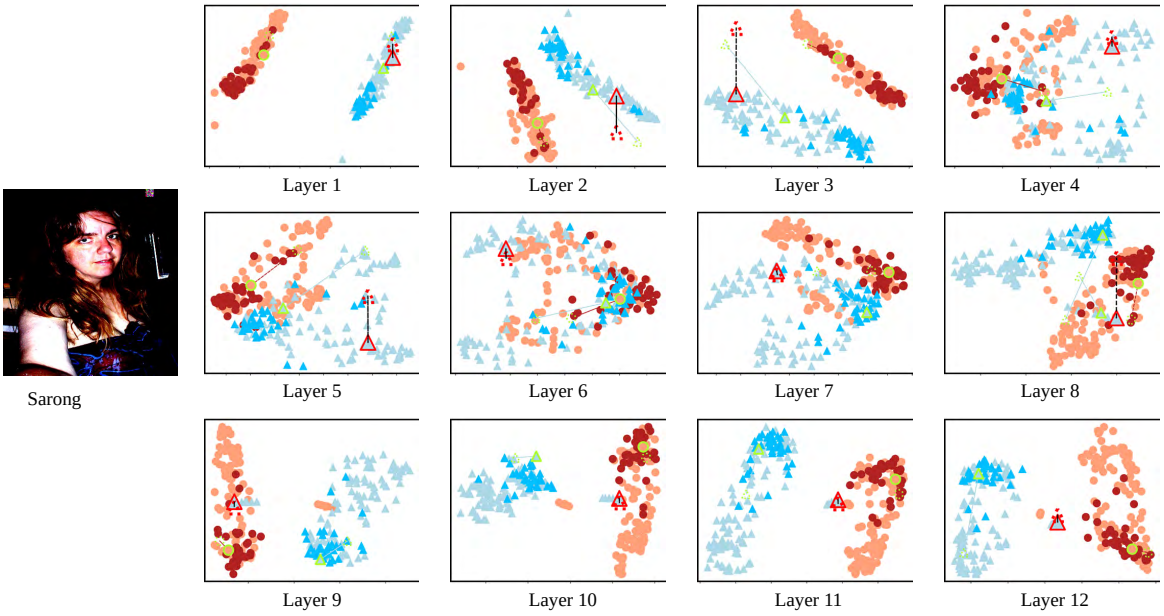


Figure 18: Query/Key Figures of Sarong Example

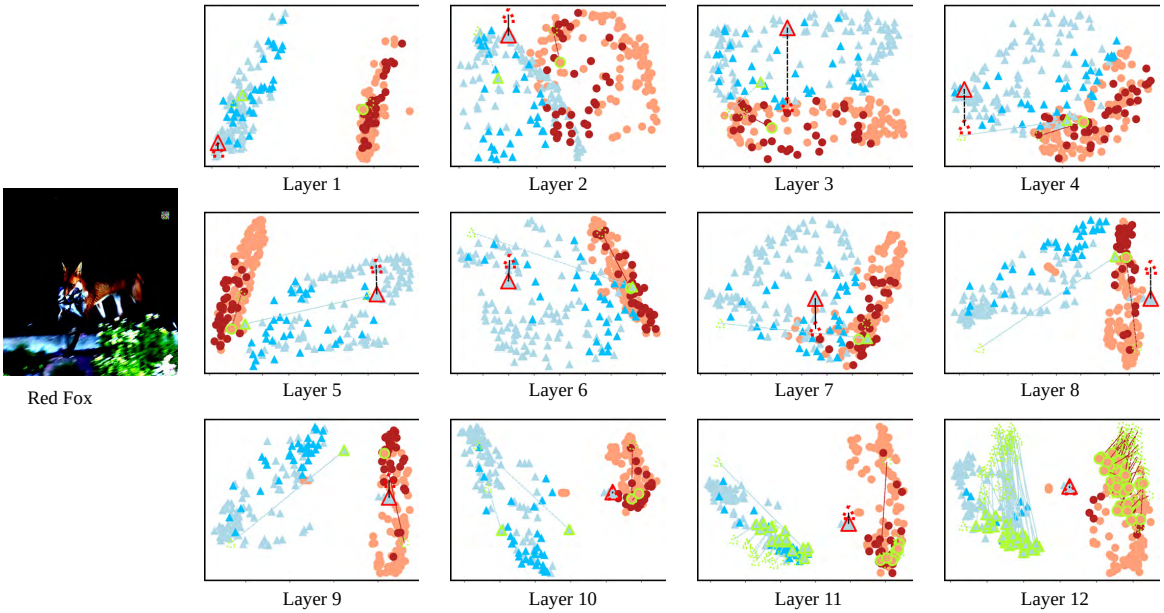


Figure 19: Query/Key Figures of Red Fox Example

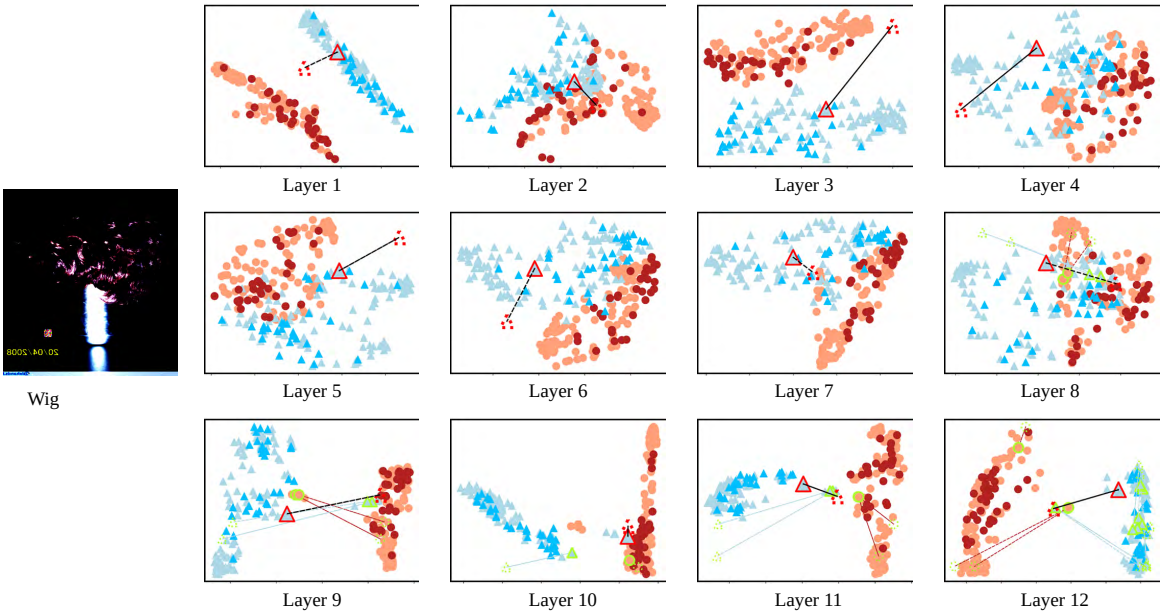


Figure 20: Query/Key Figures of Wig Example

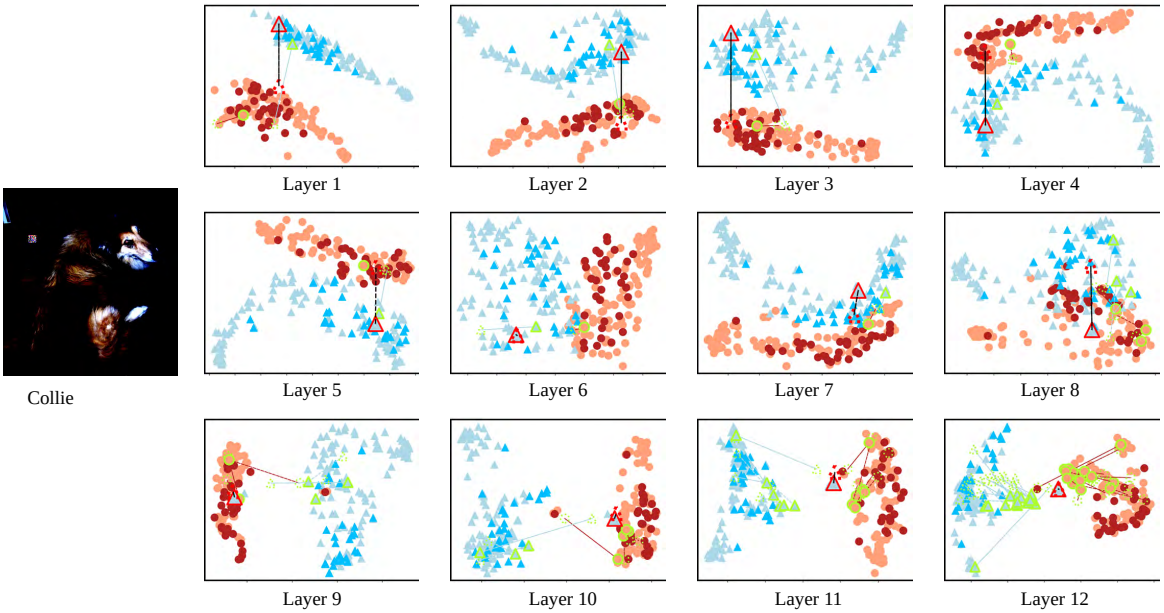


Figure 21: Query/Key Figures of Collie Example

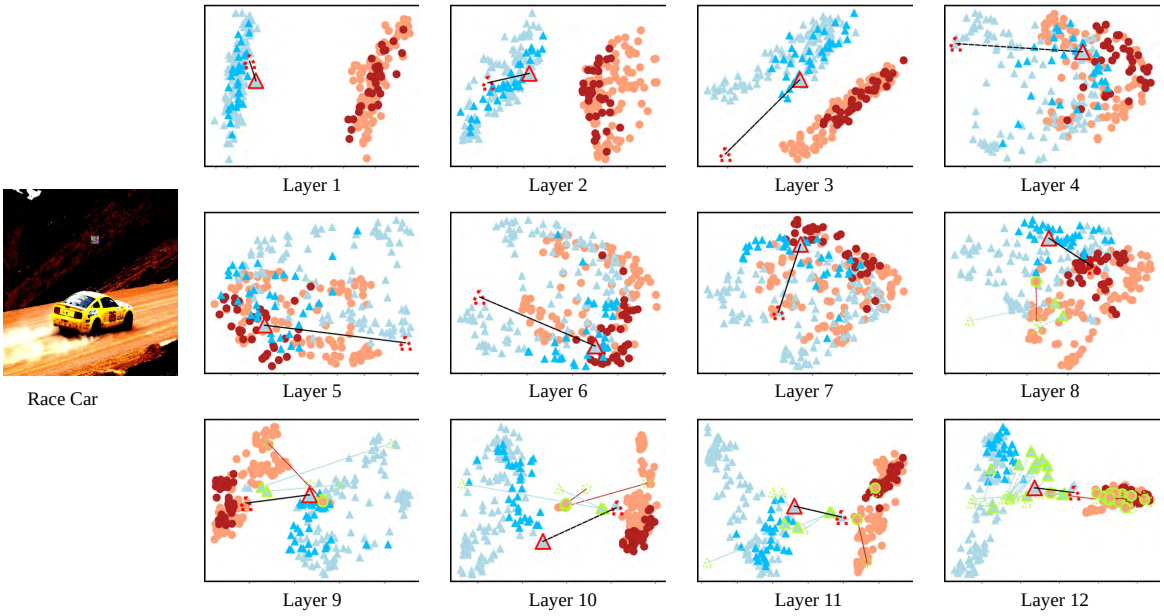


Figure 22: Query/Key Figures of Race Car Example

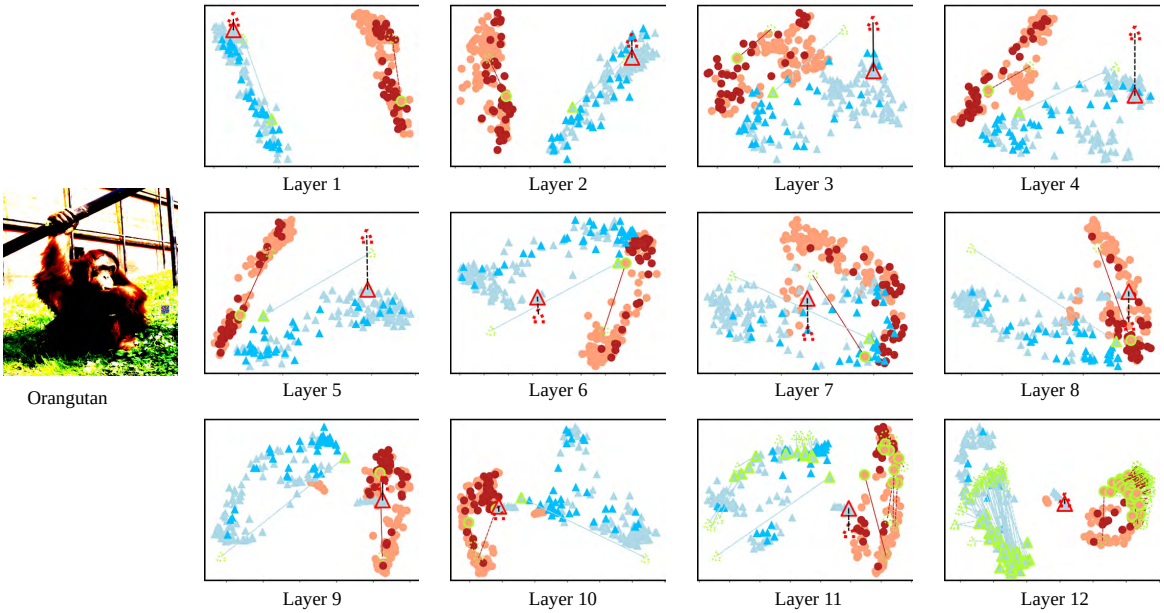


Figure 23: Query/Key Figures of Orangutan Example

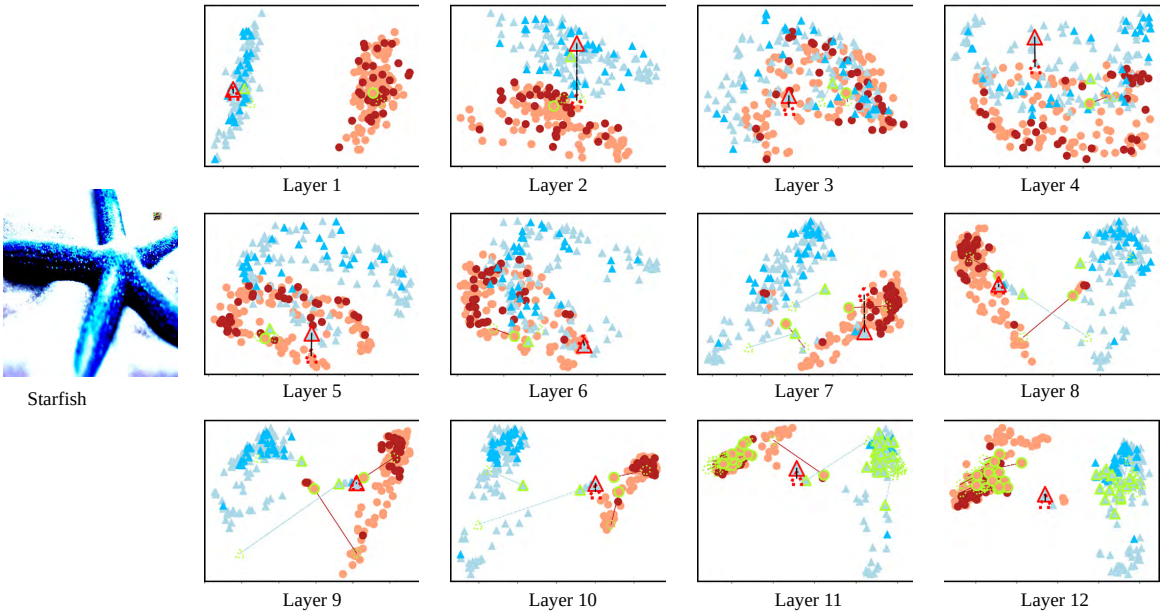


Figure 24: Query/Key Figures of Starfish Example