

# Give Me Your Attention: Dot-Product Attention Considered Harmful for Adversarial Patch Robustness

Giulio Lovisotto<sup>1,2</sup>   Nicole Finnie<sup>2</sup>   Mauricio Munoz<sup>2</sup>   Chaithanya Kumar Mummadi<sup>2,3</sup>  
Jan Hendrik Metzen<sup>2</sup>

<sup>1</sup>University of Oxford   <sup>2</sup>Bosch Center for Artificial Intelligence   <sup>3</sup>University of Freiburg

giulio.lovisotto@cs.ox.ac.uk,  
{Nicole.Finnie, AndresMauricio.MunozDelgado, ChaithanyaKumar.Mummadi, JanHendrik.Metzen}@de.bosch.com

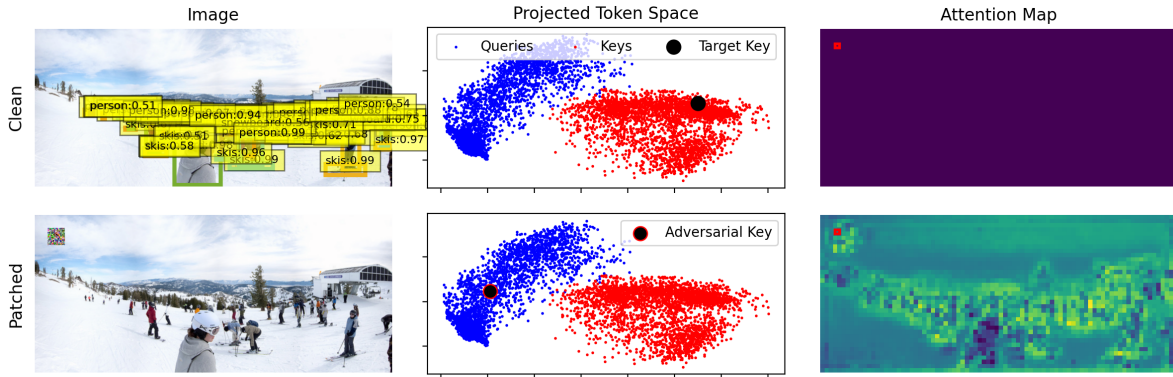


Figure 1. Comparison of clean and adversarially patched input for DETR [8]. The patch shifts a targeted key token towards the cluster of query tokens (middle column). For dot-product attention, this effectively directs the attention of all queries to the malicious token and prevents the model from detecting the remaining objects. The right column compares queries’ attention weights to the target key, marked by a red box, between clean and patched inputs and highlights large attention weights drawn by this adversarial key.

## Abstract

Neural architectures based on attention such as vision transformers are revolutionizing image recognition. Their main benefit is that attention allows reasoning about all parts of a scene jointly. In this paper, we show how the global reasoning of (scaled) dot-product attention can be the source of a major vulnerability when confronted with adversarial patch attacks. We provide a theoretical understanding of this vulnerability and relate it to an adversary’s ability to misdirect the attention of all queries to a single key token under the control of the adversarial patch. We propose novel adversarial objectives for crafting adversarial patches which target this vulnerability explicitly. We show the effectiveness of the proposed patch attacks on popular image classification (ViTs and DeITs) and object detection models (DETR). We find that adversarial patches occupying 0.5% of the input can lead to robust accuracies as low as 0% for ViT on ImageNet, and reduce the mAP of DETR on MS COCO to less than 3%.

## 1. Introduction

The attention mechanism plays a prominent role in recent success of transformers for different language and image processing tasks. Recent breakthroughs in image recognition using vision transformers [12, 23, 37] have inspired different architectures’ design for tackling tasks such as object detection [4, 8], semantic segmentation [35, 41, 46, 48], image synthesis [13, 17, 40], video understanding [3, 6, 15, 15, 19, 24, 30, 36, 49], and low-level vision tasks [9, 21, 42, 47]. These different transformers use the dot-product attention mechanism as an integral component of their architectural design to model global interactions among different input or feature patches. Understanding robustness of these dot-product attention-based networks against adversarial attacks targeting security-critical vulnerabilities is important for their deployment into real-world applications.

The increasing interest of research in transformers across vision tasks has motivated several recent works [2, 5, 7, 14,

16,27–29,34,43] to study their robustness against adversarial attacks. Some prior works [2, 5, 28, 34] have hypothesized that transformers are more robust than convolutional neural networks (CNNs) against these attacks. On the other hand, [14, 16, 27, 43] have shown that vision transformers are not an exception and are also prone to adversarial attacks. In particular, [43] shows that an adversarial attack can be tailored to transformers to achieve high adversarial transferability. These findings indicate that robustness evaluation protocols (attacks) designed for CNNs might be suboptimal for transformers. In the same line of work, we identify a principled vulnerability in the widely-used dot-product attention in transformers that can often be exploited by image-based adversarial patch attacks.

Dot-product attention computes the dot-product similarity of a query token with all key tokens, which is later normalized using the softmax operator to obtain per token attention weights. These attention weights are then multiplied with value tokens to control the value token’s contribution in the attention block. Gradient-based adversarial attacks backpropagate gradients through all the components in the architecture including the attention weights. We observe that on pretrained vision transformers, because of the softmax, the gradient flow through the attention weights is often much smaller than the flow through the value token (refer to Sec. 4.1). Consequently, gradient-based attacks with a standard adversarial objective are biased towards focusing on adversarial effects propagated through the value token and introduce little or no adversarial effect on the attention weights, thus limiting the attack’s potential.

Our work aims to adversarially affect the attention weights, even if those operate in a saturated regime of softmax where gradient-based adversarial attacks are impaired. Specifically, we propose losses that support the adversary in misdirecting the attention of most queries to a key token that corresponds to the adversarial patch, i.e., to increase the attention weights of the queries to the targeted key token. We further study necessary conditions (refer to Sec. 4.2) for a successful attack that misdirects attention weights: both (a) having projection matrices with large singular values and (b) having higher embedding dimensions, allows amplifying the effect of perturbations in a single token, and thus changing the embedding of a single key considerably. Moreover, (c) having less centered inputs (larger absolute value of input mean) to the dot-product attention results in distinct clusters of queries and keys that are distant from each other. Under this condition, moving a single key closer to the query cluster center can make the key most similar to the majority of queries simultaneously (see Figure 1).

We propose a family of adversarial losses and attacks called *Attention-Fool* that directly acts on the dot-product outputs (pre-softmax dot-product similarities). These losses optimize the adversarial patch to maximize the dot-product

similarity of all the queries to a desired key (typically the key whose token corresponds to the input region with adversarial patch). This approach maximizes the number of queries that attend to the targeted key in an attention head, please refer to Figure 2 for an illustration. We apply these losses at multiple attention heads and layers in transformers to misdirect the model’s attention from the image content to the adversarial patch in order to encourage wrong predictions. We show that our *Attention-Fool* adversarial losses improve gradient-based attacks against different vision transformers ViTs [12] and DeiT [37] for image classification and also significantly improve the attack’s effective on the object detection model DETR [8].

Our contributions can be summarized as follows: We

- Identify the necessary conditions for the existence of a vulnerability in dot-product attention layers weights, see Section 4.2.
- Provide reasons why this vulnerability may not be fully exploited by vanilla gradient-based adversarial attacks, see Section 4.1.
- Introduce Attention-Fool, a new family of losses which defines losses directly on the attention layers dot-product similarities, see Section 5.
- Show that transformers for image classification and object detection are highly sensitive to small adversarial patches, see Section 6.

## 2. Related Work

Recent successes of Vision Transformers (ViTs) [12] have inspired several works [2, 5, 14, 16, 27–29, 34, 43] that study their robustness against adversarial attacks. While some works [2, 5, 28, 28, 34] have hypothesized that ViTs are more robust than CNNs under different white- and black-box transfer attack settings (including universal adversarial patches), others [7, 27, 29, 43] have claimed that ViTs are at least as vulnerable as CNNs. In particular, [2] attributed ViT’s presumed robustness to its ability to capture global features. [5, 34] analyzed that ViTs rely on low-frequency features that are robust to adversarial perturbations. [34] have stated that ViTs have better certified robustness than CNNs and have further suggested that adversarially trained ViTs have comparable robustness to their CNN counterparts. However, they observed catastrophic overfitting for ViTs when using fast adversarial training [45], suggesting the need for improvements in adversarial training.

On the other hand, [7] have claimed that ViTs pretrained on larger datasets are at least as vulnerable as CNN counterparts. [27, 29, 43] suggested that adversarial transferability between ViTs and from ViTs to CNNs can be improved by carefully tailoring adversarial attacks to the transformer architecture. [27] explored ensembling of CNNs and ViT models to improve attack transferability. The authors of

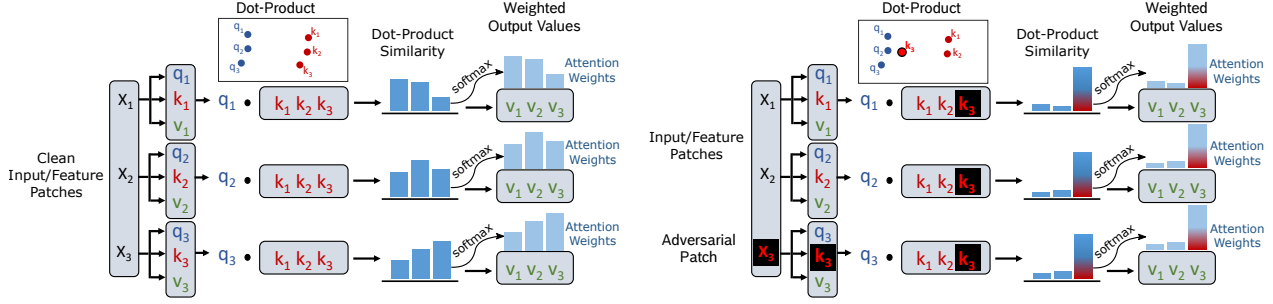


Figure 2. Example of dot-product (self-)attention mechanism for clean (left) and adversarial patch attack (right) settings. Here,  $q$ ,  $k$ , and  $v$  stand for projected queries, keys, and value tokens of input features. Left: dot-product attention computes dot-product similarities of a query with all keys, which is later normalized using softmax to obtain per token attention weights. These are multiplied with value tokens to control their contributions in an attention block. Right: *Attention-Fool* losses optimize the adversarial patch in input at  $X_3$  to maximize dot-product similarity of all the queries to the key  $k_3$  (marked in red/black), which corresponds to moving  $k_3$  closer to the queries cluster. The increase in dot-product similarity of queries with  $k_3$  misdirects the model’s attention from image content to adversarial patch.

[29] proposed a self-ensemble technique: split a single ViT model into an ensemble of networks to improve the transferability. [43] proposed an attack that skips attention gradients to generate highly transferable adversarial perturbations.

In this work, we aim to understand robustness of the widely-used dot-product attention in transformers and expose its vulnerability to adversarial patch attacks. These are constructed by tailoring adversarial objectives to specifically fool the dot-product attention mechanism. Concurrent to our work, [14, 16] also fool the attention mechanism in transformers using image patches. [16] uses a conventional adversarial patch attack to mislead the model attention to the perturbed patch to promote wrong predictions. They also show that the adversarial patch generalizes to different positions in the image. [14] optimizes the adversarial patch to increase its attention weights (post-softmax) from all other patches to attack the model. We discuss limitations of optimizing post-softmax attention weights as in [14] (refer to Section 4.1) and propose to optimize directly the pre-softmax dot-product similarities of query and keys to draw the attention to the adversarial patch.

Besides optimizing the content of the patch, there exist also methods for optimizing the location of the patch in the input. Joshi et al. [18] select the location of adversarial patches based on token saliency. Fu et al. [14] selects a salient image patch to contain the adversarial patch, based on how much attention the image patch draws in the clean image. Our work abstracts from selecting the patch location and focuses on the loss for the optimization of the patch’s content. Any patch location-selection method is orthogonal and could be combined with our Attention Fool-losses.

This work focuses on dot-product attention, widely used in transformers. We leave investigation of vulnerabilities of other attention mechanisms [20, 39, 50] to future work.

### 3. Preliminaries

We introduce the objective for patch robustness evaluation, summarize an optimization algorithm for finding adversarial patches, and recap scaled dot-product attention.

**Generic Objective Formulation.** Given a (normalized) image  $x \in [0, 1]^{3 \times h \times w}$  and associated label  $y$ , we craft an adversarial patch  $p \in [0, 1]^{3 \times p_h \times p_w}$  with  $p_h \ll h, p_w \ll w$  that maximizes the following objective:

$$\arg \max_p \mathcal{L}(f(\mathcal{F}(x, p, L)), y), \quad (1)$$

with  $L$  specifying the location of the patch  $p$  within the larger image  $x$ ,  $\mathcal{F}$  a function to apply the patch onto the image (i.e., overwriting an input region for a given size), and  $f$  being the target model. For classification tasks, we are interested in the 0-1 loss  $\mathcal{L}(x, y) = \mathcal{L}_{0,1}(x, y) = \begin{cases} 0 & x = y \\ 1 & x \neq y \end{cases}$ , which corresponds to finding patches that maximize misclassifications. We note that the constraint  $p \in [0, 1]^{3 \times p_h \times p_w}$  can be rewritten as  $\|p - 0.5\|_\infty \leq 0.5$ .

**Threat Model.** We focus on a white-box threat model, where an adversary has access to the model’s internals (this includes intermediate network layers outputs). As in [10, 14], we do not consider the imperceptibility of the patch to be a requirement. We also focus on a single-patch fixed-location threat model ( $L$  in Equation 1 is fixed a priori). Note that methods for choosing the patch location [14, 18] could be combined with the proposed approach.

**Optimization Algorithm.** For the patch optimization, throughout the paper, we use Projected Gradient Descent (PGD) [26] for  $\ell_\infty$ -norm bounded perturbations:

$$p^{t+1} = p^t + \alpha \cdot \text{sgn}(\nabla_p \mathcal{L}(f(\mathcal{F}(x, p, L)), y)). \quad (2)$$

We initialize  $p^0$  uniform randomly from  $[0, 1]^{3 \times p_h \times p_w}$ . Since the 0-1 loss  $\mathcal{L}_{0,1}$  is piecewise-constant, it is not suited for gradient-based optimization. Accordingly, a surrogate loss such as the cross-entropy  $\mathcal{L} = \mathcal{L}_{ce}$  is used typically. However, we propose alternative losses in Section 5.

**Dot-Product Attention.** In its basic form, dot-product attention [25, 38] computes, for every query, attention weights as the dot-product of the query to all keys. The softmax function is then applied over the key dimension. These attention weights are then multiplied by the values:

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^\top)V. \quad (3)$$

Here,  $Q \in \mathbb{R}^{n \times d_{\text{model}}}$ ,  $K \in \mathbb{R}^{n \times d_{\text{model}}}$ , and  $V \in \mathbb{R}^{n \times d_{\text{model}}}$  are the matrices of  $n$  queries, keys, and values, respectively. According to Vaswani et al. [38], for large values of  $d_{\text{model}}$ , the dot-product between queries and keys may grow large in magnitude. This has the effect of pushing the softmax function into the saturated regime, where it has extremely small gradients. This is due to the exponentiation of individual query-key dot-products in the softmax function. Because this can be harmful for training, they introduce scaled dot-product attention, where  $QK^\top$  is scaled by  $1/\sqrt{d_{\text{model}}}$ .

In practice, using  $H > 1$  attention heads by linearly projecting queries, keys, and values  $H$  times to  $d_k$ ,  $d_k$ , and  $d_v$  dimensions was found to be beneficial [38]. The output of the  $h$ -th attention head (AH) becomes:

$$\text{AH}_h(Q, K, V) = \text{softmax}\left(\frac{QW_Q^h(KW_K^h)^\top}{\sqrt{d_k}}\right)VW_V^h, \quad (4)$$

where  $W_Q^h \in \mathbb{R}^{d_{\text{model}} \times d_k}$ ,  $W_K^h \in \mathbb{R}^{d_{\text{model}} \times d_k}$ ,  $W_V^h \in \mathbb{R}^{d_{\text{model}} \times d_v}$  are (learned) projection matrices. The outputs of individual attention heads are concatenated and multiplied by another learned projection matrix  $W_O \in \mathbb{R}^{Hd_v \times d_{\text{model}}}$ .

A special case is self-attention with  $Q = K = V \in \mathbb{R}^{n \times d_{\text{model}}}$ , which is typically used in encoder layers of image recognition models. We define the attention weights of the  $h$ -th head on  $X$  via  $A_h(X) = \text{softmax}\left(\frac{XW_Q^h(XW_K^h)^\top}{\sqrt{d_k}}\right) \in \mathbb{R}^{n \times n}$ . The  $h$ -th self-attention head becomes:

$$\text{SelfAH}_h(X) = A_h(X)XW_V^h \quad (5)$$

## 4. Robustness of Dot-Product Attention

In this section, we first discuss why (scaled) dot-product self-attention is challenging for gradient-based adversarial attacks such as PGD. We then provide an example of an adversarial vulnerability in the attention weights themselves.

### 4.1. Gradient of Dot-Product Self-Attention

For computing  $\nabla_p \mathcal{L}(\mathcal{F}(x, p, L), y)$  in Eq. 2, it is necessary to backpropagate through the entire model. This

Table 1. Median of  $|(\nabla_X A_h(X))X|/(A_h(X)1_X)|$  over tokens and heads on a random natural image for models on 6 encoder layers. A larger version of this table is in Section A2 in Appendix.

	ViT-T	ViT-S	ViT-B	DeiT-T	DeiT-S	DETR
Layer 1	0.048	0.059	0.064	0.044	0.049	0.188
Layer 2	0.032	0.027	0.032	0.042	0.033	0.040
Layer 3	0.035	0.032	0.028	0.034	0.028	0.034
Layer 4	0.029	0.035	0.036	0.035	0.029	0.058
Layer 5	0.041	0.032	0.048	0.066	0.036	0.074
Layer 6	0.030	0.029	0.036	0.040	0.034	0.112

requires the gradient  $\nabla_X \text{SelfAH}_h(X)$  for every attention layer and head  $h$ . With the product-rule we obtain:

$$\nabla_X \text{SelfAH}_h(X) = [(\nabla_X A_h(X))X + A_h(X)1_X]W_V^h,$$

where  $1_X$  is a matrix of ones with the same shape as  $X$ .

An important property of the gradient  $\nabla_X \text{SelfAH}_h(X)$  is accordingly the element-wise ratio  $|(\nabla_X A_h(X))X|/|A_h(X)1_X|$ . We summarize the median value of this ratio over tokens and heads in Table 1 for different models and layers. As seen, the typical regime (that is, for  $> 50\%$  of the cases) is that  $(\nabla_X A_h(X))X$  is smaller than  $A_h(X)1_X$  by a factor of  $\approx 20$  for ViTs [12], DeiT [37], and for the inner encoder layers of DETR [8]. In this setting, one can approximate  $\nabla_X \text{SelfAH}_h(X) \approx (A_h(X)1_X)W_V^h$ , that is: the gradient considers the attention weights  $A_h(X)$  as effectively constant. Accordingly, gradient-based attacks such as PGD based on an end-to-end loss such as  $\mathcal{L}_{ce}$  would be biased towards focusing on adversarial effects in  $X$  that can be propagated (linearly) via the values  $V = XW_V^h$  in self-attention, while effectively ignoring potentially adverse (and non-linear) effects of  $X$  propagated via the attention weights  $A_h(X)$ . We note that also later stages of model training with gradient-based optimizers can be negatively affected by this property of dot-product attention. Studying this in more detail is left to future work.

### 4.2. Robustness of Dot-Product Attention Weights

We now study to which extent attention weights  $A_h(X)$  of dot-product attention can be affected by an adversarial patch attack. For this, we use a controlled setting with normally distributed  $X$ , where each feature has mean  $\mu$  and variance 1:  $X_j \sim \mathcal{N}(\mu \cdot \mathbf{1}, \mathbf{1})$ . Moreover, we choose  $d_k = d_{\text{model}}$  and diagonal  $W_Q = -w \cdot \mathbb{I}_{d_k}$  and  $W_K = w \cdot \mathbb{I}_{d_k}$ , that is  $W_Q$  and  $W_K$  having both scale  $w$  but opposite signs.

We study a simple threat model: the adversary can only modify  $X_0$ , the first of  $n$  entries of  $X$  ( $X$  can be considered as embedding of patches and  $X_0$  as corresponding to the embedded adversarial patch), with the constraint  $\|X_0^{adv} - X_0\|_\infty \leq \epsilon$ . Moreover, the adversaries goal is to achieve  $\left[\frac{1}{n} \sum_j (A_h(X^{adv})_{0j} \geq 0.99)\right] > 0.95$ , that is: at



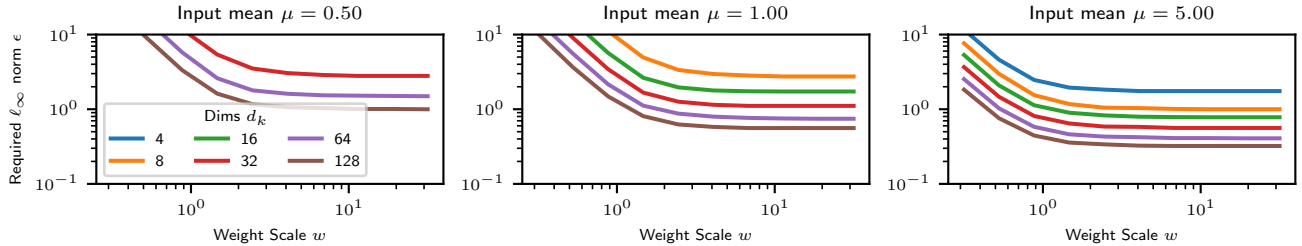


Figure 3. Minimum  $\ell_\infty$  norm perturbation  $\epsilon$  required for reaching the attacker’s goal (i.e., forcing queries to attend to the first key) on the controlled setting from Section 4.2. Increased weight scale  $w$ , higher embedding dimension  $d_k$ , and larger input mean norm  $\mu$  simplify the attack: as the three quantities increase, the  $\epsilon$  perturbation required to fulfil the goal decreases.

Table 2. Largest singular value of the projection weight matrices  $W_Q(W_K)^\top$  for randomly initialized and trained models.

	ViT/DeiT-B random	ViT-B trained	DeiT-B trained	DETR encoder	
				random	trained
Layer 1	0.80	60.96	175.28	1.27	7.93
Layer 2	0.79	22.67	65.86	1.29	10.61
Layer 3	0.80	11.76	54.19	1.25	15.35
Layer 4	0.79	5.88	44.95	1.27	45.26
Layer 5	0.79	4.91	29.36	1.28	49.82
Layer 6	1.21	4.83	28.72	1.26	29.30

least 95% of the queries need to attend to the first key with attention weights greater or equal 0.99. By design, setting  $X_0^{adv} = X_0 - \epsilon$  is a strong attack for  $\mu > 0$  because  $-\mathbf{1} \cdot \epsilon$  corresponds to the direction of  $(W_Q - W_K)(\mu \cdot \mathbf{1})$ , the difference of projected query and key mean.

We study how  $\epsilon$  needs to be chosen as a function of  $\mu$ ,  $w$ , and  $d_k$  for a successful attack with the above attack and threat model. Results are shown in Figure 3. In general, one can observe that a successful attack with a smaller perturbation amount  $\epsilon$  requires: (a) increasing the scale  $w$  of the projection matrices, (b) higher embedding dimensions  $d_k$ , and (c) less centered inputs  $X$  (larger  $|\mu|$ ). The findings (a) and (b) can be attributed to higher dimensions and larger weights allow the effect of minor perturbations in each input dimension to be amplified. Upon closer inspection (see Section B in Appendix), we can attribute finding (c) to a separation of projected keys and queries into distinct clusters for less centered inputs: all queries are close to each other, all keys are close to each other, but query-key pairs are all distant from each other. In this case, a single key can be made to be the most similar to each query, just by moving this key in the direction of the query cluster (see Figure 1 for a real data illustration).

In which regime does dot-product attention typically operate in trained image transformers? For many architectures,  $d_k$  is relatively large by design. Moreover, the input of dot-product attention is typically not centered (zero-mean) due to enabled affine transformations within the normalization layers before self-attention. Many implementa-

tions, e.g. [31], also use affine rather than linear query/key projections in the heads, where different biases in the projection of key and queries will have a similar effect as non-centered inputs. Lastly, we also observe that for many vision transformers, the product of projection weight matrices  $W_Q(W_K)^\top$  has very large maximal singular value at convergence, compared to the randomly initialized projection weight matrices (see Table 2). Large singular values can be considered to be analogous to the “large weight scales” in our controlled setting. We thus expect that dot-product attention of trained vision transformers typically operates in a setting where the attention weights  $A_h(X)$  can be a source of vulnerability to patch attacks, but gradient-based attacks such as PGD are biased towards ignoring this vulnerability (as discussed in Sec. 4.1).

## 5. Attention-Fool

We have discussed that attention weights  $A_h(X)$  can be largely affected by adversarial patches in principle but we have also shown that gradient-based attacks are impaired in exploiting this vulnerability. In order to encourage fooling of attention weights, we introduce a family of losses that are defined directly on the pre-softmax attention logits  $B_h(X) = \frac{XW_Q^h(XW_K^h)^\top}{\sqrt{d_k}}$ . We denote attacks based on these losses as *Attention-Fool*. We note that Attention-Fool losses are always maximized by the attacker, regardless of whether an attack is targeted or untargeted. Moreover, while this paper focuses on self-attention, it is straightforward to extend these losses to cross-attention.

**$\mathcal{L}_{kq}^{hl}$ : Attacking a single attention head.** We first focus on adversarially modifying attention weights of a specific head  $h$  in a specific self-attention layer  $l$ . We propose a loss  $\mathcal{L}_{kq}^{hl}$  that aims at maximizing the query attention to the  $i^*$ -th projected key, that is: maximize the number of queries that devote the majority of their attention to this key. Naturally,  $i^*$  is chosen such that its token corresponds to the region in the input where the adversarial patch has been placed.

We denote projected queries by  $P_Q^{hl} = X^{hl}W_Q^{hl} \in$

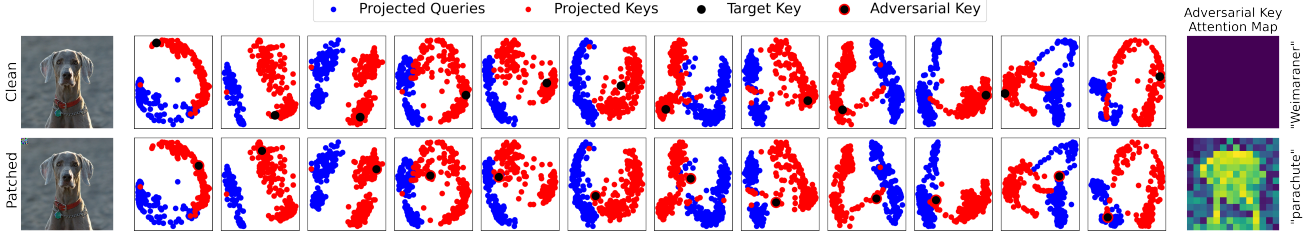


Figure 4. Embedded projected key and query tokens for clean and patched input images on each of 12 ViT layers of the attention head which showed the largest attention change. We design Attention-Fool to target all layers and heads at once. The last column reports the attention map weights of the adversarial key on the last layer – showing that the key draws a large amount of attention from queries.

$\mathbb{R}^{n \times d_k}$  and projected keys by  $P_K^{hl} = X^{hl} W_K^{hl} \in \mathbb{R}^{n \times d_k}$ .

We have  $B^{hl} = \frac{P_Q^{hl}(P_K^{hl})^\top}{\sqrt{d_k}} \in \mathbb{R}^{n \times n}$ , where the first dimension indexes queries and the second one keys. Each element of  $B^{hl}$  quantifies the dot-product similarity between a key and a query in the respective attention head  $h$  and layer  $l$ . We now set  $\mathcal{L}_{kq}^{hl} = \frac{1}{n} \sum_j B_{ji^*}^{hl}$ , where  $j$  indexes the queries. Maximizing  $\mathcal{L}_{kq}^{hl}$  thus corresponds to maximizing average dot-product similarity between queries and the target key.

**$\mathcal{L}_{kq}$ : Attacking all attention layers and heads simultaneously.** In the previous paragraph, we have introduced a loss that targets a single head and layer. However, in general fooling a single head might not be sufficient and it is also a priori unclear which head would be the most vulnerable one. The same applies to the choice of the layer, when there are multiple layers using dot-product attention. Because of this, we propose applying the loss to all layers and heads simultaneously. However, simply averaging  $\mathcal{L}_{kq}^{hl}$  over all heads and layers might not be optimal because it may favour many smaller head-wise changes overall rather than successfully fooling a subset of heads and layers to a larger extent. Because of this, we will utilize a smooth maximum  $\text{smax}(x) = \log \sum_i e^{x_i}$  over  $\mathcal{L}^{hl}$ . Specifically, we define  $\mathcal{L}_{kq}^l = \log \sum_h e^{\mathcal{L}_{kq}^{hl}} \quad \forall l$  and  $\mathcal{L}_{kq} = \log \sum_l e^{\mathcal{L}_{kq}^l}$ . We empirically compare the choice of this smooth maximum over mean and hard maximum in Section C.1 and the choice of  $l$  in  $\mathcal{L}_{kq}^l$  in Section C.2. We visualize  $\mathcal{L}_{kq}$ 's effect on each network layer in Figure 4.

In order to make the losses of different heads and layers commensurable, we also propose to normalize scales of projected keys and queries via  $\bar{P}_Q^{hl} = P_Q^{hl} / \frac{1}{n} \|P_Q^{hl}\|_{1,2}$  and  $\bar{P}_K^{hl} = P_K^{hl} / \frac{1}{n} \|P_K^{hl}\|_{1,2}$ , where  $\|X\|_{1,2} = \sum_i \sqrt{\sum_j X_{ij}^2}$  is the  $L_{1,2}$  norm. Note that the normalization is applied per individual head and makes the average  $\ell_2$  norm of queries and keys equal to 1.  $B^{hl}$  is then computed based on  $\bar{P}_Q^{hl}$  and  $\bar{P}_K^{hl}$ , making the losses  $\mathcal{L}_{kq}^{hl}$  commensurable. We empirically evaluate the effect of this normalization in Section C.3

of the supplementary material.

**$\mathcal{L}_{kq^*}$ : Targeting a special class token.** The  $\mathcal{L}_{kq}$  loss treats all queries equally and aims at misdirecting the attention of the majority of queries towards the adversarial key token. However, for many architectures not all queries are created equally, for instance for ViTs [12] and DeiT [37], there exists a special class token, which is supposed to accumulate class evidence over the layers. We propose a version of the  $\mathcal{L}_{kq}$  loss that specifically targets the attention of a certain query, for instance the one corresponding to the class token. Let  $j^*$  be the index of the query that shall be targeted. We define  $\mathcal{L}_{kq^*}^{hl} = B_{j^*i^*}^{hl}$  and generalize to  $\mathcal{L}_{kq^*}$  as above using smooth maximum over heads and layers.

In the following experiments, we will use combinations of  $\mathcal{L}_{kq}$  or  $\mathcal{L}_{kq^*}$  with standard cross-entropy loss  $\mathcal{L}_{ce}$  for the adversarial patch optimization of Equation 1. We denote the resulting set of attacks leveraging the weakness in dot-product attention as *Attention-Fool*. Note that Attention-Fool is different from concurrent work Patch-Fool [14] that defines a loss on the post-softmax attention weights, averaged of heads  $h$  and queries  $j$ :  $\mathcal{L}_{PF} = \sum_h \sum_j A_h(X)_j$ . In particular, Attention-Fool (in contrast to Patch-Fool) is not affected by the small gradient issue described in Section 4.1 since it is defined on pre-softmax attention weights.

## 6. Evaluation of Attention-Fool

We evaluate the robustness of different vision transformers against adversarial patches generated using our proposed Attention-Fool adversarial losses. We first investigate the robustness of ViT [12] and the improved DeiT [37] in Section 6.2. Then we show how Attention-Fool generalizes to DETR [8] in Section 6.3, which uses a hybrid CNN plus Transformer architecture to perform object detection.

### 6.1. Evaluation Setup

We use PGD as the optimizer to solve Equation 1. We set PGD's initial step size  $\alpha^0 = 8/255$  and schedule it with a cosine decay:  $\alpha^{(t+1)} = \alpha^{(0)} \frac{1}{2} (1 + \cos(\pi \frac{t}{N}))$ . Addition-

Table 3. Robust accuracies (%) under adversarial patch attacks on 1000 ImageNet images. Clean performance, as well robust accuracy against Patch-Fool loss [14] and Patch-RS [10] are shown as baselines. All rows in the bottom block are computed using PGD<sup>250</sup> with step size  $\alpha=8/255$  and crossentropy loss  $\mathcal{L}_{ce}$ . An Attention-Fool term is added to the loss (either  $\mathcal{L}_{kq}$  or  $\mathcal{L}_{kq*}$ ) and optionally momentum is added to PGD. Numbers in parenthesis report the improvement or degradation in robust accuracy w.r.t. the  $\mathcal{L}_{ce}$  baselines. All models use  $224 \times 224$  resolution unless marked by “384” in the model name, then they use  $384 \times 384$ . Patch size is always  $16 \times 16$ .

	ResNet50	ViT-T	ViT-B	ViT-B-384	DeiT-T	DeiT-B	DeiT-B-384
Clean	80.6	73.5	85.0	86.4	69.4	82.0	82.0
Patch-Fool loss [14]	-	0.3	17.5	47.8	3.2	52.9	68.2
Patch-RS [10]	70.8	18.6	49.8	65.9	44.0	57.3	71.2
PGD <sup>250</sup> with $\mathcal{L}_{ce}$	55.1	0.1	13.5	31.2	19.8	36.0	58.8
+ $\mathcal{L}_{kq}$	-	0.5 (+0.4)	5.0 (-8.5)	18.0 (-13.2)	13.1 (-6.7)	35.5 (-0.5)	55.1 (-3.7)
+ $\mathcal{L}_{kq*}$	-	0.3 (+0.2)	2.6 (-10.9)	13.0 (-18.2)	11.7 (-8.1)	33.7 (-2.3)	57.2 (-1.6)
+ Momentum	49.0	0.0	3.1	13.2	1.5	16.8	41.7
+ $\mathcal{L}_{kq}$	-	0.0 (-0.0)	0.1 (-3.0)	2.5 (-10.7)	0.0 (-1.5)	19.3 (+2.5)	39.8 (-1.9)
+ $\mathcal{L}_{kq*}$	-	0.0 (-0.0)	0.1 (-3.0)	1.9 (-11.3)	0.0 (-1.5)	13.1 (-3.7)	40.6 (-1.1)

ally, we add a stronger baseline to the adversarial patch robustness evaluation, we use an improved version of PGD by adding normalized-momentum in the optimization [11]:

$$m^{(t)} = \beta m^{(t-1)} + (1 - \beta) \nabla_p^t / \|\nabla_p^t\|_2,$$

and use  $m^{(t)}$  instead of the gradient  $\nabla_p^t$  in PGD.

## 6.2. Attention-Fool on Vision Transformer

We use the new Attention-Fool attack described in Section 5 to compare the effectiveness of adversarial patches optimized with the new losses  $\mathcal{L}_{kq}$  and  $\mathcal{L}_{kq*}$  with adversarial patches optimized using cross-entropy loss  $\mathcal{L}_{ce}$ . We use 1,000 images from the ImageNet 2012 [33] dataset and perform an untargeted attack, where  $y$  in Eq. 1 is the image ground truth. We also report the results on a targeted attack in Section D in Appendix. For the experiment, we place an adversarial patch of  $16 \times 16$  pixels in the top left corner, we investigate different location and sizes in Section C.4 and C.5, respectively. For the evaluation, we use pre-trained ViT models from timm [44]. We choose models which split input images into  $16 \times 16$  image sub-patches; this way the adversarial patch occupies exactly one of the image sub-patches. We consider models with input image resolution of  $224 \times 224$  or  $384 \times 384$ ; the adversarial patch covers 0.5% and 0.17% of the input, respectively. For comparison, we also perform the attack on a CNN, ResNet50, using the same setup but optimizing adversarial patches with  $\mathcal{L}_{ce}$  only. We report the resulting robust accuracies in Table 3. We find that Attention-Fool’s  $\mathcal{L}_{kq}$  and  $\mathcal{L}_{kq*}$  improve on the baseline cross-entropy on most models regardless of momentum of PGD. While Table 3 shows that  $\mathcal{L}_{kq}$  already decreases robust accuracies across most settings, using the architecture-specific Attention-Fool loss-variant  $\mathcal{L}_{kq*}$  brings more stable robust accuracy down among all models, often with large performance gains over

cross-entropy alone. Attention-Fool can bring accuracies down to 0% in ViT-T and DeiT-T models, and as low as 1.90% even in ViT-B-384, where the patch occupies only 0.17% of the input image. Notably, in contrast to prior work (see Sec. 2) we find all ViT/DeiT variants to be considerably *less* robust than a ResNet50 when the patch is placed in the top left corner, whereas placing the patch in the center could have a larger effect on a CNN.

We include additional comparisons with the “Attention-Aware Loss” of Patch-Fool (Section 4.4 in [14]). For a controlled comparison with Patch-Fool, we perform two adjustments: (i) we disregard the saliency-based selection of the patch location (Section 4.3 in [14]) and (ii) we replace Adam with PGD (momentum = 0.9). This abstracts from attack specifics and allows us to compare losses directly. The results show that Patch-Fool’s Attention-Aware loss underperforms the Attention Fool losses; confirming the limitations of optimizing post-softmax attention weights (discussed in Sec. 4.1). Moreover, we compare to Patch-RS [10], a random-search based black-box patch attack. The inferior results of Patch-RS indicate that gradient-free attacks are no viable alternative for exploiting the existent vulnerabilities in attention weights.

## 6.3. Attention-Fool on DETR

Since Attention-Fool targets dot-product attention layers, it can be adapted to target various tasks and architectures that use this attention. Here, we apply Attention-Fool to object detection with DETR [8], which combines a CNN backbone with a Transformer encoder-decoder.

**Attention-Fool Configuration.** While  $\mathcal{L}_{kq*}$  showed superior results for ViT/DeiT in Section 6.2, it is not applicable to DETR due to the absence of a class token in DETR. In contrast, we use  $\mathcal{L}_{kq}^{(1)}$ : a loss targeting all queries and heads but only the first transformer encoder layer ( $l = 1$ ) rather than all at once. We focus on  $l = 1$  because we hypothe-

Table 4. Mean Average Precisions (mAP) in presence of adversarial patches computed with  $\mathcal{L}_{ce}$  and patches computed with Attention Fool  $\mathcal{L}_{kq}^{(1)}$ , on DETR models. The Baseline Clean mAP is obtained on the clean set of images. Three different patch sizes and four different DETR models are considered, based either on R50 or R101 backbone and with and without dilation DC5.

	R50	DC5-R50	R101	DC5-R101
Clean mAP	53.00	54.25	54.41	56.74
64×64: $\mathcal{L}_{ce}$	34.91	17.67	31.76	34.71
+ $\mathcal{L}_{kq}^{(1)}$	21.03 (-13.88)	7.34 (-10.33)	2.07 (-29.70)	4.19 (-30.52)
$\mathcal{L}_{kq}^{(1)}$ only	29.52 (-5.39)	15.03 (-2.64)	2.05 (-29.71)	10.18 (-24.53)
56×56: $\mathcal{L}_{ce}$	37.74	24.13	32.06	38.28
+ $\mathcal{L}_{kq}^{(1)}$	28.19 (-9.55)	14.68 (-9.45)	3.30 (-28.76)	6.82 (-31.46)
$\mathcal{L}_{kq}^{(1)}$ only	38.14 (+0.40)	20.18 (-3.95)	5.60 (-26.46)	14.27 (-24.01)
48×48: $\mathcal{L}_{ce}$	42.08	27.69	37.89	41.32
+ $\mathcal{L}_{kq}^{(1)}$	34.88 (-7.20)	18.94 (-8.75)	9.80 (-28.09)	17.05 (-24.26)
$\mathcal{L}_{kq}^{(1)}$ only	42.19 (+0.12)	32.60 (+4.91)	12.85 (-25.04)	17.55 (-23.77)

size that the transition from CNN to transformer encoder in DETR is specifically brittle due to large activations from the CNN backbone. For a similar reason, we aggregate losses across attention heads using the maximum function rather than smax used in Section 4.

**Evaluation Setup.** We evaluate a targeted Attention-Fool attack on DETR by choosing the target to be the background class, which effectively forces missed detections (i.e., false negatives). We evaluate four pretrained DETR models from the official repository [1], with backbone either ResNet50 or ResNet101, and with or without a dilation in the ResNet layer 5 convolution (DC5). Note that DC5 models have twice the resolution in the backbone-extracted feature map, which is used as the encoder input. We select 100 images from the MS COCO 2017 validation set [22], and use the default DETR validation image loader which re-scales images to have shortest side 800 pixels. We evaluate Attention-Fool by targeting a single key token – a single unit in the CNN feature map – out of the entire token sequence which is at least 2,500- and 625-tokens long for DC5 and non-DC5 models (with final length depending on the input image resolution). We provide an ablation in Section E in appendix where we attribute the adversarial effect to this token alone. We place the adversarial patch in the image top-left centered around the 80,80 pixel. Accordingly, we target the key token with index 2,2 in non-dilated models and 4,4 in dilated ones – this ensures that the key token receptive field is centered around the adversarial patch location in the image. We record the primary COCO challenge metric, the mean Average Precision (mAP), which averages precision-recall curves scores at various Intersection-over-Union (IoU) thresholds. To not bias the evaluation towards local patch effects, we ignore detection boxes whose inter-

section with the patch box is  $>50\%$  of the detection box area. We do the same for ground truths for a fair comparison between clean and patched inputs.

**Results.** Similarly to the previous section, we use  $\mathcal{L}_{ce}$  as an attack baseline and test the improvement (degradation) in mAP as we combine it with the Attention-Fool loss  $\mathcal{L}_{kq}^{(1)}$ . We test three different patch sizes,  $64\times 64$ ,  $56\times 56$ , and  $48\times 48$ , which occupy  $<0.64\%$ ,  $<0.49\%$  and  $<0.36\%$  of the input image, respectively. Given the higher complexity of DETR (compared to ViT), we increase the number of PGD iterations to 1000 in this experiment. We report the resulting mAPs in Table 4: baseline mAP on clean images, mAP under a targeted attack which uses  $\mathcal{L}_{ce}$ , and the mAP change when we use the combined Attention-Fool  $\mathcal{L}_{ce} + \mathcal{L}_{kq}^{(1)}$  loss. Table 4 shows that the addition of  $\mathcal{L}_{kq}^{(1)}$  reduces the mAP performance across all models and all patch sizes. We find that larger models with DETR101 are more vulnerable to Attention-Fool, where the mAP can be reduced down to 2.07 – this corresponds to suppressing the detection of the vast majority of objects. See Figure 1 for an illustration.

Table 4 also presents a setting where we do not use any loss on the model’s output but solely focus on misdirecting the first encoder layer’s attention (“ $\mathcal{L}_{kq}^{(1)}$  only”). In most cases, this results in considerably lower robust accuracy than using  $\mathcal{L}_{ce}$  directly, indicating that for DETR “fooling attention is all you need” for misclassification.

## 7. Conclusion

We revisited the robustness of transformers for image recognition against patch attacks. We identified properties of the dot-product attention’s gradient that bias vanilla patch attacks to mostly ignore a core vulnerability of these attention weights. This presumably caused prior works to overestimate robustness. We propose Attention-Fool, which directly targets the dot-product attention weights and allows much tighter robustness estimates. In summary, Attention-Fool improves vanilla robustness evaluation across all considered vision transformers, and is able to fool DETR’s global object detection with a tiny remote patch.

**Limitations.** We focused on dot-product attention. Other attention mechanisms [20, 39, 50] are likely not affected to the same degree by the identified weakness. However, since dot-product attention is the predominant attention mechanism in transformers, our approach is broadly applicable.

**Potential negative societal impact.** Adversarial attacks such as ours can be used for benign purposes like reliably evaluating robustness of ML systems as well as malign ones like exploiting weaknesses of these systems. Our research provides the basis for future work on more robust attention mechanisms that can mitigate the identified vulnerability and the resulting potential for negative societal impact.



## References

- [1] DETR: End-to-End Object Detection with Transformers. <https://github.com/facebookresearch/detr>. Accessed: 2021-11-16. 8
- [2] Ahmed Aldahdooh, Wassim Hamidouche, and Olivier Deforges. Reveal of vision transformers robustness against adversarial attacks. *arXiv preprint arXiv:2106.03734*, 2021. 1, 2
- [3] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. *arXiv preprint arXiv:2103.15691*, 2021. 1
- [4] Josh Beal, Eric Kim, Eric Tzeng, Dong Huk Park, Andrew Zhai, and Dmitry Kislyuk. Toward transformer-based object detection, 2020. 1
- [5] Philipp Benz, Soomin Ham, Chaoning Zhang, Adil Karjauv, and In So Kweon. Adversarial robustness comparison of vision transformer and mlp-mixer to cnns. *arXiv preprint arXiv:2110.02797*, 2021. 1, 2
- [6] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? *arXiv preprint arXiv:2102.05095*, 2021. 1
- [7] Srinadh Bhojanapalli, Ayan Chakrabarti, Daniel Glasner, Daliang Li, Thomas Unterthiner, and Andreas Veit. Understanding robustness of transformers for image classification. *arXiv preprint arXiv:2103.14586*, 2021. 1, 2
- [8] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. 1, 2, 4, 6, 7
- [9] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12299–12310, 2021. 1
- [10] Francesco Croce, Maksym Andriushchenko, Naman D Singh, Nicolas Flammarion, and Matthias Hein. Sparse-rs: a versatile framework for query-efficient sparse black-box adversarial attacks. *arXiv preprint arXiv:2006.12834*, 2020. 3, 7
- [11] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 7
- [12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 2, 4, 6
- [13] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12873–12883, 2021. 1
- [14] Yonggan Fu, Shun Yao Zhang, Shang Wu, Cheng Wan, and Yingyan Lin. Patch-fool: Are vision transformers always robust against adversarial perturbations? In *International Conference on Learning Representations*, 2022. 1, 2, 3, 6, 7
- [15] Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2019. 1
- [16] Jindong Gu, Volker Tresp, and Yao Qin. Are vision transformers robust to patch-wise perturbations?, 2022. 1, 2, 3
- [17] Yifan Jiang, Shiyu Chang, and Zhangyang Wang. Transgan: Two transformers can make one strong gan. *arXiv preprint arXiv:2102.07074*, 1(3), 2021. 1
- [18] Ameeya Joshi, Gauri Jagatap, and Chinmay Hegde. Adversarial token attacks on vision transformers. *arXiv preprint arXiv:2110.04337*, 2021. 3
- [19] Sangho Lee, Youngjae Yu, Gunhee Kim, Thomas Breuel, Jan Kautz, and Yale Song. Parameter efficient multimodal transformers for video representation learning. *arXiv preprint arXiv:2012.04124*, 2020. 1
- [20] James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. Fnet: Mixing tokens with fourier transforms. *arXiv preprint arXiv:2105.03824*, 2021. 3, 8
- [21] Jingyun Liang, Jie Zhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1833–1844, 2021. 1
- [22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 8, 2
- [23] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021. 1
- [24] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. *arXiv preprint arXiv:2106.13230*, 2021. 1
- [25] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015. 4
- [26] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 3
- [27] Kaleel Mahmood, Rigel Mahmood, and Marten Van Dijk. On the robustness of vision transformers to adversarial examples. *arXiv preprint arXiv:2104.02610*, 2021. 1, 2
- [28] Muzammal Naseer, Kanchana Ranasinghe, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Intriguing properties of vision transformers. *arXiv preprint arXiv:2105.10497*, 2021. 1, 2
- [29] Muzammal Naseer, Kanchana Ranasinghe, Salman Khan, Fahad Shahbaz Khan, and Fatih Porikli. On improving adversarial transferability of vision transformers. *arXiv preprint arXiv:2106.04169*, 2021. 1, 2, 3

- [30] Daniel Neimark, Omri Bar, Maya Zohar, and Dotan Aselsmann. Video transformer network. *arXiv preprint arXiv:2102.00719*, 2021. 1
- [31] PyTorch. PyTorch multi-head attention implementation, 2021. <https://pytorch.org/docs/1.10.0/generated/torch.nn.MultiheadAttention.html>. 5
- [32] Peter Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20(1):53–65, 1987. 1
- [33] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 7, 2
- [34] Rulin Shao, Zhouxing Shi, Jinfeng Yi, Pin-Yu Chen, and Cho-Jui Hsieh. On the adversarial robustness of visual transformers. *arXiv preprint arXiv:2103.15670*, 2021. 1, 2
- [35] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. *arXiv preprint arXiv:2105.05633*, 2021. 1
- [36] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7464–7473, 2019. 1
- [37] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers; distillation through attention. In *International Conference on Machine Learning*, volume 139, pages 10347–10357, July 2021. 1, 2, 4, 6
- [38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 4
- [39] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020. 3, 8
- [40] Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. Sceneformer: Indoor scene generation with transformers. *arXiv preprint arXiv:2012.09793*, 2020. 1
- [41] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8741–8750, 2021. 1
- [42] Zhendong Wang, Xiaodong Cun, Jianmin Bao, and Jianzhuang Liu. Uformer: A general u-shaped transformer for image restoration. *arXiv preprint arXiv:2106.03106*, 2021. 1
- [43] Zhipeng Wei, Jingjing Chen, Micah Goldblum, Zuxuan Wu, Tom Goldstein, and Yu-Gang Jiang. Towards transferable adversarial attacks on vision transformers. *arXiv preprint arXiv:2109.04176*, 2021. 1, 2, 3
- [44] Ross Wightman. Pytorch image models, 2019. <https://github.com/rwightman/>. 7
- [45] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020. 2
- [46] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers, 2021. 1
- [47] Fuzhi Yang, Huan Yang, Jianlong Fu, Hongtao Lu, and Bain-ing Guo. Learning texture transformer network for image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5791–5800, 2020. 1
- [48] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6881–6890, 2021. 1
- [49] Luowei Zhou, Yingbo Zhou, Jason J Corso, Richard Socher, and Caiming Xiong. End-to-end dense video captioning with masked transformer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8739–8748, 2018. 1
- [50] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020. 3, 8

# Supplementary Material

## A. Gradient of Dot-Product Attention - Results

We report an extended version of Table 1 in Table A2, which includes up to 12 encoder layers (when available) and report the median of the gradient ratio  $|(\nabla_X A_h(X))X|/(A_h(X)1_X)|$ , more specifically mean and its standard error of this median across 100 images. Table A2 shows that gradients tend to flow along the  $A_h(X)1_X$  gradient term, in particular when back-propagating all the way up to the input (as in adversarial patch optimization). We also find that smaller ratios are connected to less effective patches, in fact ViT-\* models tend to have large ratios (in particular in later layers) compared to DeiT-\*, which matches lower robust accuracies in general (see Table 3).

## B. Effect of Input Mean on Attention Weight Robustness

As discussed in in Section 4.2, less centered inputs  $X$  (larger absolute value of input mean  $|\mu|$ ) make dot-product self-attention less robust to patch attacks on the controlled setting (when input variance is constant). We recap that the input mean is  $\mu = \mu \cdot 1$ , input standard deviation  $\sigma = 1$ , and  $W_Q = -w \cdot \mathbb{I}_{d_k}$  and  $W_K = w \cdot \mathbb{I}_{d_k}$ .

We now denote the query mean by  $\mu_Q = \text{Mean}(W_Q \cdot X) = -w\mu$  and key mean by  $\mu_K = \text{Mean}(W_K \cdot X) = w\mu$ . We note that the element-wise distance between query and key mean is  $|\mu_K^{(i)} - \mu_Q^{(i)}| = |w\mu_i + w\mu_i| = 2|w||\mu|$ . On the other hand, for query standard deviation, we have  $\sigma_Q = \text{StdDev}(W_Q \cdot X) = \text{StdDev}(-wX) = w1$  and for key standard deviation  $\sigma_K^2 = \text{StdDev}(W_K \cdot X) = \text{StdDev}(wX) = w1$ .

As can be seen, increasing  $|\mu|$  increases the distance between query and key cluster mean, while leaving the key's and query's standard deviation unchanged. As a result, it increases the separation of key and query clusters (see also Figure 1). On the other hand, increasing  $|w|$  has no systematic effect on the separation of query and key cluster, as  $w$ 's effect on mean and standard deviation cancels out.

We empirically quantify the query and key cluster separation using the Silhouette score [32]. Figure A1 shows this score as a function of the input mean scale  $|\mu|$  in the controlled setting. The plot confirms above's theoretical argument: increased input mean results in more separated keys and queries.

The result of this separation of keys and queries is that attention drawn by one key can increase for all queries when moving the key in the direction of the query mean (because in a sense, all queries lie in the same direction from the key as long as they are distant and have small variance). On the

other hand, if keys and queries lie intermingled, than for any direction the key moves, it will get closer to some queries at the expense of increasing distance to other queries. Because of this, for an adversarial patch attack on the attention weights, it is beneficial if keys and queries are well separated (as in Figure 1).

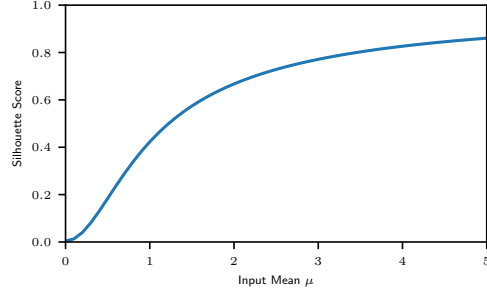


Figure A1. Silhouette score between clusters of projected keys and queries for different choices of  $\mu$  on synthetic data. Larger scores corresponds to keys and queries forming more distinct clusters.

## C. Ablation on Attention-Fool on ViT

In this section, we perform a series on ablations on Attention-Fool losses on ViT. Sections C.1, C.3 and C.2 ablate on properties of the loss, while Section C.4 and C.5 on properties of the adversarial patch.

### C.1. Reduction over Heads and Layers

As discussed in Section 5, when multiple attention layers and attention heads are present, one can aggregate per-layer and per-head  $\mathcal{L}_{kq}^{hl}$  in different ways. To study the influence of these aggregation choices, in this section we empirically test three of them: (i) smax (as in Section 5), (ii) hard maximum max and (iii) mean. Since we are aggregating along two dimensions, layers and heads, this sparks nine combinations of the three aggregations, which we empirically test on ViT models. As shown in Table A1, across different aggregation methods, choosing smax over both layers and heads generally performs well, resulting in low robust accuracies. Intuitively, using smax gives more flexibility to the optimization to choose the most vulnerable layers and heads. We also find that mean aggregation across both layers and heads is an effective choice, presumably in part due to the fact that in ViT inputs to attention layers are pre-emptively normalized, leading to smaller difference in per-head per-layer behaviour. In comparison the hard maximum max seems to be the weakest choice, leading to consistently worse results. We note from Table A1 that the best combination is rather model-specific, which suggests that a similar ablation study could be run on a model-basis. In fact, we will show in the next section how on DETR [8], Attention-Fool with max aggregation results in better performance.

Table A1. Robust accuracies (%) under adversarial patch attacks with Attention-Fool losses when choosing different ways of aggregating  $\mathcal{L}_{kq}^{hl}$  across encoder layers and attention heads. All rows are computed using PGD<sup>250</sup> with momentum and step size  $\alpha=8/255$ . Numbers in parenthesis report the improvement or degradation in robust accuracy w.r.t. the  $\text{smax}_{\text{smax}}$  default choice outlined in Section 5.

$L$ reduction	$H$ reduction	ViT-T	ViT-B	ViT-B-384	DeiT-T	DeiT-B	DeiT-B-384
smax	smax	0.00	0.10	2.50	0.00	19.30	39.80
smax	mean	0.00 (-0.00)	0.20 (+0.10)	2.70 (+0.20)	0.10 (+0.10)	17.80 (-1.50)	40.60 (+0.80)
smax	max	0.00 (-0.00)	3.30 (+3.20)	5.80 (+3.30)	0.00 (-0.00)	36.60 (+17.30)	45.90 (+6.10)
mean	smax	0.00 (-0.00)	0.00 (-0.10)	2.90 (+0.40)	0.00 (-0.00)	18.30 (-1.00)	40.40 (+0.60)
mean	mean	0.00 (-0.00)	0.10 (-0.00)	3.50 (+1.00)	0.00 (-0.00)	17.50 (-1.80)	39.60 (-0.20)
mean	max	0.00 (-0.00)	2.40 (+2.30)	9.20 (+6.70)	0.00 (-0.00)	23.80 (+4.50)	43.60 (+3.80)
max	smax	0.10 (+0.10)	15.60 (+15.50)	26.70 (+24.20)	1.20 (+1.20)	27.10 (+7.80)	45.70 (+5.90)
max	mean	0.20 (+0.20)	13.80 (+13.70)	24.60 (+22.10)	1.50 (+1.50)	26.10 (+6.80)	46.10 (+6.30)
max	max	3.70 (+3.70)	45.50 (+45.40)	59.70 (+57.20)	1.50 (+1.50)	58.50 (+39.20)	58.80 (+19.00)

Table A2. Median of  $|(\nabla_X A_h(X))X|/(A_h(X)1_X)|$  over tokens and heads, for models on 12 encoder layers. We report the mean of the ratio medians and its standard error over 100 randomly selected images from the MS COCO 2017 validation set [22] for DETR (DC5-R50), and over 100 randomly selected images from the ImageNet 2012 [33] dataset for all remaining models.

	DETR	ViT-T	ViT-S	ViT-B	DeiT-T	DeiT-S	DeiT-B
Layer 1	0.208 $\pm$ 0.008	0.079 $\pm$ 0.001	0.042 $\pm$ 0.001	0.072 $\pm$ 0.001	0.035 $\pm$ 0.000	0.045 $\pm$ 0.001	0.053 $\pm$ 0.001
Layer 2	0.035 $\pm$ 0.001	0.040 $\pm$ 0.001	0.051 $\pm$ 0.000	0.080 $\pm$ 0.001	0.045 $\pm$ 0.001	0.045 $\pm$ 0.000	0.044 $\pm$ 0.001
Layer 3	0.039 $\pm$ 0.001	0.031 $\pm$ 0.000	0.027 $\pm$ 0.000	0.047 $\pm$ 0.000	0.044 $\pm$ 0.001	0.033 $\pm$ 0.000	0.042 $\pm$ 0.000
Layer 4	0.066 $\pm$ 0.001	0.043 $\pm$ 0.001	0.028 $\pm$ 0.000	0.037 $\pm$ 0.000	0.035 $\pm$ 0.000	0.032 $\pm$ 0.000	0.057 $\pm$ 0.000
Layer 5	0.098 $\pm$ 0.001	0.037 $\pm$ 0.001	0.029 $\pm$ 0.000	0.033 $\pm$ 0.000	0.035 $\pm$ 0.000	0.033 $\pm$ 0.000	0.044 $\pm$ 0.000
Layer 6	0.147 $\pm$ 0.002	0.044 $\pm$ 0.001	0.027 $\pm$ 0.000	0.044 $\pm$ 0.001	0.070 $\pm$ 0.001	0.033 $\pm$ 0.000	0.040 $\pm$ 0.000
Layer 7	-	0.036 $\pm$ 0.000	0.031 $\pm$ 0.000	0.040 $\pm$ 0.000	0.045 $\pm$ 0.001	0.033 $\pm$ 0.001	0.040 $\pm$ 0.000
Layer 8	-	0.045 $\pm$ 0.001	0.046 $\pm$ 0.001	0.046 $\pm$ 0.001	0.050 $\pm$ 0.001	0.037 $\pm$ 0.001	0.039 $\pm$ 0.000
Layer 9	-	0.171 $\pm$ 0.005	0.087 $\pm$ 0.002	0.064 $\pm$ 0.001	0.076 $\pm$ 0.002	0.052 $\pm$ 0.001	0.049 $\pm$ 0.001
Layer 10	-	0.308 $\pm$ 0.006	0.126 $\pm$ 0.003	0.061 $\pm$ 0.001	0.082 $\pm$ 0.002	0.077 $\pm$ 0.002	0.083 $\pm$ 0.002
Layer 11	-	0.428 $\pm$ 0.009	0.280 $\pm$ 0.006	0.099 $\pm$ 0.002	0.080 $\pm$ 0.002	0.100 $\pm$ 0.003	0.192 $\pm$ 0.005
Layer 12	-	0.442 $\pm$ 0.013	0.469 $\pm$ 0.014	0.249 $\pm$ 0.006	0.212 $\pm$ 0.008	0.198 $\pm$ 0.009	0.104 $\pm$ 0.004

Table A3. Robust accuracy (%) of different vision transformers when choosing  $l$  in  $\mathcal{L}_{kq}^l$ , which targets a specific encoder layer (see Section 5). Selecting  $l$  leads to worse results compared to the baseline loss  $\mathcal{L}_{kq}$ .

	ViT-T	ViT-B	DeiT-T	DeiT-B
$\mathcal{L}_{kq}$	0.0	0.1	0.0	19.3
$\mathcal{L}_{kq}^{(l)}, l=1$	7.5 (+7.5)	0.0 (-0.1)	14.0 (+14.0)	36.3 (+17.0)
$l=2$	4.4 (+4.4)	39.1 (+39.0)	2.2 (+2.2)	56.1 (+36.8)
$l=3$	0.2 (+0.2)	21.4 (+21.3)	1.0 (+1.0)	43.5 (+24.2)
$l=4$	0.0 (-0.0)	17.2 (+17.1)	0.8 (+0.8)	22.1 (+2.8)
$l=5$	0.0 (-0.0)	18.8 (+18.7)	1.3 (+1.3)	23.9 (+4.6)
$l=6$	0.0 (-0.0)	12.4 (+12.3)	0.0 (-0.0)	24.6 (+5.3)
$l=7$	0.0 (-0.0)	9.4 (+9.3)	0.0 (-0.0)	23.9 (+4.6)
$l=8$	0.1 (+0.1)	3.2 (+3.1)	0.6 (+0.6)	18.4 (-0.9)
$l=9$	0.0 (-0.0)	0.8 (+0.7)	0.1 (+0.1)	21.1 (+1.8)
$l=10$	0.1 (+0.1)	1.3 (+1.2)	0.1 (+0.1)	23.5 (+4.2)
$l=11$	0.0 (-0.0)	1.1 (+1.0)	0.2 (+0.2)	23.1 (+3.8)
$l=12$	0.0 (-0.0)	1.0 (+0.9)	1.2 (+1.2)	25.4 (+6.1)

## C.2. Choosing Specific Layer

Section 5 introduces per-layer,  $l$ , and per-head,  $h$ , losses identified by  $\mathcal{L}_{kq}^{hl}$ , but the evaluation of Section 6.2 focuses

Table A4. Robust accuracy (%) of different vision transformers for adversarial patches of different sizes for  $\mathcal{L}_{kq}^*$ . As the patch dimension shrinks to  $8 \times 8$  (corresponding to 0.13% of total image pixels), the robust accuracies increase as expected.

Patch Size	ViT-T	ViT-B	DeiT-T	DeiT-B
$16 \times 16$	0.0	0.1	0.0	13.1
$14 \times 14$	0.0	1.9	2.3	24.2
$12 \times 12$	0.0	6.3	17.6	39.2
$10 \times 10$	5.9	22.3	38.0	52.8
$8 \times 8$	34.7	50.1	51.2	65.3
Clean Accuracy	73.6	85.0	69.5	82.0

on the aggregated loss  $\mathcal{L}_{kq}$  and its version targeting the special token  $\mathcal{L}_{kq}^*$ . To study the effectiveness of targeting specific encoder layers in the loss, here we compare using single-layer  $\mathcal{L}_{kq}^l$  with the aggregated  $\mathcal{L}_{kq}$ ; we report the results in Table A3. Targeting a single  $l$  via  $\mathcal{L}_{kq}^l$  is typically weaker than targeting all jointly via  $\mathcal{L}_{kq}$ . In addition, it is unclear how to choose  $l$  a priori without trying all options as there is no common pattern across models.



Table A5. Robust accuracies (%) under Attention Fool adversarial patch attach using the un-normalized  $P_Q^{hl}$  and  $P_K^{hl}$  introduced in Section 5. All rows are computed using PGD<sup>250</sup> with momentum and step size  $\alpha=8/255$ . Without normalization,  $\mathcal{L}_{kq}$  and  $\mathcal{L}_{kq*}$  do not improve on  $\mathcal{L}_{ce}$  baselines and in fact perform much worse, likely because individual  $\mathcal{L}_{kq}^{hl}$  losses are not commensurable with each other.

	ViT-T	ViT-B	ViT-B-384	DeiT-T	DeiT-B	DeiT-B-384
$\mathcal{L}_{ce}$	0.10	13.50	31.20	19.80	36.00	58.80
$+\mathcal{L}_{kq}$	53.10 (+53.00)	81.70 (+68.20)	84.50 (+53.30)	67.40 (+47.60)	79.80 (+43.80)	80.60 (+21.80)
$+\mathcal{L}_{kq*}$	24.40 (+24.30)	79.80 (+66.30)	82.00 (+50.80)	49.00 (+29.20)	79.60 (+43.60)	80.80 (+22.00)
+ Momentum	0.00	3.10	13.20	1.50	16.80	41.70
$+\mathcal{L}_{kq}$	50.00 (+50.00)	81.60 (+78.50)	84.30 (+71.10)	66.80 (+65.30)	78.30 (+61.50)	80.30 (+38.60)
$+\mathcal{L}_{kq*}$	21.10 (+21.10)	80.30 (+77.20)	82.10 (+68.90)	24.10 (+22.60)	78.60 (+61.80)	80.10 (+38.40)

Table A6. Robust accuracies (%) of ResNet50 and different vision transformers with the adversarial patch positioned at the center of the images. The evaluation setting is similar to Table 3 except for the position of the patch. Here, the robustness of ResNet50 has further deteriorated compared to a patch placed at the image corners. In contrast, transformers demonstrate similar vulnerability with the patch positioned at the center of the images and corner of the images under our Attention Fool loss variant  $\mathcal{L}_{kq*}$ . These results indicate that the transformers are less sensitive to location of the adversarial patch compared to CNNs.

	ResNet50	ViT-T	ViT-B	ViT-B-384	DeiT-T	DeiT-B	DeiT-B-384
$\mathcal{L}_{ce}$	41.50	0.10	14.50	28.90	23.50	44.20	66.50
$+\mathcal{L}_{kq}$	-	0.00 (-0.10)	5.80 (-8.70)	23.10 (-5.80)	22.20 (-1.30)	44.90 (+0.70)	69.10 (+2.60)
$+\mathcal{L}_{kq*}$	-	0.00 (-0.10)	4.10 (-10.40)	18.80 (-10.10)	12.60 (-10.90)	35.90 (-8.30)	67.20 (+0.70)
+ Momentum	31.10	0.00	2.40	11.10	1.60	20.90	42.30
$+\mathcal{L}_{kq}$	-	0.00 (-0.00)	0.30 (-2.10)	3.10 (-8.00)	0.10 (-1.50)	28.50 (+7.60)	48.90 (+6.60)
$+\mathcal{L}_{kq*}$	-	0.00 (-0.00)	0.20 (-2.20)	2.60 (-8.50)	0.10 (-1.50)	11.70 (-9.20)	45.90 (+3.60)

### C.3. Normalization

In Section 5 we introduced an  $\ell_{1,2}$  normalization in the computation of  $\mathcal{L}_{kq}$ , where projected queries and keys are normalized as  $\tilde{P}_Q^{hl} = P_Q^{hl} / \frac{1}{n} \|P_Q^{hl}\|_{1,2}$  and  $\tilde{P}_K^{hl} = P_K^{hl} / \frac{1}{n} \|P_K^{hl}\|_{1,2}$ . To evaluate the effect of this normalization we compute  $\mathcal{L}_{kq}$  without it, repeating the experiment reported in Table 3, and report the results in Table A5. We observe that this  $\ell_{1,2}$  normalization is very crucial and without normalization, performance of  $\mathcal{L}_{kq}$  and  $\mathcal{L}_{kq*}$  deteriorates considerably, to levels clearly below  $\mathcal{L}_{ce}$ .

### C.4. Patch Location

While Attention-Fool was designed to operate for arbitrary adversarial patch locations (as long as the adversarial patch aligns with ViT/DeiT's patch tiling), the location may have an effect on the resulting robust accuracies. In Section 6.2 we set the patch location to be the top left-most corner of the image. Here, we repeat the experiment but we place the patch in the image center, and we target the corresponding key. Specifically, in  $224 \times 224$ -resolution models we place the patch top-left corner at 96,96, while in  $384 \times 384$  models we place it at 176,176. We repeat the evaluation of Section 6.2 in this setting, and report the results in Table A6. Notably, the table shows that we obtain a significant drop in ResNet50 robust accuracy because of the different location (from 49.00% in Table 3 to 31.10% in Ta-

ble A6), while the robust accuracies of vision transformers ViT and DeiT do not change significantly. Moreover,  $\mathcal{L}_{kq*}$  improves upon  $\mathcal{L}_{ce}$  for all but the DeiT-B-384 model.

### C.5. Patch Sizes

The evaluation of Section 6.2 focuses on adversarial patches of size  $16 \times 16$ , which corresponds with the token dimension in the investigated ViT models. Here, we also investigate smaller adversarial patch sizes up to a dimension of  $8 \times 8$ , we always place the top-left corner of the patch at the (0, 0) coordinate (top-left) of the entire image. We report the robust accuracies for varying sizes in Table A4. As expected smaller patches lead to higher robust accuracies, but we find that even very small patches of  $8 \times 8$  decrease the robust accuracy significantly compared to the clean accuracy for some models.

## D. Ablation on Targeted Attacks on ViT

In Section 6.2, we reported the robust accuracies for vision transformers under an untargeted patch attacks. Here, we also evaluate a targeted attack on the same models by selecting a target class in the optimization: rather than maximizing  $\mathcal{L}_{ce}$  with the true image class  $y$  as in Eq. 1, we replace  $y$  with a target class  $y^*$  and minimize the  $\mathcal{L}_{ce}$  instead (note that the  $\mathcal{L}_{kq}$  loss is still maximized in a targeted setting). The way we combine  $\mathcal{L}_{ce}$  with the Attention-

Table A7. Adversarial patch attack success rate (%) for a targeted attack with target class “0”. The evaluation setting is similar to Table 3. Here, the attack success rate in vision transformers is larger than that of the ResNet50 model, but the improvements obtained with Attention Fool in comparison to the cross-entropy baseline are not as consistent as in the untargeted attack setting.

	ResNet50	ViT-T	ViT-B	ViT-B-384	DeiT-T	DeiT-B	DeiT-B-384
$\mathcal{L}_{ce}$	30.70	93.20	30.00	11.60	43.00	14.10	1.90
$+\mathcal{L}_{kq}$	-	94.60 (+1.40)	20.00 (-10.00)	12.20 (+0.60)	48.40 (+5.40)	14.80 (+0.70)	1.70 (-0.20)
$+\mathcal{L}_{kq*}$	-	94.30 (+1.10)	19.50 (-10.50)	13.00 (+1.40)	49.10 (+6.10)	14.60 (+0.50)	2.30 (+0.40)
+ Momentum	38.90	100.00	42.70	23.80	100.00	75.20	4.70
$+\mathcal{L}_{kq}$	-	100.00 (+0.00)	43.10 (+0.40)	21.10 (-2.70)	99.10 (-0.90)	74.60 (-0.60)	4.60 (-0.10)
$+\mathcal{L}_{kq*}$	-	100.00 (+0.00)	43.60 (+0.90)	22.40 (-1.40)	98.40 (-1.60)	77.10 (+1.90)	4.00 (-0.70)

Table A8. Mean Average Precisions (mAP) in presence of adversarial patches computed with  $\mathcal{L}_{ce}$  and Attention-Fool’s  $\mathcal{L}_{kq}^{(1)}$  patches on DETR models. Differently than in Table 4, here we replace the adversarial target key – one of the units in the backbone feature map – with its clean counterpart (the same unit computed in the non-patched image). Replacing the adversarial key removes the entirety of the adversarial effect of Attention-Fool, showing how the mAP degradation under attack can be attributed to the target key alone. Three different patch sizes and four different DETR models are considered, based either on R50 or R101 backbone and with and without dilation.

	R50	DC5-R50	R101	DC5-R101
clean mAP	53.00	54.25	54.41	56.74
64×64: $\mathcal{L}_{ce}$	38.96	17.51	33.35	35.67
$+\mathcal{L}_{kq}^{(1)}$	51.85 (+12.90)	52.92 (+35.41)	48.85 (+15.50)	56.61 (+20.94)
$\mathcal{L}_{kq}^{(1)}$ only	52.13 (+13.18)	53.48 (+35.97)	50.71 (+17.36)	57.11 (+21.44)
56×56: $\mathcal{L}_{ce}$	41.18	25.09	33.64	38.50
$+\mathcal{L}_{kq}^{(1)}$	53.34 (+12.17)	53.66 (+28.56)	53.51 (+19.87)	56.81 (+18.31)
$\mathcal{L}_{kq}^{(1)}$ only	52.73 (+11.55)	53.28 (+28.19)	52.37 (+18.74)	57.18 (+18.68)
48×48: $\mathcal{L}_{ce}$	43.61	27.84	40.45	42.16
$+\mathcal{L}_{kq}^{(1)}$	52.54 (+8.93)	52.97 (+25.13)	53.52 (+13.06)	56.90 (+14.73)
$\mathcal{L}_{kq}^{(1)}$ only	53.27 (+9.66)	53.99 (+26.16)	53.58 (+13.13)	56.90 (+14.74)

Fool variants is identical as in Section 6.2; here we choose  $y^* = 0$ . In this case, rather than reporting robust accuracies, we report attack success rate, i.e., the percentage of times the addition of the adversarial patch changed the correct classification of an image into the target class  $y^*$ . We report the results in Table A7, note that the colors and the signs of improvements/degradation are inverted in comparison to the other tables. Table A7 shows that the benefit of Attention-Fool in targeted attacks is less clear. We hypothesise that different weighting of  $\mathcal{L}_{ce}$  and  $\mathcal{L}_{kq}$  might be required in targeted settings, specifically because  $\mathcal{L}_{ce}$  is bounded by zero when minimized (while it is unbounded when maximized in the untargeted setting). Additionally, specific properties of the target class “0” on certain models might bias the evaluation; a larger, more in-depth evaluation of targeted attacks is left for future work.

## E. Adversarial Token Replacement

In Section 6.3 we showed how targeting a single key with Attention-Fool in DETR can lead to large degradation in mAP. Differently than in ViT models, because of DETR’s hybrid architecture (CNN plus Transformer) an adversarial

patch placed on the DETR image input affects a number of tokens in the encoder’s input: all those backbone outputs (tokens) whose receptive field via the CNN overlaps with the input image patch. Here, to show that the attack performance can be attributed to the individual key token that is the Attention-Fool target, we replace this key with its clean counterpart. To do so, we compute all keys on the clean image and all keys in the patched image (by forwarding the image through the CNN backbone), and we replace the target adversarial key with its clean counterpart. This is either the key indexed by 2,2 or by 4,4 in non-dilated and dilated models, respectively. We report the resulting mAPs in Table A8. The table shows how replacing this single token’s key removes a large part of the adversarial effect in all rows using  $\mathcal{L}_{kq}^{(1)}$  loss. In comparison, in rows using  $\mathcal{L}_{ce}$ , replacing this token’s key still results in low mAPs, showing that in this case the adversarial effect is not attributable to the same adversarial token alone.

## F. Additional Visualizations

We report additional visualizations akin to those in Fig. 1 and Fig. 4 in Figure A3 and Figure A2, respectively.

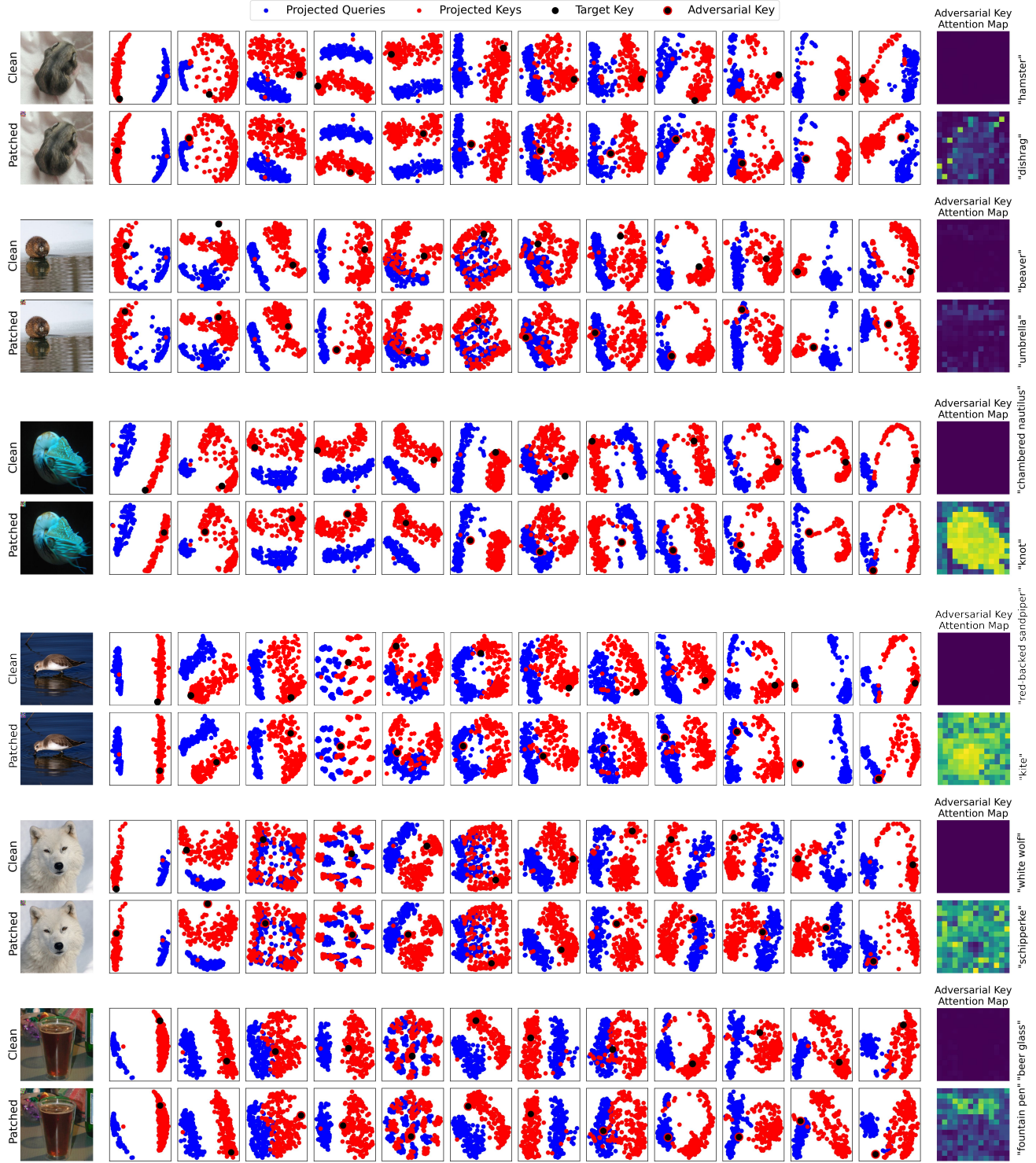


Figure A2. Embedded projected key and query tokens for clean and patched input images on each of the 12 DeiT-B layers, for a single attention head. For each image we chose an attention head which showed large amount of changes. The last column reports the attention map weights of the adversarial key on the last layer – showing that generally the key draws a large amount of attention from queries. Note that while the adversarial patch tends to focus on drawing the attention on the last layer (which is visualized in the right-most column), it can target arbitrary layers – in fact, we obtain successful mis-classifications even if the last layer attention map only has minor changes.



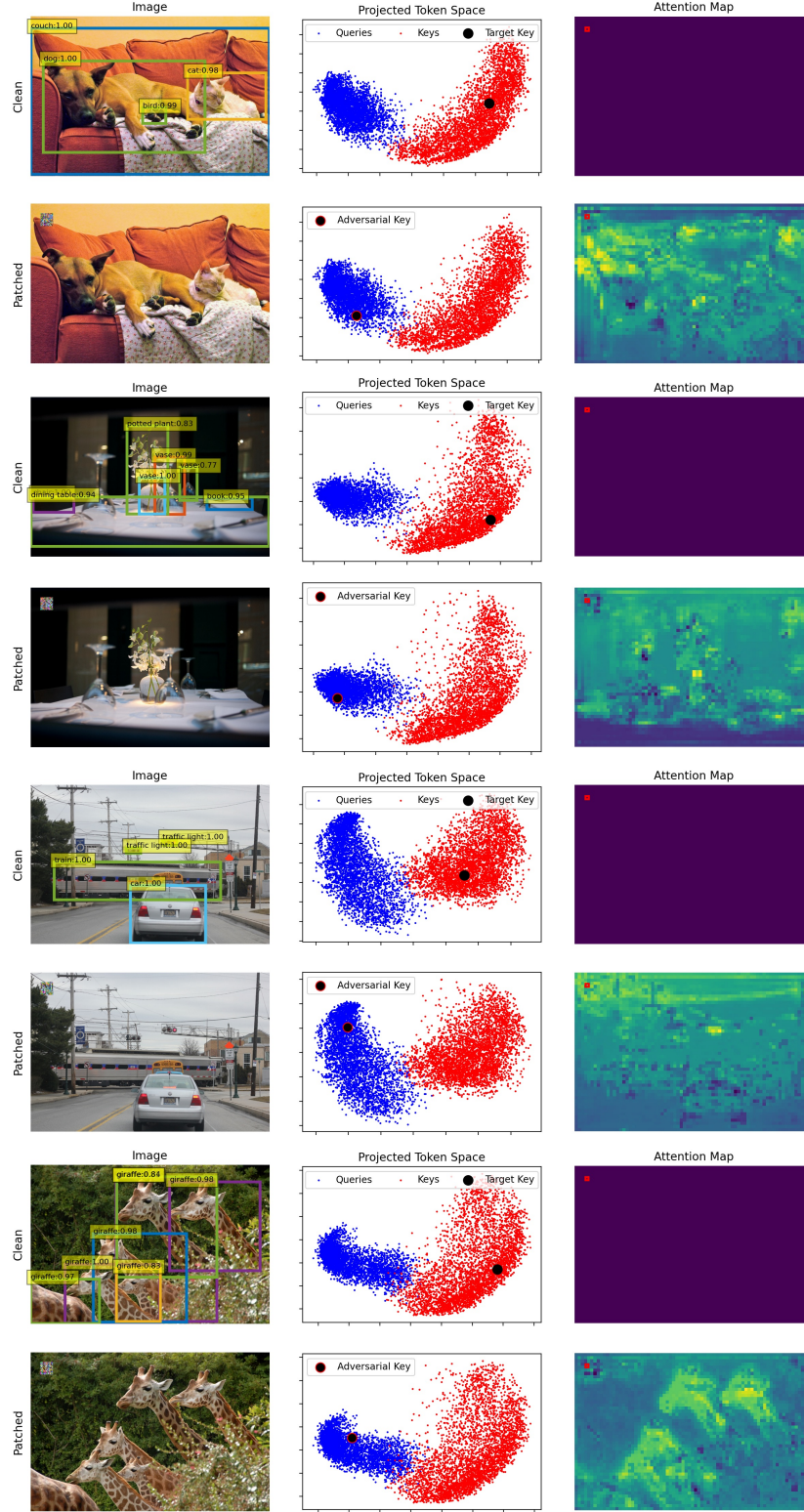


Figure A3. More comparisons of clean and adversarially patched input for DETR [8]. The patch shifts a targeted key token towards the cluster of query tokens. In dot-product attention, this directs queries attention to the malicious token and prevents the model from detecting the remaining objects. The right-most column compares queries' attention weights to the adversarial key, whose location is marked by a red box, between clean and patched inputs. These images use DETR DC5-R50 and patches are optimized with  $\mathcal{L}_{ce} + \mathcal{L}_{kq}^{(1)}$ .