Robustifying Token Attention for Vision Transformers

Yong Guo, David Stutz, Bernt Schiele
Max Planck Institute for Informatics, Saarland Informatics Campus
guoyongcs@gmail.com, \{david.stutz, schiele\}@mpi-inf.mpg.de

Abstract

Despite the success of vision transformers (ViTs), they still suffer from significant drops in accuracy in the presence of common corruptions, such as noise or blur. Interestingly, we observe that the attention mechanism of ViTs tends to rely on few important tokens, a phenomenon we call token overfocusing. More critically, these tokens are not robust to corruptions, often leading to highly diverging attention patterns. In this paper, we intend to alleviate this overfocusing issue and make attention more stable through two general techniques: First, our Token-aware Average Pooling (TAP) module encourages the local neighborhood of each token to take part in the attention mechanism. Specifically, TAP learns average pooling schemes for each token such that the information of potentially important tokens in the neighborhood can adaptively be taken into account. Second, we force the output tokens to aggregate information from a diverse set of input tokens rather than focusing on just a few by using our Attention Diversification Loss (ADL). We achieve this by penalizing high cosine similarity between the attention vectors of different tokens. In experiments, we apply our methods to a wide range of transformer architectures and improve robustness significantly. For example, we improve corruption robustness on ImageNet-C by 2.4% while improving accuracy by 0.4% based on state-of-the-art robust architecture FAN. Also, when fine-tuning on semantic segmentation tasks, we improve robustness on CityScapes-C by 2.4% and ACDC by 3.0%. Our code is available at https://github.com/guoyongcs/TAPADL.

1. Introduction

Despite the success of vision transformers (ViTs), their performance still drops significantly on common image corruptions such as ImageNet-C [24, 52, 19], adversarial examples [18, 16, 40, 56], and out-of-distribution examples as benchmarked in ImageNet-A/R/P [64, 24]. In this paper, we examine a key component of ViTs, i.e., the self-attention mechanism, to understand these performance drops. Interestingly, we discover a phenomenon we call token overfocusing, where only few important tokens are relied upon by the attention mechanism across all heads and layers. We hypothesize that this overfocusing is particularly fragile to the corruptions on input images and greatly hampers the robustness of the attention mechanism.

Starting from the state-of-the-art robust architecture FAN [67], we exemplarily investigate the last attention layer (see attention of other layers in supplementary). Specifically, Figure 1 shows a clean and a corrupted input image as well as the corresponding attention maps. These are matrices of $N \times N$, with $N$ being the number of input/output tokens. Here, the $i$-th row indicates which input tokens (columns) the $i$-th output token “attends” to – darker red indicates higher attention scores. Token overfocusing can then, informally, be defined by observing pronounced vertical lines in the attention map: First, each output token (row) focuses on only few important input tokens, ignoring most
of the other information. Second, all output tokens seem to focus on the same input tokens, leading to a very low diversity among the attention vectors in different rows. We highlight that the overfocusing issue is present throughout the entire ImageNet dataset, and also across diverse architectures (see more examples in Figure 3 and supplementary). Moreover, this overfocusing issue has also been observed in existing works. For example, Figure 5 of [10] observes very few important tokens (in deep red color) in layers 3–6; Figure 1 of [16] also reports very few important tokens (in yellow color). More critically, we find that these important tokens are extremely fragile in the presence of common corruptions. To be specific, when applying Gaussian noise on the input image, the tokens recognized as important change entirely, see Figure 1 (left, second column). Quantitatively, this can be captured by computing the cosine similarity between the clean and corrupted attention maps. Unsurprisingly, as shown by the blue box in Figure 1 (right), the cosine similarity is indeed extremely low, confirming our initial hypothesis. This motivates us to robustify the attention by alleviating the token overfocusing issue.

To this end, we encourage the attention module to focus on diverse input tokens. This can be achieved by changing the patterns in both columns and rows of an attention map, which motivates the two key components of our method. First, when comparing attention columns in Figure 1, very few tokens are important, leading to a fragile attention. To address this, we encourage output tokens to not only focus on individual input tokens but take into account the local neighborhood around these tokens, in order to make more tokens (columns) contribute meaningful information. Intuitively, an individual token itself may not be important but can be enhanced by aggregating the information from potentially important tokens located within its neighborhood. We achieve this using a learnable average pooling mechanism applied to each input token to aggregate information before computing self-attention. Second, when comparing attention rows in Figure 1, most output tokens focus on the same input tokens. Once these tokens are distorted, the whole attention may fail. To address this issue, we seek to diversify the set of input tokens (columns) that the output tokens (rows) rely on. We achieve this using a loss that explicitly penalizes high cosine similarity across rows. In Figure 1, the combination of these techniques leads to more balanced attention across columns and more diverse attention across rows. More critically, these attention maps are more stable in the light of image corruptions. Again, we quantitatively confirm this on ImageNet using the cosine similarity which is significantly higher.

Overall, we make three key contributions: 1) we propose a Token-aware Average Pooling (TAP) module that encourages the local neighborhood of tokens to participate in the self-attention mechanism. To be specific, we conduct averaging pooling to aggregate information and control it by adjusting the pooling area for each token. 2) We further develop an Attention Diversification Loss (ADL) to improve the diversity of attention received by different tokens, i.e., rows in the attention map. To this end, we compute the attention similarity between rows in each layer and minimize it during training. 3) We highlight that both TAP and ADL can be applied on top of diverse transformer architectures. In the experiments, the proposed methods consistently improve the model robustness on out-of-distribution benchmarks by a large margin while preserving a competitive improvement of clean accuracy at the same time. For example, we improve robustness against corruptions and distribution shifts on ImageNet-A/C/P/R by at least 1.5%, as shown in Table 2. In addition, the improvement also generalizes well to other downstream tasks, e.g., semantic segmentation, with an improvement of 2.4% in Table 4.

2. Related Work

The remarkable performance of ViTs on various learning tasks is largely attributed to the use of self-attention [26]. Naturally, the self-attention mechanism has been extended in various aspects, addressing shortcomings such as the heavy reliance on pre-training [42, 62, 35] or the high computational complexity [29, 28, 13, 1, 8, 3, 47] and adapting the architecture to vision tasks by considering multi-scale transformers [25, 15, 46, 48, 49, 9, 57, 53, 25, 50, 6, 58]. Similar to us, some related works [66, 10, 44, 34, 45] also investigate and visualize the self-attention in order to improve transformer architectures, e.g., to learn deeper transformers [66] or prune attention heads [45]. However, to the best of our knowledge, we are the first to report a token overfocusing issue and link it to poor robustness of ViTs.

There is also a lot of interests in understanding and improving the robustness of ViTs [5, 2, 41, 37, 4, 21, 31]. For example, [32] develops a robust vision transformer (RVT) by combining various components to boost robustness, including an improved attention scaling. FAN [67] combines token attention and channel attention [1] and can be considered state-of-the-art. Both approaches rely on modified self-attention mechanisms and can be shown to suffer from token overfocusing. Thus, our techniques can be shown to improve robustness on top of RVT and FAN, respectively. Our TAP approach also shares similarities to recent ideas of trying to introduce locality into the self-attention mechanism [54, 53, 17, 38, 61, 54, 51]. For example, CvT [53] introduces convolutions into the self-attention architecture to enhance the local information inside tokens. MViTv2 [27] exploits average pooling to extract local features and improve clean performance. However, all of these strategies use the same aggregation across all tokens, while our approach adaptively chooses an aggregation neighborhood for each token to improve robustness.
3. Robust Token Self-Attention

In the following, we focus on the attention mechanism in ViTs, aiming to improve their overall robustness. To get started, we first describe the observed token overfocusing issue where ViTs focus on very few but unstable tokens in detail in Section 3.1. Then, we propose two general techniques for alleviating this issue: First, in Section 3.2, we propose a Token-aware Average Pooling (TAP) module that encourages the neighborhood of tokens to participate in the attention mechanism. This is achieved using a learnable pooling area as illustrated in Figure 2. Second, in Section 3.3, we develop a new Attention Diversification Loss (ADL) to improve the diversity of attention patterns across tokens. Both methods can be applied to most transformer architectures and we will show they greatly improve robustness with negligible training overhead.

3.1. Token Overfocusing

As illustrated in Figure 1, we can visualize the self-attention mechanism in each layer as \( N \times N \) attention matrix. Here, \( N \) is the number of input and output tokens and each entry \((i, j)\) denotes the attention that the \( i \)-th output token (\( i \)-th row) puts on the \( j \)-th input token (\( j \)-th column) – deeper red denoting higher attention scores. To handle multi-head self-attention, we visualize this matrix by averaging across attention heads.

As baseline, Figure 1 highlights the recent FAN [67] architecture, showing the attention map of the last layer for example. We observe that the attention is generally very sparse in columns, meaning that most input tokens are not attended to and very few tokens are overfocused. More importantly, these “important” tokens are often similar across output tokens (rows). We refer to this phenomenon as token overfocusing. This leads to problems when facing corruptions such as Gaussian noise: the set of important tokens might change entirely (Figure 1, second column). This can be understood as the original tokens not capturing robust information. We can also quantitatively capture this instability by computing the cosine similarity between clean and corrupted attention maps across all ImageNet images. As shown by the blue box in Figure 1 (right), the baseline model obtains very low cosine similarities, indicating the poor robustness of standard self-attention.

We found that this phenomenon exists across diverse architectures, e.g., DeiT [43] and RVT [32], and diverse tasks, including both image classification and semantic segmentation (see visual examples in supplementary). Moreover, we highlight that our observation is consistent with existing works [10, 36]. Specifically, these works observe that the middle and deep layers (often with overfocusing issue) tend to capture global information but put focus on very few tokens, e.g., very few deep red tokens in Figure 5 of [10] and Figure 3 (Stage 4) in [36]. This further justifies the existence and importance of the token overfocusing issue. To alleviate this issue, we propose two general techniques to robustify self-attention in the remainder of this section.
3.2. Token-aware Average Pooling (TAP)

In the first part of our method, we seek to encourage more input tokens to participate in the self-attention mechanism, i.e., obtaining more columns with high scores in the attention map. To this end, we encourage each input token to explicitly aggregate useful information from its local neighborhood, in case the token itself does not contain important information. This approach is justified by existing works [53, 38, 27] and the observation that introducing any local aggregation before self-attention consistently improves robustness, see Table 1 (last column). In fact, these methods apply a fixed convolutional kernel or pooling area to all the tokens. Nevertheless, tokens often differ from each other and each token should require a specific local aggregation strategy. This motivates us to adaptively choose the right neighborhood size and aggregation strategy.

Inspired by this, we enable each token to select an appropriate area/strategy to conduct local aggregation. Specifically, we develop a Token-aware Average Pooling (TAP) that conducts average pooling and adaptively adjusts the pooling area for each token. As shown in Figure 2, we exploit a multi-branch structure that computes the weighted sum over multiple branches, each with a specific pooling area. Instead of simply changing the kernel size similar to [59], TAP changes the dilation to adjust the pooling area. The main observation behind this is that average pooling with a large kernel, without dilation, leads to extremely large overlaps between adjacent pooling regions and thereby severe redundancy in output tokens. This can, for example, be seen in Table 1 where AvgPool5x5 incurs a large drop in clean accuracy of around 1.2%. Similar to [54, 53, 17, 38], we also investigated using learnable convolutions, but observed only marginal improvement alongside a significant increase in computational cost.

Based on these observations, we build TAP based on average pooling with diverse dilations: Without loss of generality, given \( K \) branches, we consider the dilation within the range \( d \in [0, K-1] \). Here, \( d = 0 \) implies identity mapping without any computation, i.e., no local aggregation. The maximum dilation determined by \( K \) is a hyper-parameter and we find that performance and robustness improvements saturate at around \( K = 5 \) (see Figure 7 top). Within the allowed dilation range, our method includes a lightweight dilation predictor to predict which dilation (i.e., which branch in Figure 2) to utilize. Note that this can also be a weighted combination of multiple \( d \). We emphasize that this predictor is very efficient since it reduces the feature dimension (from \( C \) to \( K \) in Figure 2) such that it adds minimal computational overhead and model parameters. The same approach can also be applied to non-dilated average-pooling where \( d \) controls the kernel size, named TAP (multi-kernel). From Table 1, our TAP greatly outperforms this variant, indicating the effectiveness of using dilations for pooling.

<table>
<thead>
<tr>
<th>Model</th>
<th>#FLOPs (G)</th>
<th>#Params (M)</th>
<th>ImageNet</th>
<th>ImageNet-C ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (FAN-B-Hybrid)</td>
<td>11.7</td>
<td>50.4</td>
<td>83.9</td>
<td>46.1</td>
</tr>
<tr>
<td>+ AvgPool3x3</td>
<td>11.7</td>
<td>50.4</td>
<td>83.6</td>
<td>45.6 (-0.5)</td>
</tr>
<tr>
<td>+ AvgPool5x5</td>
<td>11.7</td>
<td>50.4</td>
<td>82.7</td>
<td>45.5 (-0.6)</td>
</tr>
<tr>
<td>+ Conv3x3</td>
<td>17.3</td>
<td>79.4</td>
<td>84.0</td>
<td>45.9 (-0.2)</td>
</tr>
<tr>
<td>+ Conv5x5</td>
<td>27.4</td>
<td>130.7</td>
<td>84.4</td>
<td>45.8 (-0.3)</td>
</tr>
<tr>
<td>+ TAP (multi-kernel)</td>
<td>11.8</td>
<td>50.7</td>
<td>84.1</td>
<td>45.5 (-0.6)</td>
</tr>
<tr>
<td>+ TAP (Ours)</td>
<td>11.8</td>
<td>50.7</td>
<td>84.3</td>
<td>44.9 (-1.2)</td>
</tr>
</tbody>
</table>

Table 1. Comparisons of local aggregation approaches based on FAN-B-Hybrid. We show that conducting average pooling for all the tokens improves the robustness but impedes clean accuracy. Introducing a convolution into each block greatly increases model complexity. In addition, we also compare a variant of our TAP, namely TAP (multi-kernel), that considers multiple kernel sizes for pooling and learns weights for each branch. Our TAP greatly outperforms this variant, indicating the effectiveness of using dilation. Moreover, TAP yields the best tradeoff between accuracy and robustness along with negligible computational overhead.

3.3. Attention Diversification Loss (ADL)

In the second part of our method, we seek to improve the diversity of attention across output tokens, i.e., encourage different rows in Figure 1 to attend to different input tokens. Based on this objective, we propose an Attention Diversification Loss (ADL) that explicitly reduces the cosine similarity of attention among different output tokens (rows). However, for this approach to work, there are several challenges to overcome. First, computing the cosine similarity between attentions is numerically tricky. For example, if two rows (i.e., output tokens) have very disjoint attention patterns, we expect a low cosine similarity close to 0. However, even for tokens that are not attended to, the attention scores will not be zero. For a large \( N \), computing dot product and adding these values up tend to result in a cosine similarity significantly above zero. To alleviate this issue, we exploit a thresholding trick to filter out those very small values and only focus on the most important ones. Let \( \mathbb{1}(\cdot) \) be the indication function, and \( \bar{A}_i^{(l)} \) be the attention vector of the \( i \)-th token (row) in the \( l \)-th layer. We introduce a threshold \( \tau \) (see ablation in Table 7) that depends on the number of tokens \( N \), i.e., \( \tau/N \). Thus, the attention after thresholding becomes

\[
\hat{A}_i^{(l)} = \mathbb{1}(A_i^{(l)} \geq \tau/N) \cdot A_i^{(l)}. \tag{1}
\]

Second, to avoid the quadratic complexity of computing similarities between pairs of \( N \) rows, we approximate it by computing the cosine similarity between each individual attention vector \( \hat{A}_i^{(l)} \) with the average attention \( \bar{A}^{(l)} := \frac{1}{N} \sum_{i=1}^{N} \hat{A}_i^{(l)} \). When considering a model with \( L \) layers, we average the ADL loss across all the layers by:

\[
L_{ADL} = \frac{1}{L} \sum_{l=1}^{L} L_{ADL}^{(l)}, \quad L_{ADL}^{(l)} = \frac{1}{N} \sum_{i=1}^{N} \frac{\bar{A}_i^{(l)} \bar{A}_i^{(l)}}{\| \bar{A}_i^{(l)} \| \| \bar{A}_i^{(l)} \|}. \tag{2}
\]
In practice, we combine our ADL with the standard cross-entropy (CE) loss and introduce a hyper-parameter $\lambda$ (see ablation in Figure 7) to control the importance of ADL:

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda \mathcal{L}_{ADL}. \quad (3)$$

We highlight that our ADL can be applied to boost the robustness on diverse tasks, including image classification and semantic segmentation (see Table 2 and Table 4).

4. Experiments

We conduct extensive experiments to verify our method on both image classification and semantic segmentation tasks. In Section 4.1, we first train classification models on ImageNet [12] and demonstrate that our models obtain significant improvement on various robustness benchmarks, including ImageNet-A [64], ImageNet-C [24], ImageNet-R [23], and ImageNet-P [24]. Then, in Section 4.2, we take our best pre-trained model and further finetune it on Cityscapes [11] for semantic segmentation. In practice, our models greatly improve mIoU on two popular robustness benchmarks, including Cityscapes-C [33] and ACDC [39], along with competitive performance on clean data. Both the code and pretrained models will be available soon.

4.1. Results on Image Classification

In this experiment, we build our method on top of two state-of-the-art robust architectures: RVT [32] and FAN [67] with the “Base” model size, i.e., RVT-B and FAN-B-Hybrid. We train the models on ImageNet and evaluate them on several robustness benchmarks. We closely follow the settings of RVT and FAN for training. Specifically, we train the models using the same augmentation schemes and adopt the batch size of 2048. We set the learning rate to $2 \times 10^{-3}$ and train all the models for 300 epochs. In all the experiments, by default, we set $K = 4$ and $\lambda = 1$ to train our models. To evaluate robustness, we consider several robustness benchmarks, including ImageNet-A [64], ImageNet-C [24], ImageNet-R [23], and ImageNet-P [24].

4.1.1 Comparisons on ImageNet

As shown in Table 2, compared with the strong baselines RVT and FAN, our models consistently improve the robustness on ImageNet-C by $>2.1\%$ and also yield comparable improvement on other robustness benchmarks, including ImageNet-A/R/SK. Moreover, our models also obtain a competitive improvement in terms of clean accuracy on ImageNet. For example, we improve the accuracy by $>0.4\%$ on both considered baseline architectures. More critically, we highlight that these improvements only come with negligible computational cost in terms of both the number of parameters and the number of floating-point operations (FLOPs). In addition, we also report the detailed corruption error on individual corruption types of ImageNet-C based on FAN-B-Hybrid. From Table 3, our best model (combining TAP and ADL together) obtains the best results on most of the corruption types. It is worth noting that our model is particularly effective against noise corruptions, e.g., yielding a large improvement of $6.25\%$ on Gaussian noise corruption. Overall, these experiments indicate that robustifying attention consistently improves robustness across different architectures and benchmarks.
### 4.1.2 Attention Stability and Visualization Results

We demonstrate that our models greatly improve attention stability against corruptions both qualitatively and quantitatively. Interestingly, in each layer, our models obtain a higher attention diversity among different attention heads.

**Attention stability.** In Figure 3 we first visualize how much the attention would be changed when facing image corruptions, e.g., Gaussian noise. We take FAN-B-Hybrid as the baseline and compare our best model with two variants that only contain TAP and ADL individually. Across diverse examples, the baseline model incurs a severe token overfocusing issue that it puts too much focus on very few tokens and comes with a significant attention shift when facing corruptions. With the help of our local aggregation module TAP, our TAP model assigns the attention to more tokens surrounding some important ones, alleviating the token overfocusing issue to some extent. Nevertheless, we still observe an attention shift between clean and corrupted examples. When training models with our ADL, the attention follows a diagonal pattern such that tokens aggregate information from the others while retaining most of the information from itself. We highlight that this diagonal pattern is somehow similar to the residual learning [22] that additionally learns a residual branch while keeping the features unchanged using an identity shortcut. Clearly, the attention becomes much more stable against corruptions. When combining TAP and ADL together, we further encourage the diagonal pattern to expand within a local region. In this way, each token puts more focus on its neighborhood beyond itself and thus obtains stronger features. Quantitatively, we also evaluate the attention stability by computing cosine similarity of attention between clean and corrupted examples across the whole ImageNet. From Figure 5, TAP yields higher attention stability than the baseline model. Thanks to the diagonal pattern in attention, the model with ADL greatly improves the similarity score by a large margin, indicating that the attention is very stable against corruptions. When combining both TAP and ADL, we can further improve attention stability.

**Attention of each head and attention diversity.** In this part, we further investigate the attention map in each individual head. As shown in Figure 4, for the baseline model, the most important tokens are always the most important ones in almost all the heads, resulting in a very low diversity of attention among different heads. By contrast, in our model, only two heads follow the diagonal pattern and the other six heads have different attention patterns from each other. We highlight that our attention is a combination of both local and global filters w.r.t. different heads. Specifically, these two diagonal heads can be regarded as local filters to extract local information since the diagonal pattern encourages tokens to aggregate information within their local neighborhood. As for the other six heads, the attention is distributed across the whole map and thus they can be regarded as global filters. From the visualization result in Figure 4, our model has a higher diversity of attention among different heads. To quantify this, following the previously discussed approach, we directly compute the cosine similarity of attention maps between any two heads in each layer (see detailed computation method in supple-
Figure 4. Attention maps of different attention heads in the last layer. For the baseline model, the most important tokens are often shared across different heads. In our model, two heads have the attention with the diagonal pattern and the other heads have specific patterns to extract different features, yielding higher attention diversity among heads (see quantitative results in Section 4.1.2).

Figure 5. Distributions of cosine similarity of intermediate attention maps between clean and corrupted examples (e.g., with Gaussian noise) on ImageNet. We demonstrate that either TAP or ADL is able to improve the stability of attention independently. When combining them together, we further improve the stability/similarity of attention against corruptions.

4.2. Results on Semantic Segmentation

In this part, we further apply our method to semantic segmentation tasks. We train the models on Cityscapes [11] and evaluate the robustness on two popular benchmarks, including Cityscapes-C [33] and ACDC [39]. Specifically, Cityscapes-C contains 16 corruption types which can be divided into 4 categories: noise, blur, weather, and digital. ACDC collects the images with adverse conditions, including night, fog, rain, and snow. In this paper, we report mIoU across diverse datasets. During training, we follow the same settings of SegFormer [55] to train our models. We demonstrate that our TAP and ADL also generalize well to segmentation tasks and significantly improve robustness.

<table>
<thead>
<tr>
<th>Model</th>
<th>Cityscapes ↑</th>
<th>Cityscapes-C ↑</th>
<th>ACDC ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepLabv3+ (R101) [7]</td>
<td>77.1</td>
<td>39.4</td>
<td>41.6</td>
</tr>
<tr>
<td>ICNet [63]</td>
<td>65.9</td>
<td>28.0</td>
<td>-</td>
</tr>
<tr>
<td>DilatedNet [60]</td>
<td>68.6</td>
<td>30.3</td>
<td>-</td>
</tr>
<tr>
<td>Swin-T [29]</td>
<td>78.1</td>
<td>47.3</td>
<td>56.3</td>
</tr>
<tr>
<td>SETR [65]</td>
<td>79.5</td>
<td>63.1</td>
<td>60.2</td>
</tr>
<tr>
<td>Segformer-B5 [55]</td>
<td>82.4</td>
<td>65.8</td>
<td>62.0</td>
</tr>
<tr>
<td>FAN-B-Hybrid [67]</td>
<td>82.2</td>
<td>67.3</td>
<td>60.6</td>
</tr>
<tr>
<td>+ TAP</td>
<td>82.7 (+0.5)</td>
<td>69.2 (+1.9)</td>
<td>62.7 (+2.1)</td>
</tr>
<tr>
<td>+ ADL</td>
<td>82.4 (+0.2)</td>
<td>69.4 (+2.1)</td>
<td>63.1 (+2.5)</td>
</tr>
<tr>
<td>+ TAP &amp; ADL</td>
<td>82.9 (+0.7)</td>
<td>69.7 (+2.4)</td>
<td>63.6 (+3.0)</td>
</tr>
</tbody>
</table>

Table 4. Comparisons of semantic segmentation models on Cityscapes validation set, Cityscapes-C, and ACDC test set. Both our TAP and ADL greatly improve the robustness. We can further improve the results when combining TAP and ADL together.

4.2.1 Quantitative Comparisons

As shown in Table 4, compared with the considered baseline model, using either our TAP or ADL individually can greatly improve the robustness on Cityscapes-C and ACDC. With the help of our effective local aggregation module TAP, we highlight that we also obtain a promising improvement of 0.5% mIoU on clean data. When combining TAP and ADL together, we further improve the performance and obtain a larger improvement of 2.4% and 3.0% on Cityscapes-C and ACDC, respectively. Moreover, our best model also significantly outperforms several popular segmentation models with comparable model size. Overall, these results indicate that the proposed two techniques not only work for image classification but also generalize well to semantic segmentation tasks.

4.2.2 Visual Comparisons

In this part, we compare the visualization results of the predicted segmentation masks based on examples of diverse robustness benchmarks. As shown in Figure 6, for the first example with snow corruptions, the baseline model cannot detect a large region of road (highlighted by the red
5. Analysis and Discussions

In the following, we present further ablation experiments and discussions. In Section 5.1, we demonstrate that the proposed two methods are general techniques and can be applied on top of diverse transformer architectures. In Section 5.2, we study the impact of the number of branches $K$ in TAP and the weight of ADL in the training loss. In Section 5.3, we investigate the strategy of how to allocate TAP layers. In practice, uniformly allocating a TAP into every block yields the best results. In Section 5.4, we study the effect of the attention threshold $\tau$ used in Eqn. (1).

5.1. Effectiveness on Diverse Architectures

Besides RVT and FAN, we additionally apply our methods on top of more transformer architectures, including DeiT [43] and Swin [29]. In this experiment, we report the accuracy on ImageNet and robustness in terms of mCE (the lower the better) on ImageNet-C. As shown in Table 5, based on DeiT-B, we greatly improve the corruption robustness by reducing the mCE by 1.9% and yield a promising improvement of 0.4% on clean data. As for Swin-B, we obtain a similar observation that our methods are particularly effective in improving corruption robustness, reducing mCE from 54.4% to 51.9%. These results indicate that our methods can generalize well across diverse architectures.

<table>
<thead>
<tr>
<th>Method</th>
<th>ImageNet</th>
<th>ImageNet-C (mCE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeiT-B [43]</td>
<td>82.0</td>
<td>48.5</td>
</tr>
<tr>
<td>+ TAP &amp; ADL</td>
<td>82.4 (+0.4)</td>
<td>46.6 (-1.9)</td>
</tr>
<tr>
<td>Swin-B [29]</td>
<td>83.4</td>
<td>54.4</td>
</tr>
<tr>
<td>+ TAP &amp; ADL</td>
<td>84.0 (+0.6)</td>
<td>51.9 (-2.5)</td>
</tr>
<tr>
<td>RVT-B [32]</td>
<td>82.6</td>
<td>46.8</td>
</tr>
<tr>
<td>+ TAP &amp; ADL</td>
<td>83.1 (+0.5)</td>
<td>44.7 (-2.1)</td>
</tr>
<tr>
<td>FAN-B-Hybrid [67]</td>
<td>83.9</td>
<td>46.1</td>
</tr>
<tr>
<td>+ TAP &amp; ADL</td>
<td>84.3 (+0.4)</td>
<td>43.7 (-2.4)</td>
</tr>
</tbody>
</table>

Table 5. Results on top of diverse architectures. We report accuracy and mean corruption error (mCE) on ImageNet and ImageNet-C, respectively. Our method consistently improves robustness and accuracy across different architectures.

5.2. Impact of Hyperparameters $K$ and $\lambda$

We conduct ablations on ImageNet-C to study the impact of two hyperparameters of our methods, including the number of branches $K$ in TAP and the weight of ADL.

The number of branches $K$. As detailed in Figure 2, we build our TAP with $K$ branches to enable tokens to consider diverse pooling areas. In fact, the value of $K$ is an important factor for the performance of our method. It is worth noting that $K = 1$ is essentially equivalent to the baseline model without TAP. As shown in Figure 7 (top), we greatly improve the robustness with a lower mCE score on ImageNet-C when gradually increasing $K$ from 1 to 4. If we further increase $K$, learning weights for too many candidate pooling areas (dilations) for each token becomes increasingly difficult and we cannot observe a significant improvement. Although additional branches only introduce minimal overhead in terms of model size and computational complexity, a large $K$ would inevitably require a larger memory footprint. Thus, we choose $K = 4$ to obtain the best results at the minimal cost of extra memory footprint.
With the standard cross-entropy loss during training. In this part, we explicitly investigate how to allocate TAP. Taking FAN-B-Hybrid as the baseline, allocating TAP into every block yields better results than non-uniform allocation strategies in terms of both accuracy and robustness. Table 6. Comparisons of accuracy on ImageNet and mCE (lower is better) on ImageNet-C against the number of branches $K$ (top) and the importance of our ADL loss $\lambda$ (bottom). Top: When only introducing TAP without ADL, our model consistently outperforms the baseline model when increasing the value of $K$ and yields the best result with $K = 4$. Bottom: When using ADL to train the model (without TAP), we observe that a too small or too large $\lambda$ reduces the benefit of our method. In practice, $\lambda = 1$ performs best in most cases.

Table 7. Comparisons of accuracy on ImageNet and mCE (lower is better) on ImageNet-C across diverse $\tau$. We take FAN-B-Hybrid as the baseline and observe that a too small or too large $\tau$ reduces the benefit of ADL. In practice, $\tau = 2$ performs best in most cases.

5.4. Effect of Attention Threshold $\tau$

According to Eqn. (1), we use a threshold $\tau/N$ with $N$ being the number of tokens to filter out very small values and only focus on the most important ones in the attention map via $A_i^{(l)} = 1(A_i^{(l)} \geq \tau/N) \cdot A_i^{(l)}$. Here, we explicitly study the effect of the attention threshold $\tau$ for computing our ADL. As detailed in Table 7, when using ADL to train the model (without TAP), our ADL only brings marginal improvements in terms of clean performance on ImageNet. However, our ADL becomes particularly effective in improving robustness, e.g., greatly reducing mCE on ImageNet-C. In practice, a too small or too large $\tau$ reduces the benefit of our ADL. In our experiments, we set $\tau = 2$ performs to obtain the best results.

6. Conclusion

In this paper, we address the token overfocusing issue in vision transformers (ViTs) such that ViTs tend to rely on very few important tokens in the attention mechanism. In fact, the attention is not robust and often obtains highly diverging attention patterns in the presence of corruptions. To alleviate this, we propose two general techniques. First, our Token-aware Average Pooling (TAP) module encourages the local neighborhood of tokens to take part in the self-attention by learning an adaptive average pooling scheme for each token. Second, our Attention Diversification Loss (ADL) explicitly reduces the cosine similarity of attention among tokens. In practice, we apply our methods to diverse architectures and obtain a significant improvement of robustness on different benchmarks and learning tasks.
References


[25] Zilong Huang, Youcheng Ben, Guozhong Luo, Pei Cheng, Gang Yu, and Bin Fu. Shuffle transformer: Rethinking...


[45] Elena Voita, David Talbot, Fedor Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In Annual Meeting of the Association for Computational Linguistics (ACL), 2019. 2


