# TOKEN-LABEL ALIGNMENT FOR VISION TRANSFORMERS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

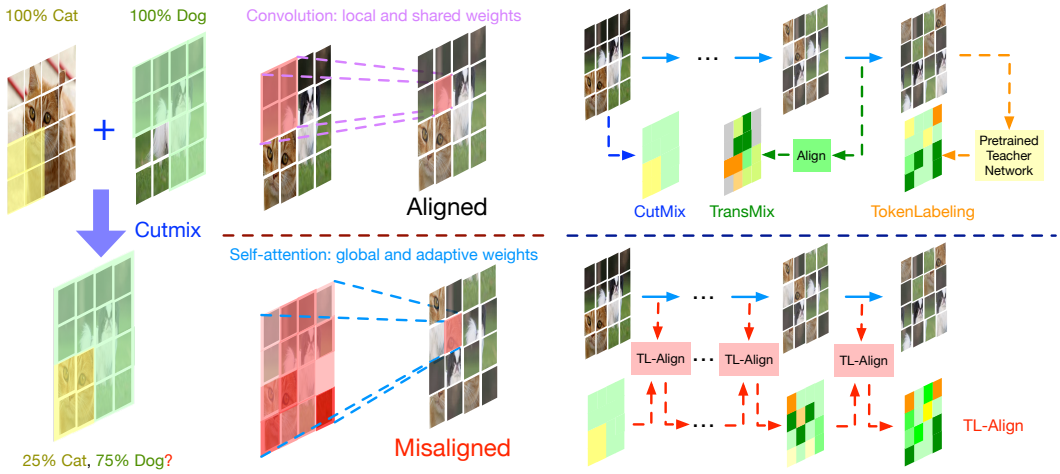
Data mixing strategies (e.g., CutMix) have shown the ability to greatly improve the performance of convolutional neural networks (CNNs). They mix two images as inputs for training and assign them with a mixed label with the same ratio. While they are shown effective for vision transformers (ViTs), we identify a token fluctuation phenomenon that has suppressed the potential of data mixing strategies. We empirically observe that the contributions of input tokens fluctuate as forward propagating, which might induce a different mixing ratio in the output tokens. The training target computed by the original data mixing strategy can thus be inaccurate, resulting in less effective training. To address this, we propose a token-label alignment (TL-Align) method to trace the correspondence between transformed tokens and the original tokens to maintain a label for each token. We reuse the computed attention at each layer for efficient token-label alignment, introducing only negligible additional training costs. Extensive experiments demonstrate that our method improves the performance of ViTs on image classification, semantic segmentation, objective detection, and transfer learning tasks.

## 1 INTRODUCTION

The recent developments of vision transformers (ViTs) have revolutionized the computer vision field and set new state-of-the-arts in a variety of tasks, such as image classification (Dosovitskiy et al., 2020; Touvron et al., 2021b; Liu et al., 2021a; Chu et al., 2021), object detection (Carion et al., 2020; Zhu et al., 2020; Dai et al., 2021a;b), and semantic segmentation (Li et al., 2017; Strudel et al., 2021; Zheng et al., 2021; Cheng et al., 2021). The successful structure of alternative spatial mixing and channel mixing in ViTs also motivates the arising of high-performance MLP-like deep architectures (Tolstikhin et al., 2021; Touvron et al., 2021a; Tang et al., 2022; Wei et al., 2022) and promotes the evolution of better CNNs (Ding et al., 2022; Liu et al., 2022; Guo et al., 2022). In addition to architecture designs, an improved training strategy can also greatly boost the performance of a trained deep model (Jiang et al., 2021; Touvron et al., 2022; Chen et al., 2022; 2021b).

The training of modern deep architecture almost all adopts data mixing strategies for data augmentation (Walawalkar et al., 2020; Uddin et al., 2020; Kim et al., 2020; Verma et al., 2019; Yun et al., 2019; Zhang et al., 2018), which have been proven to consistently improve the generalization performance. They randomly mix two images as well as their labels with the same mixing ratio to produce mixed data. As the most commonly used data mixing strategy, CutMix (Yun et al., 2019) performs a copy-and-paste operation on the spatial domain to produce spatially mixed images. While data mixing strategies have been widely studied for CNNs (Walawalkar et al., 2020; Uddin et al., 2020; Kim et al., 2020), few works have explored their compatibilities with ViTs (Chen et al., 2021b). We find that self-attention in ViTs causes a fluctuation of the original spatial structure. Unlike the translation equivalence that ensures a global label consistency for CNNs, self-attention in ViTs undermines this global consistency and causes a misalignment between the token and label. This misalignment induces a different mixing ratio in the output tokens. The training targets computed by the original data mixing strategies can then be inaccurate, resulting in less effective training.

To address this, we propose a token-label alignment (TL-Align) method for ViTs to obtain a more accurate target for training. We present an overview of our method in Figure 1. We first assign a label to each input token in the mixed image according to the source of the token. We then trace the correspondence between the input tokens and the transformed tokens and align the labels ac-




(a) CutMix (b) Token misalignment for ViTs (c) Different ways to obtain the mixed labels

Figure 1: An overview of the proposed TL-Align method. (a) CutMix-like methods (Yun et al., 2019) are widely used in model training, which spatially mix the tokens and their labels in the input space. (b) They are originally designed for CNNs and assume the processed tokens are spatially aligned with the input tokens. We show that it does not hold true for ViTs due to the global receptive field and the adaptive weights. (c) Compared with existing methods, our method can effectively and efficiently align the tokens and labels without requiring a pretrained teacher network.

cordingly. We assume that only the spatial self-attention and residual connection operation alter the presence of input tokens since channel MLP and layer normalization process each token independently. We reuse the computed attentions to linearly mix the labels of input tokens to obtain those of transformed tokens. The token-label alignment is performed iteratively to obtain a label for each output token. For class-token-based classification (e.g., ViT (Dosovitskiy et al., 2020) and DeiT (Touvron et al., 2021b)), we directly use the aligned label for the output class token as the training target. For global-pooling-based classification (e.g., Swin (Liu et al., 2021a)), we similarly average the labels of output tokens as the training target. The proposed TL-Align is only used for training to improve performance and introduces no additional workload for inference. We apply the proposed TL-Align to various ViT variants with CutMix including plain ViTs (DeiT (Touvron et al., 2021b)) and hierarchical ViTs (Swin (Liu et al., 2021a)). A consistent performance boost is observed across different models on ImageNet-1K (Deng et al., 2009). Specifically, our TL-Align improves DeiT-S by 0.8% using the same training recipe. We also evaluated the ImageNet-pretrained models on various downstream tasks including semantic segmentation, objection detection, and transfer learning. Experimental results verify the generalization ability of our method.

## 2 Related Work

**Vision Transformer.** Transformers have been widely used in natural language processing and achieved great success on many language tasks. Recently, Vision Transformers (ViTs) have aroused extensive interest in computer vision due to their competitive performance compared with CNNs (Dosovitskiy et al., 2020; Touvron et al., 2021b; Liu et al., 2021a; Chu et al., 2021). Dosovitskiy et al. (2020) firstly introduced transformers into the image classification task. They split the input image into non-overlapped patches and then feed them into the transformer encoders. Liu et al. (2021a) proposed a shifted windowing scheme to produce hierarchical feature maps suitable for dense prediction tasks. The great potential of vision transformer has motivated its adaptation to many challenging tasks including object detection (Dai et al., 2021a; Zhu et al., 2020; Carion et al., 2020), segmentation (Cheng et al., 2021; Strudel et al., 2021), image enhancement (Chen et al., 2021a; Li et al., 2021) and video understanding (Liu et al., 2021b; Arnab et al., 2021).

Recently, some efforts have been devoted to producing better training targets to improve the performance of vision transformers (Jiang et al., 2021; Touvron et al., 2022). For example, DeiT (Touvron et al., 2021b) introduces a knowledge distillation procedure to reduce the training cost of ViTs and achieves a better accuracy/speed trade-off. TokenLabeling (Jiang et al., 2021) employs a pretrained

teacher annotator to predict a label for each token for dense knowledge distillation. Differently, we do not require a pretrained network to obtain the training targets. Our TL-Align maintains an aligned label for each token layer by layer and can be trained efficiently in an end-to-end manner.

**Data Mixing Strategy.** As an important type of data augmentation, data mixing strategies have demonstrated a consistent improvement in the generalization performance of CNNs. Zhang et al. (2018) first proposed to combine a training pair to create augmented samples for model regularization. They perform linear interpolations on both the input images and associated targets. Following MixUp, CutMix (Yun et al., 2019) also utilizes the mixture of two input images but adopts a region copy-and-paste operation. Later methods including Puzzle Mix (Kim et al., 2020), SaliencyMix (Uddin et al., 2020) and Attentive CutMix (Walawalkar et al., 2020) leverage the salient regions for informative mixture generation. Recently, Yang et al. (2022) proposed a RecursiveMix strategy which employs the historical input-prediction-label triplets for scale-invariant feature learning. Despite the better performance, a drawback of these methods is the heavily increased training cost due to the saliency extraction or historical information exploitation.

Most existing data mixing methods are originally designed for CNNs, and their effectiveness on ViTs has not been well explored. TransMix (Chen et al., 2021b) utilizes the class attention map at the last layer to re-weight the mixing targets. They assume that the output tokens keep spatial correspondence with the input tokens. However, we identify a token fluctuation phenomenon for ViTs which may cause a mismatch between the tokens and their labels. This mismatch leads to inaccurate label assignments in both the original CutMix and TransMix. To address this, we propose to align the label space with the token space by tracing their correspondence in a layerwise manner.

# 3 Proposed Approach

## 3.1 Preliminaries

The convolution neural network (CNN) has been the dominant architecture for computer vision in the deep learning era, greatly improving the performance of many tasks. Its monopoly has been challenged by the recent emergence of vision transformers (ViTs), which first "patchify" each image into tokens and process them with alternating self-attention (SA) and multi-layer perceptron (MLP).

In addition to architecture design, training strategy also has a large effect on the model performance, especially the data augmentation strategy. Data mixing (Walawalkar et al., 2020; Uddin et al., 2020; Kim et al., 2020; Verma et al., 2019; Yun et al., 2019; Zhang et al., 2018) is an important set of data augmentation for the training of both CNNs and ViTs, as it significantly improves the generalization ability of models. As the most commonly used data mixing strategy, CutMix (Yun et al., 2019) aims to create virtual training samples from the given training samples $(\mathbf{X}, y)$, where $\mathbf{X} \in \mathcal{R}^{H\times W\times C}$ denotes the input image and $y$ is the corresponding label. CutMix randomly selects a local region from one input $\mathbf{X}_1$ and uses it to replace the pixels in the same region of another input $\mathbf{X}_2$ to generate a new sample $\tilde{\mathbf{X}}$. Similarly, the label $\tilde{y}$ of $\tilde{\mathbf{x}}$ is also the combination of the original labels $y_1$ and $y_2$:

$$\begin{aligned} \tilde{\mathbf{X}} &= \mathbf{M} \odot \mathbf{X}_1 + (\mathbf{1} - \mathbf{M}) \odot \mathbf{X}_2 \\ \tilde{y} &= \lambda y_1 + (1-\lambda) y_2 \end{aligned} \tag{1}$$

where $M \in \{0,1\}^{H\times W}$ is a binary mask indicating the image each pixel belongs to, $\mathbf{1}$ is an all-one matrix, and $\odot$ is the element-wise multiplication. $\lambda$ reflects the mixing ratio of two labels and is the proportion of pixels cropped from $\mathbf{X}_1$ in the mixed image $\tilde{\mathbf{X}}$. For a cropped region $[r_x, r_x + r_w] \times [r_y, r_y + r_h]$ from $\mathbf{X}_1$, we compute $\lambda = \frac{r_w r_h}{WH}$ to obtain the initial mixed target $\tilde{y}$.

## 3.2 The Token Fluctuation Phenomenon

CutMix is originally designed for CNNs and assumes the feature extraction process does not alter the mixing ratio. However, we discover that different from CNNs, self-attention in ViTs can lead to the fluctuation of some tokens. The fluctuation further results in the mismatch between the token space and label space, which hinders the effective training of the network.

Formally, we use $\mathbf{z}_i$ to denote a token of the image $\mathbf{Z}$, i.e., $\mathbf{z}_i$ is the transposed $i$-th column vector of $\mathbf{Z}$. We can then compute the $i$-th transformed token $\hat{\mathbf{z}}_i$ after the spatial operation as $\hat{\mathbf{z}}_i = \sum_{j=1}^{N} w^s_{i,j} \mathbf{z}_j$, where $w^s_{i,j}$ is the $i,j$-th element of the computed spatial mixing matrix $\mathbf{w}^s(\mathbf{z})$.

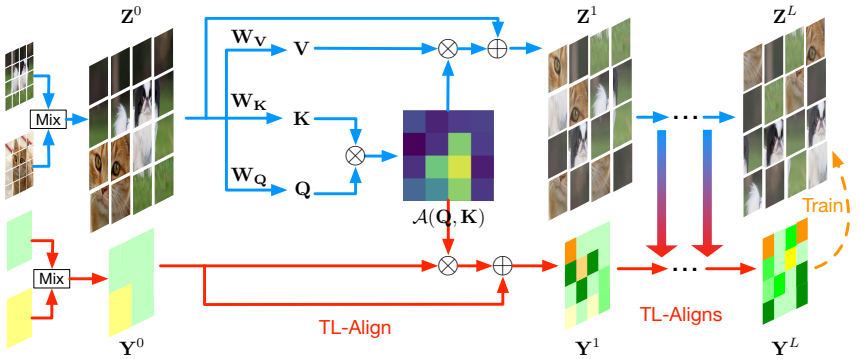



Figure 2: Illustration of the proposed TL-Align. We trace the correspondence between the input tokens and the transformed tokens and align the labels accordingly. We reuse the computed attentions to linearly mix the labels of input tokens to obtain those of transformed tokens. The token-label alignment is performed iteratively to obtain a label for each output token.

With the assumption of the linear information integration, we define the contribution of an original token $\mathbf{z}_i$ to an mixed token $\hat{\mathbf{z}}_j$ as $c(\mathbf{z}_i, \hat{\mathbf{z}}_j) = \frac{|w^s_{i,j}|}{\sum_{k=1}^{N} |w^s_{k,j}|}$, where $|\cdot|$ denotes the absolute value. We can then compute the presence of a token $\mathbf{z}_i$ in all the mixed image tokens as:

$$p(\mathbf{z}_i) = \sum_{j=1}^{N} c(\mathbf{z}_i, \hat{\mathbf{z}}_j) = \sum_{j=1}^{N} \frac{|w^s_{i,j}|}{\sum_{k=1}^{N} |w^s_{k,j}|}. \tag{2}$$

For non-strided depth-wise convolution, each token is multiplied by each element in the convolutional kernel due to the translation invariance. We thus have the following equations:

$$\sum_{l=1}^{N} |w^s_{i,l}| = \sum_{j=1}^{N} |w^s_{k,j}| = \sum_{k=1,l=1}^{M} |K_{k,l}|, \quad \forall i, j \in \mathbf{P}_{NE}, \tag{3}$$

where $\mathbf{P}_{NE}$ denotes the set of positions that are not at the edge of the image, $K_{k,l}$ denotes the value of the $k, l$-th position of the convolution kernel $\mathbf{K}$ and $M$ is the kernel size. We can infer that $p(\mathbf{z}_i) = 1,\ \forall i \in \mathbf{P}_{NE}$, i.e., the effect of all the internal tokens does not change during the convolution process. However, for self-attention in ViTs, Eq. (3) does not hold due to the non-existence of translation invariance. The fluctuation of $p(\mathbf{z})$ is further amplified by the input dependency of the spatial mixing matrix $\mathbf{w}^s(\mathbf{z})$ induced by self-attention. As an extreme case, we may obtain $p(\mathbf{z}) \sim 0$ for certain tokens. The fluctuation of tokens will alter the proportion of mixing (i.e., $\lambda$) and the network might even completely ignore one of the mixed images. The actual label of the processed tokens can then deviate from the mixed label computed by Eq. (1), resulting in less effective training.

### 3.3 Token-Label Alignment

In vision transformers, each token interacts with other tokens globally using the self-attention mechanism. The input-dependent weights empower ViTs with more flexibility but also result in a mismatch between the processed token and the initial token. To address this, we propose a token-label alignment (TL-Align) method to trace the correspondence between the input tokens and transformed tokens to obtain the aligned labels for the resulting representations, as illustrated in Figure 2.

Specifically, ViTs first split the mixed input $\tilde{\mathbf{X}}$ after CutMix (Eq. (1)) to a sequence of $N$ non-overlapped patches and then flatten them to obtain the original image tokens $\{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \cdots, \tilde{\mathbf{x}}_N\}$. We then project them into a proper dimension and add positional embeddings:

$$\mathbf{Z}^0 = [\tilde{\mathbf{z}}_{cls}; \tilde{\mathbf{x}}_1 \cdot \mathbf{E}; \tilde{\mathbf{x}}_2 \cdot \mathbf{E}; \cdots; \tilde{\mathbf{x}}_N \cdot \mathbf{E}] + \mathbf{E}_{pos}, \tag{4}$$

where $\tilde{\mathbf{z}}_{cls} \in \mathcal{R}^{1\times d}$ denotes the class token, N is the number of tokens, $\mathbf{E}$ represents the patch projector, and $\mathbf{E}_{pos} \in \mathcal{R}^{(N+1)\times d}$ is the position embeddings. Note that we adopt the process of the original transformer architecture (Dosovitskiy et al., 2020) as an example without loss of generality. Other models may omit the class token and use a relative positional embedding instead, which does not affect the utility of the proposed TL-Align method.

We first assign each token $\mathbf{z}_i \in \mathcal{R}^{1\times d}$ with a label embedding $\mathbf{y}_i \in \mathcal{R}^{1\times C}$:

$$\mathbf{Y}^0 = [\tilde{\mathbf{y}}^0_{cls}; \tilde{\mathbf{y}}^0_1; \tilde{\mathbf{y}}^0_2; ...; \tilde{\mathbf{y}}^0_N], \tag{5}$$

where the sum of elements in each $\mathbf{y}_i$ equals 1 (i.e., $\sum_{j=1}^{C} y_{i,j} = 1$) and $y_{i,j}$ indicates how much the $i$-th token belong to the $j$-th class. We initialize the label embedding following the conventional data mixing paradigm. For example, when using CutMix to mix two images $\mathbf{X}_1$ and $\mathbf{X}_2$ from the $j$-th class and the $k$-th class with a mixing ratio of $\lambda$, we set $\tilde{y}_{cls,j} = \lambda$ and $\tilde{y}_{cls,k} = 1 - \lambda$ for the class token. For each patch token, we set $\tilde{y}_{i,j} = 1$ if it comes from $\mathbf{X}_1$ and $\tilde{y}_{i,k} = 1$ if it comes from $\mathbf{X}_2$. If a patch token contains both the mixed images, we use the mixing ratio within this patch as the label. For MixUp, we can simply set all label embeddings $\{\tilde{\mathbf{y}}_i\}$ with $\tilde{y}_{,j} = \lambda$ and $\tilde{y}_{,j} = 1 - \lambda$.

We perform TL-Align in a layer-wise manner and compute the aligned labels based on the operation on the tokens. Formally, ViTs use self-attention to perform spatial mixing of the input tokens $\mathbf{Z}$:

$$\begin{aligned} &\mathbf{Q} = \mathbf{Z}\cdot\mathbf{W}_\mathbf{Q}, \mathbf{K} = \mathbf{Z}\cdot\mathbf{W}_\mathbf{K}, \mathbf{V} = \mathbf{Z}\cdot\mathbf{W}_\mathbf{V}, \\ &\mathcal{A}(\mathbf{Q},\mathbf{K}) = \text{Softmax}(\mathbf{Q}\cdot\mathbf{K}^T/\sqrt{d}), \\ &\hat{\mathbf{Z}} = \text{SA}(\mathbf{Z}) = \mathcal{A}(\mathbf{Q},\mathbf{K})\cdot\mathbf{V}. \end{aligned} \tag{6}$$

To align the labels, we update the label embeddings $\mathbf{Y}$ using the same attention matrix $\mathcal{A}(\mathbf{Q},\mathbf{K})$:

$$\hat{\mathbf{Y}} = \mathcal{A}(\mathbf{Q},\mathbf{K})\cdot\mathbf{Y}. \tag{7}$$

ViTs usually adopt multi-head self-attention (MSA) to perform multiple self-attentions parallelly:

$$\hat{\mathbf{Z}} = \text{MSA}(\mathbf{Z}) = [\text{SA}_1(\mathbf{Z}); \text{SA}_2(\mathbf{Z}); \cdots; \text{SA}_H(\mathbf{Z})]\cdot\mathbf{w}_h, \tag{8}$$

where $H$ is the number of heads and $\mathbf{w}_h \in \mathcal{R}^{d\times d}$. We then adapt our label alignment to MSA by simply taking the average of all the attention matrices for alignment:

$$\hat{\mathbf{Y}} = \text{TL-Align-S}(\mathbf{Z},\mathbf{Y}) := \frac{1}{H}\sum_{i=1}^{H}\mathcal{A}_i(\mathbf{Q},\mathbf{K})\cdot\mathbf{Y}, \tag{9}$$

where $\mathcal{A}_i$ is the attention matrix corresponding to the $i$-th head $\text{SA}_i$.

Each transformer block $l$ processes the tokens by both spatial and channel mixing:

$$\begin{aligned} &\hat{\mathbf{Z}}^{l-1} = \text{MSA}(\text{LN}(\mathbf{Z}^{l-1})), \;\; \mathbf{Z}'^{l-1} = \hat{\mathbf{Z}}^{l-1} + \mathbf{Z}^{l-1}, \\ &\hat{\mathbf{Z}}^{l} = \text{MLP}(\text{LN}(\mathbf{Z}'^{l-1})), \;\; \mathbf{Z}^{l} = \hat{\mathbf{Z}}^{l} + \mathbf{Z}'^{l-1}, \end{aligned} \tag{10}$$

where MLP and LN denote the MLP module and layer normalization (Ba et al., 2016), respectively. Our TL-Align then aligns the label embeddings in a similar manner:

$$\begin{aligned} &\hat{\mathbf{Y}}^{l-1} = \text{TL-Align-S}(\mathbf{Y}^{l-1}), \;\; \mathbf{Y}'^{l-1} = \text{Norm}(\hat{\mathbf{Y}}^{l-1} + \mathbf{Y}^{l-1}), \\ &\hat{\mathbf{Y}}^{l} = \mathbf{Y}'^{l-1}, \;\; \mathbf{Y}^{l} = \text{Norm}(\hat{\mathbf{Y}}^{l} + \mathbf{Y}'^{l-1}), \end{aligned} \tag{11}$$

where Norm denotes the normalization operation. We implement Norm by a simple average.

Hierarchical vision transformers such as Swin (Liu et al., 2021a) further introduce a patch aggregation operation to merge multiple patches. They usually concatenate multiple tokens across the channels to reduce the spatial resolution. Instead of concatenation, we simply add the label embeddings of the merged tokens followed by normalization as the aligned labels.

We synchronously align the labels with the processed tokens layer by layer and obtain the aligned tokens $\mathbf{Z}^L$ and labels $\mathbf{Y}^L$. The final representation of the image $\mathbf{z}$ is either the class token $\mathbf{z}^L_{cls}$ (Dosovitskiy et al., 2020; Touvron et al., 2021b) or the average pooling of all the spatial tokens $\frac{1}{N}\sum_{i=1}^{N}\mathbf{z}^L_i$ (Liu et al., 2021a). The aligned label $\mathbf{y}_{align}$ for the image is then $\mathbf{y}^L_{cls}$ or $\frac{1}{N}\sum_{i=1}^{N}\mathbf{y}^L_i$ depending on the specific model. We then adopt the aligned label $\mathbf{y}_{align}$ to train the network and can adapt to different loss functions and training schemes:

$$J = J(\mathbf{z}, \text{stop-gradient}(\mathbf{y}_{align})). \tag{12}$$

We do not back-propagate through the aligned label as they only serve as a more accurate target.

Our TL-Align serves as a plug-and-play module on various vision transformers while only introducing negligible training costs. We adjust the label of each token adaptively during the layer-by-layer propagation and preserve alignment between tokens and labels throughout the forward process. TL-Align is only used during training and introduces no additional computation cost when inference.

Table 1: **Results on ImageNet classification task.** We compare the parameters, FLOPs and accuracy of different vision transformer backbones without and with our TL-Align.

| Model | Image Size | Params | FLOPs | Top-1 Acc.(%) | Top-5 Acc.(%) |
|---|---|---|---|---|---|
| DeiT-T | $224^2$ | 5.7M | 1.6G | 72.2 | 91.3 |
| +TL-Align | | | | **73.2** | **91.7** |
| DeiT-S | $224^2$ | 22M | 4.6G | 79.8 | 95.0 |
| +TL-Align | | | | **80.6** | 95.0 |
| DeiT-B | $224^2$ | 86M | 17.5G | 81.8 | 95.5 |
| +TL-Align | | | | **82.3** | **95.8** |
| Swin-T | $224^2$ | 29M | 4.5G | 81.2 | 95.5 |
| +TL-Align | | | | **81.4** | **95.7** |
| Swin-S | $224^2$ | 50M | 8.8G | 83.0 | 96.3 |
| +TL-Align | | | | **83.4** | **96.5** |
| Swin-B | $224^2$ | 88M | 15.4G | 83.5 | 96.4 |
| +TL-Align | | | | **83.7** | **96.5** |

Table 2: **Comparison of our TL-Align with other training strategies on ImageNet.**

| Method | Params | Speed (image/s) | Acc.(%) |
|---|---|---|---|
| Vanilla | 22M | 322 | 76.4 |
| CutMix | 22M | 322 | 79.8 |
| Puzzle-Mix | 22M | 139 | 79.8 |
| SaliencyMix | 22M | 314 | 79.2 |
| Attentive-CutMix | 46M | 239 | 77.5 |
| TransMix | 22M | 322 | 80.1 |
| CutMix + TL-Align | 22M | 311 | **80.6** |

Table 3: **Results on semantic segmentation on the ADE20K dataset.**

| Backbone | Params | FLOPs | mIoU | mIoU (MS) | mAcc |
|---|---|---|---|---|---|
| DeiT-S | 58M | 1032G | 43.8 | 45.1 | 55.2 |
| +TL-Align | | | **44.5** | **45.7** | **55.5** |
| Swin-T | 60M | 945G | 44.4 | 45.8 | 55.6 |
| +TL-Align | | | **44.7** | **46.5** | **56.4** |
| Swin-S | 81M | 1038G | 47.6 | 49.5 | 58.8 |
| +TL-Align | | | **48.0** | **49.7** | **59.5** |
| Swin-B | 121M | 1188G | 48.1 | 49.7 | 59.1 |
| +TL-Align | | | **48.3** | **50.1** | **59.7** |

# 4 EXPERIMENTS

In this section, we conducted extensive experiments to evaluate the proposed TL-Align method. We demonstrate the improvement of TL-Align on various vision transformers and compare it with state-of-the-art training strategies concerning accuracy, network complexity, and training speed. We examine the transferability on downstream tasks including semantic segmentation, object detection, and transfer learning. We further provide in-depth analysis to evaluate the effectiveness of TL-Align.

## 4.1 IMAGENET CLASSIFICATION

**Implementation Details.** We first evaluate our TL-Align method on ImageNet for image classification. We conduct experiments on various transformer architectures: three variants of DeiT (Touvron et al., 2021b) (DeiT-T, DeiT-S, and DeiT-B), and three variants of Swin Transformer (Liu et al., 2021a) (Swin-T, Swin-S, and Swin-B). For tiny and small models, we train the models from scratch for 300 epochs following the same training recipe as (Touvron et al., 2021b) and (Liu et al., 2021a) for fair comparisons. For large models (i.e., Deit-B and Swin-B), we finetune the official pre-trained models for 40 epochs with a constant learning rate of 1e-5 and a weight decay of 1e-8.

**Performance on Different Architectures.** As shown in Table 1, TL-Align steadily improves the performance of different vision transformer architectures. Specifically, TL-Align boosts the top-1 accuracy of DeiT-T, DeiT-S, and DeiT-B by 1.0%, 0.8%, and 0.5%, respectively, in a parameter-free manner. Moreover, our method is generalizable and can be directly applied to hierarchical vision transformers like Swin. It is worth noting that most existing methods need either architecture modifications (adding a class token in (Chen et al., 2021b)) or extra computations (saliency map extraction in (Uddin et al., 2020)) when applied to Swin. In contrast, our TL-Align method can be used as a plug-and-play module and achieves consistent improvement on variants of Swin.

**Comparison with Other Training Strategies.** We also compare our method with the state-of-the-art training strategies for data mixing on DeiT-S, including CutMix (Yun et al., 2019), Puzzle-Mix (Kim et al., 2020), SaliencyMix (Uddin et al., 2020), Attentive-CutMix (Walawalkar et al., 2020), and TransMix (Chen et al., 2021b). Specifically, we train the DeiT-S model while only disabling CutMix as the baseline method, which is denoted as Vanilla in Table 2. Moreover, since TransMix (Chen et al., 2021b) reports the EMA accuracy with different hyperparameters, we repro-

Table 4: **Experimental results on object detection and instance segmentation on COCO.**

| Backbone | Params | FLOPs | Schedule | $AP^{box}$ | $AP^{box}_{50}$ | $AP^{box}_{75}$ | $AP^{mask}$ | $AP^{mask}_{50}$ | $AP^{mask}_{75}$ |
|---|---|---|---|---|---|---|---|---|---|
| Swin-T | 86M | 745G | 3x | 50.4 | 69.2 | 54.7 | 43.7 | 66.6 | 47.3 |
| +TL-Align | | | | **50.5** | **69.4** | **54.9** | **43.8** | 66.6 | 47.3 |
| Swin-S | 107M | 838G | 3x | 51.9 | 70.7 | 56.3 | 45.0 | 68.2 | 48.8 |
| +TL-Align | | | | **52.2** | **71.1** | **56.7** | **45.2** | **68.4** | **49.1** |
| Swin-B | 145M | 982G | 3x | 51.9 | 70.5 | 56.4 | 45.0 | 68.1 | 48.9 |
| +TL-Align | | | | **52.3** | **71.2** | **56.9** | **45.3** | **68.7** | **49.1** |

Table 5: **The accuracy and model complexity on different transfer learning datasets**.

| Model | Params | FLOPs | C-10 | C-100 | Flowers | Cars |
|---|---|---|---|---|---|---|
| ResNet50 | 26M | 4.1G | - | - | 96.2 | 90.0 |
| ViT-B/16 | 86M | 55.4G | 98.1 | 87.1 | 89.5 | - |
| ViT-L/16 | 307M | 190.7G | 97.9 | 86.4 | 89.7 | - |
| Deit-T | 5.7M | 1.6G | 97.6 | 85.7 | 97.1 | 90.1 |
| +TL-Align | 5.7M | 1.6G | **97.8** | **86.4** | **97.9** | **90.7** |
| Deit-S | 22M | 4.6G | 97.9 | 90.2 | 98.1 | 91.4 |
| +TL-Align | 22M | 4.6G | **98.8** | **90.4** | **98.3** | **91.8** |
| Deit-B | 86M | 17.5G | 99.1 | 90.8 | 98.4 | 92.1 |
| +TL-Align | 86M | 17.5G | 99.1 | 90.5 | **98.6** | **93.0** |

duce it under the same training recipe (Touvron et al., 2021b) for a fair comparison. As demonstrated in Table 2, TL-Align shows significantly better performance than the other mixup variants while maintaining the number of parameters and training speed. Puzzle-Mix obtains the same classification accuracy as CutMix but results in a much lower training speed as it relies on an extra model to get the optimal solution. SaliencyMix and Attentive-CutMix lead to performance degeneration when built upon DeiT-S backbone. Notably, our method also achieves higher top-1 accuracy than ViT-targeted TransMix. Due to the token fluctuation phenomenon, the class token attention utilization in TransMix can not reflect the actual contribution of different tokens. Differently, TL-Align obtains accurate alignment of the tokens and labels, resulting in improved performance.

### 4.2 Downstream Tasks

**Semantic Segmentation.** We evaluate our TL-Align on ADE20K dataset (Lin et al., 2014) for semantic segmentation. We adopt DeiT-S and three variants of Swin Transformer as backbones equipped with UpperNet for segmentation. As presented in Table 3, TL-Align improves the segmentation performance on both DeiT and Swin at different model scales, showing its effectiveness.

**Object Detection and Instance Segmentation.** We also examine the performance of TL-Align on object detection and instance segmentation on the COCO 2017 dataset (Lin et al., 2014). We apply our TL-Align to Swin (Liu et al., 2021a) due to the advantage of the hierarchical representations on object detection tasks. We adopt the Cascade Mask-RCNN (Cai & Vasconcelos, 2018) framework and use the training strategy of 3x schedule. As shown in Table 4, we observe consistent improvements on all variants of Swin Transformer. This demonstrates the advantages of our method for learning token-level meaningful features suitable for dense prediction tasks.

**Transfer Learning.** We further evaluate the transferred classification performance of TL-Align on CIFAR-10 (Krizhevsky et al., 2009), CIFAR-100 (Krizhevsky et al., 2009), Flowers (Nilsback & Zisserman, 2008) and Cars (Krause et al., 2013). We use pre-trained models on ImageNet and finetune them on these datasets following existing works (Touvron et al., 2021b). We compare the performance with and without TL-Align on three variants of DeiT (Touvron et al., 2021b), as shown in Table 5. TL-Align obtains significant performance gains for all variants on the four datasets.

### 4.3 Performance Analysis and Visualization

**Effectiveness of Token-Label Alignment.** We first quantize the difference between the original targets and aligned labels and investigate its correlation with the model. Specifically, we compute the Root Mean Square Error (RMSE) between the original targets and labels obtained by our TL-

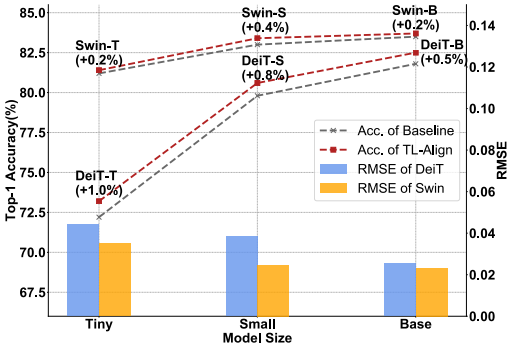



Figure 3: **The Root Mean Square Error (RMSE) between original CutMix targets and labels obtained by T-L Align.** We show results on variants of DeiT and Swin.

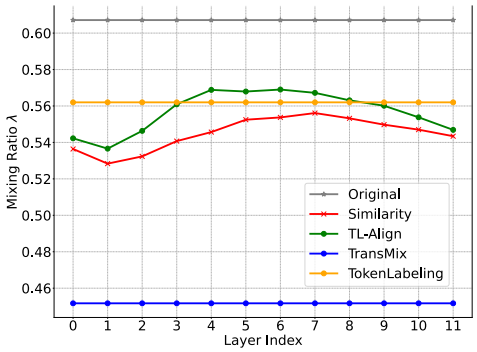



Figure 4: **Visualization of mixing ratio $\lambda$ of fluctuating tokens from different layers.** We compare the results of TL-Align with CutMix, token similarity, TransMix, and TokenLabeling.

Table 6: **Comparison results of model generalization ability and robustness.** We evaluate them on various out-of-distribution/corrupted datasets and against adversarial attacks.

| Model | FLOPs | Params | ImageNet | | Generalization | Robustness | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Top-1↑ | Top-5↑ | IN-V2↑ | IN-A↑ | IN-C↓ | IN-R↑ | AutoAttack↑ |
| DeiT-T | 5.7M | 1.6G | 72.2 | 91.3 | 60.4 | 7.7 | 69.1 | 34.1 | 3.9 |
| +TL-Align | 5.7M | 1.6G | **73.2** | **91.7** | **61.4** | 6.1 | **68.0** | **34.6** | **4.4** |
| DeiT-S | 22M | 4.6G | 79.8 | 95.0 | 68.5 | 18.9 | 54.7 | 42.5 | 6.9 |
| +TL-Align | 22M | 4.6G | **80.6** | 95.0 | **68.9** | **19.2** | **53.2** | **43.2** | **7.5** |
| DeiT-B | 86M | 17.5G | 81.8 | 95.5 | 70.5 | 27.9 | 48.5 | 45.3 | - |
| +TL-Align | 86M | 17.5G | **82.3** | **95.8** | **70.9** | **29.0** | **47.1** | 44.4 | - |

Align. As shown in Figure 3, the RMSE decreases when enlarging the model size. This indicates that larger models demonstrates less token fluctuation. Moreover, the RMSE for Swin Transformer tends to be lower compared with DeiT of a similar model size. This is due to the adopted local-window self-attention in Swin which preserves more local information. These observations are consistent with our experimental results: the improvements on small models and DeiT-like backbones tend to be more significant as they encounter more token fluctuation.

**Visualization of the Layer-wise Mixing Ratio of Fluctuated Tokens.** To investigate the effectiveness of TL-Align, we compute a similarity-based "ground-truth" mixing ratio for each layer. Specifically, we compute the similarities of tokens between the mixed and unmixed images and use them as the label of each token. We compare them with the mixing ratios produced by TL-Align, CutMix (Yun et al., 2019), TransMix (Chen et al., 2021b), and TokenLabeling (Jiang et al., 2021). As shown in Figure 4, the similarity-based mixing ratio changes at each layer, resulting from token fluctuation. However, CutMix, TransMix, and TokenLabeling assume the output tokens keep spatial correspondence with the input tokens and compute a fixed mixing ratio. TL-Align assigns dynamic labels to tokens using layer-wise alignment, which is more accurate compared with other methods.

**Evaluation of Robustness and Generalization.** We further conduct experiments to validate the generalization ability and robustness of TL-Align, as shown in Table 6. For robustness evaluation, we employ four corrupted and out-of-distribution datasets including ImageNet-A, ImageNet-C and ImageNet-R. We also adopt AutoAttack (Croce & Hein, 2020) to evaluate the adversarial robustness on ImageNet validation set. Due to the memory limitation, we do not experiment with DeiT-B on AutoAttack. We use mean Corruption Error (mCE, lower is better) for ImageNet-C and Top-1 Accuracy for others as the evaluation metric. For generalization evaluation, we adopt the ImageNet-V2 dataset (Recht et al., 2019). We see that TL-Align improves both the robustness and generalization ability, demonstrating the superiority of adopting TL-Align for pre-training.

**Ablation Study on different Data Mixing Strategies.** Due to the efficiency of the proposed layer-wise alignment, TL-Align can be directly applied to a wide range of data mixing strategies. We adopt MixUp, CutMix, a random mixing strategy and a block-wise mixing strategy to evaluate the generalizability of TL-Align. The random mixing and block-wise mixing strategies are inspired by MAE (He et al., 2022) and BEiT (Bao et al., 2021) and we replace the masking operation with image mixing on patch-level and block-level (both of size 16×16) respectively. The comparison

Table 7: **Ablation of applying TL-Align to different data mixing strategies for DeiT-S training.**

| MixUp | CutMix | Random | Block-wise | Top-1 Acc.(%) | +TL-Align Top-1 Acc.(%) |
|---|---|---|---|---|---|
| × | × | × | × | 76.4 | - |
| × | ✓ | × | × | 79.8 | 80.6 |
| ✓ | ✓ | × | × | 79.8 | 80.2 |
| × | × | ✓ | × | 79.7 | 80.2 |
| × | × | × | ✓ | 80.0 | 80.3 |

Table 8: **Ablation of different token-label alignment operations.**

| Alignment | Top-1 Acc.(%) |
|---|---|
| None (DeiT-S baseline) | 79.8 |
| TL-Align-S (Layer 12) | 80.1 |
| TL-Align-S (Layer 2,4,6,8) | 80.2 |
| Normalization Disabled | 80.3 |
| Default (TL-Align) | **80.6** |

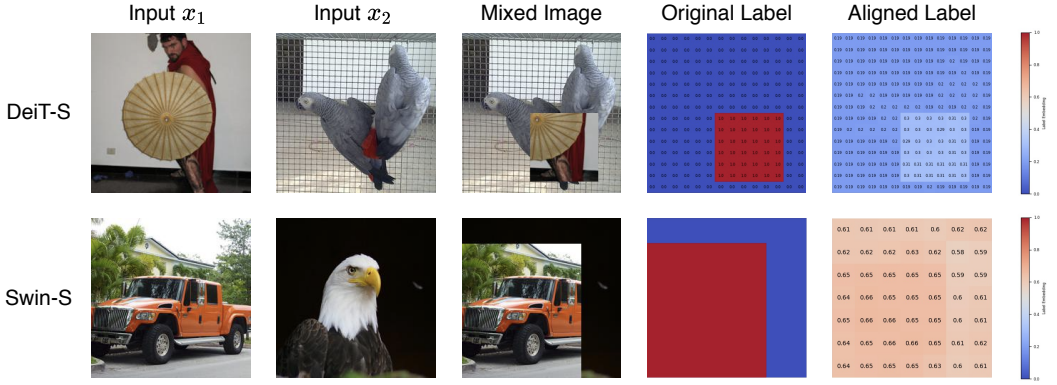



Figure 5: **The visualization of results on DeiT-S and Swin-S.** We visualize the input images, the mixed image, the original label embedding, and the label embedding after token-label alignment.

results of training DeiT-S with and without our approach is demonstrated in Table 7. Specifically, TL-Align improves CutMix by 0.8%, MixUp+CutMix by 0.4%, random mixing by 0.5% and block-wise mixing by 0.3% respectively, further verifying the generalizability of the proposed TL-Align.

**Ablation Study on Different Label Alignment Operations.** Our TL-Align aligns the labels with tokens transformed by spatial self-attention and residual connection layer-by-layer. To investigate the effect of reusing attention maps and normalization, we conduct an ablation study regarding different alignment operations on DeiT-S. We try aligning the labels only by using the attention map of Layer 12, which is equivalent to TransMix (Chen et al., 2021b). We also test the performance of applying alignment to several middle transformer layers and disabling normalization. As presented in Table 8, incomplete alignment at a part of layers marginally boosts the performance as it cannot well handle the token fluctuation issue. Disabling normalization leads to 0.3% accuracy drop due to the inaccurate alignment at the presence of residual connections. This demonstrates the significance of the token-label alignment by attention utilization and normalization in a layer-wise manner.

**Visualizations of Aligned Labels.** We visualize the labels obtained by TL-Align on DeiT-S (Touvron et al., 2021b) and Swin-S (Liu et al., 2021a) as shown in Figure 5. Specifically, the aligned label embedding is obtained after the final transformer block for both DeiT-S and Swin-S. The value of the label embedding represents the probability of the belonged class of the corresponding token. We use red to denote larger probabilities towards the first image and blue for the second image. We observe that the aligned labels can deviate from the original labels and result in different mixing ratios for training. Therefore, using the original ratio as the training target may produce false training signals and lead to inferior performance. We see that our TL-Align can correct the labels when the images are mixed with uninformative tokens. More visualization results are included in Appendix B.

## 5 Conclusion

In this paper, we have presented a token-label alignment method for training better vision transformers. As important subsets of data augmentation methods, data mixing strategies are able to consistently improve the performance of both CNNs and ViTs. We identify a token fading issue for ViTs and address this by tracing the correspondence between transformed tokens and the original tokens to obtain a label for each output token to obtain more accurate training signals. Experimental results have demonstrated that the proposed TL-Align method can uniformly improve the performance of various ViT models. The generalization of TL-Align to other architectures such as MLP-like models remains unknown and is a promising future direction.

## REFERENCES


Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *ICCV*, pp. 6836–6846, 2021.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv*, abs/1607.06450, 2016.

Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, pp. 6154–6162, 2018.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, pp. 213–229, 2020.

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 *CVPR*, pp. 12299–12310, 2021a.

Jie-Neng Chen, Shuyang Sun, Ju He, Philip Torr, Alan Yuille, and Song Bai. Transmix: Attend to mix for vision transformers. *arXiv preprint arXiv:2111.09833*, 2021b.

Tianlong Chen, Zhenyu Zhang, Yu Cheng, Ahmed Awadallah, and Zhangyang Wang. The principle of diversity: Training stronger vision transformers calls for reducing all levels of redundancy. In *CVPR*, 2022.

Bowen Cheng, Alexander G Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. In *NeurIPS*, 2021.

Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. In *NeurIPS*, 2021.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pp. 2206–2216. PMLR, 2020.

Xiyang Dai, Yinpeng Chen, Jianwei Yang, Pengchuan Zhang, Lu Yuan, and Lei Zhang. Dynamic detr: End-to-end object detection with dynamic attention. In *ICCV*, pp. 2988–2997, 2021a.

Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. Up-detr: Unsupervised pre-training for object detection with transformers. In *CVPR*, pp. 1601–1610, 2021b.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pp. 248–255. Ieee, 2009.

Xiaohan Ding, Xiangyu Zhang, Yizhuang Zhou, Jungong Han, Guiguang Ding, and Jian Sun. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *CVPR*, 2022.

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. In *ICLR*, 2020.

Meng-Hao Guo, Cheng-Ze Lu, Zheng-Ning Liu, Ming-Ming Cheng, and Shi-Min Hu. Visual attention network. *arXiv preprint arXiv:2202.09741*, 2022.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8340–8349, 2021a.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15262–15271, 2021b.

Zi-Hang Jiang, Qibin Hou, Li Yuan, Daquan Zhou, Yujun Shi, Xiaojie Jin, Anran Wang, and Jiashi Feng. All tokens matter: Token labeling for training better vision transformers. In *NeurIPS*, 2021.

Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *ICML*, pp. 5275–5285, 2020.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCVW*, pp. 554–561, 2013.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Wenbo Li, Xin Lu, Jiangbo Lu, Xiangyu Zhang, and Jiaya Jia. On efficient transformer and image pre-training for low-level vision. *arXiv preprint arXiv:2112.10175*, 2021.

Xiaoxiao Li, Ziwei Liu, Ping Luo, Chen Change Loy, and Xiaoou Tang. Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade. In *CVPR*, pp. 3193–3202, 2017.

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 *ECCV*, pp. 740–755, 2014.

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. In *ICCV*, 2021a.

Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. *arXiv preprint arXiv:2106.13230*, 2021b.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *arXiv preprint arXiv:2201.03545*, 2022.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, 2008.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pp. 5389–5400. PMLR, 2019.

Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *ICCV*, 2021.

Yehui Tang, Kai Han, Jianyuan Guo, Chang Xu, Yanxi Li, Chao Xu, and Yunhe Wang. An image patch is a wave: Quantum inspired vision mlp. In *CVPR*, 2022.

Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *NeurIPS*, 34, 2021.

Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, et al. Resmlp: Feedforward networks for image classification with data-efficient training. *arXiv preprint arXiv:2105.03404*, 2021a.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, pp. 10347–10357, 2021b.

Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit. *arXiv preprint arXiv:2204.07118*, 2022.

AFM Uddin, Mst Monira, Wheemyung Shin, TaeChoong Chung, Sung-Ho Bae, et al. Saliencymix: A saliency guided data augmentation strategy for better regularization. *arXiv preprint arXiv:2006.01791*, 2020.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *ICML*, pp. 6438–6447, 2019.

Devesh Walawalkar, Zhiqiang Shen, Zechun Liu, and Marios Savvides. Attentive cutmix: An enhanced data augmentation approach for deep learning based image classification. *arXiv preprint arXiv:2003.13048*, 2020.

Guoqiang Wei, Zhizheng Zhang, Cuiling Lan, Yan Lu, and Zhibo Chen. Activemlp: An mlp-like architecture with active token mixer. *arXiv preprint arXiv:2203.06108*, 2022.

Lingfeng Yang, Xiang Li, Borui Zhao, Renjie Song, and Jian Yang. Recursivemix: Mixed learning with history. *arXiv preprint arXiv:2203.06844*, 2022.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, pp. 6023–6032, 2019.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

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 *CVPR*, pp. 6881–6890, 2021.

Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2020.

## A GENERALIZING TL-ALIGN BEYOND VITS

ViTs can achieve better accuracy/computation trade-off than conventional CNNs, where one of the working mechanisms is the alternation between spatial mixing (e.g., SA) and channel mixing (e.g., MLP) (Tolstikhin et al., 2021). Based on this, some works have explored different spatial mixing strategies in addition to self-attention, including spatial MLP (Tolstikhin et al., 2021; Touvron et al., 2021a; Tang et al., 2022; Wei et al., 2022) and depth-wise convolution (Ding et al., 2022; Liu et al., 2022; Guo et al., 2022). For an image $\mathbf{X} \in \mathbb{R}^{H\times W\times C}$, they first perform patch-wise image tokenization to obtain a tokenized image representation $\mathbf{Z} \in \mathbb{R}^{N\times d}$, where $N$ is the number of tokens and $d$ is the number of channels. We formulate modern deep vision networks into various compositions of five operations:

- Spatial mixing: $\mathbf{Z} \leftarrow \mathbf{W}^s(\mathbf{Z}) \cdot \mathbf{Z}$, where $\mathbf{W}^s(\mathbf{Z}) \in \mathbb{R}^{N\times N}$.
- Channel mixing: $\mathbf{Z} \leftarrow \mathbf{Z} \cdot \mathbf{W}^c(\mathbf{Z})$, where $\mathbf{W}^c(\mathbf{Z}) \in \mathbb{R}^{d\times d}$.
- Point-wise transformation: $\mathbf{Z} \leftarrow f(\mathbf{Z})$, where $f$ is a point-wise operation such as bias adding and normalization.
- Residual connection: $\mathbf{Z} \leftarrow \mathbf{Z} + g(\mathbf{Z})$, where $g$ can be one or a composition of the above operations.
- Spatial aggregation: $\mathbf{Z} \leftarrow \text{Aggre}(\{\mathbf{Z}_i\})$, where Aggre typically concatenates multiple tokens across the feature dimension.

For example, MLP-Mixer (Tolstikhin et al., 2021) adopts $\mathbf{W}^s(\mathbf{Z}) = W^s$, where $W^s \in \mathbb{R}^{N\times N}$ is a learnable parameter matrix. ConvNeXt (Liu et al., 2022) adopts $\mathbf{W}^s(\mathbf{Z}) = T(\mathbf{K})$, where $\mathbf{K} \in \mathbb{R}^{7\times 7}$ is a convolutional kernel and $T$ transforms the kernel into a equivalent matrix for direct multiplication.

The proposed TL-Align can be generalized to different architectures by applying the corresponding operations on the label embeddings. We initialize the label embedding following Eq. (5). We detail the label embedding updating for different operations in Table 9. The $\text{Norm}(\cdot)$ operation denotes that we normalize each row vector so that the sum of all elements equals to 1.

Table 9: Updating of the label embeddings for different operations on the tokens.

| Operation | Token Processing | Label Alignment | Example |
|---|---|---|---|
| Spatial mixing | $\mathbf{Z} \leftarrow \mathbf{W}^s(\mathbf{Z}) \cdot \mathbf{Z}$ | $\mathbf{Y} \leftarrow \text{Norm}(\mathbf{W}^s(\mathbf{Z})) \cdot \mathbf{Y}$ | Spatial attention |
| Channel mixing | $\mathbf{Z} \leftarrow \mathbf{Z} \cdot \mathbf{W}^c(\mathbf{Z})$ | $\mathbf{Y} \leftarrow \mathbf{Y}$ | Channel MLP |
| Point-wise transformation | $\mathbf{Z} \leftarrow f(\mathbf{Z})$ | $\mathbf{Y} \leftarrow \mathbf{Y}$ | Layer normalization |
| Residual connection | $\mathbf{Z} \leftarrow \mathbf{Z} + g(\mathbf{Z})$ | $\mathbf{Y} \leftarrow \text{Norm}(\mathbf{Y} + g(\mathbf{Y}))$ | Residual connection |
| Spatial aggregation | $\mathbf{Z} \leftarrow \text{Aggre}(\{\mathbf{Z}_i\})$ | $\mathbf{Y} \leftarrow \text{Norm}(\sum_i \mathbf{Y}_i)$ | Patch merging |

For spatial mixing, we accordingly mix the token embeddings using the same weights as the token processing. For example, for a processed token $\hat{\mathbf{z}} = 0.5 \cdot \mathbf{z}_1 + 0.5 \cdot \mathbf{z}_2$, we similarly compute the aligned label as $\hat{\mathbf{y}} = 0.5 \cdot \mathbf{y}_1 + 0.5 \cdot \mathbf{y}_2$, assuming the label information is linearly addable. As channel mixing and point-wise transformation only reorganize information within each token, they do not alter the label embedding. For residual connection, we similarly add a residual connection to the label embedding before normalization. Spatial aggregation is similar to spatial mixing and also aggregates information among multiple tokens. Therefore, we also need to align the labels by adding their label embeddings before normalization.

We leave the experiments for generalized TL-Align for future works.

## B MORE VISUALIZATIONS

We provide more visualization results of obtained labels of the proposed token-label alignment method in Figure 6. We visualize the input images, the mixed image, the original label embedding, and the label embedding after token-label alignment. Specifically, the aligned label embedding is obtained after the final transformer block for both DeiT-S and Swin-S. The size of the original label embedding is equivalent to the number of input tokens, i.e. $14 \times 14$ for DeiT-S and $56 \times 56$ for

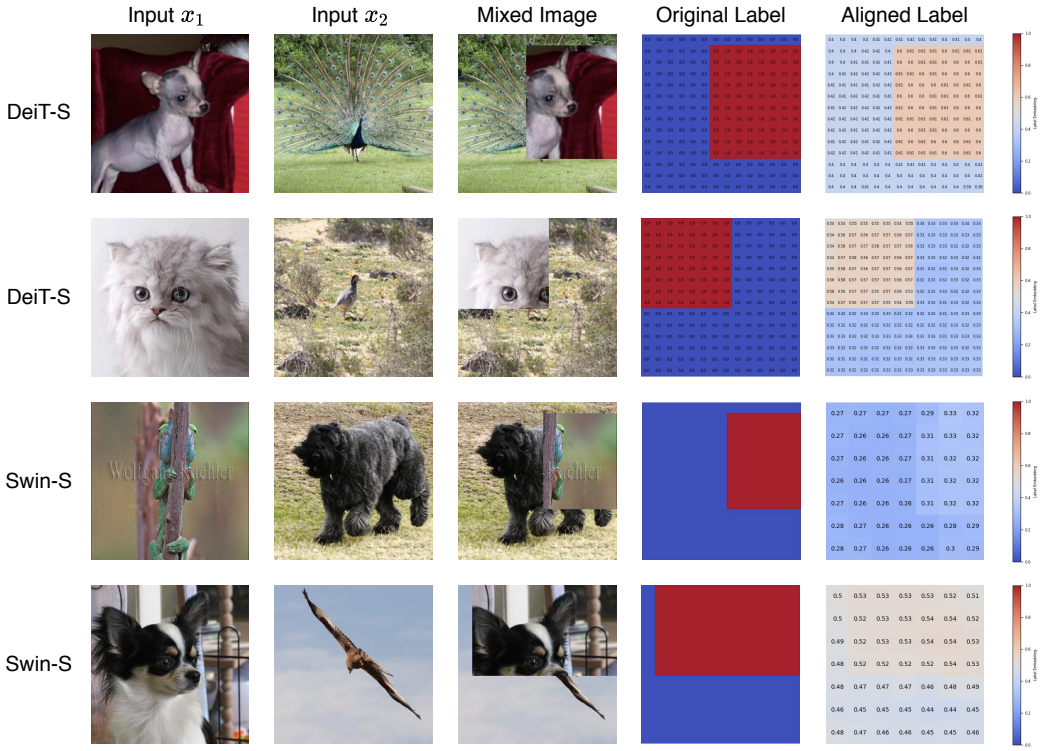



Figure 6: More visualization results on DeiT-S and Swin-S. We visualize the input images, the mixed image, the original label embedding and the label embedding after token-label alignment.

Swin-Transformer since they employ different patch size for patch embbedding. The size of the aligned label embedding is equivalent to the number of output tokens, i.e. $14 \times 14$ for DeiT-S and $7 \times 7$ for Swin-Transformer due to patch merging. The value of the label embedding represents the probability the corresponding token belongs to each class, which is viewed by color. Red stands for the class of the first input image while blue stands for the class of the second input image. We observe that the aligned labels can deviate from the original labels and result in different mixing ratios during training. Therefore, using the original mixing ratio as the training target produces false training signals and might lead to inferior performance.

## C Details about Datasets

We evaluate our method on ImageNet for image classification task, on ADE20K for semantic segmentation and COCO 2017 for object detection and instance segmentation. ImageNet contains about 1.2 million training and 50K validation images from 1K categories. ADE20K contains 20K training images and 2K validation images from 150 semantic categories. COCO 2017 dataset consists of 118K training images and 5K validation images from 80 different categories.

We further conduct experiments to evaluate the robustness and the generalization ability of the TL-Align pretrained models. For robustness, we consider ImageNet-A, ImageNet-C, ImageNet-R and under AutoAttack. ImageNet-A (Hendrycks et al., 2021b) consists of naturally adversarial examples from real-world challenging scenarios. ImageNet-C (Hendrycks & Dietterich, 2019) is used to evaluate the model robustness to diverse image corruptions. ImageNet-R (Hendrycks et al., 2021a) contains various artistic renditions of 200 ImageNet classes. which contains new test sets of ImageNet following the same labeling protocol. AutoAttack (Croce & Hein, 2020) is a novel adversarial attacks benchmark to test the adversarial robustness on ImageNet validation set. To evaluate the generalization ability, we adopt the ImageNet-V2 dataset (Recht et al., 2019), which contains new test sets of ImageNet following the same labeling protocol.