TokenMixup: Efficient Attention-guided Token-level Data Augmentation for Transformers

Anonymous Author(s)
Affiliation
Address
email

Abstract

Mixup is a commonly adopted data augmentation technique for image classification. Recent advances in mixup methods primarily focus on mixing based on saliency. However, many saliency detectors require intense computation and are especially burdensome for parameter-heavy transformer models. To this end, we propose TokenMixup, an efficient attention-guided token-level data augmentation method that aims to maximize the saliency of a mixed set of tokens. TokenMixup provides $\times 15$ faster saliency-aware data augmentation compared to gradient-based mixup methods. Moreover, we introduce a variant of TokenMixup which mixes tokens within a single instance, thereby enabling multi-scale feature augmentation. Experiments show that our methods significantly improve the baseline models' performance on CIFAR and ImageNet-1K, while being more efficient than previous methods. We also reach state-of-the-art performance on CIFAR-100 among from-scratch transformer models. Code will be released.

1 Introduction

Various data augmentation methods have been proposed for computer vision tasks. One of the most successful augmentation methods is mixup [1], which is commonly applied to image classification. It attempts to augment the input by taking a convex combination of two random instances, and by reassigning the ground truth label correspondingly. Building on mixup, many subsequent works [2–9] have been introduced. They focus on mixing instances in a more meaningful way (*e.g.*, saliency-aware mixup, manifold level mixup, submodular diversity maximization *etc.*), while a great majority attempts to perform mixup based on saliency.

These saliency-aware mixup methods often entail gradient computation. To detect salient regions, the input is forward-propagated once and back-propagated to extract the gradient map. However, such a mechanism tends to be computationally heavy. This is especially pertinent to transformer-based models [10], as they generally retain high parameter volume. Keeping this in mind, we aim to present a saliency-aware mixup method favorable for transformers. Specifically, we regard self-attention as the inherent saliency detector, which serves as an efficient means for augmentation.

We accordingly propose **TokenMixup**, an efficient attention-guided token-level mixup method. Our objective is to mix intermediate token sets so that the saliency level of a batch is maximized. To detect salient tokens, we take advantage of the attention map to achieve a $\times 15$ speed-up compared to the gradient-based saliency estimator, without compromising model performance. The batch saliency is maximized by optimally matching instance pairs, accomplished by an algorithm that exactly solves the optimization problem. We also introduce ScoreNet, a simple module that measures the difficulty of an instance. Based on ScoreNet, TokenMixup is applied selectively based on sample difficulty, which can be viewed as a type of curriculum learning method. To the best of our knowledge, this is the first attempt in literature to discuss mixup in a curriculum learning perspective. Furthermore, we

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

present a novel mixup approach that combines intermediate tokens across transformer layers within a single instance. We name this Vertical TokenMixup (VTM) since it *vertically* mixes tokens to provide rich multi-scale information to the attention layer.

Experiments on image classification datasets imply that our methods are both effective and efficient. Also, an advantage of TokenMixup is that it can be used in parallel with other mixup methods. In combination with those methods, we improve our baselines' performance by significant margins on CIFAR-10, CIFAR-100, and ImageNet-1K. Especially for CIFAR-100, we achieve a new state-of-the-art performance among transformer-based models that do not use pre-trained models.

Then, our contributions are fourfold:

- We propose TokenMixup, an efficient attention-guided token-level mixup for transformers. By incorporating the self-attention map, we achieve $\times 15$ faster saliency-aware data augmentation compared to gradient-based mixup methods.
- We incorporate curriculum learning into mixup, to adaptively perform TokenMixup in an intermediate layer based on the confidence score of ScoreNet.
- We also studied a variant, Vertical TokenMixup, which performs mixup with a single sample and enables multi-scale feature augmentation.
- We achieve state-of-the-art performance on CIFAR-100 among transformer models trained from scratch. We also gain significant improvements over baseline models for image classification.

2 Related Works

Mixup. Input Mixup [1] is a data augmentation method widely adopted for image classification. The classification model is trained with a convex combination of input images and labels. A special case of Input Mixup is CutMix [2], which can be seen as a pixel-wise mixup method with binary masks. Recent advances in these mixup methods mainly focus on appropriately utilizing saliency information, and on mixing in image feature levels.
The core motivation of saliency-based mixup is that salient regions should be preserved when mixed, to retain a sufficient amount of information and learn more natural feature representations. SaliencyMix [5] adopts various saliency detectors to directly extract salient regions. Puzzle Mix [3] attempts to mix images using saliency information while retaining local statistics. Co-Mixup [4] maximizes gradient-based saliency while encouraging supermodular diversity of the mixed images. SuperMix [9] takes advantage of supervised signals to mix input images based on saliency.
On the other hand, Manifold Mixup [6] has provided theoretical grounds on the advantages of mixing images in the higher level of features. Manifold mixing helps the model learn flatter representation and smoother decision boundaries. Moreover, it is noted that it enhances robustness to adversarial attacks. Recently, AlignMix [7] was proposed to further align geometric properties of objects by mixing instances in the feature space. Furthermore, StyleMix [8] proposed to manipulate content and style information disparately to generate more abundant and robust samples.

Vision Transformers. Originating from the natural language processing field, the Transformer [10] model enjoyed tremendous success in numerous computer vision applications, *e.g.*, image classification, object detection, human-object interaction detection and object tracking *et cetera*. Image classification is one of the most fundamental computer vision tasks. Starting from the ViT [11], many works have attempted to assimilate the convolutional operation with transformer blocks [12–17]. Other variants have effectively incorporated the convolutional neural network into the transformer to achieve superior performance [18–22].

3 Methods

Here, we introduce TokenMixup, a simple but effective mixup method for transformer models. The goal of our method is to augment intermediate tokens while maximizing the saliency level. This can be accomplished in three major steps: 1) Sample difficulty assessment, 2) Attention-guided saliency detection & optimal assignment, 3) Token-level mixup. The following subsections provide detailed descriptions of the steps.

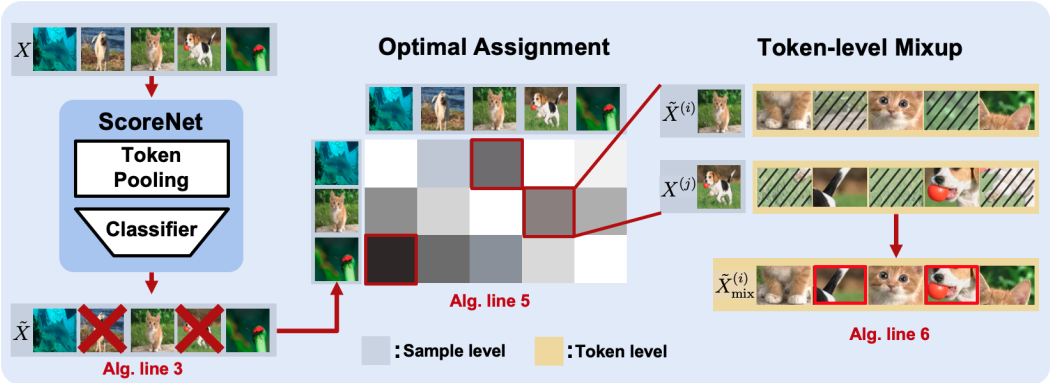

Figure 1: **TokenMixup.** Batch samples X are first filtered with respect to difficulty scores evaluated by ScoreNet in $\mathcal{F}$, resulting in *easy* samples $\tilde{X}$. Then, images in $\tilde{X}$ are optimally paired with samples in X via Hungarian matching, so that overall saliency of the mini-batch is maximized after mixup. For a matched pair $(\tilde{X}^{(i)}, X^{(j)})$, tokens of $\tilde{X}^{(i)}$ with sufficiently lower saliency levels than the tokens of $X^{(j)}$ are replaced, resulting in a new token set $\tilde{X}^{(i)}_{\text{mix}}$ with a improved saliency level.

3.1 Sample Difficulty Assessment

To adaptively apply augmentation based on sample difficulty, a function $\mathcal{F}$ needs to be defined which evaluates the difficulty of each sample. We use a parameterized module, ScoreNet, as the difficulty measurer. ScoreNet is a simple MLP that predicts the target value $Y^{(i)}$ based on the intermediate tokens $X^{(i)}$, where i is the index of the sample in a batch (See the supplement for figure). Given the prediction by ScoreNet, the difficulty score is computed by prediction loss defined as

$$\mathcal{F}(X^{(i)}, Y^{(i)}) = \text{CrossEntropy}(\text{ScoreNet}(X^{(i)}), Y^{(i)}). \tag{1}$$

If the score is greater than a threshold τ, the sample is deemed sufficiently hard and no mixup is performed. On the other hand, the set of *easy* samples with $\mathcal{F}(X^{(i)}, Y^{(i)}) < \tau$ denoted as $\tilde{X}$ will be augmented by Token-level mixup after pairing with other samples in a mini-batch, as illustrated in Figure 1. Note that ScoreNet is trained simultaneously with the main model and learning ScoreNet to assess sample difficulty can be viewed as an auxiliary task.

3.2 Attention-guided Saliency Detection & Optimal Assignment

Several works [15, 23–25] have discussed the imbalanced information of tokens. Due to this imbalance, random token mixing would potentially cause significant information loss and meaningless token replacements (See Section 5.3 for relevant analysis). Thus, we aim to mix tokens based on saliency.

Attention-guided saliency detection. Instead of the computationally heavy gradient-based saliency detectors [3, 4], we take advantage of the attention map, an inherent saliency approximator within transformer modules. Specifically, we can infer the saliency of the tokens in the ith layer via Attention Rollout [26], which computes the attention imposed on the tokens from layer i to $i + \ell$ ($\ell \geq 0$) as

$$A = \Phi^{(i)} \cdot \Phi^{(i+1)} \cdots \Phi^{(i+\ell)}, \tag{2}$$

such that $\Phi^{(i)} = \frac{1}{H}\sum_{h=1}^{H} \Phi_h^{(i)}$ where H is the number of heads in the multi-head attention layer, and $\Phi_h^{(i)}$ is the hth attention head of layer i. However, in order to retrieve A, the samples need to be propagated for the subsequent ℓ layers with stop-gradient. Since each additional forward propagation is an overhead, we approximate Attention Rollout by setting $\ell = 0$. That is, we only use the attention map of the following layer as the saliency estimator. We find this a sufficient approximation of the full Attention Rollout. See section 5.1 for analysis on the soundness of the approximated saliency map. Then, the saliency score S_t can be computed as

$$S_t = \frac{1}{n}\sum_{i=1}^{n} A_{i,t}, \tag{3}$$

where $t = 1, 2, \ldots, n$ denotes the token index.

Algorithm 1 TokenMixup

Input: $X \in \mathbb{R}^{\mathrm{b}\times\mathrm{n}\times\mathrm{d}}$, $Y \in \mathbb{R}^{\mathrm{b}\times\mathrm{c}}$, τ, ρ
Output: $X \in \mathbb{R}^{\mathrm{b}\times\mathrm{n}\times\mathrm{d}}$, $Y \in \mathbb{R}^{\mathrm{b}\times\mathrm{c}}$

$U^{(i)} \leftarrow$ evaluate difficulty of $X^{(i)}$ s.t. $i = 1, 2, \ldots, b$ ▹ Eq. (1)
$S_t^{(i)} \leftarrow$ evaluate saliency of each token with ℓ-step attention rollout ▹ Eq. (3)
$\tilde{X}^{(i)}, \tilde{Y}^{(i)}, \tilde{S}^{(i)} \leftarrow$ select easy samples w.r.t U and τ s.t. $i = 1, 2, \ldots, b'$
$C_{ij} \leftarrow \sum_t \max(S_t^{(j)} - \tilde{S}_t^{(i)} - \rho,\ 0)$ s.t. $i = 1, 2, \ldots, b'$ and $j = 1, 2, \ldots, b$ ▹ Eq. (6)
$\sigma(m) \leftarrow \mathrm{HungarianMatching}(C_{ij})$ s.t. $m = 1, 2, \ldots, b'$ and $\sigma(m) \in \{1, 2, \ldots, b\}$
$\tilde{X}_{\mathrm{mix}}^{(i)} \leftarrow \text{Mix-token}(\tilde{X}^{(i)}, X^{(\sigma(i))}; \rho)$ s.t. $i = 1, 2, \ldots, b'$ ▹ Eq. (10)
$\tilde{Y}_{\mathrm{mix}}^{(i)} \leftarrow \mathrm{Relabel}(\tilde{Y}^{(i)}, Y^{(\sigma(i))})$ ▹ Eq. (11)
restore $\tilde{X}_{\mathrm{mix}}^{(i)}, \tilde{Y}_{\mathrm{mix}}^{(i)}$ to X, Y s.t. $i = 1, 2, \ldots, b'$
return X, Y

Optimal assignment. Based on the estimated saliency of tokens, we aim to maximize the overall saliency level by optimally assigning a different mixup target for each instance. So, we first define $P_{i,j} \in \mathbb{R}^{\mathrm{n}}$ as the saliency difference between a random instance pair (i, j), which is computed as

$$P_{i,j} \triangleq S_j - S_i, \tag{4}$$

where $i, j = 1, 2, \ldots, b$ and $S_i \in \mathbb{R}^{\mathrm{n}}$ is the token saliency map from the ith instance. Then, we represent our objective as an optimization problem written as

$$\max_{\sigma \in \mathcal{M}} \max_{r_i \in \mathbb{B}^{\mathrm{n}}} \sum_{i=1}^{b} (P_{i,\sigma(i)} - \rho)^\top \cdot r_i, \tag{5}$$

where $\mathcal{M}$ is the set of all possible batch permutations of b instances, and $\sigma \in \mathcal{M}$ refers to an arbitrary permutation. ρ is the threshold hyperparameter that controls the minimum saliency gain required for a token to be mixed, and r_i is a binary decision vector for the ith instance. *i.e.*, token t of i is replaced with token t of $\sigma(i)$ when $r_{i,t} = 1$, and preserved when $r_{i,t} = 0$ $(t = 1, 2, \ldots, n)$.

The above optimization problem can be exactly solved by utilizing the Hungarian Matching algorithm [27]. The matching algorithm requires the score matrix, $C \in \mathbb{R}^{\mathrm{b}\times\mathrm{b}}$, which is computed as

$$C_{i,j} = \sum_{t=1}^{n} \max(P_{i,j}^{(t)} - \rho,\ 0), \tag{6}$$

where each item in $C_{i,j}$ refers to the maximum saliency gain resulting from mixing $\tilde{X}^{(i)}$ and $X^{(j)}$. The actual maximum gain will be reached when $\rho = 0$, but we set a positive threshold to control the minimum saliency gain required for each token to be replaced. Then, by applying the Hungarian algorithm, we can find the optimal batch permutation σ^* such that

$$\sigma^* = \arg\max_{\sigma} \sum_{i=1}^{n} C_{i,\sigma(i)}. \tag{7}$$

Then, by mixing $\tilde{X}^{(i)}$ and $X^{(\sigma^*(i))}$, our objective is optimized.

By incorporating sample difficulty assessment (section 3.1) into (5), our final objective is written as

$$\begin{aligned} &\max_{\sigma \in \mathcal{M}} \max_{r_i \in \mathbb{B}^{\mathrm{n}}} \sum_{i=1}^{b} \mathbb{1}_{\{\mathcal{F}(X^{(i)}, Y^{(i)}) < \tau\}} (P_{i,\sigma(i)} - \rho)^\top r_i \\ &\text{s.t. } \mathbf{1}^\top r_i \le n \times \mathbb{1}_{\{\mathcal{F}(X^{(i)}, Y^{(i)}) < \tau\}}, \end{aligned} \tag{8}$$

where $\mathcal{F}$ is the difficulty measurer, and τ is the difficulty threshold. $\mathbb{1}$ denotes the indicator function, while $\mathbf{1}$ is a one vector. If the sample is hard, *i.e.*, $\mathcal{F}(X^{(i)}, Y^{(i)}) \ge \tau$, the constraint becomes $\mathbf{1}^\top r_i \le 0$, enforcing all elements of r_i to be 0. In that case, none of the tokens will be mixed. On the other hand, if the sample is easy, *i.e.*, $\mathcal{F}(X^{(i)}, Y^{(i)}) < \tau$, the constraint is $\mathbf{1}^\top r_i \le n$ which is satisfied by any random r_i vector. Then, the objective becomes identical to (5).

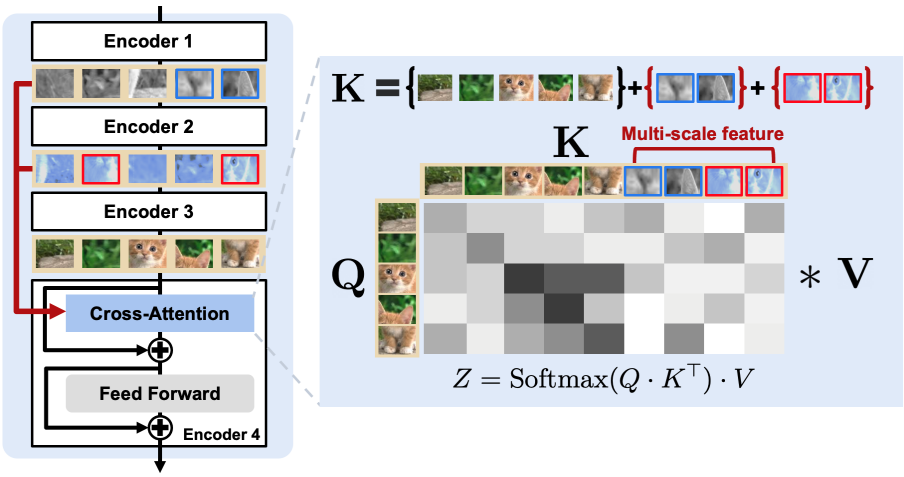

Figure 2: **Vertical TokenMixup**. By applying VTM, the κ most salient tokens from each previous layer are brought up to form the key and value tokens. By utilizing a cross-attention mechanism, the dimension is preserved while being able to aggregate multi-scale information for token augmentation. The figure demonstrates the case when VTM is applied to layer 4.

3.3 Token-level Mixup

Now, the remaining question is *how* the tokens of $\tilde{X}^{(i)}$ and $X^{(\sigma^*(i))}$ are going to be mixed. As defined in objective (5), we use a hard replacement of tokens with respect to decision vector r_i. Then, mixing is performed by replacing token index t from the original instance i, with token t from the paired instance $j = \sigma^*(i)$, so that $P^{(t)}_{i,j} > \rho$ is satisfied. In other words, only the token indices that increase saliency level greater than ρ are mixed. This is done by defining an appropriate binary mask $M_t \in \mathbb{B}^{\mathrm{n}}$ $(\mathrm{t} = 1, 2, \ldots, \mathrm{n})$ containing

$$M_t = \begin{cases} 0 & , \; S^{(j)}_t - \tilde{S}^{(i)}_t > \rho \\ 1 & , \; \text{otherwise} \end{cases} \tag{9}$$

$\tilde{S}^{(i)}_t$ is the saliency score of $\tilde{X}^{(i)}_t$. Then, the mask is employed as

$$\tilde{X}^{(i)}_{\text{mix}} = M \odot \tilde{X}^{(i)} + (1 - M) \odot X^{(j)}, \tag{10}$$

where operator $\odot$ denotes element-wise multiplication and $\tilde{X}^{(i)}$, $X^{(j)}$ refer to the paired instances. We also reassign labels based on the replaced tokens' saliency score. The new mixed label is computed as

$$\tilde{Y}^{(i)}_{\text{mix}} = \frac{\sum_{t=1}^{n} M_t \cdot \tilde{S}^{(i)}_t}{\sum_{t=1}^{n} M_t \cdot \tilde{S}^{(i)}_t + (1 - M_t) \cdot S^{(j)}_t} \cdot \tilde{Y}^{(i)} + \frac{\sum_{t=1}^{n} (1 - M_t) \cdot S^{(j)}_t}{\sum_{t=1}^{n} M_t \cdot \tilde{S}^{(i)}_t + (1 - M_t) \cdot S^{(j)}_t} \cdot Y^{(j)}. \tag{11}$$

Consequently, relatively unimportant tokens will be replaced with salient tokens from another instance, and its label will be adjusted with respect to the change in overall saliency level. See Figure 1 for visualization, and Algorithm 1. More detailed pseudocode is provided in the supplement.

3.4 Vertical TokenMixup

Mixing multiple instances is not the only way to mixup. We also present a simple variant of TokenMixup, which provides token-level augmentation with a single input sample by utilizing the tokens from previous layers. Similar to (5), we define a new objective for Vertical TokenMixup as

$$\max_{l \in \mathcal{L}} \; \max_{r_i \in \mathbb{B}^{\mathrm{n}}} \; \sum_{i=1}^{b} (P_{i,l(i)} - \rho)^\top \cdot r_i, \tag{12}$$

where $\mathcal{L}$ is the set of previous layer indices, and $l(i)$ returns an arbitrary layer index for the ith instance. This objective can be optimized by a similar scheme as in Algorithm 1. However, considering that the

tokens from each layer with identical index tend to contain similar information, saliency difference matrix $P_{i,l(i)} = S_{l(i)} - S_i$ (as in (4)) will most likely be constant across indices. This may lead to meaningless mixup without much increase in saliency level. Therefore, we take a different approach to mix tokens *vertically*.

The simplest method would be to concatenate all tokens from previous layers and apply self-attention. But due to the quadratic complexity of self-attention, naive concatenation may induce excessive computation. To reduce overhead, we selectively pool the κ most salient tokens from each previous layer and concatenate them to the original token set. Also, to preserve input dimension, we adopt the cross-attention mechanism. If X_1 is the original token set and X_2 is the concatenated token set, vertical cross-attention is expressed as

$$Z = \mathrm{Softmax}(X_1 W_q \cdot (X_2 W_k)^\top) \cdot X_2 W_v, \tag{13}$$

where W_q, W_k, W_v denote the projection layers for query, key, and value, respectively. By setting $\kappa \ll n$, where n is the number of tokens per layer, complexity is reduced from $O(|\mathcal{L}|^2 \cdot n^2)$ to $O(n^2 + (|\mathcal{L}| - 1) \cdot \kappa \cdot n)$. See Figure 2 for visualization, and the supplement for pseudocode.

3.5 Discussions

TokenMixup as Curriculum Learning. A general framework for curriculum learning consists of two main components: Difficulty Measurer and Training Scheduler [28]. TokenMixup satisfies these conditions, as ScoreNet measures the difficulty of each input, and augmenting the tokens based on input difficulty naturally schedules training. In the early training phase where the model parameter is not optimized, most instances will be evaluated *difficult* and no TokenMixup is performed. As the model converges, on the other hand, many samples will become *easy*, triggering augmentation to make the task more challenging overall. Also, TokenMixup is applied discriminatively based on individual sample difficulty which enables more intricate curriculum scheduling. See section 5.3 for empirical demonstration. These properties will be especially useful when other mixup methods [1–4] are used in parallel. For instance, if input mixup is already applied, further augmentation may be needless. In such a case, ScoreNet will regard the sample sufficiently difficult, and no further mixup will be performed.

4 Experiments

4.1 Preliminaries

Baseline. In experiments on CIFAR [29], we used Compact Convolution Transformer (CCT) [18] as our baselines. For ImageNet-1k [30] experiments, we used the vanilla ViT-B/16 [11] as baselines. Then, our augmentation methods are evaluated on three representative image classification datasets: CIFAR-10, CIFAR-100, and ImageNet-1K.

Experimental setup. We evaluate TokenMixup and its variant, Vertical TokenMixup (VTM), on CIFAR and ImageNet with different settings. To avoid confusion, we denote the original TokenMixup as Horizontal TokenMixup (HTM) in the following sections. For CIFAR experiments, we adopt the 1500-epoch version of CCT. We modified the learning rate scheduler and positional embedding type to achieve better performance than original papers (denoted $*$ in Table 1). Other experiment settings follow [18], and all experiments on CIFAR datasets were conducted on a single RTX A6000 GPU. For ImageNet-1K experiments, we used the ViT-B/16 model pre-trained on ImageNet-21k. We took advantage of the officially released pre-trained weights. Experiments for Horizontal TokenMixup were conducted on a single NVIDIA A100 GPU, and 4 RTX 3090 GPUs were used in parallel for Vertical TokenMixup. Other experiment settings are reported in the supplement.

4.2 Experimental Results

First, we demonstrate the image classification performance on CIFAR-10 and CIFAR-100 when TokenMixup is applied to CCT. From Table 1, we can see that Horizontal TokenMixup improves its strict baseline trained in the exact same setting, by a significant margin of 0.68%. Vertical TokenMixup also surpasses the baseline performance by a margin of 0.55%. We outperform the previous state-of-the-art model by a huge margin of 0.83%. In the case of CIFAR-10, we achieved higher accuracy of 97.65% and 97.84% with HTM and VTM, respectively. Considering the highly saturated performance on this dataset, a 0.36% improvement over the baseline is not trivial.

Table 1: **Experimental results on CIFAR.** Experiment results with TokenMixup on CCT are compared with state-of-the-art models that do not use pretrained models for initialization. Top-1 validation accuracy is reported for each model, and baseline models denoted with * reports retrained performance. Also, the number in the parenthesis indicates the number of epochs used for training. Best performances are highlighted in yellow.

Models	**# Params**	**MACs**	**CIFAR-10**	**CIFAR-100**
Convolutional Network based				
ResNet18	11.18 M	0.04 G	90.27	63.41
ResNet50	23.53 M	0.08 G	90.60	61.68
ResNet1001-v2 [31]	10.33 M	1.55 G	95.08	77.29
MobileNetV2/2.0 [32]	8.72 M	0.02 G	91.02	67.44
WRN-28-10 [33]	36.5 M	-	96.00	80.75
WRN-40-4 [33]	8.9 M	-	95.47	78.82
ResNeXt-29-8×64d [34]	34.4 M	-	96.35	82.23
ResNeXt-29-16×64d [34]	68.1 M	-	96.42	82.69
Transformer based				
ViT-Lite-6/4 [18]	3.19 M	0.21 G	90.94	69.20
ViT-Lite-7/4 [18]	3.72 M	0.24 G	91.38	69.74
NesT-T [35]	17.0 M	-	96.04	78.69
NesT-B [35]	68.0 M	-	97.20	82.56
CVT-6/4 [18]	3.19 M	0.20 G	92.58	72.25
CCT-7/3x1(1500) [18]	3.76 M	0.95 G	97.48	82.72
CCT-7/3x1(1500) *	3.78 M	0.95 G	97.48	82.87
CCT-7/3x1(1500) + **Horizontal TM (ours)**	3.81 M	0.95 G	**97.65**	**83.55**
CCT-7/3x1(1500) + **Vertical TM (ours)**	3.78 M	0.95 G	**97.84**	**83.42**

Table 2: **Experimental results on ImageNet.** Experiment results with TokenMixup on ViT are compared. We report Top-1 and Top-5 validation accuracy on ImageNet-1k. For baseline, we used ViT-B/16 (224×224) pretrained on ImageNet-21k. † denotes fine-tuned performance officially reported by the authors of ViT.

Models	**# Params**	**MACs**	**Top-1 Acc.**	**Top-5 Acc.**
Convolutional Network based				
ResNet50 [36]	25.5 M	4.3 G	76.20	-
ResNet152 [36]	60.19 M	11.58 G	78.57	-
DenseNet-264 [37]	-	-	79.20	94.71
WRN-50-2-bottleneck [33]	68.9 M	-	78.10	93.97
ReGNetY-4G [38]	21 M	2.0 G	80.0	-
ResNeXt-50-2×40d [34]	-	-	77.00	-
ResNeXt-101-64x4d [34]	-	-	79.60	94.70
Transformer based				
ViT-B/16-224† [11]	86.6 M	16.9 G	81.2	-
ViT-B/16-224 + **Horizontal TM (ours)**	87.3 M	16.9 G	**82.38**	**96.29**
ViT-B/16-224 + **Vertical TM (ours)**	86.6 M	16.9 G	**82.31**	**96.30**

Performance on ImageNet-1K is evaluated on the vanilla ViT [11] model. The officially reported Top-1 accuracy is 81.2%, which is achieved by using weights pre-trained on ImageNet-21K and fine-tuning on ImageNet-1K. By fine-tuning the model further by employing our augmentation methods, we achieve Top-1 accuracy of 82.38% with Horizontal TokenMixup, and 82.31% with Vertical TokenMixup. Also note that HTM marginally increases parameter size by only 0.7M, while VTM does not require any additional parameters.

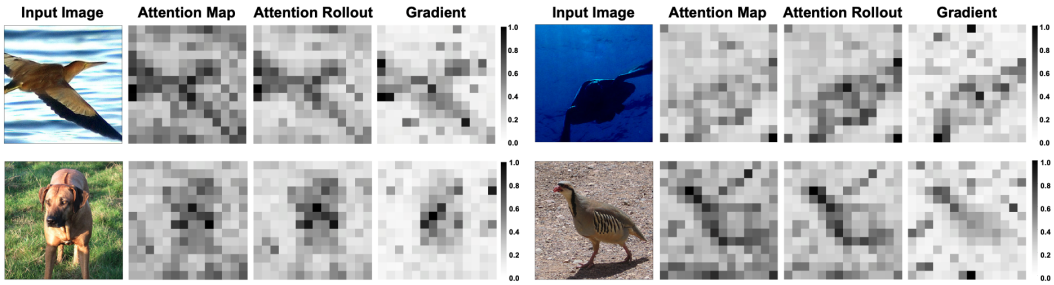

Figure 3: **Qualitative comparison of saliency detectors.** The saliency map derived for each method is provided. We can observe that the methods render similar outputs. Note, the values are minmax scaled to [0,1].

Table 3: **Saliency detector comparison.** Our method achieves high accuracy while being ×15 as fast as the gradient method. Also, we observe only ×3 latency compared to the random baseline.

Saliency type	**Gradient-based**	**Attention-based**	**Random baseline**
Avg. Latency (ms)	236 (×14.8)	16 (×1.0)	5 (×0.3)
CIFAR-100 Acc.	83.18	**83.55**	83.05

5 Analysis

Here, we analyze TokenMixup to answer the following research questions: **Q1**. Is the Attention an appropriate saliency approximation? **Q2**. What are the effects of each component in TokenMixup? **Q3**. What properties does TokenMixup have? Note, all analyses are done on the CIFAR-100 dataset.

5.1 Soundness of Saliency Estimation

In TokenMixup, we estimate the saliency using the attention map from the subsequent layer. The soundness of this approximation would be one of the main questions regarding saliency detection. So, we provide relevant analyses on the appropriateness of our saliency approximation method to answer the research question **Q1**.

Qualitative examples. Here, we provide a qualitative analysis on the performance of each saliency detector. In Figure 3, we can observe that there is no significant difference between the full attention rollout and the 1-step approximation, *i.e.*, the subsequent attention map. Slight discrepancy in the sharpness of the distribution can be observed, but it does not greatly affect our method that utilizes saliency threshold ρ. Moreover, attention-based methods seem to be more accurate than the gradient-based detector in certain cases. See the supplement for additional qualitative examples.

Efficiency comparison. We compare the efficiency of three saliency detectors: gradient-based, attention-based, and random baseline. For the gradient-based detector, we follow the method used in Co-Mixup [4], and random baseline refers to a random selection of tokens. In Table 3, we report the average latency of saliency detection per iteration in the CIFAR-100 experiment setting with batch size of 128. Compared to the gradient-based method, attention-based saliency detection with 1-step rollout achieves approximately ×15 speed-up while demonstrating even better performance.

5.2 Component Analysis

In this section, we analyze the effect of each component in Horizontal TokenMixup and Vertical TokenMixup so as to answer **Q2**.

In Table 4, we provide ablation studies on components of HTM: Sample difficulty assessment (ScoreNet), Optimal assignment, Token-level mixup, and Saliency-based label reassignment. By ablating ScoreNet, all instances are mixed regardless of sample difficulty. Note, ScoreNet still receives supervision signals. By getting rid of optimal assignment, mixup pairs are not assigned via Hungarian matching; pairs are randomly selected. Token-level mixup and label reassignment is relevant to the method by which tokens are selected. Then, by ablating mixup, we randomly select tokens to replace, while label reassignment ablation derives new labels based on the number of tokens that have been replaced. We also conducted sensitivity tests on key hyperparameters τ, ρ, and κ, but tables and discussions were moved to the supplement due to spatial constraints.

Table 4: **Ablation study on CIFAR-100.** Performance on CIFAR-100 is evaluated by ablating each component one-by-one. We observe that optimal assignment and label reassignment plays a significant role.

Ablated Component	CIFAR-100 Accuracy
CCT-7/3x1 + HTM	**83.55** (- 0.00)
(–) Sample difficulty assessment (ScoreNet)	82.85 (- 0.70)
(–) Optimal assignment	82.61 (- 0.94)
(–) Token-level mixup	83.05 (- 0.50)
(–) Saliency-based label reassignment	82.62 (- 0.93)

Table 5: **Mixup combinations.** CCT performance on CIFAR-100 is compared for mixup method combinations. * denotes values reported in the original paper. Best performances among comparable settings are highlighted.

Mixup Combination	CIFAR-100 Accuracy
CCT-7/3x1	76.80
CCT-7/3x1 + Manifold Mixup	76.72
CCT-7/3x1 + **Horizontal TM**	77.34
CCT-7/3x1 + **Vertical TM**	77.77
CCT-7/3x1 + Input Mixup & Cutmix *	82.72
CCT-7/3x1 + Input Mixup & Cutmix + Manifold Mixup	80.08
CCT-7/3x1 + Input Mixup & Cutmix + **Horizontal TM**	**83.55**
CCT-7/3x1 + Input Mixup & Cutmix + **Vertical TM**	**83.42**

5.3 What TokenMixup Learns

Here, we answer our research question **Q3** by analyzing what our mixup methods are learning. We demonstrate a curriculum learning perspective in terms of HTM, and compare our method with a direct mixup alternative, manifold mixup [6].

A Curriculum Learning perspective. In Figure 4 we plot the trend in the number of samples per batch TokenMixup is activated based on ScoreNet. In the first few epochs (regime (A)), the number of mixed samples is relatively low, as there is less need for augmentation in the early phase of learning. But as the learnable parameters warm up, the number increases abruptly. In regime (B), the average number of mixed tokens slowly increases as model performance steadily improves. In the last learning phase (regime (C)), mixup frequency converges.

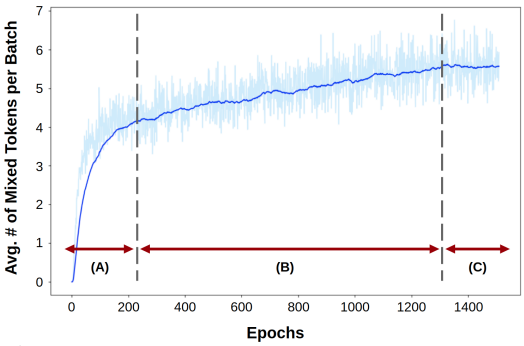

Figure 4: **Trend in the number of mixed instances.**

Comparison with Manifold Mixup. TokenMixup is applied to the intermediate tokens, similarly to manifold mixup [6] which is an input mixup method applied to intermediate features. By using manifold mixup, transformer tokens are mixed via linear interpolation, instead of the hard replacement of tokens as in TokenMixup. In Table 5, our methods are compared with manifold mixup, along with various combinations of original mixup methods (*i.e.*, input mixup and cutmix). The best setting in the CCT paper adopted input mixup and cutmix. By applying HTM and VTM on top of that setting, we achieved state-of-the-art performance. On the other hand, by applying manifold mixup, we observed severe deterioration in accuracy. We conjecture that soft mixing of transformer tokens leads to over-smoothing of features, which may be inadequate for self-attention layers.

6 Conclusion

We proposed TokenMixup for transformers, which adopts the attention map in place of computationally heavy gradient-based saliency detectors. By adopting the attention as the saliency map, we achieve ×15 speed-up compared to the gradient-based method with even better performance. We also introduce a novel perspective of curriculum learning to mixup, which enables adaptive augmentation based on sample difficulty and training schedule. To add diversity, Vertical TokenMixup is introduced, which mixes tokens from different layers for multi-scale feature augmentation with a single sample. With TokenMixup, we achieve state-of-the-art performance on CIFAR-100, and improve baselines by significant margins on CIFAR-10 and ImageNet-1K.

References

[1] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization, 2017.

[2] 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*, 2019.

[3] Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *ICML*, 2020.

[4] Jang-Hyun Kim, Wonho Choo, Hosan Jeong, and Hyun Oh Song. Co-mixup: Saliency guided joint mixup with supermodular diversity, 2021.

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

[6] 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*, 2019.

[7] Shashanka Venkataramanan, Yannis Avrithis, Ewa Kijak, and Laurent Amsaleg. Alignmix: Improving representation by interpolating aligned features. In *ICLR*, 2022.

[8] Minui Hong, Jinwoo Choi, and Gunhee Kim. Stylemix: Separating content and style for enhanced data augmentation. In *CVPR*, 2021.

[9] Ali Dabouei, Sobhan Soleymani, Fariborz Taherkhani, and Nasser M Nasrabadi. Supermix: Supervising the mixing data augmentation. In *CVPR*, 2021.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[11] 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*, 2021.

[12] 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*, 2021.

[13] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, 2021.

[14] Kun Yuan, Shaopeng Guo, Ziwei Liu, Aojun Zhou, Fengwei Yu, and Wei Wu. Incorporating convolution designs into visual transformers. In *ICCV*, 2021.

[15] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. In *ICCV*, 2021.

[16] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *NeurIPS*, 2021.

[17] Stéphane d'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *ICML*, 2021.

[18] Ali Hassani, Steven Walton, Nikhil Shah, Abulikemu Abuduweili, Jiachen Li, and Humphrey Shi. Escaping the big data paradigm with compact transformers, 2021.

[19] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *ICCV*, 2021.

[20] Weijian Xu, Yifan Xu, Tyler Chang, and Zhuowen Tu. Co-scale conv-attentional image transformers. In *ICCV*, 2021.

[21] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Conditional positional encodings for vision transformers, 2021.

[22] 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. *NeurIPS*, 2021.

[23] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *NeurIPS*, 2021.

[24] Byungseok Roh, JaeWoong Shin, Wuhyun Shin, and Saehoon Kim. Sparse detr: Efficient end-to-end object detection with learnable sparsity. In *ICLR*, 2021.

[25] Tao Wang, Li Yuan, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. Pnp-detr: towards efficient visual analysis with transformers. In *ICCV*, 2021.

[26] Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers, 2020.

[27] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 1957.

[28] Xin Wang, Yudong Chen, and Wenwu Zhu. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

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

[30] 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. *IJCV*, 2015.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.

[32] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

[33] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks, 2016.

[34] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

[35] Zizhao Zhang, Han Zhang, Long Zhao, Ting Chen, Sercan Arik, and Tomas Pfister. Nested hierarchical transformer: Towards accurate, data-efficient and interpretable visual understanding. In *AAAI*, 2022.

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[37] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[38] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *CVPR*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 3.
   (b) Did you describe the limitations of your work? [Yes] See Appendix
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The codes will be released if accepted.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4 and Appendix for technical details
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A] Our experiment settings do not conduct each experiment multiple times.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 4 and Appendix for technical details.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Appendix or Section 4 baseline.
   (b) Did you mention the license of the assets? [Yes] See Appendix.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See Appendix.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] See Appendix.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See Appendix.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]