Efficient Visual Transformer by Learnable Token Merging

Yancheng Wang and Yingzhen Yang

Abstract—Self-attention and transformers have been widely used in deep learning. Recent efforts have been devoted to incorporating transformer blocks into different neural architectures, including those with convolutions, leading to various visual transformers for computer vision tasks. In this paper, we propose a novel and compact transformer block, Transformer with Learnable Token Merging (LTM), or LTM-Transformer. LTM-Transformer performs token merging in a learnable scheme. LTM-Transformer is compatible with many popular and compact transformer networks, and it reduces the FLOPs and the inference time of the visual transformers while maintaining or even improving the prediction accuracy. In the experiments, we replace all the transformer blocks in popular visual transformers, including MobileViT, EfficientViT, ViT, and Swin, with LTM-Transformer blocks, leading to LTM-Transformer networks with different backbones. The LTM-Transformer is motivated by reduction of Information Bottleneck, and a novel and separable variational upper bound for the IB loss is derived. The architecture of the mask module in our LTM blocks, which generates the token merging mask, is designed to reduce the derived upper bound for the IB loss. Extensive results on computer vision tasks evidence that LTM-Transformer renders compact and efficient visual transformers with comparable or much better prediction accuracy than the original visual transformers. The code of the LTM-Transformer is available at https://github.com/Statistical-Deep-Learning/LTM.

Index Terms—Visual Transformers, Learnable Token Merging, Information Bottleneck, Variational Upper Bound, Compact Transformer Networks

1 Introduction

BUILDING upon the success of Transformer in natural language processing [1], visual transformers have demonstrated remarkable performance across a wide range of tasks [2], [3], [4], [5], [6], [7], [8]. However, the achievements of visual transformers are accompanied with heavy computational costs [3], [9], making their deployment impractical under resource-limited scenarios. The aforementioned limitations have spurred recent research endeavors aimed at developing efficient visual transformers. In this paper, we study the problem of accelerating visual transformers by token merging.

Token merging is an effective method for reducing the FLOPs and improving the inference speed of visual transformers [10], [11], [12], [13], [14], [15]. However, most existing token merging methods [13], [14], [15], [16] largely sacrifice the prediction accuracy of the original transformer networks for reduced computation costs [15], [17]. These methods [13], [15] generally focus on identifying and merging similar tokens by averaging their features. However, such merging strategies, which are based solely on feature similarity, can potentially diminish the informative features in the tokens that are critical to the prediction tasks. Therefore, it remains an interesting and important question that if we can perform token merging while preserving a compelling performance of the visual transformers after token merging. To this end, we propose a novel transformer block, Transformer with Learnable Token Merging, or LTM-Transformer, which learns how to merge tokens while exhibiting a compelling generalization capability of the transformer with merged tokens.

• *Yancheng Wang and Yingzhen Yang are with School of Computing and Augmented Intelligence, Arizona State University, Tempe, AZ, 85281. E-mail: ywan1053@asu.edu, yingzhen.yang@asu.edu*

Motivation. Due to the fact that the FLOPs of a visual transformer largely depend on the number of tokens in all the transformer blocks, the FLOPs of a visual transformer can be significantly reduced by reducing the number of tokens in all the transformer blocks. As reviewed in Section 2.1, the existing token merging methods do not have a principled way of merging original tokens based on their informativeness and importance for classification tasks. **Our goal is to design a principled informative token merging method to merge the output tokens of every transformer block into fewer tokens without largely sacrificing the prediction accuracy of the original visual transformer, and more informative original tokens contribute more to the merged tokens in the token merging process.** However, directly merging the output tokens, even by carefully designed methods [13], [14], [15], would adversely affect the performance of the model. In this paper, we propose to maintain a compelling prediction accuracy of a visual transformer with token merging by an informative token merging process. In our LTM-Transformer block, the original attention output tokens of a transformer block are merged into less target tokens, and every target token is an informative weighted average of the original output tokens. All the target tokens, or the merged tokens, are the final attention output tokens for the LTM-Transformer block, which are fed to an MLP to produce the output of the LTM-Transformer block as illustrated by Figure 1.

Such a token merging process in LTM-Transformer is primarily inspired by the well-known presence of considerable redundancy in the original output tokens of transformer blocks [15], [16]. As different tokens have varying importance in modeling the visual features at a particular transformer block, it is natural to compute an informative aggregation of the original attention output tokens as the final (target) attention output tokens so that more informative and more important tokens contribute more to the merged tokens with a larger weight in the weighted average in the aggregation process.

Such an idea of informative token merging can also be viewed from the perspective of Information Bottleneck (IB). Let X be the input image. Let Z be the original attention output tokens, which are merged into the merged tokens denoted by $\tilde{X}$, and let Y be the ground truth training labels for a classification task. $\tilde{X}$ has fewer tokens than Z. The principle of IB is to maximize the mutual information between $\tilde{X}$ and Y while minimizing the mutual information between $\tilde{X}$ and X. That is, IB encourages the network to learn the merged tokens more correlated with the class labels while reducing their correlation with the input. Extensive empirical and theoretical works have evidenced that models respecting the IB principle enjoy compelling generalization. With the informative token merging process in LTM-Transformer, the merged tokens $\tilde{X}$ are the informative aggregation of the original attention output tokens Z, so $\tilde{X}$ are less correlated with the training images and in this manner the IB principle is better adhered. This is reflected in Table 6 and Table 7 in Section 4.4.2, where a model for ablation study with token merging, ToMe, enjoys less IB loss than the vanilla transformer, MobileViT-S. This observation indicates that the IB principle is better respected by the token merging process in ToMe. In order to further decrease the IB loss, we propose an Information Bottleneck (IB) inspired token merging process, where a LTM-Transformer block generates an informative token merging mask, which reduces the IB loss for visual transformers. For example, our model termed "LTM-MobileViT-S" in Table 6 and Table 7 is the visual transformer with the IB loss reduced by replacing all the transformer blocks in MobileViT-S with LTM-Transformer blocks so that more informative merged tokens are generated by the proposed informative token merging process. While ToMe hurts the prediction accuracy compared to the vanilla model, our LTM-Transformer enjoys even higher top-1 accuracy than the vanilla MobileViT-S, and we have the same observations for MobileViT-XS and EfficientViT.

Contributions. The contributions of this paper are presented as follows.

First, we present a novel and compact transformer block termed Transformer with Information Bottleneck inspired Token Merging, or LTM. Our LTM block generates an informative token merging mask which reduces the IB loss. The LTM blocks can be used to replace all the transformer blocks in many popular vision transformers, rendering compact vision transformers with competitive performance. The effectiveness of LTM is evidenced by replacing all the transformer blocks in popular vision transformers, including MobileViT [18], EfficientViT [6], ViT [3], and Swin [4], with LTM blocks, for image classification, object detection and instance segmentation tasks.

Second, we propose an informative token merging process for vision transformers, which can reduce the IB loss. As a first step, we derive a novel and *separable* variational upper bound for the IB loss associated with token merging, which is $I(\tilde{X}(G), X) - I(\tilde{X}(G), Y)$ where $I(\cdot,\cdot)$ denotes mutual information and G is the token merging mask in LTM. We then view a transformer with multiple LTM blocks as an iterative process for the reduction of the IB loss by gradient descent, and every LTM block simulates one-step gradient descent on the variational upper bound for the IB loss. Inspired by this understanding, the token merging mask at the current layer is generated from the token merging mask at the previous layer and the input tokens at the current layer by a learnable mask module, following the formula of gradient descent as in (3) in Section 3.2. As a result, such informative token merging process generated in a network with LTM blocks enjoys reduced IB loss, which is evidenced in our ablation study in Section 4.4.2. Due to the separability of the variational upper bound for the IB loss, a neural network with LTM blocks can be trained in an end-to-end manner with standard SGD.

Visualization for Informative Token Merging. Figure 4 illustrates the informative token merging process by LTM with a comparison to LTMP. We remark that LTMP uses uniform merging weights, so their token merging process would not weight informative tokens property, resulting in their inferior performance in classification.

It is worthwhile to mention that our LTM models can be either fine-tuned from pre-trained backbones or trained from scratch. As evidenced in Table 1, our LTM models always outperform the current state-of-the-art token merging methods, including the fine-tuning-based method LTMP [14], when fine-tuned for the same number of epochs. We remark that as shown in Table 6, the baseline token merging method, ToMe, and LTMP, can already reduce the IB loss. By replacing all the transformer blocks with our LTM blocks, the networks with LTM exhibit even smaller IB loss and enjoy higher classification accuracy and less FLOPs, either trained from scratch or fine-tuned from pre-trained models. Furthermore, as shown in Table 2, our LTM models also outperform all the competing token merging methods when trained from scratch. Importantly, extensive experiment results on various computer vision tasks demonstrate the compelling performance of LTM networks compared to the competing baselines.

This paper is organized as follows. The related works in efficient vision transformers and compression of vision transformers by pruning or token merging are discussed in Section 2. The formulation of LTM is detailed in Section 3. The effectiveness of LTM is demonstrated in Section 4 for image classification, object detection and instance segmentation tasks, by replacing all the transformer blocks of various popular vision transformers, including MobileViT [18], EfficientViT [6], ViT [3], and Swin [4], with LTM blocks. We conclude the paper in Section 5. We use $[n]$ to denote natural numbers between 1 and n inclusively.

2 Related Works

2.1 Efficient Visual Transformers

Visual transformer models have recently achieved superior performance on a variety of computer vision applications [3], [4], [5], [19], [20]. However, visual transformers often encounter high computational demands due to the quadratic complexity of the point-wise attention and numerous Multi-Layer Perceptron (MLP) layers. To mitigate the challenges of high computational costs, various strategies have been developed [2], [5], primarily aimed at refining the network architectures and incorporating sparse mechanisms for efficient computation. These include the integration of convolutions into transformer networks [6], [18], [21], the use of knowledge distillation for training more efficient transformers [22], [23], [24], and compressing existing visual transformers with methods such as pruning [25], [26], [27]. Techniques for compressing visual transformers generally fall into three categories: (1) Channel Pruning, which targets the elimination of superfluous heads and channels within ViT blocks [25], [28], [29]; (2) Block Pruning, which involves removing redundant transformer blocks [26], [30]; (3) Token Pruning and Token Merging, which prune less important tokens and merge similar tokens in the input of transformer blocks [15], [16], [27], [31].

In this paper, we focus on learning to merge tokens guided by the information bottleneck theory of deep learning and primarily review existing works on Token Pruning and Merging [13], [14], [15], [16], [31]. DynamicViT [16] observes that the prediction in visual transformers is only based on a subset of the most informative tokens and proposes a hierarchical token sparsification framework to prune redundant tokens. ToMe [15] proposes a graph-based matching algorithm that combines similar tokens in each visual transformer block of a pre-trained visual transformer. LTMP [14] learns threshold masking modules that dynamically determine which tokens to merge and prune in a unified framework similar to DynamicViT. ToFu [13] also combines token pruning and token merging. Instead of averagely merging similar tokens, ToFu proposes a conventional average merging module to improve the quality of merged tokens.

2.2 Related Works about Information Bottleneck

[32] provides the first in-depth analysis of conventional information bottleneck (IB) theories and deep learning to establish the connection between the nonlinearity of neural networks and the compression phase of training. Building on the theory of IB, [33] proposes a probabilistic attention module reducing mutual information between the input and the masked representation while increasing mutual information between the masked representation and the task label. Further exploring the mechanics of IB in deep learning, [34] finds that self-attention mechanisms can be interpreted as iterative steps in optimizing the IB objective, which explains the advantages of self-attention in learning robust representation. Distinct from most existing methods that implicitly incorporate the IB principle, our work adopts a direct and innovative approach. We aim to optimize a novel and separable variational upper bound of the IB loss with a learnable token merging method. The proposed LTM-Transformer leads to compelling performance on many popular visual transformer architectures with lower computation cost, benefiting from the learnable token merging mechanism guided by the IB principle.

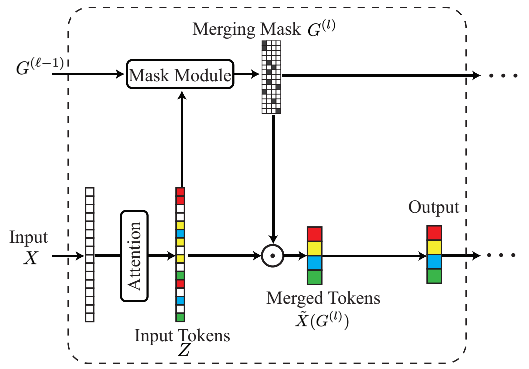
3 Formulation

In this section, we first illustrate how to perform token merging using a token merging mask. We then describe how to generate the token merging mask from a learnable mask module in a LTM-Transformer block, as well as the training algorithm of a neural network with LTM-Transformer blocks. We derive a novel and separable variational upper bound for the IB loss, and the token merging masks are generated to reduce such variational upper bound for the IB loss.

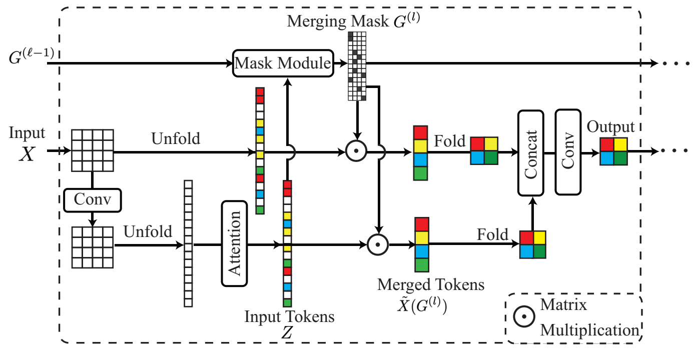
3.1 Token Merging by Learnable Token Merging Masks

Given the input feature tokens $X \in \mathbb{R}^{N \times D}$ where N is the number of tokens and D is the token dimension, the LTM block first applies the self-attention module on the input feature tokens by $Z = \mathrm{ATTN}(X) \in \mathbb{R}^{N \times D}$, where $\mathrm{ATTN}(\cdot)$ is the regular QKV self-attention operation [35]. As illustrated in Figure 1, every LTM block has a learnable mask module that generates the token merging mask $G^{(\ell)}$ where ℓ is the index of the current layer or block. The LTM block merges the N tokens of Z into P tokens with $P < N$ by multiplying Z with the token merging mask $G^{(\ell)} \in \mathbb{R}^{N \times P}$. We set $P = \lceil r \times N \rceil$, where $r \in (0,1)$ is termed the compression ratio for LTM, and a smaller r renders less merged tokens after token merging. The token merging mask $G^{(\ell)}$ of the ℓ-th transformer block is generated by the token merging mask $G^{(\ell-1)}$ of the previous layer and the feature tokens Z, which is motivated by reducing the IB loss and detailed in Section 3.2. The token merging mask $G^{(1)}$ for the first transformer block is generated by applying an existing learnable token merging method, LTMP [14], which generates a binarized token merging mask $M \in [0,1]^{N \times P}$ using Gumbel-Softmax with $N \times P$ learnable parameters. After obtaining the merging mask $G^{(\ell)}$, the features tokens of Z are merged into P tokens by $\tilde{X}(G^{(\ell)}) = \left(Z^\top G^{(\ell)}\right)^\top \in \mathbb{R}^{P \times D}$, which is then passed to the following MLP layers in the transformer block.

In addition to merging tokens in regular transformer blocks such as ViT [35] and Swin [4], the LTM-Transformer block can also be applied to efficient transformer blocks widely applied in efficient visual transformer architectures such as MobileViT [18] and EfficientViT [6]. Regular transformer blocks obtain the output by sequentially applying the attention operation and the MLP on the input feature tokens. However, efficient transformer blocks usually contain residual connections following the design of residual connections in Convolutional Neural Networks (CNNs). That is, these blocks maintain the same shapes for the input X and the self-attention output Z and concatenate them to produce the output features of the current transformer block. As a result, we cannot only merge the tokens of Z. Instead, our LTM-Transformer block merges the tokens of both X and Z so that the number of merged tokens for X and Z is the same. To this end, we apply the same token merging mask $G^{(\ell)}$ to merge both X and Z. As a result, the compressed X and Z are of the same shape after the token merging process, and they can still be concatenated, which is illustrated in Figure 1(b). In addition, transformer blocks in the efficient visual transformers are usually accompanied with convolution operations so that they need to maintain the feature tokens in a three-dimensional format $X \in \mathbb{R}^{H \times W \times D}$ as illustrated in Figure 1(b). To apply our token merging method on efficient transformer blocks, we set the number of merged tokens after token merging as $P = H' \times W'$, where r is the compression ratio, and $H' = \lceil H \times \sqrt{r} \rceil, W' = \lceil W \times \sqrt{r} \rceil$.



(a) LTM block for regular transformers, such as ViT and Swin.



(b) LTM block for efficient transformers, such as MobileViT and EfficientViT.

Fig. 1: Overall framework of Learnable Token Merging (LTM)-Transformer block for regular transformer blocks such as ViT and Swin (a), and efficient transformer blocks such as MobileViT and EfficientViT (b).

Therefore, the merged tokens can still be reshaped into three-dimensional features for later convolution operations. The analysis about the inference computation cost, or the FLOPs, of the LTM transformer block for token merging in both regular transformers and efficient transformers as illustrated in Figure 1 is detailed below.

Computation Cost Analysis of LTM-Transformer for Token Merging. We hereby analyze the additional inference computation cost, or the FLOPs, of the LTM transformer block for token merging in both regular transformers and efficient transformers as illustrated in Figure 1. Let D be the dimension of input tokens and N be the number of tokens. The FLOPs of the token merging in an LTM transformer block in regular visual transformers is $6CDP+3C+ND^2+NDP$, where $6CDP+3C+ND^2$ is the FLOPs for calculating the merging mask and NDP is the cost for applying the merging mask on the input tokens. In the LTM transformer block of efficient visual transformers, the additional FLOPs of the token merging is $6CDP+3C+ND^2+2NDP$, since the merging mask will be applied to both the input tokens and the merged tokens.

3.2 Generating Token Merging Mask by Reducing the Variational Upper Bound for the IB Loss

We describe how to generate the token merging mask in a LTM-Transformer block in this subsection, and the generation of the token merging mask is inspired by reduction of the IB loss. We first introduce the setup where the IB loss can be specified.

Given the training data $\{X_i,y_i\}_{i=1}^n$ where X_i is the i-th input training feature and y_i is the corresponding class label. Let Z_i be the self-attention output tokens of the X_i, and $\tilde{X}_i(G)=(Z_iG)^\top$ is the merged tokens with G being the token merging mask. We first specify how to compute the IB loss, $\mathrm{IB}(G)=I(\tilde{X}(G),X)-I(\tilde{X}(G),Y)$ which depends on G and other network parameters, X is a random variable representing the input feature which takes values in $\{X_i\}_{i=1}^n$, $\tilde{X}(G)$ is a random variable representing the merged tokens which takes values in $\left\{\tilde{X}_i(G)\right\}_{i=1}^n$. Y is a random variable representing the class label which takes values in $\{y_i\}_{i=1}^n$. We first compute the class centroids $\left\{\tilde{\mathcal{C}}_a\right\}_{a=1}^C$ and $\{\mathcal{C}_b\}_{b=1}^C$ for the merged tokens and the input features by averaging the merged tokens and the input features in each class, where C is the number of classes. We also abbreviate $\tilde{X}(G)$ as $\tilde{X}$ for simplicity of the notations. Then we define the probability that $\tilde{X}$ belongs to class $\tilde{\mathcal{C}}_a$ as $\Pr\left[\tilde{X}\in a\right]=\frac{1}{n}\sum\limits_{i=1}^n\phi(\tilde{X}_i,a)$ with $\phi(\tilde{X}_i,a)=\frac{\exp\left(-\left\|\tilde{X}_i-\tilde{\mathcal{C}}_a\right\|_2^2\right)}{\sum_{a=1}^A\exp\left(-\left\|\tilde{X}_i-\tilde{\mathcal{C}}_a\right\|_2^2\right)}$. Similarly, we define the probability that X_i belongs to class $\mathcal{C}_b$ as $\Pr[X\in b]=\frac{1}{n}\sum\limits_{i=1}^n\phi(X_i,b)$. Moreover, we have the joint probabilities $\Pr\left[\tilde{X}\in a,X\in b\right]=\frac{1}{n}\sum\limits_{i=1}^n\phi(\tilde{X}_i,a)\phi(X_i,b)$ and $\Pr\left[\tilde{X}\in a,Y=y\right]=\frac{1}{n}\sum\limits_{i=1}^n\phi(\tilde{X}_i,a)\mathbb{1}_{\{y_i=y\}}$ where $\mathbb{1}_{\{\}}$ is an indicator function. As a result, we can compute the mutual information $I(\tilde{X}(G),X)$ and $I(\tilde{X}(G),Y)$ by

$$I(\tilde{X}(G),X)=\sum_{a=1}^C\sum_{b=1}^C\Pr\left[\tilde{X}(G)\in a,X\in b\right]\log\frac{\Pr\left[\tilde{X}(G)\in a,X\in b\right]}{\Pr\left[\tilde{X}(G)\in a\right]\Pr\left[X\in b\right]},$$

$$I(\tilde{X}(G),Y)=\sum_{a=1}^C\sum_{y=1}^C\Pr\left[\tilde{X}(G)\in a,Y=y\right]\log\frac{\Pr\left[\tilde{X}(G)\in a,Y=y\right]}{\Pr\left[\tilde{X}(G)\in a\right]\Pr\left[Y=y\right]},$$

and then compute the IB loss $\mathrm{IB}(G)$. As explained in our motivation, we aim to perform token merging while can reduce the IB loss. However, directly optimizing the IB loss in the standard SGD training is difficult as the IB loss is not separable. Given a variational distribution $Q(\tilde{X}\in a|Y=y)$ for $y,a\in[C]$ computed by Eq. (5) in Section 1.3 of the supplementary, the following theorem gives a variational upper bound, $\mathrm{IBB}(G)$, for the IB loss $\mathrm{IB}(G)$. $\mathrm{IBB}(G)$ is separable and thus compatible with SGD training with minibatches.

Theorem 3.1.

$$\mathrm{IB}(G)\le\mathrm{IBB}(G)-C_0, \tag{1}$$

where C_0 is a constant only depending on the input training features $\{X_i\}_{i=1}^n$, and

$$\mathrm{IBB}(G):=\frac{1}{n}\sum_{i=1}^n\sum_{a=1}^C\sum_{b=1}^C\phi(\tilde{X}_i(G),a)\phi(X_i,b)\log\phi(X_i,b)$$
$$-\frac{1}{n}\sum_{i=1}^n\sum_{a=1}^C\sum_{y=1}^C\phi(\tilde{X}_i(G),a)\mathbb{1}_{\{y_i=y\}}\log Q(\tilde{X}\in a|Y=y).$$

Proposition 3.2. *Suppose $\tilde{X}_i(G) = \left(Z_i^\top G\right)^\top \in \mathbb{R}^{P\times D}$ with $Z_i \in \mathbb{R}^{N\times D}$ being the self-attention output tokens for the i-th training feature and $G \in \mathbb{R}^{N\times P}$ is the token merging mask where N is the number of tokens, D is the token dimension, P is the number of merged tokens after token merging, and $\tilde{X}_i(G)$ denotes the merged tokens. At step ℓ of gradient descent on* $\mathrm{IBB}(G)$, *we have*

$$\begin{aligned} G^{(\ell)} &= G^{(\ell-1)} - \eta\nabla_G \mathrm{IBB}(G^{(\ell-1)}) \\ &= G^{(\ell-1)} - \frac{2\eta}{n}\sum_{i=1}^{n}\sum_{a=1}^{C} Z_i \frac{S_{ia}^{(l-1)}}{(\gamma_i^{(l-1)})^2}\left(\gamma_i^{(l-1)}\mathcal{C}_a - \zeta_i^{(l-1)}\right)\psi_{i,a},\ \ell\geq 2, \end{aligned} \tag{2}$$

where $S_{ia}^{(\ell)} := \exp\left(-\left\|\tilde{X}_i(G^{(\ell)}) - \tilde{\mathcal{C}}_a\right\|_2^2\right)$ *for* $i \in [n]$ *and* $a \in [C]$, $\gamma_i^{(\ell)} := \sum_{a=1}^{C} S_{ia}^{(\ell)}$, $\zeta_i^{(\ell)} := \sum_{a=1}^{C} S_{ia}^{(\ell)}\tilde{\mathcal{C}}_a$ *for* $i \in [n]$, $\psi_{i,a} := \sum_{b=1}^{C}\phi(X_i,b)\log\phi(X_i,b) - \sum_{y=1}^{C}\mathrm{I\!I}_{\{y_i=y\}}\log Q(\tilde{X}\in a|Y=y)$.

The proofs of Theorem 3.1 and Proposition 3.2 are deferred to Section 1 of the supplementary. Inspired by Proposition 3.2, we can understand a transformer with token merging and multiple transformer blocks as an iterative process which reduces $\mathrm{IBU}(G)$ by gradient descent, where the ℓ-th transformer block performs one-step gradient descent on $\mathrm{IBU}(G)$ according to (2). The mask module of at the ℓ-th LTM block generates the token merging mask $G^{(\ell)}$ from $G^{(\ell-1)}$, the token merging mask of the previous block, through (2). To improve the flexibility of the token merging mask, an MLP is applied on Z_i. Moreover, as IBU and ∇_GIBU are separable, (2) can be performed on a minibatch $\mathcal{B}_j \subseteq \{1,\ldots,n\}$, which is compatible with minibatch-based training with SGD for a transformer network with LTM blocks. In practice, the mask module of the ℓ-th LTM block generates $G^{(\ell)}$ by

$$\tilde{G}^{(\ell)} = G^{(\ell-1)} - \frac{2\eta}{n}\sum_{i\in\mathcal{B}_j}\sum_{a=1}^{C} Z_i \frac{S_{ia}^{(l-1)}}{(\gamma_i^{(l-1)})^2}\left(\gamma_i^{(l-1)}\mathcal{C}_a - \zeta_i^{(l-1)}\right)\psi_{i,a}, \tag{3}$$

$$G^{(\ell)} = \tilde{G}^{(\ell)} \circ M^{(\ell)} \tag{4}$$

where $M^{(\ell)} \in [0,1]^{N\times P}$ is a binarized token merging mask generated by LTMP [14] for the ℓ-th LTM block by applying the Gumbel-Softmax operation on $N \times P$ learnable parameters. The mask $M^{(\ell)}$ in our LTM serves as a learnable token merging mask module. Since the update formulation in Eq.(3) does not incorporate any trainable parameters, the number of the trainable parameters of an LTM block is the same as the number of the trainable parameters in a transformer block with LTMP, which is $N \times P$.

The training loss of our LTM models are the regular cross-entropy loss. The derived variational upper bound for the IB loss, IBB, is used to design the token mask module, where the token merging mask $G^{(\ell)}$ is generated by Eq. (3) and Eq. (4). Algorithm 1 describes the training process of a neural network with LTM-Transformer blocks using the standard cross-entropy loss for a classification problem. It is remarked that all the MLP layers of the mask modules in all the LTM-Transformer blocks, along with other network parameters, are updated with standard SGD. In

Algorithm 1 Training Algorithm of LTM-Transformers

```
1:  Initialize the weights of the network by $\mathcal{W} = \mathcal{W}(0)$ through
    random initialization, set $t_{\text{train}}$ which is the number of
    training epochs.
2:  for $t \leftarrow 1$ to $t_{\text{train}}$ do
3:    if $t < t_{\text{warm}}$ then
4:      Perform gradient descent by a standard step of SGD
        without applying token merging in LTM transformer
        blocks.
5:    else
6:      Update $\tau(\tilde{X}_i, a)$ for all the classes $a \in [C]$ and $i \in [n]$.
7:      for $j \leftarrow 1$ to $J$ do
8:        Forward step: generate $\left\{G^{(\ell)}\right\}$ for all
          the LTM blocks by Eq. (3) using the
          minibatch $\mathcal{B}_j$, the updated $\left\{\tau(\tilde{X}_i, a)\right\}_{i\in\mathcal{B}_j, a\in[C]}$,
          $\left\{Q^{(t-1)}(\tilde{X}\in a|Y=y)\right\}_{a\in[C], y\in[C]}$, and
          $\left\{\tilde{\mathcal{C}}_a^{(t-1)}\right\}_{a=1}^{C}$, as well as the output of the network
9:        Backward step: update the MLP layers of the mask
          modules in all the LTM blocks as well as all the other
          weights in the neural network by a standard step of
          SGD on the cross-entropy loss
10:     end for
11:     Compute $Q^{(t)}(\tilde{X}\in a|Y=y)$ by Eq. (5) in Section 1.3
        of the supplementary, and update the class centroids
        $\left\{\tilde{\mathcal{C}}_a^{(t)}\right\}_{a=1}^{C}$.
12:   end if
13: end for
14: return The trained weights $\mathcal{W}$ of the network
```

order to generate the token merging masks for all the LTM-Transformer blocks before a new epoch starts, we update the variational distribution $Q^{(t)}$ at the end of the previous epoch.

4 Experimental Results

In this section, LTM-Transformers are assessed for the image classification task on the ImageNet-1k dataset. The results in Section 4.1 indicate that LTM outperforms existing state-of-the-art networks while maintaining a more compact architecture either fine-tuned from pre-trained backbones or trained from scratch. In addition, LTM is compared with existing methods on token merging and shows better performance with lower computation costs. Furthermore, in Sections 4.2 and 4.3, we demonstrate that the use of LTM-MobileViT and LTM-EfficientViT as feature extraction backbones leads to superior mAP and reduced FLOPs compared to the baseline models for the tasks of object detection and semantic segmentation. Comprehensive ablation studies are performed in Section 4.4. In Section 4.4.1, we study the impact of compression ratio on LTM. In Section 4.4.2, we perform ablation studies on the effects of LTM-Transformer in reducing the IB loss and the IB bound. In Section 4.4.3, we study IB loss and IB bound at different layers of an LTM-Transformer. In Section 4.4.4, we illustrate the training loss and the test loss in the training process of LTM-Transformers. In Section 4.4.5, we visualize merged tokens and their merging weights by LTM for selected images from ImageNet. In Section 4.4.6, we study the impact of compression ratio at different layers of the LTM-Transformer. Additional experiment results are presented in Section 2 of the supplementary of our paper. In

Methods	# Params.	FLOPs	Training Time (minutes/epoch)	Inference Time (ms/batch)	Top-1 Accuracy (%)					
					0	1	5	10	25	50
MobileViT-XS [18]	2.3 M	0.70 G	-	11.3	74.80	-	-	-	-	-
ToMe-MobileViT-XS [15]	2.3 M	0.54 G	-	10.4	72.73	-	-	-	-	-
ToFu-MobileViT-XS [13]	2.3 M	0.54 G	-	10.7	73.32	-	-	-	-	-
LTMP-MobileViT-XS [14]	2.3 M	0.56 G	16.0	10.9	-	73.91	73.79	73.98	74.05	74.18
LTM-MobileViT-XS (Fine-tuned)	2.3 M	0.52 G	16.5	10.7	-	**74.25**	**74.31**	**74.54**	**74.70**	**74.95**
MobileViT-S [18]	5.6 M	1.40 G	-	15.1	78.40	-	-	-	-	-
ToMe-MobileViT-S [15]	5.6 M	1.22 G	-	14.2	76.72	-	-	-	-	-
ToFu-MobileViT-S [13]	5.6 M	1.22 G	-	14.4	77.24	-	-	-	-	-
LTMP-MobileViT-S [14]	5.6 M	1.26 G	18.1	14.5	-	77.53	77.69	77.82	78.03	78.14
LTM-MobileViT-S (Fine-tuned)	5.6 M	1.17 G	19.0	14.1	-	**77.72**	**78.15**	**78.34**	**78.85**	**79.05**
EfficientViT-B1 [r224] [6]	9.1 M	0.52 G	-	10.0	79.40	-	-	-	-	-
ToMe-EfficientViT-B1 [r224] [15]	9.1 M	0.47 G	-	9.6	78.81	-	-	-	-	-
ToFuEfficientViT-B1 [r224] [13]	9.1 M	0.47 G	-	9.8	79.04	-	-	-	-	-
LTMP-EfficientViT-B1 [r224] [14]	9.1 M	0.50 G	13.8	9.8	-	79.21	79.31	79.32	79.36	79.40
LTM-EfficientViT-B1 [r224] (Fine-tuned)	9.1 M	0.44 G	14.6	9.6	-	**79.39**	**79.62**	**79.85**	**80.07**	**80.22**
Swin-T [4]	29.0 M	4.50 G	-	20.8	81.30	-	-	-	-	-
ToMe-Swin-T [15]	29.0 M	3.91 G	-	17.5	79.28	-	-	-	-	-
ToFuSwin-T [13]	29.0 M	3.91 G	-	17.8	79.65	-	-	-	-	-
LTMP-Swin-T [14]	29.0 M	3.95 G	21.0	17.9	-	79.78	79.96	80.09	80.24	80.30
LTM-Swin-T (Fine-tuned)	29.0 M	3.82 G	21.9	17.0	-	**80.06**	**80.46**	**80.79**	**81.20**	**81.38**
Swin-B [4]	88.0 M	15.4 G	-	33.9	83.50	-	-	-	-	-
ToMe-Swin-B [15]	88.0 M	13.0 G	-	29.9	81.87	-	-	-	-	-
ToFu-Swin-B [13]	88.0 M	13.0 G	-	30.1	82.04	-	-	-	-	-
LTMP-Swin-B [14]	88.0 M	13.2 G	27.2	30.4	-	82.24	82.39	82.45	82.51	82.55
LTM-Swin-B (Fine-tuned)	88.0 M	12.0 G	28.9	29.6	-	**82.50**	**82.72**	**82.88**	**83.43**	**83.64**
ViT-S [35]	22.1 M	4.30 G	-	22.5	81.20	-	-	-	-	-
ToMe-ViT-S [15]	22.1 M	3.82 G	-	18.4	80.04	-	-	-	-	-
ToFu-ViT-S [13]	22.1 M	3.82 G	-	18.7	80.19	-	-	-	-	-
LTMP-ViT-S [14]	22.1 M	3.89 G	21.4	19.0	-	80.32	80.40	80.35	80.41	80.50
LTM-ViT-S (Fine-tuned)	22.1 M	3.70 G	22.7	18.2	-	**80.47**	**80.69**	**80.94**	**81.27**	**81.55**
ViT-B [35]	86.5 M	17.58 G	-	37.2	83.74	-	-	-	-	-
ToMe-ViT-B [15]	86.5 M	13.12 G	-	31.0	82.86	-	-	-	-	-
ToFu-ViT-B [13]	86.5 M	13.12 G	-	31.5	83.22	-	-	-	-	-
LTMP-ViT-B [14]	86.5 M	13.46 G	27.2	32.7	-	83.29	83.40	83.44	83.50	83.55
LTM-ViT-B (Fine-tuned)	86.5 M	12.85 G	28.4	30.7	-	**83.35**	**83.57**	**83.76**	**83.91**	**83.96**

TABLE 1: Performance comparison between LTM with competing token merging baselines, ToMe [15], ToFu [13], and LTMP [14] in fine-tuning setup on ImageNet. Among the compared methods, ToMe and ToFu do not require training. Both LTM models and LTMP models are fine-tuned for 1, 5, 10, 25, and 50 epochs for fair comparisons.

Section 2.1 of the supplementary, we evaluate the transfer learning capability of the LTM-Transformer. In Section 2.2 of the supplementary, we evaluate the effectiveness of fine-tuning the LTM-Transformer with LoRA. In Section 2.3 of the supplementary, we evaluate the throughput of the LTM-Transformer with variant batch sizes.

4.1 Image Classification

Implementation details. In this section, we evaluate LTM models for ImageNet [36] classification. We employ MobileViT-S [18], MobileViT-XS [18], EfficientViT-B1 [6], ViT-S [35], ViT-B [35], Swin-T [4], and Swin-B [4] as backbone architectures. We substitute the conventional transformer blocks in these backbones with LTM blocks. All the experiments are conducted on four NVIDIA A100 GPUs with a total batch size of 1024 images. Following prior works [4], our training incorporates popular data augmentation methods such as RandAugment, Mixup, Cutmix, and random erasing. We set η in Eq. (2) to 1 in all the experiments. In addition, we apply a softmax operation on the token merging mask at each layer to ensure the aggregation weights for each merged token sum to 1. In all our experiments, we set the value of compression ratio $r = 0.7$ for all our LTM models. We set the compression ratio of the compared methods to 0.75 to see how LTM-Transformers compare with competing token merging methods with an even more significant reduction in the number of tokens. A study on the impact of the compression ratio r on the performance of the LTM model is performed in Table 5 in Section 4.4.1.

We conduct the experiments of LTM for the token merging of vision transformers under two different training setups, which are the fine-tuning setup and the training-from-scratch setup. The experiments in the fine-tuning setup are conducted following the state-of-the-art token merging method, LTMP [14]. The training-from-scratch setup is designed to explore the potential of training LTM-Transformers from the beginning while reducing the IB loss with token merging, and the training with different backbones follows the same training settings as the original training process of the corresponding backbones [4], [6], [18], [35].

Fine-Tuning Setup. Our proposed LTM can be straightforwardly applied to token merging with pre-trained models using the fine-tuning setup as in the existing state-of-the-art token merging method, LTMP [14]. In the fine-tuning setup, LTM models are not trained from scratch, and token merging for a pre-trained visual transformer is performed by simply changing all the transformer blocks of the pre-trained models to LTM-Transformer blocks. Following the settings in LTMP [14], the token merging mask modules are added to the original transformer blocks, and all the pre-trained weights are loaded as the initialization for the LTM models. In the fine-tuning process, the pre-trained weights are not updated and only the weights in the token merging mask modules, $\left\{M^{(\ell)}\right\}$, are updated. We fine-tune the LTM models for 1, 5, 10, 25, and 50 epochs, respectively, and compare them with LTMP models fine-tuned for the same number of epochs. Note that LTM models and LTMP models with the same backbones have the same number

Model	# Params	FLOPs	Top-1
MobileViT-XS	2.3 M	0.7 G	74.8
ToMe-MobileViT-XS [15]	2.3 M	0.54 G	72.7
ToFu-MobileViT-XS [13]	2.3 M	0.54 G	73.3
LTMP-MobileViT-XS [14]	2.3 M	0.56 G	73.9
LTM-MobileViT-XS (Ours)	2.3 M	0.52 G	**75.8**
MobileViT-S	5.6 M	1.4 G	78.4
ToMe-MobileViT-S [15]	5.6 M	1.22 G	76.7
ToFu-MobileViT-S [13]	5.6 M	1.22 G	77.2
LTMP-MobileViT-S [14]	5.6 M	1.26 G	77.5
LTM-MobileViT-S (Ours)	5.6 M	1.17 G	**79.7**
EfficientViT-B1 [r224] [6]	9.1 M	0.52 G	79.4
S^2ViTE-EfficientViT-B1 [r224] [37]	8.2 M	0.47 G	79.0
SPViT-EfficientViT-B1 [r224] [27]	9.2 M	0.49 G	79.3
SAViT-EfficientViT-B1 [r224] [29]	8.4 M	0.47 G	79.2
ToMe-EfficientViT-B1 [r224] [15]	9.1 M	0.47 G	78.8
ToFu-EfficientViT-B1 [r224] [13]	9.1 M	0.47 G	79.0
LTMP-EfficientViT-B1 [r224] [14]	9.1 M	0.50 G	79.2
LTM-EfficientViT-B1 [r224] (Ours)	9.1 M	0.44 G	**80.2**
EfficientViT-B1 [r288] [6]	9.1 M	0.86 G	80.4
ToMe-EfficientViT-B1 [r288] [15]	9.1 M	0.73 G	79.7
ToFu-EfficientViT-B1 [r288] [13]	9.1 M	0.73 G	79.8
LTMP-EfficientViT-B1 [r288] [14]	9.1 M	0.76 G	80.0
LTM-EfficientViT-B1 [r288] (Ours)	9.1 M	0.70 G	**81.0**
ViT-S [3]	22.1 M	4.3 G	81.2
LTM-ViT-S (Ours)	22.1 M	3.7 G	**81.8**
ViT-B [3]	86.5 M	17.6 G	83.7
LTM-ViT-B (Ours)	86.5 M	12.9 G	**83.9**
Swin-T [4]	29.0 M	4.5 G	81.3
LTM-Swin-T (Ours)	29.0 M	3.8 G	**81.8**
Swin-B [4]	88.0 M	15.4 G	83.5
LTM-Swin-B (Ours)	88.0 M	12.0 G	**83.8**

TABLE 2: Comparisons with baseline methods on ImageNet-1k validation set for the train-from-scratch setup.

of parameters. It is observed that LTM models significantly outperform LTMP models fine-tuned for the same number of epochs. For example, the LTM-Swin-B fine-tuned for 50 epochs outperforms the LTMP-Swin-B fine-tuned for 50 epochs by 1.09% in top-1 accuracy. The LTM-Swin-T fine-tuned for 50 epochs outperforms the LTMP-Swin-T fine-tuned for 50 epochs by 1.08% in top-1 accuracy.

In addition, we also compare the LTM with three token merging methods, ToMe [15], ToFu [13], and LTMP [14], to demonstrate the superiority of our LTM. The results are shown in Table 1. The inference time of all the models is also evaluated on the validation set of ImageNet-1k and reported in milliseconds (ms) per batch for an evaluation batch size of 128 on one Nvidia A100 GPU. We also evaluate the training cost of our LTM models compared with the other fine-tuning based token merging method, LTMP [14]. The training time is evaluated on 4 NVIDIA A100 GPUs with an effective batch size of 512 images. We report the average training time per epoch. The training overhead of LTM-Transformers mainly comes from the computation of $\left\{\phi(\tilde{X}_i, a)\right\}_{i \in \mathcal{B}_j, a \in [C]}$, $\left\{Q^{(t-1)}(\tilde{X} \in a | Y = y)\right\}_{a \in [C], y \in [C]}$, and $\left\{\tilde{\mathcal{C}}_a^{(t-1)}\right\}_{a=1}^{C}$ as described in Algorithm 1. It is observed that LTM only brings a marginal training time overhead compared to the competing token merging method.

Training-from-Scratch Setup. As shown in Table 2, LTM models show reduced FLOPs and enhanced accuracy compared to their original visual transformer counterparts. For instance, LTM-MobileViT-S not only reduces the FLOPs of MobileViT-S from 1.4G to 1.17G but also improves the top-1 accuracy by 1.3%. Similarly, LTM-Swin-T achieves a 0.5% accuracy increase while lowering the FLOPs from 4.5G to 3.7G compared to the original Swin-T. Moreover, we compare LTM models against models compressed by current state-of-the-art weight pruning methods for efficient visual transformers, including S^2ViTE [37], SPViT [27], and SAViT [29] on EfficientViT-B1 (r224). To apply S^2ViTE, SPViT, and SAViT on EfficientViT-B1 (r224), we first run their pruning process following the standard implementation in their papers [27], [29], [37] on the ImageNet training data. After obtaining the pruned networks, we fine-tune the pruned models following the same setting as stated in their papers [27], [29], [37]. It is observed that LTM-EfficientViT-B1 outperforms all pruned models by at least 0.9% in top-1 accuracy with even less FLOPs.

Feature backbone	# Params.	FLOPs	mAP
MobileNetv3 [38]	4.9 M	1.4 G	22.0
MobileNetv2 [39]	4.3 M	1.6 G	22.1
MobileNetv1 [40]	5.1 M	2.6 G	22.2
MixNet [41]	4.5 M	2.2 G	22.3
MNASNet [42]	4.9 M	1.7 G	23.0
YoloV5-N (640×640) [43]	1.9 M	4.5 G	28.0
Vidt [44]	7.0 M	6.7 G	28.7
MobileViT-XS	2.7 M	1.7 G	24.8
LTM-MobileViT-XS(Ours)	2.7 M	1.5 G	**25.4**
MobileViT-S	5.7 M	2.4 G	27.7
LTM-MobileViT-S(Ours)	5.7 M	2.1 G	**28.4**
EfficientViT	9.9 M	1.5 G	28.4
LTM-EfficientViT(Ours)	9.9 M	1.4 G	**28.9**

TABLE 3: Object detection performance with SSDLite.

4.2 Object Detection

Implementation details. We incorporate ImageNet pre-trained models, that are LTM-MobileViT-XS, LTM-MobileViT-S, and LTM-EfficientViT, with the single-shot object detection backbone, SSDLite [39], to evaluate on the MS-COCO dataset [45], which comprises 117k training images and 5k validation images. We fine-tune all pre-trained LTM-Transformers within the object detection framework at a standard input resolution of 320×320. These models undergo a training period of 200 epochs using the AdamW optimizer, adhering to the training protocols established in [18]. Employing a cosine learning rate scheduler, the initial learning rate of 0.0009 is gradually reduced to $1.6e^{-6}$. For the object localization, we utilize a smooth ℓ^1 loss, and for classification, cross-entropy losses are applied. The evaluation of performance on the validation set is conducted using the mAP metric with an IoU range from 0.50 to 0.95 in increments of 0.05.

Methods	mAP^{box}	AP$^b_{50}$	AP$^b_{75}$	mAP^m	AP$^m_{50}$	AP$^m_{75}$
EViT [21]	32.8	54.4	34.5	31.0	51.2	32.2
EfficientViT-B1 [6]	33.5	55.4	34.8	31.9	52.3	32.7
LTM-EfficientViT-B1	**34.3**	**56.1**	**35.2**	**32.8**	**52.8**	**33.1**

TABLE 4: Instance Segmentation Results on COCO.

Results. We conduct a comparative study of our LTM Transformers against other lightweight feature backbones within the SSDLite object detection framework. The results, as detailed in Table 3, illustrate significant improvements in object detection performance when the feature backbone is upgraded to include LTM-Transformer blocks. For example, substituting MobileViT-S with LTM-MobileViT-S enhances the mAP by 0.7% while concurrently reducing FLOPs by

0.3G. In addition, SSDLite equipped with LTM-EfficientViT achieves a substantial performance increase of 6.9% while maintaining the same FLOPs as MobileNetV3.

4.3 Instance Segmentation

In this section, we assess the efficacy of LTM when applied to instance segmentation tasks using the COCO dataset [45]. We utilize Mask R-CNN [46] equipped with a Feature Pyramid Network (FPN) as the segmentation head, built on the LTM-EfficientViT-B1 feature backbone. For comparative analysis, we include EfficientViT-B1 [6] and EViT [21] as baseline models. Both our models and the baselines are trained on the training split of the COCO dataset and evaluated on the validation split, adhering to the protocols established by [47]. The training duration is set to 12 epochs, consistent with the $1\times$ schedule described in [47]. The AdamW optimizer is employed for training following the practices of [21]. We initiate the learning rate at 0.001, which is then gradually reduced following a cosine learning rate schedule. Performance metrics reported include the mean bounding box Average Precision (mAP^b) and mean mask Average Precision (mAP^m), along with bounding box Average Precision (AP^b) and mask Average Precision (AP^m) at IoU thresholds of 0.5 and 0.75. The findings, detailed in Table 4, demonstrate that LTM-EfficientViT-B1 consistently enhances segmentation performance across various thresholds. For example, LTM-EfficientViT-B1 outperforms EfficientViT-B1 by 0.8% and 0.9% in mAP^b and mAP^m, respectively.

4.4 Ablation Study

4.4.1 Study on the Impact of Compression Ratio

We conduct an ablation study on the compression ratio of token merging on ViT-B. We train LTM-ViT-B models with different compression ratios in both the train-from-scratch setup and fine-tuning setup. In the fine-tuning setup, all models are fine-tuned for 50 epochs. It is observed from the results in Table 5 that although a smaller compression ratio can result in a slight accuracy drop, the LTM-ViT-B with a compression ratio of 0.65 can still achieve better performance than the original ViT-B model either fine-tuned from the pre-trained checkpoint or trained from scratch.

Methods	FLOPs (G)	r	Train-from-scratch	Fine-tuning
ViT-B	17.58	1.00	83.74	83.74
LTM-ViT-B	16.55	0.95	84.43	84.32
LTM-ViT-B	15.25	0.90	84.46	84.29
LTM-ViT-B	14.19	0.85	84.33	84.35
LTM-ViT-B	14.89	0.80	84.15	84.20
LTM-ViT-B	13.49	0.75	83.95	84.05
LTM-ViT-B	12.85	0.70	83.87	83.96
LTM-ViT-B	11.95	0.65	83.75	83.81
LTM-ViT-B	11.03	0.60	83.53	83.56
LTM-ViT-B	10.15	0.55	83.07	83.14
LTM-ViT-B	9.63	0.50	82.87	82.95
LTM-ViT-B	8.77	0.45	83.46	83.51
LTM-ViT-B	8.30	0.40	82.23	82.37

TABLE 5: Performance comparison between LTM-ViT-B with different compression ratios. The models in the fine-tuning setup are fine-tuned for 50 epochs.

4.4.2 Study on the Effects of LTM in Reducing IB Loss

We study the effectiveness of LTM in reducing the IB loss and the variational upper bound of IB loss, which is the IB bound, across three vision transformers, including MobileViT-S, MobileViT-XS, and EfficientViT (r224). We compare the performance of the vision transformers with the baseline token merging method, ToME [15], LTMP [14], and the corresponding LTM-Transformer models with all the transformer blocks replaced with the LTM blocks. The ablation study results for the fine-tuning setup are shown in Table 6. The ablation study results for the train-from-scratch setup are shown in Table 7. The results indicate that although ToMe and LTMP reduce the IB loss and the IB bound in both the fine-tuning setup and the train-from-scratch setup, thereby adhering to the IB principle, which aims to enhance the correlation of features with class labels while reducing their correlation with the input, LTM can further decrease the IB loss and IB bound. In particular, our LTM models improve the vanilla vision transformers, the ToMe models, and the LTMP models by a large margin in terms of both IB loss and top-1 accuracy for both the fine-tuning setup and the train-from-scratch setup. For instance, the LTMP-ViT-B fine-tuned for 50 epochs reduces the IB loss of ViT-B by 0.00333. The LTM-ViT-B fine-tuned for 50 epochs further reduces the IB loss of LTMP-ViT-B fine-tuned for 50 epochs by 0.00866.

4.4.3 Study on the IB loss and IB bound at Different Layers

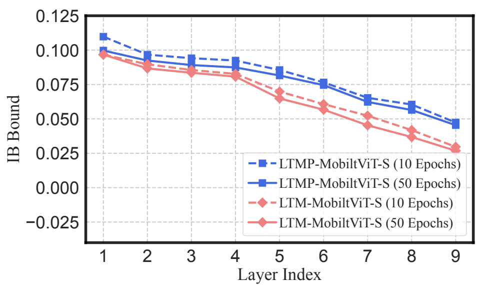
To study how the IB loss $\text{IB}(G)$, and the variational upper bound for the IB loss, $\text{IBB}(G)$, decrease with respect to layer index ℓ of a LTM-Transformer network, $\text{IB}(G)$ and $\text{IBB}(G)$ for both MobileViT-S and LTM-MobileViT-S across different transformer layers are illustrated in Figure 2. Both models contain 9 transformer layers. It is observed from Figure 2 that both $\text{IB}(G)$ and $\text{IBB}(G)$ decrease in deeper layers with larger layer indices of MobileViT-S and LTM-MobileViT-S. This observation suggests that features in deeper layers correlate more closely with the class labels and less with the input features, adhering to the IB principle. Moreover, LTM-MobileViT-S reduces both $\text{IB}(G)$ and $\text{IBB}(G)$ to lower levels in deeper layers compared to MobileViT-S. These observations evidence that the mask module in the LTM-Transformer block, which generates the informative token merging mask by (3) can effectively reduce both $\text{IB}(G)$ and $\text{IBB}(G)$, better adhering to the IB principle than the vanilla MobileViT-S. At the early stage of the training after 100 epochs, the IB bound and the IB loss of LTM-MobileViT-S are similar to those of the MobileViT-S. After training for 300 epochs, the IB bound and the IB loss of LTM-MobileViT-S are much smaller than those of the MobileViT-S.

4.4.4 Training Loss and Test Loss of LTM-Transformers

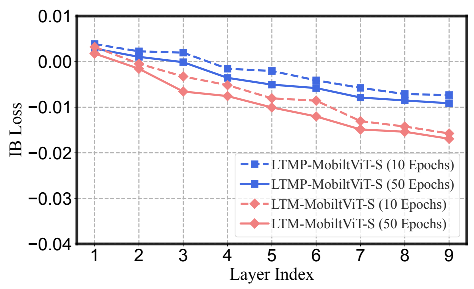
In this section, we illustrate the training loss and the test loss of LTM-MobileViT-XS, LTM-MobileViT-S, and LTM-EfficientViT-B1. In comparison, we also illustrate the training loss and test loss of MobileViT-XS, MobileViT-S, and EfficientViT-B1. All the models are trained for 300 epochs for each cycle with two cycles. The plots are illustrated in Figure 3 for the second cycle. It can be observed that LTM-Transformer networks achieve lower training losses and test losses at the end of the training, which demonstrates the

Model	FLOPs	Top-1				IB Bound				IB Loss			
		0	1	10	50	0	1	10	50	0	1	10	50
MobileViT-S	1.40 G	78.40	-	-	-	0.05782	-	-	-	-0.00432	-	-	-
ToMe-MobileViT-S	1.22 G	76.72	-	-	-	0.04931	-	-	-	-0.00525	-	-	-
LTMP-MobileViT-S	1.26 G	-	77.53	77.82	78.14	-	0.04902	0.04735	0.04542	-	-0.00765	-0.00874	-0.00913
LTM-MobileViT-S	1.17 G	-	**77.72**	**78.34**	**79.05**	-	**0.03095**	**0.02967**	**0.02683**	-	**-0.01430**	**-0.01576**	**-0.01692**
EfficientViT-B1	0.52 G	79.40	-	-	-	0.06014	-	-	-	-0.00451	-	-	-
ToMe-EfficientViT-B1	0.47 G	78.81	-	-	-	0.04642	-	-	-	-0.00732	-	-	-
LTMP-EfficientViT-B1	0.52 G	-	79.21	79.32	79.40	-	0.04537	0.04219	0.03970	-	-0.00802	-0.00916	-0.00995
LTM-EfficientViT-B1	0.44 G	-	**79.39**	**79.62**	**80.22**	-	**0.02874**	**0.02703**	**0.02635**	-	**-0.01585**	**-0.01664**	**-0.01704**
ViT-B	17.58 G	83.74	-	-	-	0.05539	-	-	-	-0.00419	-	-	-
ToMe-ViT-B	13.12 G	82.86	-	-	-	0.04583	-	-	-	-0.00647	-	-	-
LTMP-ViT-B	13.46 G	-	83.29	83.44	83.55	-	0.04392	0.04275	0.04086	-	-0.00665	-0.00693	-0.00752
LTM-ViT-B	12.85 G	-	**83.35**	**83.76**	**83.96**	-	**0.03732**	**0.03506**	**0.03082**	-	**-0.01425**	**-0.01572**	**-0.01618**

TABLE 6: Ablation study on the effects of LTM in reducing IB loss in the fine-tuning setup. Both LTMP models and LTM models fine-tuned for 1, 10, 50 epochs are evaluated.

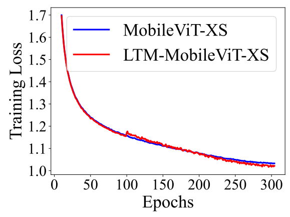


(a) IB bound (IBB(G)) comparison between MobileViT-S and LTM-MobileViT-S.

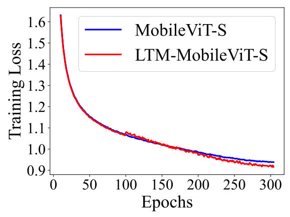


(b) IB loss (IB(G)) comparison between MobileViT-S and LTM-MobileViT-S.

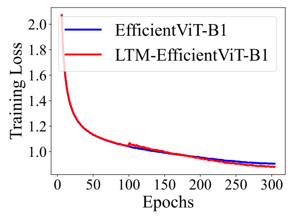
Fig. 2: IB bound and IB loss comparison between MobileViT-S and LTM-MobileViT-S at different transformer layers.



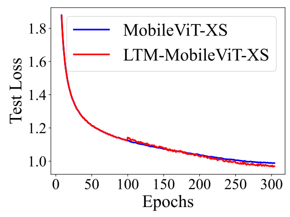
(a) Training loss comparison between MobileViT-XS and LTM-MobileViT-XS.



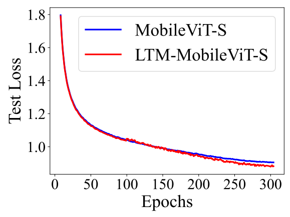
(b) Training loss comparison between MobileViT-S and LTM-MobileViT-S.



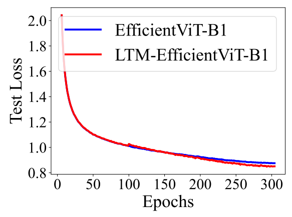
(c) Training loss comparison between EfficientViT-B1 and LTM-EfficietViT-B1.



(d) Test loss comparison between MobileViT-XS and LTM-MobileViT-XS.



(e) Test loss comparison between MobileViT-S and LTM-MobileViT-S.



(f) Test loss comparison between EfficientViT-B1 and LTM-EfficietViT-B1.

Fig. 3: Training loss and test loss comparison between LTM-Transformer networks and corresponding baseline models.

Model	FLOPs	Top-1	IB Bound	IB Loss
MobileViT-S	1.40 G	78.40	0.05782	-0.00432
ToMe-MobileViT-S	1.22 G	76.72	0.04931	-0.00525
LTMP-MobileViT-S	1.26 G	78.14	0.04542	-0.00913
LTM-MobileViT-S	1.17 G	**79.68**	**0.02425**	**-0.01725**
EfficientViT-B1	0.52 G	79.40	0.06014	-0.00451
ToMe-EfficientViT-B1	0.47 G	78.81	0.04642	-0.00732
LTMP-EfficientViT-B1	0.52 G	79.40	0.03970	-0.00995
LTM-EfficientViT-B1	0.44 G	**80.20**	**0.02689**	**-0.01730**
ViT-B	17.58 G	83.74	0.05539	-0.00419
ToMe-ViT-B	13.12 G	82.86	0.04583	-0.00647
LTMP-ViT-B	13.46 G	83.55	0.04086	-0.00752
LTM-ViT-B	12.85 G	**83.87**	**0.03094**	**-0.01636**

TABLE 7: Ablation study on the effects of LTM in reducing IB loss in the train-from-scratch setup.

benefit of LTM-Transformers in improving the performance of the visual transformers through the IB-inspired token merging.

4.4.5 *Visualization Results*

To study the effectiveness of LTM-Transformer in selecting informative tokens during the token merging process, we visualize the token merging masks in the first transformer block of LTMP-MobileViT-S and LTM-MobileViT-S for particular target tokens in selected images from ImageNet in Figure 4. Each image is divided into 16×16 tokens. For each example, we select only the most representative target token

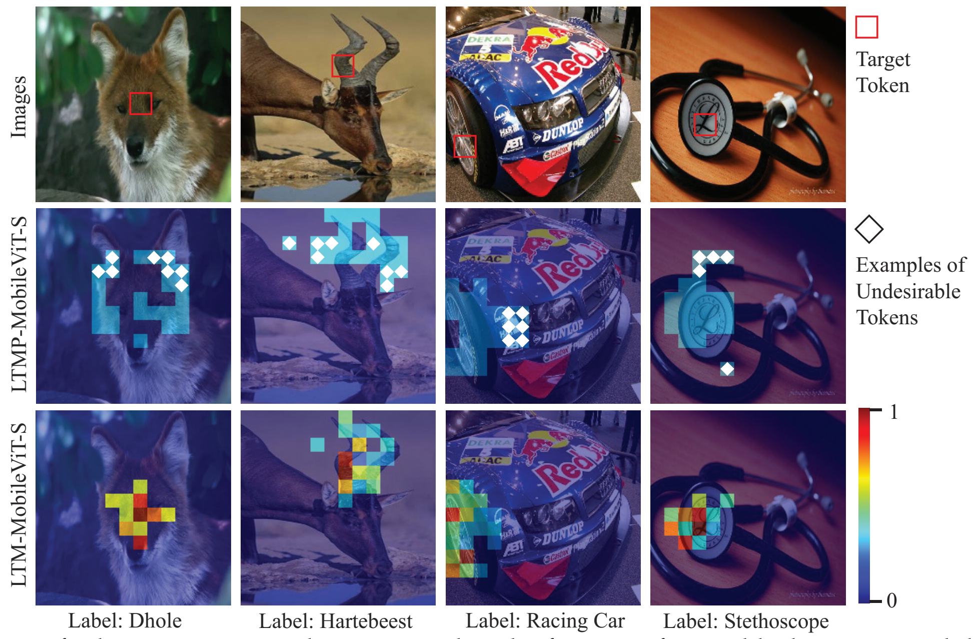


Fig. 4: Comparisons of token merging weights generated in the first transformer block in LTM-MobileViT-S and LTMP-MobileViT-S. The target tokens resulting from the token merging process are marked with red boxes in the first row. The token merging weights for tokens merged into the target tokens by LTMP-MobileViT-S are illustrated in the second row. The token merging weights for tokens merged into the target tokens by LTM-MobileViT-S are illustrated in the third row. Tokens that are more semantically similar to the target tokens should receive higher merging weights so as to achieve informative token merging. Examples of the undesirable tokens, which are not semantically similar to the target tokens, are marked with white diamonds. In contrast with LTMP, the heatmaps of the merging weights generated by LTM illustrate that LTM usually assigns larger weights to tokens that are more semantically similar to the target tokens.

that encapsulates the critical features of the objects in the image, and the target token is a weighted average of several self-attention output tokens with the token merging weights in the token merging mask. The input images are illustrated in the first row. Every target token is marked in red boxes in the first row, which is calculated by $\tilde{X}(G^{(\ell)}) = \left(Z^\top G^{(\ell)}\right)^\top$ as described in Section 3.1. The third row illustrates the token merging mask $G^{(\ell)}$ generated by the LTM for the target tokens with heatmaps. We remark that the token merging masks for LTM and LTMP contain the token merging weights for these two methods. The token merging masks generated by LTMP for the same target tokens are illustrated in the second row. The class labels for each image are presented at the bottom of each column. Tokens that are more semantically similar to the target token are expected to receive larger token merging weights in the token merging masks, so that the original tokens can contribute to the target token in accordance to their informativeness with respect to the target token. The visualization results illustrate that the mask module in the LTM-Transformer block usually assigns higher merging weights to tokens that are more semantically similar to the target token. In contrast, LTMP assigns uniform merging weights to all the selected tokens, which usually contain some undesirable tokens that are not semantically similar or even relevant to the target tokens. In the example of the dhole in the first column, the LTM-Transformer block assigns larger merging weights to tokens around the eyes of the hole as the target token is also around the eyes. In contrast, the competing baseline LTMP mistakenly merges tokens on the ears of the hole into the target token. In the example of the hartebeest in the second column, the LTM-Transformer block assigns larger weights to tokens on the twisted horns of the hartebeest as the target token is also on the twisted horns of the hartebeest. The uneven distribution of token merging weights computed by LTM is consistent with the goal of informative token merging in our LTM-Transformer, where more informative original tokens, whose features are more similar to that of the target token, should receive higher token merging weights than other original tokens. In contrast, the competing baseline LTMP mistakenly merges tokens in the background around the twisted horns of the hartebeest. In the example of the racing car in the third column, the LTM-Transformer block assigns larger weights to tokens on the wheel of the car as the target token is also on the wheel. In contrast, the competing baseline LTMP mistakenly merges tokens around the light of the car. In the example of the Stethoscope in the fourth column, the LTM-Transformer block assigns larger weights to tokens on the diaphragm of the stethoscope. In contrast, the competing baseline LTMP mistakenly merges tokens on the top of the diaphragm of the stethoscope in the background.

In addition, the visualization results illustrate that LTMP usually assigns larger weights to tokens that are more semantically similar to the target tokens. For instance, in the example of the hartebeest, tokens on the left horn of the hartebeest have larger token merging weights than tokens on the right horn of the hartebeest because the target token is on the left horn, which exhibits visual patterns that are not shared on the right horn. Similarly, in the example of the stethoscope, tokens around the center of the diaphragm have larger merging weights than tokens on the boundary of the diaphragm because the target token is in the center of the diaphragm, which exhibits visual patterns that are not shared on the boundary of the diaphragm.

4.4.6 Study on the Impacts of Compression Ratio at Different Layers of the LTM-Transformer

In our experiments, the compression ratio r for all the layers in the LTM-Transformer is set to a fixed value of 0.7. To study how the compression ratio at different layers in the LTM-Transformer influences the IB loss at the corresponding layer, we train ablation models by changing the compression ratios at different layers from 0.7 to 0.3, 0.5, and 0.9. For each of the ablation models, we only change the compression ratio at a specific layer and keep the compression ratios for other layers fixed at 0.7. The ablation study is performed for all 9 layers of the LTM-MobileViT-S in the fine-tuning setup, and all ablation models are fine-tuned for 50 epochs. Let ℓ denote the layer index where the compression ratio is changed. The IB loss at the corresponding layer where the compression ratio is changed and the top-1 accuracy of the ablation models are shown in Table 8. It is observed that reducing the compression ratio from $r = 0.9$ to $r = 0.7$ can reduce the IB loss at the corresponding layer and increase the top-1 accuracy, benefiting from merging redundant tokens. However, further reducing the compression ratio to $r = 0.5$ and $r = 0.3$ leads to an increased IB loss and reduced top-1 accuracy, especially at bottom layers where $\ell \leq 5$. This is attributed to the excessive merging of tokens, potentially resulting in the loss of informative features that are critical for the classification task.

Layer	$r = 0.3$		$r = 0.5$		$r = 0.7$		$r = 0.9$	
Index ℓ	Top-1	IB Loss	Top-1	IB Loss	Top-1	IB Loss	Top-1	IB Loss
1	78.22	0.00214	78.70	0.00190	79.05	0.00178	79.02	0.00185
2	78.43	-0.00137	78.85	-0.00140	79.05	-0.00156	79.01	-0.00148
3	78.55	-0.00622	78.90	-0.00630	79.05	-0.00648	78.98	-0.00641
4	78.57	-0.00725	79.05	-0.00742	79.02	-0.00762	78.98	-0.00750
5	78.75	-0.00938	78.83	-0.00950	79.05	-0.01005	79.00	-0.00977
6	78.95	-0.01185	79.02	-0.01197	79.05	-0.01204	79.04	-0.01199
7	79.00	-0.01430	79.04	-0.01464	79.05	-0.01488	79.01	-0.01486
8	79.00	-0.01525	79.03	-0.01556	79.05	-0.01592	79.05	-0.01592
9	79.04	-0.01718	79.03	-0.01704	79.05	-0.01725	79.00	-0.01728

TABLE 8: Ablation study on the impacts of compression ratio in reducing the IB loss at different layers of the LTM-MobileViT-S. All ablation models are fine-tuned for 50 epochs. ℓ denotes the layer index where the compression ratio is changed. The IB loss at the ℓ-th layer and the top-1 accuracy of the ablation models are shown.

In addition, we have performed an ablation study on evaluating the transfer learning capability of the LTM-Transformer in Section 2.1 of the supplementary. The results show that LTM does not affect the transfer learning capability of the original model. Moreover, we have shown that the parameter-efficient post-training method, LoRA [48], can significantly reduce the training costs of the LTM-Transformer without scarfing the performance in Section 2.2 of the supplementary. In Section 2.3 of the supplementary, we have shown that the LTM-Transformer exhibits faster inference speed than the models compressed by competing token merging methods across different batch sizes. In Section 3 of the supplementary, we have discussed the potential future work of this paper for improved training efficiency.

5 Conclusion

In this paper, we propose a novel transformer block, Transformer with Learnable Token Merging, or LTM-Transformer. LTM-Transformer blocks perform token merging so as to render a transformer network with less FLOPs and faster inference speed. A LTM-Transformer block generates an informative token merging mask for token merging in a learnable manner, which is inspired by the reduction of the Information-Bottleneck loss. A network with LTM-Transformer blocks can either be trained from scratch or fine-tuned from the pre-trained backbone, and it enjoys a reduction of IB loss and reduced FLOPs while maintaining a compelling prediction accuracy. We demonstrate the effectiveness of LTM-Transformer by replacing all the transformer blocks in several popular visual transformers with LTM-Transformer blocks. Extensive experiments on various tasks demonstrate the effectiveness of the LTM-Transformer.

References

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

[2] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z. Jiang, F. E. Tay, J. Feng, and S. Yan, "Tokens-to-token vit: Training vision transformers from scratch on imagenet," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021.

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[4] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 2021, pp. 9992–10 002.

[5] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," *ICLR*, 2021.

[6] H. Cai, J. Li, M. Hu, C. Gan, and S. Han, "Efficientvit: Lightweight multi-scale attention for high-resolution dense prediction," in *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*. IEEE, 2023, pp. 17 256–17 267.

[7] Y. Wang, P. Li, and Y. Yang, "Visual transformer with differentiable channel selection: an information bottleneck inspired approach," in *Forty-first International Conference on Machine Learning*, 2024.

[8] Y. Wang, R. Goel, U. Nath, A. Silva, T. Wu, and Y. Yang, "Learning low-rank feature for thorax disease classification," *Advances in Neural Information Processing Systems*, vol. 37, pp. 117 133–117 163, 2024.

[9] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International conference on machine learning*. PMLR, 2021.

[10] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems*, 2015, pp. 1135–1143.

[11] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[12] W. Sun, Y. Jiang, S. Zhang, and Y. Liu, "Cascade pruning: Towards class imbalance in convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9860–9869.

[13] M. Kim, S. Gao, Y.-C. Hsu, Y. Shen, and H. Jin, "Token fusion: Bridging the gap between token pruning and token merging," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 1383–1392.

[14] M. Bonnaerens and J. Dambre, "Learned thresholds token merging and pruning for vision transformers," *Transactions on Machine Learning Research*, 2023. [Online]. Available: https://openreview.net/forum?id=WYKTCKpImz

[15] D. Bolya, C.-Y. Fu, X. Dai, P. Zhang, C. Feichtenhofer, and J. Hoffman, "Token merging: Your vit but faster," *ICLR*, 2023.

[16] Y. Rao, W. Zhao, B. Liu, J. Lu, J. Zhou, and C. Hsieh, “Dynamicvit: Efficient vision transformers with dynamic token sparsification,” in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 2021, pp. 13 937–13 949.

[17] I. Bello, B. Zoph, A. Vaswani, J. Shlens, and Q. V. Le, “Attention augmented convolutional networks,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 3286–3295.

[18] S. Mehta and M. Rastegari, “Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer,” *ICLR*, 2022.

[19] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” in *European Conference on Computer Vision*, 2020.

[20] Y. Wang, N. Xu, C. Chen, and Y. Yang, “Adaptive cross-layer attention for image restoration,” *arXiv preprint arXiv:2203.03619*, 2022.

[21] X. Liu, H. Peng, N. Zheng, Y. Yang, H. Hu, and Y. Yuan, “Efficientvit: Memory efficient vision transformer with cascaded group attention,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 14 420–14 430.

[22] B. Graham, A. El-Nouby, H. Touvron, P. Stock, A. Joulin, H. Jégou, and M. Douze, “Levit: a vision transformer in convnet’s clothing for faster inference,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 12 259–12 269.

[23] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, “Designing network design spaces,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020.

[24] C. Gong, D. Wang, M. Li, X. Chen, Z. Yan, Y. Tian, qiang liu, and V. Chandra, “NASVit: Neural architecture search for efficient vision transformers with gradient conflict aware supernet training,” in *International Conference on Learning Representations*, 2022.

[25] T. Chen, Y. Cheng, Z. Gan, L. Yuan, L. Zhang, and Z. Wang, “Chasing sparsity in vision transformers: An end-to-end exploration,” *Advances in Neural Information Processing Systems*, 2021.

[26] F. Yu, K. Huang, M. Wang, Y. Cheng, W. Chu, and L. Cui, “Width & depth pruning for vision transformers,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 3, 2022, pp. 3143–3151.

[27] Z. Kong, P. Dong, X. Ma, X. Meng, W. Niu, M. Sun, X. Shen, G. Yuan, B. Ren, H. Tang, M. Qin, and Y. Wang, “Spvit: Enabling faster vision transformers via latency-aware soft token pruning,” in *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XI*, ser. Lecture Notes in Computer Science, vol. 13671. Springer, 2022, pp. 620–640.

[28] A. Chavan, Z. Shen, Z. Liu, Z. Liu, K.-T. Cheng, and E. P. Xing, “Vision transformer slimming: Multi-dimension searching in continuous optimization space,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[29] C. Zheng, Z. Li, K. Zhang, Z. Yang, W. Tan, J. Xiao, Y. Ren, and S. Pu, “Savit: Structure-aware vision transformer pruning via collaborative optimization,” in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[30] S. Yu, T. Chen, J. Shen, H. Yuan, J. Tan, S. Yang, J. Liu, and Z. Wang, “Unified visual transformer compression,” *ICLR*, 2022.

[31] Z. Wang, H. Luo, P. Wang, F. Ding, F. Wang, and H. Li, “Vtc-lfc: Vision transformer compression with low-frequency components,” *Advances in Neural Information Processing Systems*, vol. 35, 2022.

[32] A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey, and D. D. Cox, “On the information bottleneck theory of deep learning,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2019, no. 12, p. 124020, 2019.

[33] Q. Lai, Y. Li, A. Zeng, M. Liu, H. Sun, and Q. Xu, “Information bottleneck approach to spatial attention learning,” in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*. ijcai.org, 2021, pp. 779–785.

[34] D. Zhou, Z. Yu, E. Xie, C. Xiao, A. Anandkumar, J. Feng, and J. M. Alvarez, “Understanding the robustness in vision transformers,” in *International Conference on Machine Learning*. PMLR, 2022, pp. 27 378–27 394.

[35] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, “An image is worth 16x16 words: Transformers for image recognition at scale,” in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and L. Fei-Fei, “Imagenet large scale visual recognition challenge,” *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[37] T. Chen, Y. Cheng, Z. Gan, L. Yuan, L. Zhang, and Z. Wang, “Chasing sparsity in vision transformers: An end-to-end exploration,” *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[38] A. Howard, R. Pang, H. Adam, Q. V. Le, M. Sandler, B. Chen, W. Wang, L. Chen, M. Tan, G. Chu, V. Vasudevan, and Y. Zhu, “Searching for mobilenetv3,” in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, pp. 1314–1324.

[39] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *IEEE conference on computer vision and pattern recognition*, 2018.

[40] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *CoRR*, vol. abs/1704.04861, 2017.

[41] M. Tan and Q. V. Le, “Mixconv: Mixed depthwise convolutional kernels,” in *British Machine Vision Conference (BMVC)*, 2019.

[42] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, “Mnasnet: Platform-aware neural architecture search for mobile,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2820–2828.

[43] J. Redmon and A. Farhadi, “Yolo9000: better, faster, stronger,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[44] H. Song, D. Sun, S. Chun, V. Jampani, D. Han, B. Heo, W. Kim, and M.-H. Yang, “Vidt: An efficient and effective fully transformer-based object detector,” *ICLR*, 2022.

[45] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European conference on computer vision*. Springer, 2014.

[46] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” in *IEEE International Conference on Computer Vision*, 2017.

[47] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, “Mmdetection: Open mmlab detection toolbox and benchmark,” *CoRR*, vol. abs/1906.07155, 2019.

[48] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “Lora: Low-rank adaptation of large language models,” in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Yancheng Wang is a Ph.D. student in Computer Science at Arizona State University. He received his B.S. degree in Mathematics and Applied Mathematics from Xi’an Jiaotong University in 2018. His research interest is in developing robust and efficient deep learning methods, such as neural architecture search, and their applications to image and graph data.

Yingzhen Yang Dr. Yingzhen Yang has been an assistant professor at SCAI since 2019, and his research areas include deep learning and statistical machine learning with more than 70 papers published in these areas. He is the recipient of the Best Paper Finalist for ECCV (European Conference on Computer Vision) in 2016. He has served as a senior program committee member, a committee member or a reviewer for premier conferences and journals in machine learning, deep learning, and artificial intelligence, including Journal of Machine Learning Research (JMLR), Journal of Artificial Intelligence Research (JAIR), IEEE Transactions on Image Processing (TIP), the International Joint Conferences on Artificial Intelligence (IJCAI), the Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence, the International Conference on Machine Learning (ICML), the Annual Conference on Neural Information Processing Systems (NeurIPS), the International Conference on Learning Representations (ICLR).