RaftMLP: How Much Can Be Done Without Attention and with Less Spatial Locality?

Yuki Tatsunami¹,² and Masato Taki¹

¹ Rikkyo University, Tokyo, Japan
² AnyTech Co., Ltd., Tokyo, Japan

Abstract. For the past ten years, CNN has reigned supreme in the world of computer vision, but recently, Transformer has been on the rise. However, the quadratic computational cost of self-attention has become a serious problem in practice applications. There has been much research on architectures without CNN and self-attention in this context. In particular, MLP-Mixer is a simple architecture designed using MLPs and hit an accuracy comparable to the Vision Transformer. However, the only inductive bias in this architecture is the embedding of tokens. This leaves open the possibility of incorporating a non-convolutional (or non-local) inductive bias into the architecture, so we used two simple ideas to incorporate inductive bias into the MLP-Mixer while taking advantage of its ability to capture global correlations. A way is to divide the token-mixing block vertically and horizontally. Another way is to make spatial correlations denser among some channels of token-mixing. With this approach, we were able to improve the accuracy of the MLP-Mixer while reducing its parameters and computational complexity. The small model that is RaftMLP-S is comparable to the state-of-the-art global MLP-based model in terms of parameters and efficiency per calculation. Our source code is available at https://github.com/okojoalg/raft-mlp.

Keywords: Image classification · Network architecture · Multilayer perceptron.

1 Introduction

In the past decade, CNN-based deep architectures have been developed in the computer vision domain. The first of these models was AlexNet [24], followed by other well-known models such as VGG [34], GoogLeNet [35], and ResNet [15]. These CNN-based models have exhibited high accuracy in various tasks, including image classification, object detection, semantic segmentation, and image generation. Adopting convolution, they employ the inherent inductive bias of images. Meanwhile, Transformer [45] has been winning success in recent years in the field of Natural Language Processing (NLP). Inspired by this success, Vision Transformer (ViT) [11] has been proposed. ViT is a Transformer-based visual
model that replaces CNN with the self-attention mechanism. The main idea of ViT is to divide the image into patches based on their spatial locations and apply the Transformer using these patches as tokens. Immediately after the ViT paper appeared, various related works \cite{1,4,10,12,13,29,46,56,52,55} have been done. They have shown that Transformer-based models are competitive with or even exceed CNN-based models in various image recognition and generation tasks. Although Transformer-based models have a reduced inductive bias for images compared to CNN-based models, they compensate for this lack by using a vast array of parameters and computational complexity instead. Moreover, it is successful because it can capture global correlations due to replacing the local receptive fields of convolution with global attention.

More recently, there has been a growing interest in improving the computational complexity of computationally intensive self-attention. Some works \cite{31,40,41} claim that Multi Layer Perceptron (MLP) alone is sufficient for image tasks without self-attention. In particular, MLP-Mixer \cite{40} has performed a wide variety of MLP-based experiments, and the accuracy of image classification is not better than ViT, but the results are comparable. The MLP-based model, like ViT, first decomposes an image into tokens. A combined operation of MLP, transposition, and activation functions follows the tokenization. The significant point to note is that the transposition operation switches from token-mixing block to channel-mixing block and vice versa. While the channel-mixing block is equivalent to 1x1 convolution in CNN, the token-mixing block is a module that can capture the global correlations between tokens.
The wonderful thing about the MLP-Mixer is that it exhibited the possibility of competing with the existing models with a simple architecture without convolution nor self-attention. In particular, the fact that a simple MLP-based model could compete with current models leads us to think about successors to convolution. This idea has triggered the interest of many researchers on whether computer vision tasks can outgrow the classical convolution paradigm that has been in the mainstream for ten years. Motivated by the MLP-Mixer, some architectures have been proposed that inject convolutional local structures in pursuit of accuracy. We call the models with such structures local MLP-based models. In contrast, models such as MLP-Mixer, which adopt a design to capture global correlations without local operation, are called global MLP-based models. The global MLP-based model, including MLP-Mixer, has a shortcoming with the models. Unlike convolution, the resolution of the images used for training and inference is fixed, and thwarts the application to downstream tasks such as object detection and semantic segmentation. This paper aims to achieve cost-effectiveness with fewer resources in developing a global MLP-based model. The contributions of this study are as follows.

**Spatial structure** As shown in Fig. 1, we propose a module in which the token mixing block is divided into vertical and horizontal mixing blocks in series. In the standard MLP-Mixer, the relevance of patches has no inductive bias in the vertical and horizontal directions in the original two-dimensional image. In our proposed model, we implicitly assume as an inductive bias that patch sequences aligned horizontally have similar correlations with other horizontally aligned patch sequences. The same can be said for vertically aligned patch sequences—additionally, groups of channels are jointed in tensors before inputting into vertical-mixing and horizontal-mixing blocks. Jointed channels are shared with both mixing blocks. Thus, we assume that there are objects and their visual patterns are often distributed linearly over an image and geometrical relation among some channels.

**Multi-scale patch embedding** While ViT and MLP-Mixer patch embedding was a simple method; we added a hierarchical structure. That is multi-scale patch embedding, which embeds information around the patch in the original patch embedding, as shown in Fig. 3. The multi-scale patch embedding method, which also embeds information around the patch in the embedding of the original patch, helped us increase the accuracy at the cost of a small amount of computation and memory consumption.

We will demonstrate that the proposed model with a simple inductive bias without excessive spatial locality as convolution is superior to MLP-Mixer and comparable to global MLP-based models. In addition, we will mention that the proposed method is a model that can achieve accuracy at a reduced cost compared to previous studies. In the appendix, we will study the applicability of the proposed model to downstream tasks such as semantic segmentation, instance segmentation, and object detection. The results will encourage the future possibilities of architectures without self-attention and with less spatial locality.
2 Related Work

Transformer-based models  Originally proposed for NLP, Transformer [45] soon began to be applied to other domains, including visual tasks. In particular, in image recognition, the attention-augmented convolution has been introduced in [3,19,48]. Stand-alone attention for visual task, rather than an augmentation to convolution, is studied in [33], where it was shown that fully self-attentional version of ResNet-50 outperforms the original ResNet in ImageNet classification task.

More Transformer-like architectures, process input tokens by self-attention, rather than augmenting CNNs by attention, were studied in [6] and [11]. In particular, in [11], ViT based on a BERT-type pure Transformer was proposed to deal with high-resolution inputs such as the ImageNet dataset. ViT was pre-trained using a large-scale dataset and transferred to ImageNet, which gave superior results compared to state-of-the-art CNNs.

Inspired by ViT, various transformer-like architectures have been proposed. The most relevant one to our study is CrossFormer [47], which includes a hierarchical structure and Cross-scale Embedding for patch embedding at each level. Cross-scale Embedding effectively injects inductive biases for image domain by using convolution with multiple kernel sizes to perform patch embedding, and it resembles our proposed Multi-scale Patch Embedding in the basic idea. In addition, CrossFormer also proposes a method called Long Short Distance Attention, in which self-attention is divided into two parts, one for long-distance and one for short-distance.

Global MLP-based models  Recently, several alternatives to CNN-based architectures have been proposed that are simple, yet competitive with CNN despite not using convolution or self-attention [40,31,41]. MLP-Mixer [40] replaces the self-attention layer of ViT with simple cross-tokens MLP. Despite its simplicity, MLP-Mixer achieves results that are competitive with ViT. gMLP [28] which consists of an MLP-based module with multiplicative gating is an alternative to MLP-Mixer, achieves higher accuracy than MLP-Mixer with fewer parameters. Vision Permutator [17] focused on mixing in vertical and horizontal directions like our work. Unlike ours, which employs a serialized structure, the Vision Permutator incorporates a parallelized structure, which results in higher accuracy with fewer parameters than the MLP-Mixer. sMLP [39] also shares the idea of decomposing token mixing into vertical and horizontal information mixing. These mixings are performed in parallel and the results are added and output from the module. Another direction of global mixing is CCS-MLP [49] as an example. To achieve translation invariance, CCS-MLP introduces circulant token mixing instead of vanilla token mixing MLP.

Local MLP-based models  Moving to a generic inductive bias like Transformer and MLP has attractive possibilities, but its lack of an inductive bias like convolution means that its pre-training requires vast amounts of data compared to CNNs. In order to achieve good performance without large datasets, MLP-based
RaftMLP architectures have been proposed as an alternative to MLPs such as S$^2$-MLP [50], S$^2$-MLPv2 [51], AS-MLP [26], CycleMLP [5], and ConvMLP [25], which incorporate local structures. Although these models have the name of MLP, their essential motivation is the same as CNN in that they use the local structure of the models to extract patterns efficiently. Hence, we call these MLP-based architectures local MLP-based models. In contrast, architectures that mainly utilize MLPs to capture global correlations, such as MLP-Mixer and our study, are called global MLP-based models.

3 RaftMLP

In this section, we describe MLP-Mixer on which RaftMLP is based and the method adopted for RaftMLP.

3.1 Background

MLP-Mixer [40] splits an inputted image into patches of the same size immediately after input and is followed by MLPs that maintain the patch structure. There are two types of MLP: The first one is the token-mixing block, another is the channel-mixing block. We split an image with height $h$ and width $w$ into tokens with height and width $p$. If $h$ and $w$ are divisible by $p$, by viewing this image as a collection of these tokens, we can regard the image as a data array of height $h' = h/p$, width $w' = w/p$ and channel $cp^2$ where $c$ denotes channel of the inputted image. The number of a token is then $s = hw/p^2$. The token-mixing block is map $\mathbb{R}^s \rightarrow \mathbb{R}^s$ that acts across axes of a token. In contrast, the channel-mixing block is map $\mathbb{R}^c \rightarrow \mathbb{R}^c$ that acts across axes of a channel as well where $c$ is the number of channels. Both blocks contain the same modules: Layer Normalization (LN) [2] for each channel, Gaussian Error Linear Units (GELU) [16] and MLP. Concretely, the following equation gives the blocks

$$X_{\text{output}} = X_{\text{input}} + W_2 \text{GELU}(W_1 \text{LN}(X_{\text{input}})),$$

where $X_{\text{input}}$ denotes input tensor, $X_{\text{output}}$ denotes output tensor, $W_1 \in \mathbb{R}^{a \times ac_a}$, $W_2 \in \mathbb{R}^{ac_a \times a}$ denote matrices of MLP layer, and $a$ denotes expansion factor. For simplicity, the bias term in MLP was omitted. In token-mixing block, $a = s$ and in channel-mixing block, $a = c$. Moreover, the token-axis and channel-axis are permuted between both mixings. In this way, MLP-Mixer [40] is composed of transposition and two types of mixing blocks.

3.2 Vertical-mixing and Horizontal-mixing Block

In the previous subsection, we discussed the token-mixing block. The original token-mixing block does not reflect any two-dimensional structure of an input image, such as height or width direction. In other words, the inductive bias for images is not included in the token-mixing block. MLP-Mixer [40] therefore...
has no inductive bias for images except for how the first patches are made. We decompose this token-mixing block into two blocks that mix vertical and horizontal axes respectively and incorporate inductive bias for image domain. The following describes our method.

The vertical-mixing block is map $R^{h'} \to R^{h'}$ that acts across the vertical axis. Precisely, this map captures correlations along the horizontal axis, utilizing the same MLP along the channel and horizontal dimensions. The map also applies layer normalization for each channel, GELU, and the residual connection. The components of this mixing block are the same as the original token-mixing block.

Similarly, the horizontal-mixing block is map $R^{w'} \to R^{w'}$, and shuffle the horizontal axis. The structure is dual, only replacing vertical and horizontal axes. We propose replacing token-mixing with a successive application of vertical-mixing and horizontal-mixing, assuming meaningful correlations along vertical and horizontal directions of 2D images. This structure is shown in Fig. 1. The formula is as follows:

$$U_{*,j,k} = X_{*,j,k} + W_{2,\text{ver}} \text{GELU}(W_{1,\text{ver}} \text{LN}(X_{*,j,k})), \quad \forall j = 1, \ldots, w', \forall k = 1, \ldots, c,$$

$$Y_{i,*,k} = U_{i,*,k} + W_{2,\text{hor}} \text{GELU}(W_{1,\text{hor}} \text{LN}(U_{i,*,k})), \quad \forall i = 1, \ldots, h', \forall k = 1, \ldots, c,$$

where $W_{1,\text{ver}} \in R^{h' \times h'}, W_{2,\text{ver}} \in R^{h' \times h'}, W_{1,\text{hor}} \in R^{w' \times w'},$ and $W_{2,\text{hor}} \in R^{w' \times w'}$ denote MLP weight matrices and $U, X,$ and $Y$ denote feature tensors.

3.3 Channel Raft

Let us assume that several groups of feature map channels have correlations originating from spatial properties. Under this assumption, some feature maps would have some patterns across vertical or horizontal directions. To capture
such spatial correlations, we integrate feature maps into the vertical and horizontal shuffle. As shown in Fig. 2, this can be carried out by arranging the feature maps in $h'r \times w'r$, which is reshaping the $h' \times w' \times c$ tensor into a $h'r \times w'r \times c'$ tensor with $c' = c/r^2$ channels. We then perform the vertical-mixing and the horizontal-mixing blocks for this new tensor. In this case, the layer normalization done in each mixing is for the original channel. We refer to this structure as channel raft. The combination of vertical- and horizontal-mixing blocks and the channel raft is called raft-token-mixing block in this paper. The pseudo-code for the raft-token-mixing block is given in Listing 1.1. The combination of raft-token-mixing block and the channel-mixing block is referred to as RaftMLP block.

```python
# b: size of mini-batch, h: height, w: width,
# c: channel, r: size of raft, o: c//r,
# e: expansion factor,
# x: input tensor of shape (h, w, c)

def __init__(self):
    self.lnv = nn.LayerNorm(c)
    self.lnh = nn.LayerNorm(c)
    self.fnv1 = nn.Linear(r * h, r * h * e)
    self.fnv2 = nn.Linear(r * h * e, r * h)
    self.fnh1 = nn.Linear(r * w, r * w * e)
    self.fnh2 = nn.Linear(r * w * e, r * w)

def forward(self, x):
    y = self.lnv(x)
    y = rearrange(y, 'b (h w) (r o) -> b (o w) (r h)')
    y = self.fcv1(y)
    y = F.gelu(y)
    y = self.fcv2(y)
    y = rearrange(y, 'b (o w) (r h) -> b (h w) (r o)')
    y = x + y
```

Fig. 3. A visualization of the concept of multi-scale-embedding.
3.4 Multi-scale Patch Embedding

The majority of both Transformer-based models and MLP-based models are based on patch embedding. We propose an extension of this method named multi-scale patch embedding, which is a patch embedding method that better represents the layered structure of an image. The main idea of the proposed method is twofold. The first is to cut out patches in such a way that the regions overlap. The second is to concatenate the channels of multiple-size patches and then project them by a linear embedding layer. The outline of the method is shown in Fig. 3, and the details are explained below. First, let \( r \) be an arbitrary even number. The method performs zero-padding of \((2^m - 1)r/2\) width on the top, bottom, left, and right sides then cut out the patch with \(2^mr\) on one side and \(r\) stride. In the case of \( m = 0 \), the patch is cut out the same way as in conventional patch embedding. After this patch embedding, the height \( h' = h/p \) and width \( w' = w/p \) of the tensor is the same, and the output channel is \(2^2m r^2\).

Here, we describe the implementation of multi-scale patch embedding. Multi-scale patch embedding is a generalization of conventional patch embedding, but it is also slightly different from convolution. However, by injecting a layered structure into the embedding, it can be said to incorporate the inductive bias for images. As the \( m \) increases, the computational complexity increases, so we should be careful to decide which \( m \) patch cutout to use. Our method is similar to convolutional embedding, but it slightly differs because it uses a linear layer projection after concatenating. See the appendix for code details.

3.5 Hierarchical Design

In the proposed method, hierarchical design is introduced. Our architecture used a four-level hierarchical structure with channel raft and multi-scale patch embedding to effectively reduce the number of parameters and improve the accuracy. The hierarchical design is shown in Fig. 1. In this architecture, the number of levels is \( L = 4 \), and at level \( l \), after extracting a feature map of \( h/2^{l+1} \times w/2^{l+1} \times c_l \) by multi-scale patch embedding, the RaftMLP block is repeated \( k_l \) times. The embedding is done using multi-scale patch embedding, but for \( l = 1, 2, 3 \), the feature maps for \( m = 0, 1 \) are concatenated, and for \( l = 4 \), conventional patch embedding is used. We prepared a hierarchical RaftMLP model with multiple scales. By settling \( c'_l \), the number of channels for the level \( l \), and \( N_l \), the number of
RaftMLP blocks for the level, we developed models for three scales: RaftMLP-S, RaftMLP-M, and RaftMLP-L. The common settings for all three models are vertical dilation expansion factor $e_{\text{ver}} = 2$, horizontal dilation expansion factor $e_{\text{hor}} = 2$, channel dilation expansion factor $e_{\text{can}} = 4$, and channel raft size $r = 2$. For patch embedding at each level, multi-scale patch embedding is utilized, but for the $l = 1, 2, 3$ level, patch cutting is performed for $m = 0, 1$ and then concatenated. For the final level, conventional patch embedding to reduce parameters and computational complexity is utilized. For the output head, a classifier with linear layers and softmax is applied after global average pooling. Refer to the appendix for other settings. Our experiments show that the performance of image classification improves as the scale is increased.

3.6 Impact of Channel Raft on Computational Costs

We will discuss the computational complexity of channel raft, ignoring normalization and activation functions. Here, let $h'$ denote the height of the patch placement, $w'$ the width of the patch placement, and $e$ the expansion factor.

**Number of parameters** The MLPs parameter for a conventional token-mixing block is

$$h'w'(2eh'w' + e + 1).$$

(4)

In contrast, the parameter used for a vertical-mixing block is

$$h'r(2eh'r + e + 1),$$

(5)

and the parameter used for a horizontal-mixing block is

$$w'r(2ew'r + e + 1).$$

(6)

In other words, the total number of parameters required for a raft-token-mixing block is

$$h'r(2eh'r + e + 1) + w'r(2ew'r + e + 1).$$

(7)

This means that if we assume $h' = w'$ and ignore $e + 1$, the parameters required for a conventional token-mixing block in the proposed method are $2(r/h')^2$ times for a conventional token-mixing. In short, if we choose $r$ to satisfy $r < h'/\sqrt{2}$, the memory cost can be reduced.

**Number of multiply-accumulate** If we ignore the bias term, the MLPs used for a conventional token-mixing block require $e(h'w')^4$ multiply-accumulates. By contrast, a raft-token-mixing block requires only $er^4(h'^4 + w'^4)$. Assuming $h' = w'$, a raft-token-mixing requires only multiply-accumulate of $2r^4/h'^4$ ratio to conventional token-mixing block. To put it plainly, if $r$ is chosen so that $r < h'/2^{1/4}$, then multiply-accumulation has an advantage over a conventional token-mixing block.
Table 1. Accuracy of the models to be compared with the accuracy of the models derived from the experiments with ImageNet-1k. The throughput measurement infers 16 images per batch using a single V100 GPU. Performance have been not measured for S²-MLP-deep because the code is not publicly available.

| Backbone Model | #params × FLOPs (M) | Top-1 Acc. (%) | Top-5 Acc. (%) | Throughtput (image/s) |
|----------------|---------------------|-----------------|----------------|----------------------|
| **Low-resource Models** (≤50P) | | | | |
| CNN | ResNet-18 [15] | 11.7 | 1.8 | 69.8 | 89.1 | 4190 |
| MobileNetV3 [18] | 5.4 | 0.2 | 71.7 | - | 1896 |
| EfficientNet-B0 [17] | 5.3 | 0.4 | 77.1 | - | 1275 |
| Local MLP | CycleMLP-B1 [5] | 15.2 | 2.1 | 78.9 | - | 364 |
| ConvMLP-S [25] | 9.0 | 2.4 | 76.8 | - | 1929 |
| **Middle-Low-resource Models** (50P < #params × FLOPs < 150P) | | | | |
| CNN | ResNet-50 [15] | 25.6 | 3.8 | 76.3 | 92.2 | 1652 |
| EfficientNet-B4 [37] | 19.0 | 4.2 | 82.6 | 96.3 | 465 |
| Transformer | DeiT-S [42] | 22.1 | 4.6 | 81.2 | - | 1583 |
| TNT-ViT-14 [52] | 21.5 | 6.1 | 81.7 | - | 849 |
| AS-MLP-Ti [26] | 28.0 | 4.4 | 81.3 | - | 805 |
| CycleMLP-B3 [5] | 17.4 | 3.9 | 79.0 | - | 1410 |
| SST-MLP-deep [50] | 51.0 | 9.7 | 80.7 | 95.4 | - |
| ConvMLP-L [25] | 42.7 | 9.9 | 80.2 | - | 928 |
| **Middle-High-resource Models** (150P < #params × FLOPs < 500P) | | | | |
| CNN | ResNet-152 [15] | 60.0 | 11.0 | 77.8 | 93.8 | 548 |
| EfficientNet-B0 [37] | 30.0 | 9.9 | 83.7 | - | 249 |
| EfficientNetV2-S [39] | 23.0 | 8.8 | 83.9 | - | 549 |
| PVC-S [46] | 44.2 | 6.7 | 81.2 | - | 742 |
| Transformer | Swin-S [29] | 50.0 | 8.7 | 83.0 | - | 559 |
| Nest-S [55] | 38.0 | 10.4 | 81.3 | - | 521 |
| SST-MLP-deep [50] | 51.0 | 9.7 | 80.7 | 95.4 | - |
| ConvMLP-B3 [5] | 38.0 | 6.9 | 82.4 | - | 364 |
| AS-MLP-S [26] | 50.0 | 8.5 | 83.1 | - | 442 |
| ConvMLP-L [25] | 42.7 | 9.9 | 80.2 | - | 928 |
| **High-resource Models** (Models with #params × FLOPs > 500P) | | | | |
| Transformer | ViT-B/16 [11] | 86.6 | 55.5 | 77.9 | - | 762 |
| DeiT-B [42] | 86.6 | 17.6 | 81.8 | - | 789 |
| CaiT-S36 [43] | 68.2 | 13.9 | 83.3 | - | 335 |
| Nest-B [55] | 68.0 | 17.9 | 81.8 | - | 412 |
| **Global MLP** | RaftMLP-S (ours) | 21.4 | 4.3 | 78.8 | 93.0 | 873 |
| RaftMLP-B (ours) | 36.2 | 6.5 | 79.4 | 94.3 | 658 |
| RaftMLP-L (ours) | 36.2 | 6.5 | 79.4 | 94.3 | 658 |
4 Experimental Evaluation

In this section, we exhibit experiments for image classification with RaftMLP. In the principal part of this experiment, we utilize the Imagenet-1k dataset [8] to train three types of RaftMLP and compare them with MLP-based models and Transformers-based models mainly. We also carry out an ablation study to demonstrate the effectiveness of our proposed method, and as a downstream task, we evaluate transfer learning of RaftMLP for image classification. Besides, We conduct experiments employing RaftMLP as the backbone for object detection and semantic segmentation.

4.1 ImageNet-1k

To evaluate the training results of our proposed classification models, RaftMLP-S, RaftMLP-M and RaftMLP-L, we train them on ImagNet-1k dataset [8]. This dataset consists of about 1.2 million training images and about 50,000 validation images assigned 1000 category labels. We also describe how the training is set up below. We employ AdamW [30] with weight decay 0.05 and learning schedule: maximum learning rate \( \frac{\text{batchsize}}{512} \times 5 \times 10^{-4} \), linear warmup on first 5 epochs, and after cosine decay to \( 10^{-5} \) on the following 300 epochs to train our models. Moreover, we adopt some augmentations and regularizations; random horizontal flip, color jitter, Mixup [54] with \( \alpha = 0.8 \), CutMix [53] with \( \alpha = 1.0 \), Cutout [9] of rate 0.25, Rand-Augment [7], stochastic depth [20] of rate 0.1, and label smoothing [36] 0.1. These settings refer to the training strategy of DeiT [42]. The other settings are changed for each experiment. Additionally, all training in this experiment is performed on a Linux machine with 8 RTX Quadro 8000 cards. The results of trained models are showed in Table 1. In Fig. 4, we compare our method with other global MLP-based models in terms of accuracy against the number of parameters and computational complexity. Fig. 4 reveals that RaftMLP-S is a cost-effective method.

4.2 Ablation Study

In order to verify the effectiveness of the two methods we propose, we carry out ablation studies. The setup for these experiments is the same as in Subsection 4.1.

Channel Raft (CR) We have carried out experiments to verify the effectiveness of channel rafts. Table 2 compares and verifies MLP-Mixer and MLP-Mixer with the token mixing block replaced by channel rafts. Although we have prepared architectures for \( r = 1, 2, 4 \) cases, \( r = 1 \) case has no raft structure but is just a conventional token-mixing block vertically and horizontally separated. Table 2 has shown that channel rafts effectively improve accuracy and costless channel raft structure such as \( r = 2 \) is more efficient for training than increasing \( r \).
**Fig. 4.** Accuracy per parameter and accuracy per FLOPs for the family of global MLP-based models

**Table 2.** An ablation experiment of channel raft. Note that Mixer-B/16 is experimented with our implementation

| Model                      | r | #Mparams | GFLOPs | Top-1 Acc. |
|----------------------------|---|----------|--------|------------|
| Mixer-B/16                 |   | 59.9     | 12.6   | 74.3%      |
| Mixer-B/16 with CR         | 1 | 58.1     | 11.4   | 77.0%      |
|                            | 2 | 58.2     | 11.6   | 78.3%      |
|                            | 4 | 58.4     | 12.0   | 78.0%      |

**Multi-scale Patch Embedding (MSPE)** RaftMLP-M is composed of three multiscale patch embeddings and a conventional patch embedding. To evaluate the effect of multi-scale patch embedding, we compared RaftMLP-M with the model with multi-scale patch embeddings replaced by conventional patch embeddings in RaftMLP-M. The result is shown on Table 3. As a result of comparing the models with and without multi-scale patch embedding, RaftMLP-M with multi-scale patch embedding improves the accuracy by 0.7% compared to the model without multi-scale patch embedding.

**Table 3.** An ablation experiment of multi-scale patch embedding

| Model                | #Mparams | GFLOPs | Top-1 Acc. |
|----------------------|----------|--------|------------|
| RaftMLP-M            | 21.4     | 4.3    | 78.8%      |
| RaftMLP-M without MSPE | 20.0   | 3.8    | 78.1%      |
4.3 Transfer Learning

The study of transfer learning is conducted on CIFAR-10/CIFAR-100 [23], Oxford 102 Flowers [32], Stanford Cars [22] and iNaturalist [44] to evaluate the transfer capabilities of RaftMLP pre-trained on ImageNet-1k [8]. The fine-tuning experiments adopt batch size 256, weight decay $10^{-4}$ and learning schedule: maximum learning rate $10^{-4}$, linear warmup on first 10 epochs, and after cosine decay to $10^{-5}$ on the following 40 epochs. We also do not use stochastic depth [20] and Cutout [9] in this experiment. The rest of the settings are equivalent to Subsection 4.1. In our experiments, we also resize all images to the exact resolution 224 × 224 as ImageNet-1k. The experiment is shown in Table 4. We achieve that RaftMLP-L is more accurate than Mixer-B/16 in all datasets.

| Dataset            | Mixer-B/16 | RaftMLP-S | RaftMLP-M | RaftMLP-L |
|--------------------|-----------|-----------|-----------|-----------|
| CIFAR-10           | 97.7%     | 97.4%     | 97.7%     | 98.1%     |
| CIFAR-100          | 85.0%     | 85.1%     | 86.8%     | 86.8%     |
| Oxford 102 Flowers | 97.8%     | 97.1%     | 97.9%     | 98.4%     |
| Stanford Cars      | 84.3%     | 84.7%     | 87.6%     | 89.0%     |
| iNaturalist18      | 55.6%     | 56.7%     | 61.7%     | 62.9%     |
| iNaturalist19      | 64.1%     | 65.4%     | 69.2%     | 70.1%     |

5 Discussion

The above experimental results show that even an architecture that does not use convolution but has a simple inductive bias for images like vertical and horizontal decomposition can achieve performance competing with Transformers. This is a candidate for minimal inductive biases to improve MLP-based models without convolution. Also, Our method does not require as much computational cost as Transformer. In addition, the computational cost is as expensive as or less than that of CNN. The main reason for the reduced computational cost is that it does not require self-attention. The fact that only simple operations such as MLP are needed without self-attention nor convolution means that MLP-based models will be widely used in applied fields since they do not require special software or hardware carefully designed to reduce computational weight. Furthermore, the raft-token-mixing block has the lead over the token-mixing block of MLP-Mixer in terms of computational complexity when the number of patches is large. As we described in Section 3, substituting the token-mixing block as the raft-token-mixing block reduces parameters from the square of the patches to several times patches. In other words, the more the resolution of images is, the more dramatically parameters are reduced with RaftMLP. The hierarchical design adopted in
this paper contributes to the reduction of parameters and computational complexity. Since multi-scale embedding leads to better performance with less cost, our proposal will make it realistic to compose architectures that do not depend on convolution. Meanwhile, the experimental results in the appendix suggest that the proposed model is not very effective for some downstream tasks. As shown in the appendix, the feature map of global MLP-based models differs from the feature map of CNNs in that it is visualized as a different appearance from the input image. Such feature maps are not expected to work entirely in convolution-based architectures such as RetinaNet [27], Mask R-CNN [14], and Semantic FPN [21]. Global MLP-based models will require their specialized frameworks for object detection, instance segmentation, and semantic segmentation.

6 Conclusion

In conclusion, the result has demonstrated that the introduction of the raft-token-mixing block improves accuracy when trained on the ImageNet-1K dataset [8], as compared to plain MLP-Mixer [40]. Although the raft-token-mixing decreases the number of parameters and FLOPs only lightly compared to MLP-Mixer [40], it contributes to the improvement in accuracy in return. We conclude that adding a non-convolutional and non-self-attentional inductive bias to the token-mixing block of MLP-Mixer can improve the accuracy of the model. In addition, due to the introduction of hierarchical structures and multi-scale patch embedding, RaftMLP-S with lower computational complexity and number of parameters have achieved accuracy comparable to the state-of-the-art global MLP-based model with similar computational complexity and number of parameters. We have explicated that it is more cost-effective than the Transformer-based models and well-known CNNs.

However, global MLP-based models have not yet fully explored their potential. Inducing other utilitarian inductive biases, e.g., parallel invariance, may improve the accuracy of global MLP-based models. Further insight into these aspects is left to future work.

Acknowledgements We thank the people who support us, belonging to Graduate School of Artificial Intelligence and Science, Rikkyo University.

References

1. Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., Schmid, C.: ViViT: A video vision transformer. In: ICCV (2021)
2. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. In: NeurIPS (2016)
3. Bello, I., Zoph, B., Vaswani, A., Shlens, J., Le, Q.V.: Attention augmented convolutional networks. In: ICCV. pp. 3286–3295 (2019)
4. Chen, C.F., Fan, Q., Panda, R.: CrossViT: Cross-attention multi-scale vision transformer for image classification. In: ICCV (2021)
5. Chen, S., Xie, E., Ge, C., Liang, D., Luo, P.: CycleMLP: A MLP-like architecture for dense prediction. arXiv preprint arXiv:2107.10224 (2021)
6. Cordonnier, J.B., Loukas, A., Jaggi, M.: On the relationship between self-attention and convolutional layers. In: ICLR (2019)
7. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: RandAugment: Practical automated data augmentation with a reduced search space. In: CVPR Workshops. pp. 702–703 (2020)
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR. pp. 248–255 (2009)
9. DeVries, T., Taylor, G.W.: Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552 (2017)
10. Ding, M., Yang, Z., Hong, W., Zheng, W., Zhou, C., Yin, D., Lin, J., Zou, X., Shao, Z., Yang, H., et al.: Cogview: Mastering text-to-image generation via transformers. In: NeurIPS (2021)
11. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2021)
12. El-Nouby, A., Touvron, H., Caron, M., Bojanowski, P., Douze, M., Joulin, A., Laptev, I., Neverova, N., Synnaeve, G., Verbeek, J., et al.: Xcit: Cross-covariance image transformers. arXiv preprint arXiv:2106.09681 (2021)
13. Han, K., Xiao, A., Wu, E., Guo, J., Xu, C., Wang, Y.: Transformer in transformer. In: NeurIPS (2021)
14. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: ICCV. pp. 2961–2969 (2017)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
16. Hendrycks, D., Gimpel, K.: Gaussian error linear units (GELUs). arXiv preprint arXiv:1606.08415 (2016)
17. Hou, Q., Jiang, Z., Yuan, L., Cheng, M.M., Yan, S., Feng, J.: Vision Permutator: A permutable MLP-like architecture for visual recognition. arXiv preprint arXiv:2106.12368 (2021)
18. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Others: Searching for MobileNetV3. In: ICCV. pp. 1314–1324 (2019)
19. Hu, J., Shen, L., Albanie, S., Sun, G., Vedaldi, A.: Gather-Excite: Exploiting feature context in convolutional neural networks. In: NeurIPS (2018)
20. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: ECCV. pp. 646–661 (2016)
21. Kirillov, A., Girshick, R., He, K., Dollár, P.: Panoptic feature pyramid networks. In: CVPR. pp. 6399–6408 (2019)
22. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3D object representations for fine-grained categorization. In: ICCV Workshops. pp. 554–561 (2013)
23. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Tech. rep., University of Toronto (2009)
24. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NeurIPS. vol. 25, pp. 1097–1105 (2012)
25. Li, J., Hassani, A., Walton, S., Shi, H.: ConvMLP: Hierarchical convolutional MLPs for vision. arXiv preprint arXiv:2109.04454 (2021)
26. Lian, D., Yu, Z., Sun, X., Gao, S.: AS-MLP: An axial shifted MLP architecture for vision. arXiv preprint arXiv:2107.08391 (2021)
27. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object
detection. In: ICCV. pp. 2980–2988 (2017)
28. Liu, H., Dai, Z., So, D.R., Le, Q.V.: Pay attention to MLPs. arXiv preprint
arXiv:2105.08050 (2021)
29. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin trans-
former: Hierarchical vision transformer using shifted windows. In: ICCV (2021)
30. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2019)
31. Melas-Kyrizaki, L.: Do you even need attention? a stack of feed-forward layers does
surprisingly well on imagenet. arXiv preprint arXiv:2105.02723 (2021)
32. Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number
of classes. In: ICGVIP. pp. 722–729 (2008)
33. Ramachandran, P., Parmar, N., Vaswani, A., Bello, I., Levskaya, A., Shlens, J.: 
Stand-alone self-attention in vision models. In: NeurIPS (2019)
34. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale
image recognition. In: ICLR (2015)
35. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., 
Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: CVPR. pp. 1–
9 (2015)
36. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the incep-
tion architecture for computer vision. In: CVPR. pp. 2818–2826 (2016)
37. Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural
networks. In: ICML. pp. 6105–6114 (2019)
38. Tan, M., Le, Q.V.: EfficientNetV2: Smaller models and faster training. In: ICML
(2021)
39. Tang, C., Zhao, Y., Wang, G., Luo, C., Xie, W., Zeng, W.: Sparse mlp for im-
age recognition: Is self-attention really necessary? arXiv preprint arXiv:2109.05422
(2021)
40. Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., 
Yung, J., Keysers, D., Uszkoreit, J., Lucic, M., et al.: Mlp-mixer: An all-mlp archi-
tecture for vision. arXiv preprint arXiv:2105.01601 (2021)
41. Touvron, H., Bojanowski, P., Caron, M., Cord, M., El-Nouby, A., Grave, E., Joulin,
A., Synnaeve, G., Verbeek, J., Jégou, H.: Resmlp: Feedforward networks for image
classification with data-efficient training. arXiv preprint arXiv:2105.03404 (2021)
42. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training
data-efficient image transformers & distillation through attention. In: ICML (2021)
43. Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., Jégou, H.: Going deeper
with image transformers. In: ICCV. pp. 32–42 (2021)
44. Van Horn, G., Mac Aodha, O., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam,
H., Perona, P., Belongie, S.: The iNaturalist species classification and detection
dataset. In: CVPR. pp. 8769–8778 (2018)
45. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser,
L., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017)
46. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao,
L.: Pyramid vision transformer: A versatile backbone for dense prediction without
convolutions. In: ICCV (2021)
47. Wang, W., Yao, L., Chen, L., Cai, D., He, X., Liu, W.: CrossFormer: A versatile
vision transformer based on cross-scale attention. arXiv preprint arXiv:2108.00154
(2021)
48. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: CVPR.
pp. 7794–7803 (2018)
49. Yu, T., Li, X., Cai, Y., Sun, M., Li, P.: Rethinking token-mixing mlp for mlp-based vision backbone. arXiv preprint arXiv:2106.14882 (2021)
50. Yu, T., Li, X., Cai, Y., Sun, M., Li, P.: S$^2$-MLP: Spatial-shift MLP architecture for vision. arXiv preprint arXiv:2106.07477 (2021)
51. Yu, T., Li, X., Cai, Y., Sun, M., Li, P.: S$^2$-mlpv2: Improved spatial-shift mlp architecture for vision. arXiv preprint arXiv:2108.01072 (2021)
52. Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Tay, F.E., Feng, J., Yan, S.: Tokens-to-Token ViT: Training vision transformers from scratch on ImageNet. In: ICCV. pp. 558–567 (2021)
53. Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.: Cutmix: Regularization strategy to train strong classifiers with localizable features. In: ICCV. pp. 6023–6032 (2019)
54. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: ICLR (2018)
55. Zhang, Z., Zhang, H., Zhao, L., Chen, T., Pfister, T.: Aggregating nested transformers. arXiv preprint arXiv:2105.12723 (2021)
56. Zhou, D., Kang, B., Jin, X., Yang, L., Lian, X., Jiang, Z., Hou, Q., Feng, J.: Deepvit: Towards deeper vision transformer. arXiv preprint arXiv:2103.11886 (2021)
Supplementary Material for "RaftMLP: How Much Can Be Done Without Attention and with Less Spatial Locality?"

Yuki Tatsunami¹,²[0000−0002−7889−8143] and Masato Taki¹[0000−0002−5375−7862]

¹ Rikkyo University, Tokyo, Japan
² AnyTech Co., Ltd., Tokyo, Japan

{y.tatsunami, taki_m}@rikkyo.ac.jp

A More pseudocode

This section describes the pseudo-code for the methods discussed in this paper. The pseudo-code for Multi-scale Patch Embedding is detailed in Listing 1.1.

```
# b: size of mini-batch, h: height, w: width,
# kernels: list of kernel sizes for unfold.
# e.g., [4, 8]
def __init__(self, in_channels, out_channels, kernels):
    mlp_in_channels = 0
    for k in kernels:
        mlp_in_channels += k ** 2
        mlp_in_channels *= in_channels
    self.embeddings = nn.ModuleList([nn.Sequential(*[
            nn.Unfold(kernel_size=k,
            stride=self.stride,
            padding=(k - self.stride) // 2),
            Rearrange("b c hw -> b hw c")
        ]) for k in kernels])
    self.fc = nn.Linear(mlp_in_channels, out_channels)

def forward(self, input):
    b, _, h, w = input.shape
    outputs = []
    for emb in self.embeddings:
        output = emb(input)
        outputs.append(output)
    return self.fc(torch.cat(outputs, dim=2))
```

Listing 1.1: Pseudocode of multi-scale patch embedding (Pytorch-like)
## B Downstream Task Application

In this section, we discuss the application of our model to downstream tasks. In order to apply our models to downstream tasks such as semantic segmentation, instance segmentation, and object detection, various resolutions need to be supported. Therefore, we insert bicubic interpolation before and after the raft-token-mixing block, as shown in Fig. 1. In the bicubic interpolation before the block, we convert the input to the resolution used for pre-training. The bicubic interpolation after the block restores the resolution before the first bicubic interpolation. Moreover, since the resolution of input images is not always divisible by the patch size, we apply bicubic interpolation to obtain the resolution before multi-scale embedding that is a factor of the patch size. This method can be applied to other global MLP-based models such as MLP-Mixer too.

![Fig. 1: Application of RaftMLP block utilizing bicubic interpolation](image)

### B.1 Object Detection

For the evaluation of object detection and instance segmentation, we compose a model in which the backbones of RetinaNet [?] and Mask R-CNN [?], which are both standard implementations on the object detection framework mmdetection [?], are replaced by RaftMLP and MLP-Mixer. For the dataset, we used MS COCO [?], which is one of the most popular benchmark datasets for object detection. The training setup is similar to ConvMLP [?], with AdamW as the optimizer, learning rate set to $10^{-4}$, weight decay set to $10^{-4}$, and 12 epochs of training with a batch size of 16. The results are compared with PureMLP [?], ResNet [?], and ConvMLP [?], and a summary is provided in Fig. 2. See Appendix B.4 for more details.

### B.2 Semantic Segmentation

We replace the backbone of Semantic FPN [?] implemented on mmsegmentation [?] with RaftMLP and MLP-Mixer and evaluate their performances on the segmentation task. We adopt AdamW as the optimizer with a learning rate of $2.0 \times 10^{-4}$ and a weight decay of $10^{-4}$. The learning schedule follows the polynomial decay
learning rate policy with a power of 0.9. We use the famous ADE20K dataset to train the model, with the input image randomly resized and cropped to a resolution of $512 \times 512$. The model had trained for 40000 iterations. The above settings follow ConvMLP [?]. A summary of the experimental results is shown in Fig. 2. See Appendix B.4 for more details.

![Fig. 2: The above compares the training results of RetinaNet and Mask R-CNN on MS COCO and Semantic FPN on ADE20K. We compare the results with ResNet, PureMLP, ConvMLP, Mixer, and RaftMLP as backbones in each case. RetinaNet uses AP for bounding boxes, Mask R-CNN AP for bounding boxes and segmentation, and Semantic FPN uses mIoU as their metric.](image-url)
B.3 Details of Architectures

RaftMLP The details of the architectures of RaftMLP-S, RaftMLP-M, and RaftMLP-L used in this paper are details in Table 1.

Object Detection, Instance Segmentation and Semantic Segmentation For the RetinaNet [?] and Mask R-CNN [?] and Semantic FPN [?] we used, we consulted the results of [?], which is the same setup for ResNet, PureMLP, and ConvMLP, which are our comparison. The backbones we have experimented with are RaftMLP and Mixer-B/16. All of the architectures we have arranged use Feature Pyramid Network [?]. Therefore, we must clearly state what feature pyramid was input to these architectures from the backbones RaftMLP and Mixer-B/16. RaftMLP utilizes the output immediately after the first multiscale patch embedding and the outputs of Level-2 to Level-4 as feature maps to be input to the detector and segmentor. Similarly, Mixer-B/16, along with RaftMLP, uses the output immediately after the patch embedding and the outputs of Block-4, Block-10, and Block-12 as before-mentioned feature maps.

B.4 Details of Quantitative Results

Object Detection and Instance Segmentation Table 2 contains the detailed results of the experiment for RetinaNet performed in Subsection B.1, Table 3 includes the detailed results of the experiment for Mask R-CNN worked in Subsection B.1. The results of RetinaNet are not doing as well overall as PureMLP, even with RaftMLP, which guarantees some spatial structure. In particular, it struggles to detect small objects. This result can be seen in B.6, where RaftMLP adds artifacts to the feature map, harming object detection.

Semantic Segmentation Table 4 contains the detailed results of the experiment performed in Subsection B.2.

B.5 Qualitative Results

Object Detection and Instance Segmentation Fig. 3a shows the ground truth for an sample of MS COCO validation dataset. Fig. 3b shows the inference result for RetinaNet with ResNet-50 as the backbone to be installed in mmdetection, and Fig. 3c, 3d, 3e, and 3f inference results for the four RetinaNets trained in Subsection B.1. Fig. 3g shows the inference result for Mask R-CNN with ResNet-50 as the backbone to be installed in mmdetection, and Fig. 3h, 3i, 3j, and 3k inference results for the four Mask R-CNNs trained in Subsection B.1. Despite the lack of precision, the figures reveal that results of Global MLP-based models for the object detection and instance segmentation tasks are satisfactory.

Semantic Segmentation Fig. 4a presents an image with ADE20k validation dataset overlaid with its ground truth. Fig. 4b shows the inference results of the model applying ResNet-50, which mmsegmentation provides, as the backbone of the Semantic FPN. Fig. 4c, 4d, 4e, and 4f show the inference results for the four models we trained in Subsection B.2 experiment.
Fig. 3: Qualitative results of object detection and instance segmentation

(a) Ground Truth
(b) RetinaNet|ResNet-50
(c) RetinaNet|Mixer-B/16
(d) RetinaNet|RaftMLP-S
(e) RetinaNet|RaftMLP-M
(f) RetinaNet|RaftMLP-L
Fig. 3: Qualitative results of object detection and instance segmentation
Fig. 4: Qualitative results of semantic segmentation

B.6 Visualization

We used an image with ImageNet to visualize and compare its feature map. The image used as input was the ferret image on the left of Fig. 5, which was input to pre-trained ResNet-50, Mixer-B/16, and RaftMLP-M. Some of the outputs of the intermediate layers are summarized on the right side of Fig. 5. For ResNet-50, we used the output of layers 1 through 4; for Mixer-B/16, we used the output of Blocks 2, 4, 10, and 12; for RaftMLP-M, we used the output of each Level. We have also included further intermediate layer outputs for the three models,
see Fig. 6, 7, 8, and 9 for ResNet-50, Fig. 10, 11, 12 and 13 for Mixer-B/16, and Fig. 14, 15, 16, and 17 for RaftMLP-M.

As mentioned in Section 5, the appearance of features in the middle layer of global MLP-based models is different from that of the convolutional base represented by ResNet. We believe this is why global MLP-based models do not perform well when selected as the backbone of existing architectures for object detection, instance segmentation, and semantic segmentation. The feature map of RaftMLP-M is different from that of ResNet in that the lower layers have feature maps that capture the features of the ferret. In contrast, the upper layers have feature maps with visible artifacts of vertical and horizontal lines. The feature maps of Mixer-B/16 do not capture the features of the ferret, and they are overall shuffled and have many similar feature maps. Tasks such as object detection, semantic segmentation, or even image generation will require innovations specific to global MLP-based models. The occurrence of artifacts might have a minor impact on classification, where global average pooling is used. However, for tasks such as segmentation and image generation, it becomes a severe problem. Hence, it will be necessary to design architectures and loss functions that do not emit this artifact. Or else, convolution-based methods such as RetinaNet, Mask R-CNN, and Semantic FPN may be insufficient to recover the whole shuffled information by global MLP-based models. To recover the global shuffled information by the global MLP-based model, global MLP-based models may lack a module that can capture the global relations, such as self-attention modules and token-mixing blocks.
Table 1: Specific settings on the model architectures of hierarchy RaftMLP in different scales. \( l \) denotes level and \( c'_l \) denotes the number of basic channels in RaftMLP for level \( l \).

| Model     | RaftMLP-S | RaftMLP-M | RaftMLP-L |
|-----------|-----------|-----------|-----------|
| \( l = 1 \) | RaftMLP \( \times 2 \) \( c'_1 = 64 \) | RaftMLP \( \times 2 \) \( c'_1 = 96 \) | RaftMLP \( \times 2 \) \( c'_1 = 128 \) |
| \( l = 2 \) | RaftMLP \( \times 2 \) \( c'_2 = 128 \) | RaftMLP \( \times 2 \) \( c'_2 = 192 \) | RaftMLP \( \times 2 \) \( c'_2 = 192 \) |
| \( l = 3 \) | RaftMLP \( \times 6 \) \( c'_3 = 256 \) | RaftMLP \( \times 6 \) \( c'_3 = 384 \) | RaftMLP \( \times 6 \) \( c'_3 = 512 \) |
| \( l = 4 \) | RaftMLP \( \times 2 \) \( c'_4 = 512 \) | RaftMLP \( \times 2 \) \( c'_4 = 768 \) | RaftMLP \( \times 2 \) \( c'_4 = 1024 \) |

Table 2: Comparison of RetinaNet metrics trained on MS COCO with each ResNet, PureMLP, ConvMLP, RaftMLP, and Mixer as the backbone.

| Backbone     | #MParams | \( AP^b \) | \( AP^{50}_b \) | \( AP^{75}_b \) | \( AP^S_b \) | \( AP^M_b \) | \( AP^L_b \) |
|--------------|----------|------------|----------------|----------------|------------|------------|------------|
| ResNet-18 [? ,?] | 21.3 | 31.8 | 49.6 | 33.6 | 16.3 | 34.3 | 43.2 |
| PureMLP-S [?] | 17.6 | 27.1 | 44.2 | 28.3 | 13.6 | 29.2 | 36.4 |
| ConvMLP-S [?] | 18.7 | 37.2 | 56.4 | 39.8 | 20.1 | 40.7 | 50.4 |
| RaftMLP-S | 19.6 | 17.7 | 33.3 | 16.5 | 4.5 | 14.1 | 32.4 |
| ResNet-50 [? ,?] | 37.7 | 36.3 | 55.3 | 38.6 | 19.3 | 40.0 | 48.8 |
| PureMLP-M [?] | 25.9 | 28.0 | 45.6 | 29.0 | 14.5 | 29.9 | 37.8 |
| ConvMLP-M [?] | 27.1 | 39.4 | 58.7 | 42.0 | 21.5 | 43.2 | 52.5 |
| RaftMLP-M | 27.1 | 19.3 | 36.3 | 17.8 | 5.2 | 15.9 | 35.1 |
| ResNet-101 [? ,?] | 56.7 | 38.5 | 57.8 | 41.2 | 21.4 | 42.6 | 51.1 |
| PureMLP-L [?] | 50.1 | 28.8 | 46.8 | 29.9 | 15.0 | 31.0 | 38.4 |
| ConvMLP-L [?] | 52.9 | 40.2 | 59.3 | 43.3 | 23.5 | 43.8 | 53.3 |
| RaftMLP-L | 52.9 | 19.5 | 36.8 | 18.1 | 5.0 | 16.1 | 35.4 |
| Mixer-B/16 [?] | 70.3 | 10.7 | 20.0 | 10.1 | 0.1 | 6.7 | 25.8 |
Table 3: Comparison of Mask R-CNN metrics trained on MS COCO with each ResNet, PureMLP, ConvMLP, RaftMLP and Mixer as the backbone.

| Backbone    | #MParams | $AP^{b}$ | $AP^{50}_{b}$ | $AP^{75}_{b}$ | $AP^{m}$ | $AP^{50}_{m}$ | $AP^{75}_{m}$ |
|-------------|-----------|----------|----------------|---------------|----------|---------------|---------------|
| ResNet-18 [?,?] | 31.2      | 34.0     | 54.0           | 36.7          | 31.2     | 51.0          | 32.7          |
| PureMLP-S [?]  | 27.5      | 25.1     | 45.1           | 25.1          | 25.0     | 42.8          | 26.0          |
| ConvMLP-S [?]  | 28.7      | 38.4     | 59.8           | 41.8          | 35.7     | 56.7          | 38.2          |
| RaftMLP-S      | 29.5      | 21.8     | 40.2           | 21.0          | 19.7     | 36.5          | 19.1          |
| ResNet-50 [?,?] | 44.2      | 38.0     | 58.6           | 41.4          | 34.4     | 55.1          | 36.7          |
| PureMLP-M [?]  | 35.8      | 25.8     | 46.1           | 25.8          | 25.6     | 43.5          | 26.5          |
| ConvMLP-M [?]  | 37.1      | 40.6     | 61.7           | 44.5          | 37.2     | 58.8          | 39.8          |
| RaftMLP-M      | 40.9      | 23.4     | 42.5           | 22.7          | 21.1     | 38.8          | 20.8          |
| ResNet-101 [?,?] | 63.2      | 40.4     | 61.1           | 44.2          | 36.4     | 57.7          | 38.8          |
| PureMLP-L [?]  | 59.5      | 26.5     | 45.0           | 27.4          | 26.7     | 47.5          | 26.8          |
| ConvMLP-L [?]  | 62.2      | 41.7     | 62.8           | 45.5          | 38.2     | 59.9          | 41.1          |
| RaftMLP-L      | 55.5      | 24.2     | 43.9           | 23.7          | 21.6     | 39.7          | 23.7          |
| Mixer-B/16 [?] | 79.8      | 11.9     | 22.8           | 11.2          | 9.5      | 19.1          | 8.5           |

Table 4: Comparison of Semantic FPN metrics trained on MS COCO with each ResNet, PureMLP, ConvMLP, RaftMLP and Mixer as the backbone.

| Backbone    | #MParams | mIoU |
|-------------|-----------|------|
| ResNet-18 [?,?] | 15.5       | 32.9 |
| PureMLP-S [?]  | 11.6       | 23.9 |
| ConvMLP-S [?]  | 12.8       | 35.8 |
| RaftMLP-S      | 13.6       | 30.7 |
| ResNet-50 [?,?] | 28.5       | 36.7 |
| PureMLP-M [?]  | 19.9       | 25.2 |
| ConvMLP-M [?]  | 21.1       | 38.6 |
| RaftMLP-M      | 25.0       | 32.3 |
| ResNet-101 [?,?] | 47.5       | 38.8 |
| PureMLP-L [?]  | 43.6       | 26.3 |
| ConvMLP-L [?]  | 46.3       | 40.0 |
| RaftMLP-L      | 39.6       | 33.3 |
| Mixer-B/16 [?] | 63.9       | 28.1 |
Fig. 6: Part of the feature maps output from Layer-1 of ResNet-50 with the ferret images as input
Fig. 7: All the feature maps output from Layer-2 of ResNet-50 with the ferret images as input.
Fig. 8: All the feature maps output from Layer-3 of ResNet-50 with the ferret images as input.
Fig. 9: All the feature maps output from Layer-4 of ResNet-50 with the ferret images as input

Fig. 10: All the feature maps output from Block-2 of Mixer-B/16 with the ferret images as input
Fig. 11: All the feature maps output from Block-4 of Mixer-B/16 with the ferret images as input.
Fig. 12: All the feature maps output from Block-10 of Mixer-B/16 with the ferret images as input.
Fig. 13: All the feature maps output from Block-12 of Mixer-B/16 with the ferret images as input
Fig. 14: All the feature maps output from Level-1 of RaftMLP-M with the ferret images as input
Fig. 15: All the feature maps output from Level-2 of RaftMLP-M with the ferret images as input.

Fig. 16: All the feature maps output from Level-3 of RaftMLP-M with the ferret images as input.
Fig. 17: All the feature maps output from Level-4 of RaftMLP-M with the ferret images as input