S^2-MLP: Spatial-Shift MLP Architecture for Vision

Tan Yu, Xu Li, Yunfeng Cai, Mingming Sun, Ping Li
Cognitive Computing Lab
Baidu Research
10900 NE 8th St. Bellevue, Washington 98004, USA
No.10 Xibeiwang East Road, Beijing 100193, China
{tanyu01,lixu13,caiyunfeng,sunmingming01,liping11}@baidu.com

Abstract

Recently, visual Transformer (ViT) and its following works abandon the convolution and exploit the self-attention operation, attaining a comparable or even higher accuracy than CNN. More recently, MLP-mixer abandons both the convolution and the self-attention operation, proposing an architecture containing only MLP layers. To achieve cross-patch communications, it devises an additional token-mixing MLP besides the channel-mixing MLP. It achieves promising results when training on an extremely large-scale dataset such as JFT-300M. But it cannot achieve as outstanding performance as its CNN and ViT counterparts when training on medium-scale datasets such as ImageNet-1K. The performance drop of MLP-mixer motivates us to rethink the token-mixing MLP. We discover that token-mixing operation in MLP-mixer is a variant of depthwise convolution with a global reception field and spatial-specific configuration. In this paper, we propose a novel pure MLP architecture, spatial-shift MLP (S^2-MLP). Different from MLP-mixer, our S^2-MLP only contains channel-mixing MLP. We devise a spatial-shift operation for achieving the communication between patches. It has a local reception field and is spatial-agnostic. Meanwhile, it is parameter-free and efficient for computation. The proposed S^2-MLP attains higher recognition accuracy than MLP-mixer when training on ImageNet-1K dataset. Meanwhile, S^2-MLP accomplishes as excellent performance as ViT on ImageNet-1K dataset with considerably simpler architecture and fewer FLOPs and parameters.

1. Introduction

In the past years, convolutional neural networks (CNN) [23, 14] have achieved great success in computer vision. Recently, inspired by the triumph of Transformer [41] in natural language processing, vision Transformer (ViT) [9] is proposed. It replaces the convolution operation in CNN with the self-attention operation used in Transformer to model the visual relations between local patches in different spatial locations of the image. ViT and its following works [38, 46, 42, 28, 13, 44, 39] have achieved comparable or even better performance than CNNs. Compared with CNN demanding a meticulous design for convolution kernel, ViT stacks several standard Transformer blocks, taking less hand-crafted manipulation and reducing the inductive biases.

More recently, MLP-mixer [36] proposes a simpler alternative based entirely on multi-layer perceptrons (MLP) to further reduce the inductive biases. The basic block in MLP-mixer consists of two components: channel-mixing MLP and token-mixing MLP. The channel-mixing MLP projects the feature map along the channel dimension and thus conducts the communications between different channels. In parallel, the token-mixing dimension projects the feature map along the spatial dimension and exploits the communications between spatial locations. When training on the ultra large-scale dataset such as JFT-300M [33], MLP-mixer attains promising recognition accuracy. But there is still an accuracy gap between MLP-mixer and ViT when training on medium-scale datasets including ImageNet-1K and ImageNet21K [8]. Specifically, Mixer-Base-16 [36] achieves only a 76.44% top-1 accuracy on ImageNet-1K, whereas ViT-Base-16 [9] achieves a 79.67% top-1 accuracy.

The unsatisfactory performance of MLP-mixer on ImageNet-1K motivates us to rethink the mixing-token MLP. Given N patch features in the matrix form, $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]$, the token-mixing MLP conducts $\mathbf{XW}$ where $\mathbf{W} \in \mathbb{R}^{N \times M}$ is the weight matrix. It is easy to observe that each column of $\mathbf{XW}$, the output of the token-mixing MLP, is a weighted summation of patch features (columns in $\mathbf{X}$). The weights in summation can be regarded as the attention in Transformer. But the self-attention in Transformer is data-dependent, whereas the weights for summation in token-mixing MLP is agnostic to the input. To some extent, the weighted summation is more like depthwise convolution [5, 19, 20]. But the depthwise convolution only has a local reception field. In contrast, token-mixing MLP has

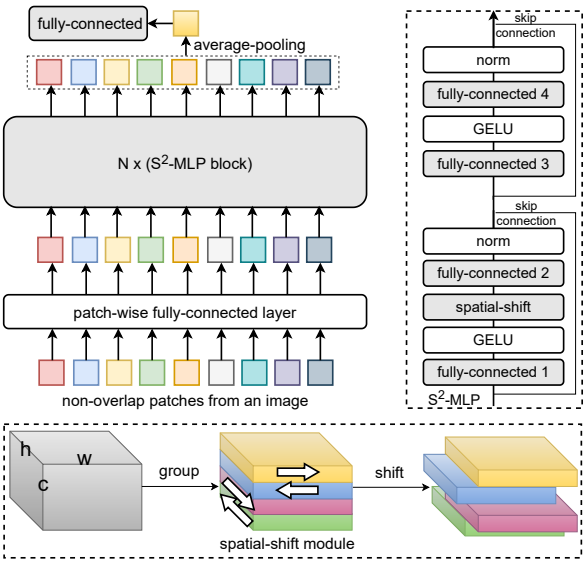

Figure 1. The architecture of the proposed spatial-shift multi-layer perceptions (S^2-MLP) model. Non-overlap patches cropped from an image are the input of the model. They go through a stack of S^2-MLP blocks which are further aggregated into a single feature vector through global average pooling. After that, the feature vector is fed into a fully-connected layer for predicting the label. An S^2-MLP block contains four fully-connected layers, two GELU layers [15], two layer normalization [1], two skip connections [14], and a spatial-shift module. The proposed spatial-shift module groups c channels into several groups. Then it shifts different groups of channels in different directions.

a global reception field. Besides, the depthwise convolution kernel is shared among different locations, whereas the weights for summation in token-mixing MLP are different for different locations. Without the limitation of the local reception field and the spatial-agnostic constraint, the token-mixing MLP is more flexible and has a stronger fitting capability. But the freedom from breaking a chain is accompanied by losing the spatially-invariant property.

In this work, we propose a spatial-shift MLP (S^2-MLP) architecture, a conceptually simple architecture containing only channel-mixing MLPs. To conduct communication between spatial locations, we adopt a spatial shift operation, which is parameter-free and efficient for computation. Meanwhile, the spatial shift is spatial-agnostic and meanwhile maintains a local reception field. Figure 1 illustrates the architecture of the proposed S^2-MLP. It crops an image into $w \times h$ non-overlap patches. For each patch, it obtains the path embedding through a fully-connected layer. The wh patches further go through a stack of S^2-MLP blocks for feature extraction. Each S^2-MLP block contains four fully-connected layers. The fully-connected layer used in each S^2-MLP block serves as the same function as the channel-mixing MLP used in MLP-mixer. But our S^2-MLP does not need token-mixing MLP. Instead, the communications between different spatial locations are achieved through the proposed spatial-shift module. It is parameter-free and simply shifts channels from a patch to its adjoining patches. Despite that the spatial-shift module only supports communications between adjacent patches, stacking a series of S^2-MLP blocks makes the long-range communications feasible.

The proposed S^2-MLP is frustratingly simple and elegant in architecture. It attains considerably higher recognition accuracy than MLP-mixer on ImageNet1K dataset with a comparable scale of parameters and FLOPs. Meanwhile, it achieves a comparable recognition accuracy with respect to ViT on ImageNet1K dataset with a considerably simpler structure, fewer parameters and FLOPs.

2. Related Work

Transformer-based vision models. Visual Transformer (ViT) [9] is the first work to build a purely Transformer-based vision backbone. Through training on an extremely large-scale dataset, JFT-300M [33], it has achieved promising results compared with *de facto* vision backbone, convolutional neural network. DeiT [38] adopts the advanced training and augmentation strategy and achieves excellent performance when training on ImageNet-1K only. Recently, several works further improve the performance of visual Transformer from multiple perspectives. For instance, PVT [42] uses a progressive shrinking pyramid to reduce computations of large feature maps. T2T [46] progressively tokenizes the image to model the local structure information of the image. TNT [13] constructs another Transformer within the outer-level Transformer to model the local patch. CPVT [7] proposes a conditional positional encoding to effectively encode the spatial locations of patches. Visual Longformer [47] adopts the global tokens to boost efficiency. PiT [16] investigates the spatial dimension conversion and integrates pooling layers between self-attention blocks. Swin-Transformer [28] adopts a hierarchical architecture of high flexibility to model the image at various scales. Twins [6] utilizes a hierarchical structure consists of a locally-grouped self-attention and a global sub-sampled attention. CaiT [40] builds and optimizes deeper transformer networks for image classification. Multi-scale vision Transformer [10] utilizes Transformer for video recognition. Multi-view vision Transformer [4] achieves excellent performance in 3D object recognition.

MLP-based vision models. MLP-Mixer [36] proposes a conceptually and technically simple architecture solely based on MLP layers. To model the communications between spatial locations, it proposes a token-mixing MLP. Despite that MLP-Mixer has achieved promising results when training on a huge-scale dataset JFT-300M, it is not as good as its visual Transformer counterparts when training on a medium-scale dataset including ImageNet-1K and ImageNet-21K. FF [30] adopts a similar architecture but inherits the positional embedding from ViT. Res-MLP [37] also designs a pure MLP

architecture. It proposes an affine transform layer which facilities stacking a huge number of MLP blocks. Using a deeper architecture than MLP-mixer, Res-MLP achieves a higher accuracy than MLP-mixer and a comparable recognition accuracy as ViT. gMLP [27] designs a gating operation to enhance the communications between spatial locations and achieves a comparable recognition accuracy compared with DeiT. EA [12] replaces the self-attention module with an external attention through external memories learned from the training data. It is implemented by a cascade of two linear layers. CCS-MLP [45] rethinks the design of token-mixing MLP and proposes a channel-specific circulant token-mixing MLP. More advanced works [25, 3, 18, 32] adapt hierarchical pyramid to enhance the representing power.

3. Method

3.1. Preliminary

Layer Normalization (LN) [1]. Given a c-dimensional vector $\mathbf{x} = [x_1, \cdots, x_c]$, layer normalization computes the mean $\mu = \frac{1}{c}\sum_{i=1}^{c} x_i$ and the standard deviation $\sigma = \sqrt{\frac{1}{c}\sum_{i=1}^{c}(x_i - \mu)^2}$. It normalizes each entry in $\mathbf{x}$ by $\bar{x}_i = \frac{x_i - \mu}{\sigma}, \ \forall i \in [1, c]$.

Gaussian Error Linear Units (GELU) is defined as $\mathrm{GELU}(x) = x\Phi(x)$, where $\Phi(x)$ is the standard Gaussian cumulative distribution function defined as $\Phi(x) = \frac{1}{2}[1 + \mathrm{erf}(x/\sqrt{2})]$.

MLP-Mixer [36] stacks N blocks. Each block consists of two types of MLP layers: channel-mixing MLP and token-mixing MLP. We denote a patch feature by $\mathbf{p}_i \in \mathbb{R}^c$ and an image with n patch features by $\mathbf{P} = [\mathbf{p}_1, \cdots, \mathbf{p}_n]$. Channel-mixing MLP projects $\mathbf{P}$ along the channel dimension:

$$\hat{\mathbf{P}} = \mathbf{P} + \mathbf{W}_2 \mathrm{GELU}(\mathbf{W}_1 \mathrm{LN}(\mathbf{P})). \tag{1}$$

Token-mixing MLP projects the channel-mixed patch features $\hat{\mathbf{P}}$ along the spatial dimension:

$$\bar{\mathbf{P}} = \hat{\mathbf{P}} + \mathrm{GELU}(\mathrm{LN}(\hat{\mathbf{P}})\mathbf{W}_3)\mathbf{W}_4, \tag{2}$$

where $\mathbf{W}_3 \in \mathbb{R}^{N \times \bar{N}}$ and $\mathbf{W}_4 \in \mathbb{R}^{\bar{N} \times N}$.

3.2. Spatial-Shift MLP Architecture

As shown in Figure 1, our spatial-shift MLP backbone consists of a patch-wise fully-connected layer, N S^2-MLP blocks, and a fully-connected layer for classification. The proposed spatial-shift operation is closely related to Shift [43], 4-connected Shift [2] and TSM [26]. Our spatial-shift operation can be regarded as a special version of 4-Connected Shift without origin element information. Different from the 4-connected shift residual block [2] in a fc-shift-fc structure, our S^2-MLP block, as visualized in Figure 1, takes another two fully-connected layers only for mixing channels after a fc-shift-fc structure. Besides, 4-connected shift residual network exploits convolution in the early layer, whereas ours adopts a pure-MLP structure.

Patch-wise fully-connected layer. We denote an image by $I \in \mathbb{R}^{W \times H \times 3}$. It is uniformly split into $w \times h$ patches, $\mathcal{P} = \{\boldsymbol{\mathcal{P}}_i\}_{i=1}^{wh}$, where $\boldsymbol{\mathcal{P}}_i \in \mathbb{R}^{p \times p \times 3}$, $w = \frac{W}{p}$, and $h = \frac{H}{p}$. For each patch $\boldsymbol{\mathcal{P}}_i$, we unfold it into a vector $\mathbf{p}_i \in \mathbb{R}^{3p^2}$ and project it into an embedding vector $\mathbf{e}_i$ through a fully-connected layer followed by a layer normalization:

$$\mathbf{e}_i = \mathrm{LN}(\mathbf{W}_0 \mathbf{p}_i + \mathbf{b}_0), \tag{3}$$

where $\mathbf{W}_0 \in \mathbb{R}^{c \times 3p^2}$ and $\mathbf{b}_0 \in \mathbb{R}^c$ are parameters of the fully-connected layer and $\mathrm{LN}(\cdot)$ is the layer normalization.

S^2-MLP block. Our architecture stacks N S^2-MLP of the same size and structure. Each spatial-shift block contains four fully-connected layers, two layer-normalization layers, two GELU layers, two skip-connections, and the proposed spatial-shift module. It is worth noting that all fully-connected layers used in our S^2-MLP only serve to mix the channels. We do not use the token-mixing MLP in MLP-mixer. Since the fully-connected layer is well known, and we have already introduced layer normalization and GELU above, we only focus on the proposed spatial-shift module here. We denote the feature map in the input of our spatial-shift module by $\boldsymbol{\mathcal{T}} \in \mathcal{R}^{w \times h \times c}$, where w denotes the width, h represents the height, and c is the number of channels. The spatial-shift operation can be decomposed into two steps: 1) split the channels into several groups, and 2) shift each group of channels in different directions.

Group. We uniformly split $\boldsymbol{\mathcal{T}}$ along the channel dimension and obtain g thinner tensors $\{\boldsymbol{\mathcal{T}}_\tau\}_{\tau=1}^{g}$ where $\boldsymbol{\mathcal{T}}_\tau \in \mathcal{R}^{w \times h \times c/g}$. It is worth noting that the number of groups, g, is dependent on the design of the shifting directions in the second step. For instance, by default, we only shift along four directions, and thus g is set as 4 in this configuration.

Spatial-shift operation. For the first group of channels, $\boldsymbol{\mathcal{T}}_1$, we shift it along the wide dimension by $+1$. In parallel, we shift the second group of channels, $\boldsymbol{\mathcal{T}}_2$, along the wide dimension by -1. Similarly, we shift $\boldsymbol{\mathcal{T}}_3$ along the height dimension by $+1$ and $\boldsymbol{\mathcal{T}}_4$ along the height dimension by -1. We clarify the formulation of the spatial-shift operation in Eq. (4) and demonstrate the pseudocode in Algorithm 1.

$$\begin{aligned}
&\boldsymbol{\mathcal{T}}_1[1:w,:,:] \leftarrow \boldsymbol{\mathcal{T}}_1[0:w-1,:,:], \\
&\boldsymbol{\mathcal{T}}_2[0:w-1,:,:] \leftarrow \boldsymbol{\mathcal{T}}_2[1:w,:,:], \\
&\boldsymbol{\mathcal{T}}_3[:,1:h,:] \leftarrow \boldsymbol{\mathcal{T}}_3[:,0:h-1,:], \\
&\boldsymbol{\mathcal{T}}_4[:,0:h-1,:] \leftarrow \boldsymbol{\mathcal{T}}_4[:,1:h,:].
\end{aligned} \tag{4}$$

After spatially shifting, each patch absorbs the visual content from its adjoining patches. The spatial-shift operation is parameter-free and makes the communication between

Algorithm 1 Pseudocode of our spatial-shift operation.

```
def spatial_shift(x):
  w,h,c = x.size()
  x[1:,:,:c/4] = x[:w-1,:,:c/4]
  x[:w-1,:,c/4:c/2] = x[1:,:,c/4:c/2]
  x[:,1:,c/2:c*3/4] = x[:,:h-1,c/2:c*3/4]
  x[:,:h-1,3*c/4:] = x[:,1:,3*c/4:]
  return x
```

different spatial locations feasible. The above mentioned spatial-shift manner is one of most straightforward ways for shifting. We also evaluate other manners. Surprisingly, the above simple manner has achieved excellent performance. Using the spatial-shift operation, we no longer need token-mixer as MLP-mixer. We only need channel-mixer to project the patch-wise feature along the channel dimension. Note that the spatial-shift operation in a single block is only able to gain the visual content from adjacent patches and cannot have access to visual content of all patches in the image. But we stack N S^2-MLP blocks, the global visual content will be gradually diffused to every patch.

3.3. Relations with depthwise convolution

Depthwise convolution. Given a feature map defined as a tensor $\mathcal{T} \in \mathbb{R}^{w\times h\times c}$, depthwise convolution [5, 19, 20] utilize a two dimensional convolution kernel $\mathbf{K}_i$ separably on each two-dimensional slice of the tensor $\mathcal{T}[:,:,i] \in \mathbb{R}^{w\times h}$ where $i \in [1, c]$. Depthwise convolution takes cheap computational cost and thus is widely used in efficient neural network for fast inference.

Relations. The proposed spatial shift is inspired by the temporal shift proposed in TSM [26]. It is originally utilized for modeling the temporal relations between adjacent frames, which shifts the channels along the temporal dimension. In this work, we extend it to the two-dimensional spatial scenario. In fact, the spatial-shift operation is equal to a depthwise convolution with a fixed and group-specific kernel weights. Let denote a set of depthwise convolution kernels as $\mathcal{K} = \{\mathbf{K}_1, \cdots, \mathbf{K}_c\}$. If we set

$$\mathbf{K}_i = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \forall i \in (0, \frac{c}{4}],$$

$$\mathbf{K}_j = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad \forall j \in (\frac{c}{4}, \frac{c}{2}],$$

$$\mathbf{K}_k = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \forall k \in (\frac{c}{2}, \frac{3c}{4}],$$

$$\mathbf{K}_l = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad \forall l \in (\frac{3c}{4}, c],$$

the depthwise convolution based on the group of kernels $\mathcal{K}$ is equivalent to our spatial-shift operation.

That is, our spatial-shift operation is a variant of depthwise convolution with the fixed weights defined above. Meanwhile, the spatial-shift operation shares kernel weights within each group of channels. As we mentioned in the introduction, token-mixing MLP in MLP-mixer is a global-reception and spatial-specific depthwise convolution. Meanwhile, compared with our spatial shift and vanilla depthwise convolution, the weights for summation in token-mixing are shared cross channels for a specific spatial location. In contrast, the vanilla depthwise convolution learns different convolution kernels for different channels, and our spatial-shift operation shares the weights within the group and adopts different weights for different groups. In other words, both our spatial-shift operation and token-mixing MLP in MLP-mixer are variants of depthwise convolution. We summarize their relations and differences in Table 1.

	wights	reception field	spatial	channel
TM	learned	global	specific	agnostic
S^2	fixed	local	agnostic	group-specific
DC	learned	local	agnostic	specific

Table 1. Relations among token-mixing (TM), spatial-shift (S^2) and depthwise convolution (DC).

3.4. Complexity Analysis

Patch-wise fully-connected layer (PFL) projects each cropped patch, $\boldsymbol{\mathcal{P}} \in \mathbb{R}^{p\times p\times 3}$, into a c-dimensional vector. The weights of PFL satisfy $\mathbf{W}_0 \in \mathbb{R}^{c\times 3p^2}$ and $\mathbf{b}_0 \in \mathbb{R}^c$. Thus, the number of parameters in PFL is

$$\text{Params}_{\text{PFL}} = (3p^2 + 1)c. \tag{5}$$

The total number of patches is $M = w \times h = \frac{W}{p} \times \frac{H}{p}$ where W is the width and H is the height of the input image. In this case, the floating operations (FLOPs) in PFL is

$$\text{FLOPs}_{\text{PFL}} = 3Mcp^2. \tag{6}$$

It is worth noting that, following previous works [38, 13], we only consider the multiplication operation between float numbers when counting FLOPs.

S^2-MLP blocks. Our S^2-MLP architecture consists of N S^2-MLP blocks. The input and output of all blocks are of the same size. We denote the input of the i-th S^2-MLP block by a tensor $\boldsymbol{\mathcal{T}}_{\text{in}}^{(i)}$ and the output by $\boldsymbol{\mathcal{T}}_{\text{out}}^{(i)}$. Then, they satisfy

$$\boldsymbol{\mathcal{T}}_{\text{in}}^{(i)}, \boldsymbol{\mathcal{T}}_{\text{out}}^{(i)} \in \mathbb{R}^{w\times h\times c}, \quad \forall i \in [1, N]. \tag{7}$$

Meanwhile, all S^2-MLP blocks take the same operation and are of the same configuration. This leads to the fact that all blocks take the same computational cost and the same number of parameters. To obtain the total number of parameters

and FLOPs of the proposed S^2-MLP architecture, we only need count that for each basic block.

Only fully-connected layers contain parameters. As shown in Figure 1, S^2-MLP contains four fully-connected layers. We denote the weights of the first two fully-connected layer as $\{\mathbf{W}_1, \mathbf{b}_1\}$ and $\{\mathbf{W}_2, \mathbf{b}_2\}$ where $\mathbf{W}_1 \in \mathbb{R}^{c\times c}$ and $\mathbf{W}_2 \in \mathbb{R}^{c\times c}$. These two fully-connected layer keep the feature dimension unchanged. We denote the weights of the third fully-connected layer as $\{\mathbf{W}_3, \mathbf{b}_3\}$ where $\mathbf{W}_3 \in \mathbb{R}^{\bar{c}\times c}$ and $\mathbf{b}_3 \in \mathbb{R}^{\bar{c}}$. $\bar{c}$ denotes the hidden size. Following ViT and MLP-mixer, we set $\bar{c} = rc$ where r is the expansion ratio which is set as 4, by default. In this step, the feature dimension of each patch increases from c to $\bar{c}$. In contrast, the fourth fully-connected layer reduces the dimension of each patch from $\bar{c}$ back to c. Its weights $\mathbf{W}_4 \in \mathbb{R}^{c\times\bar{c}}$ and $\mathbf{b}_4 \in \mathbb{R}^{c}$. The number of parameters per S^2-MLP block is the total number of entries in $\{\mathbf{W}_i, \mathbf{b}_i\}_{i=1}^4$ is

$$\text{Params}_{\text{S}^2} = c(2c{+}2\bar{c}){+}3c{+}\bar{c} = c^2(2r{+}2){+}c(3{+}r). \quad (8)$$

Meanwhile, the total FLOPs of fully-connected layers in each S^2-MLP block is

$$\text{FLOPs}_{\text{S}^2} = M(2c^2 + 2c\bar{c}) = Mc^2(2r + 2). \quad (9)$$

Fully-connected classification layer (FCL) takes input the c-dimensional vector from average-pooling M patch features in the output of the last S^2-MLP block. It outputs k-dimensional score vector where k is the number of classes. Hence, the number of parameters in FCL is

$$\text{Params}_{\text{FCL}} = (c + 1)k. \quad (10)$$

Meanwhile, the FLOPs of FCL is

$$\text{FLOPs}_{\text{FCL}} = Mck. \quad (11)$$

By adding up the number of parameters in the patch-wise fully-connected layer, N S^2-MLP blocks, and the fully-connected classification layer, we obtain the total number of parameters of thw whole architecture:

$$\text{Params} = \text{Params}_{\text{PFL}} + N * \text{Params}_{\text{S}^2} + \text{Params}_{\text{FCL}}.$$

And the total number of FLOPs is

$$\text{FLOPs} = \text{FLOPs}_{\text{PFL}} + N * \text{FLOPs}_{\text{S}^2} + \text{FLOPs}_{\text{FCL}}.$$

3.5. Implementation

We set the cropped patch size ($p \times p$) as 16×16. We reshape input image into the 224×224 size. Thus, the number of patches $M = (224/16)^2 = 196$. We set expansion ratio $r = 4$. We attempt two types of settings: 1) wide settings and 2) deep settings. The wide settings follow the base model of MLP-mixer [36]. The wide settings set the number of S^2-MLP blocks (N) as 12 and the hidden size c as 768. Note that MLP-mixer also implements the large model and the huge model. Nevertheless, our limited computing resources cannot afford the expensive cost of investigating the large and huge models on ImageNet-1K dataset. The deep settings follow ResMLP-36 [37]. The deep settings set the number of S^2-MLP blocks (N) as 36 and the hidden size c as 384. We summarize the hyperparameters, the number of parameters, and FLOPs of two settings in Table 2.

Settings	M	N	c	r	p	Para.	FLOPs
wide	196	12	768	4	16	71M	14B
deep	196	36	384	4	16	51M	10.5B

Table 2. The hyper-parameters, the number of parameters and FLOPs. Following MLP-Mixer [36], the number of parameters excludes the weights of the fully-connected layer for classification.

4. Experiments

Datasets. We evaluate the performance of S^2-MLP on the widely used benchmark, ImageNet-1K [8]. It consists of 1.2 million training images from one thousand categories and 50 thousand validation images with 50 images in each category. Meanwhile, due to limited computing resources, the ablation study is only conducted on its subset ImageNet100. It only contains images of randomly selected 100 categories. ImageNet100 contains 0.1 million training images and 5 thousand images for validation.

Training details. We adopt the training strategy provided by DeiT [38]. To be specific, we train our model using AdamW [29] with weight decay 0.05 and a batch size of 1024. We use a linear warmup and cosine decay. The initial learning rate is 1e-3 and gradually drops to 1e-5 in 300 epochs. We also use label smoothing [34], DropPath [24], and repeated augmentation [17]. All training is conducted on a Linux server equipped with four NVIDIA Tesla V100 GPU cards. The whole training process of the proposed S^2-MLP on ImageNet-1K dataset takes around 4.5 days. S^2-MLP is implemented in PaddlePaddle platform developed by Baidu.

4.1. Main results

The main results are summarized in Table 3. As shown in the table, compared with ViT [9], Mixer-B/16 is not competitive in terms of accuracy. In contrast, the proposed S^2-MLP has obtained a comparable accuracy with respect to ViT. Meanwhile, Mixer-B/16 and our S^2-MLP take considerably fewer parameters and FLOPs, making them more attractive compared with ViT when efficiency is important. Meanwhile, we note that, by introducing some hard-crafted design, following Transformer-based works such as PVT-Large [42], TNT-B [13], T2T-ViT$_t$-24 [46], CaiT [40], Swin-B [28], and Nest-B [48] have considerably improved ViT. MLP-based models including the proposed S^2-MLP cannot

Model	Resolution	Top-1 (%)	Top5 (%)	Params (M)	FLOPs (B)
CNN-based					
ResNet50 [14]	224×224	76.2	92.9	25.6	4.1
ResNet152 [14]	224×224	78.3	94.1	60.2	11.5
RegNetY-8GF [31]	224×224	79.0	–	39.2	8.0
RegNetY-16GF [31]	224×224	80.4	–	83.6	15.9
EfficientNet-B3 [35]	300×300	81.6	95.7	12	1.8
EfficientNet-B5 [35]	456×456	84.0	96.8	30	9.9
Transformer-based					
ViT-B/16 [9]	384×384	77.9		86.4	55.5
ViT-B/16* [9, 36]	224×224	79.7	–	86.4	17.6
DeiT-B/16 [38]	224×224	81.8	–	86.4	17.6
PiT-B/16 [16]	224×224	82.0	–	73.8	12.5
PVT-Large [42]	224×224	82.3	–	61.4	9.8
CPVT-B [7]	224×224	82.3	–	88	17.6
TNT-B [13]	224×224	82.8	96.3	65.6	14.1
T2T-ViT$_t$-24 [46]	224×224	82.6	–	65.1	15.0
CaiT-S32 [40]	224×224	83.3	–	68	13.9
Swin-B [28]	224×224	83.3	–	88	15.4
Nest-B [48]	224×224	83.8	–	68	17.9
Container [11]	224×224	82.7	–	22.1	8.1
MLP-based ($c = 768$, $N = 12$)					
Mixer-B/16 [36]	224×224	76.4	–	59	11.6
FF [30]	224×224	74.9	–	59	11.6
S^2-MLP-wide (ours)	224×224	80.0	94.8	71	14.0
MLP-based ($c = 384$, $N = 36$)					
ResMLP-36 [37]	224×224	79.7	–	45	8.9
S^2-MLP-deep (ours)	224×224	80.7	95.4	51	10.5

Table 3. Results on ImageNet-1K without extra data. ViT-B/16* denotes the ViT-B/16 model in MLP-mixer [36] with extra regularization.

achieve as high recognition accuracy as the state-of-the-art Transformer-based vision models such as CaiT, Swin-B and Nest-B. Meanwhile, the state-of-the-art Transformer-base vision model, Nest-B, cannot achieve better trade-off between the recognition accuracy and efficiency compared with the state-of-the-art CNN model, EfficientNet-B5 [35].

After that, we compare our S^2-MLP architecture with its MLP counterparts which are recently proposed including MLP-mixer, FF [30], ResMLP-36 [37]. Among them, MLP-mixer, FF and ResMLP-36 adopt a similar structure. A difference between ResMLP-36 and MLP-mixer is that ResMLP-36 develops an affine transformation layer to replace the layer normalization for a more stable training. Meanwhile, ResMLP-36 stacks more MLP layers than MLP-mixer but uses a smaller hidden size. Specifically, ResMLP-36 adopts 36 MLP layers with a 384 hidden size. In contrast, MLP-Mixer uses 12 MLP layers with a 768 hidden size. Through a trade-off between the number of MLP layers and hidden size, ResMLP-36 leads to a higher accuracy.

Our wide model, S^2-MLP-wide adopts the wide settings in Table 2. Specifically, same as MLP-mixer and FF, S^2-MLP-wide adopts 12 blocks with hidden size 768. As shown in Table 3, compared with MLP-mixer and FF, the proposed S^2-MLP-wide achieves a considerably higher recognition accuracy. Specifically, MLP-mixer only achieves a 76.4% top-1 accuracy and FF only achieves a 74.9% accuracy. In contrast, the top-1 accuracy of the proposed S^2-MLP-wide is 80.0%. In parallel, our deep model, S^2-MLP-deep adopts the deep settings in Table 2. Specifically, same as ResMLP, S^2-MLP-deep adopts 36 blocks with hidden size 384. Meanwhile, we also use the affine transformation proposed in ResMLP to replace layer normalization for a fair comparison. As shown in Table 3, compared with ResMLP-36, our S^2-MLP-deep achieves higher recognition accuracy. Another drawback of MLP-mixer and ResMLP is that, the size of the weight matrix in token-mixer MLP, $\mathbf{W} \in \mathbb{R}^{N \times N}$ ($N = wh$), is dependent on the feature map size. **That is, the structure of MLP-mixer as well as ResMLP varies as the input scale changes.** Thus, MLP-mixer and ResMLP trained on the feature map of 14×14 size generated from an image of 224×224 size can not process the feature map of 28×28 size from an image of 448×448 size. In contrast, the architecture of our S^2-MLP is invariant to the input scale.

Transfer learning. We testify the performance of our S^2-

method	scale	C10	C100	Car
ViT-B/16 [9]	384	98.1	87.1	–
ViT-L/16 [9]	384	97.9	86.4	–
DeiT-B [38]	224	99.1	90.8	92.1
ResMLP-S12 [37]	224	98.1	87.0	84.6
ResMLP-S24 [37]	224	98.7	89.5	89.5
ResMLP-36*	224	98.7	88.5	91.0
S^2-MLP-deep	224	98.8	89.4	93.1

Table 4. The performance of transfer learning on CIFAR10 (C10), CIFAR100 (C100) and Car. ResMLP-36* denotes the performance of fine-tuned ResMLP-36 with the same settings as S^2-MLP-deep.

MLP in transfer leanring. We load the pre-trained model on ImageNet-1K dataset and fine-tune it on the target datasets including CIFAR10/100 [22] and Stanford Car [21]. Among them, CIFAR10 and CIFAR100 contain $50,000$ training images. In contrast, Stanford Car only contains $8,144$ training images, which is indeed a good testbed for over-fitting. As shown in Table 4, using 224×224-scale images, our S^2-MLP-deep achieves better performance than ViT-B/16 and ViT-L/16 using 384×384-scale images. Meanwhile, our S^2-MLP-deep achieves the comparable performance as DeiT-B [38] with considerably fewer FLOPs. Meanwhile, we compare with ResMLP-S12/24 [37] and ResMLP-36 fine-tuned based on the same settings as our S^2-MLP-deep. As shown in the table, on CIFAR100 and Car datasets, our S^2-MLP-deep considerably outperforms ResMLP-36. Especially, on Sanford Car dataset, our S^2-MLP-deep achieves a 93.1 top-1 accuracy whereas ResMLP-36 only obtains a 91.0 accuracy. The considerably worse performance of ResMLP-36 on the small scale Car dataset reveals the fact that the token-mixing MLP is more prone to over-fitting compared with our spatial-shift operation.

configuration	Top-1 (%)
depthwise conv	80.5
spatial shift (ours)	80.7

Table 5. The performance of the 3×3 depthwise convolution.

Depthwise convolution. As we mentioned in Section 3.3, our spatial-shift operation is a variant of 3×3 depthwise convolution with a fixed kernel. We believe that when replacing the spatial-shift operation with a depthwise convolution, it can still achieve a good performance. But the 3×3 depthwise convolution will bring more parameters and FLOPs compared with our paramter-free and computation-free spatial-shift operation. To validate it, we conduct the experiment by replacing the spatial-shift operation with a 3×3 depthwise convolution. As shown in Table 5, interestingly, the proposed spatial-shift operation equivalent to 3×3 depthwise convolution with pre-defined kernel achieves comparable accuracy as the 3×3 depthwise convolution with learned weights from the data.

4.2. Ablation study

Due to limited computing resources, the ablation study is conducted on ImageNet100, which is a subset of ImageNet-1K containing images of randomly selected 100 categories. Meanwhile, due to the limited space, the ablation study in this section only includes that with the wide settings. We move that with the deep settings in the supplementary material. We only change one hyperparameter each time and keep the others the same as the wide settings in Table 2.

r	Top-1 (%)	Top-5 (%)	Para. (M)	FLOPs (B)
1	86.1	96.7	29	5.7
2	86.4	96.9	43	8.4
3	87.0	96.8	57	11
4	87.1	97.1	71	14
5	86.6	96.8	86	17

Table 6. The influence of the expansion ratio, r.

Expansion ratio. Recall that the weights of the third layer and the fourth fully-connected layer, $\mathbf{W}_3 \in \mathbb{R}^{rc \times c}$ and $\mathbf{W}_4 \in \mathbb{R}^{c \times rc}$. r determines the modeling capability of these two fully-connected layers in each S^2-MLP block. Table 6 shows the influence of r. As shown in the table, the top-1 accuracy increases from 86.1% to 87.0% as r increases from 1 to 3. Meanwhile, the number of parameters increases from 29M to 57M accordingly. But the accuracy saturates and even turns worse when r surpasses 3. This might be due to the fact that ImagetNet100 is too small and our model suffers from over-fitting when r is large.

c	Top-1 (%)	Top-5 (%)	Para. (M)	FLOPs (B)
192	79.7	94.7	4.3	0.9
384	85.3	96.6	17	3.5
576	85.7	96.7	38	7.9
768	87.1	97.1	71	14
960	87.0	97.0	106	20

Table 7. The influence of the hidden size, c.

Hidden size. The hidden size (c) in MLPs of S^2-MLP blocks also determine the modeling capability of the proposed S^2-MLP architecture. In Table 7, we show the influence of c. As shown in the table, the top-1 recognition accuracy increases from 79.7% to 87.1% as the hidden size c increases from 192 and 768, and the number of parameters increases from 4.3M to 71M, and FLOPs increases from 0.9G to 20G. Meanwhile, the recognition accuracy saturates when c surpasses 768. Taking both accuracy and efficiency into consideration, we set $c = 768$, by default.

Shifting directions. By default, we split 768 channels into four groups and shift them along four directions as Figure 2 (a). We also attempt other shifting settings. (b) splits the channels into 8 groups, and shift them along eight directions. (c), (d), (e), and (f) split the channels into two groups, and

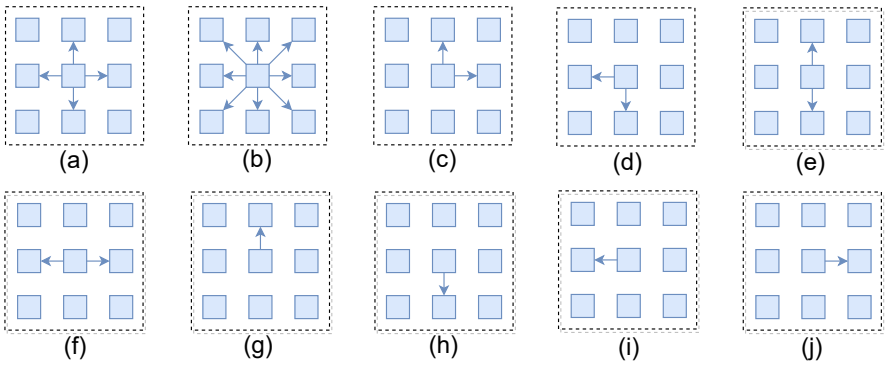

Figure 2. Ten different shifting settings. (a) is the default option which shifts channels along four directions. (b) shifts channels along eight directions. (c),(d),(e), and (f) shift channels in two directions. (g), (h), (i), and (j) shift channels along a single direction.

Settings	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	w/o
Top-1 (%)	87.1	87.0	85.0	85.1	79.5	80.5	77.7	77.5	78.3	78.4	56.7
Top-5 (%)	97.1	97.1	96.1	96.2	93.1	93.7	92.7	92.5	93.4	93.4	81.0

Table 8. The influence of shifting directions.

shift them along two directions. (g), (h), (i), and (j) shift all channels along a single direction. In Table 8, we show the recognition accuracy of our S^2-MLP with shifting from (a) to (j). Meanwhile, we also show that achieved by S^2-MLP without (w/o) shifting. As shown in the table, without shifting, the network performs poorly due to a lack of communications between patches. Meanwhile, comparing (e) with (f), we discover that the horizontal shifting is more useful than the vertical shifting. Comparing (c) with (e)/(f), we observe that shifting in two dimensions (both horizontal and vertical) will be helpful than shifting in a single dimension (horizontal or vertical). Moreover, comparing (a) and (b), we conclude that shifting along four directions is enough. Overall, the default shifting configuration, (a), the most natural way for shifting, achieves excellent performance.

Input scale. The input image is resized into $W \times H$ before being fed into the network. When the patch size p is fixed, the image of larger scale will generate more patches, which will inevitably bring more computational cost. But a larger scale is beneficial for modeling fine-grained details in the image, and generally leads to higher recognition accuracy. Table 9 shows the influence of the input image scale. As shown in the table, when $W \times H$ increases from 112×112 to 336×336, the top-1 recognition accuracy improves from 80.6% to 88.2%, the number of parameters keeps unchanged since the network does not change, and the FLOPs also increases from 3.5G to 31G. When the input scale increases from 224×224 to 384×384, the gain in recognition accuracy is not significant, but the FLOPs is doubled.

$W \times H$	Top-1 (%)	Top-5 (%)	Para.	FLOPs
112×112	80.6	94.2	71M	3.5B
224×224	87.1	97.1	71M	14B
384×384	88.2	97.6	71M	31B

Table 9. The influence of the input image scale.

When the input image scale is fixed, the increase of patch size will reduce the number of patches. The larger-size patch enjoys high efficiency but is not good at capturing the fine-level details. Thus, the larger-size patches cannot achieve as high accuracy as their smaller counterparts. As shown in Table 10, when p increases from 16 to 32, it reduces FLOPs from 14B to 3.5B. But it also leads to that the top-1 recognition accuracy drops from 87.1% to 81.0%.

$p \times p$	Top-1 (%)	Top-5 (%)	Para.	FLOPs
32×32	81.0	94.6	73M	3.5B
16×16	87.1	97.1	71M	14B

Table 10. The influence of the patch size.

5. Conclusion

In this paper, we propose a spatial shift MLP (S^2-MLP) architecture. It adopts a pure MLP structure without convolution and self-attention. To achieve the communications between spatial locations, we adopt a spatial shift operation, which is simple, parameter-free, and efficient. On ImageNet-1K dataset, it achieves considerably higher recognition accuracy than the pioneering work, ViT, with a comparable number of parameters and FLOPs. Meanwhile, it takes a much simpler architecture, less number of parameters and FLOPs compared with its ViT counterpart. Moreover, we also discuss the relations among the spatial shifting operation, token-mixing MLP in MLP-mixer, and the depthwise convolution. We discover that both token-mixing MLP and the proposed spatial-shift operation are variants of the depthwise convolution. We hope that these results and discussions could inspire further research to discover simpler and more effective vision architecture in the future.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Andrew Brown, Pascal Mettes, and Marcel Worring. 4-connected shift residual networks. In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 1990–1997, Seoul, Korea, 2019.

[3] Shoufa Chen, Enze Xie, Chongjian Ge, Ding Liang, and Ping Luo. Cyclemlp: A mlp-like architecture for dense prediction. *arXiv preprint arXiv:2107.10224*, 2021.

[4] Shuo Chen, Tan Yu, and Ping Li. Mvt: Multi-view vision transformer for 3d object recognition. In *Proceedings of the 32th British Machine Vision Conference (BMVC)*, 2021.

[5] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017*, pages 1800–1807, Honolulu, HI, 2017.

[6] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. *arXiv preprint arXiv:2104.13840*, 2021.

[7] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Conditional positional encodings for vision transformers. *arXiv preprint arXiv:2102.10882*, 2021.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *Proceedings of the 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, Miami, FL, 2009.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, Virtual Event, 2021.

[10] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. *arXiv preprint arXiv:2104.11227*, 2021.

[11] Peng Gao, Jiasen Lu, Hongsheng Li, Roozbeh Mottaghi, and Aniruddha Kembhavi. Container: Context aggregation network. *arXiv preprint arXiv:2106.01401*, 2021.

[12] Meng-Hao Guo, Zheng-Ning Liu, Tai-Jiang Mu, and Shi-Min Hu. Beyond self-attention: External attention using two linear layers for visual tasks. *arXiv preprint arXiv:2105.02358*, 2021.

[13] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Las Vegas, NV, 2016.

[15] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUS). *arXiv preprint arXiv:1606.08415*, 2016.

[16] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. *arXiv: 2103.16302*, 2021.

[17] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8126–8135, Seattle, WA, 2020.

[18] Qibin Hou, Zihang Jiang, Li Yuan, Ming-Ming Cheng, Shuicheng Yan, and Jiashi Feng. Vision permutator: A permutable mlp-like architecture for visual recognition. *arXiv preprint arXiv:2106.12368*, 2021.

[19] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[20] Lukasz Kaiser, Aidan N. Gomez, and François Chollet. Depthwise separable convolutions for neural machine translation. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, Vancouver, Canada, 2018.

[21] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013.

[22] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1106–1114, Lake Tahoe, NV, 2012.

[24] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

[25] Dongze Lian, Zehao Yu, Xing Sun, and Shenghua Gao. As-mlp: An axial shifted mlp architecture for vision. *arXiv preprint arXiv:2107.08391*, 2021.

[26] Ji Lin, Chuang Gan, and Song Han. TSM: temporal shift module for efficient video understanding. In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7082–7092, Seoul, Korea, 2019.

[27] Hanxiao Liu, Zihang Dai, David R So, and Quoc V Le. Pay attention to mlps. *arXiv preprint arXiv:2105.08050*, 2021.

[28] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.

[29] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, New Orleans, LA, 2019.

[30] Luke Melas-Kyriazi. Do you even need attention? a stack of feed-forward layers does surprisingly well on imagenet. *arXiv preprint arXiv:2105.02723*, 2021.

[31] Ilija Radosavovic, Raj Prateek Kosaraju, Ross B. Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10425–10433, Seattle, WA, 2020.

[32] Yongming Rao, Wenliang Zhao, Zheng Zhu, Jiwen Lu, and Jie Zhou. Global filter networks for image classification. *arXiv preprint arXiv:2107.00645*, 2021.

[33] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 843–852, Venice, Italy, 2017.

[34] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, Las Vegas, NV, 2016.

[35] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 6105–6114, Long Beach, CA, 2019.

[36] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. MLP-Mixer: An all-MLP architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021.

[37] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, and Hervé Jégou. ResMLP: Feedforward networks for image classification with data-efficient training. *arXiv preprint arXiv:2105.03404*, 2021.

[38] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

[39] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. *arXiv preprint arXiv:2103.17239*, 2021.

[40] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. *arXiv preprint arXiv:2103.17239*, 2021.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008, Long Beach, CA, 2017.

[42] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv preprint arXiv:2102.12122*, 2021.

[43] Bichen Wu, Alvin Wan, Xiangyu Yue, Peter Jin, Sicheng Zhao, Noah Golmant, Amir Gholaminejad, Joseph Gonzalez, and Kurt Keutzer. Shift: A zero flop, zero parameter alternative to spatial convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9127–9135, 2018.

[44] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. CvT: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808*, 2021.

[45] Tan Yu, Xu Li, Yunfeng Cai, Mingming Sun, and Ping Li. Rethinking token-mixing mlp for mlp-based vision backbone. In *Proceedings of the 32th British Machine Vision Conference (BMVC)*, 2021.

[46] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token ViT: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021.

[47] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. *arXiv preprint arXiv:2103.15358*, 2021.

[48] Zizhao Zhang, Han Zhang, Long Zhao, Ting Chen, and Tomas Pfister. Aggregating nested transformers. *arXiv preprint arXiv:2105.12723*, 2021.