AS-MLP: AN AXIAL SHIFTED MLP ARCHITECTURE FOR VISION

Anonymous authors
Paper under double-blind review

ABSTRACT

An Axial Shifted MLP architecture (AS-MLP) is proposed in this paper. Different from MLP-Mixer, where the global spatial feature is encoded for the information flow through matrix transposition and one token-mixing MLP, we pay more attention to the local features communication. By axially shifting channels of the feature map, AS-MLP is able to obtain the information flow from different axial directions, which captures the local dependencies. Such an operation enables us to utilize a pure MLP architecture to achieve the same local receptive field as CNN-like architecture. We can also design the receptive field size and dilation of blocks of AS-MLP, *etc*, just like designing those of convolution kernels. With the proposed AS-MLP architecture, our model obtains 83.3% Top-1 accuracy with 88M parameters and 15.2 GFLOPs on the ImageNet-1K dataset. Such a simple yet effective architecture outperforms all MLP-based architectures and achieves competitive performance compared to the transformer-based architectures (*e.g.*, Swin Transformer) even with slightly lower FLOPs. In addition, AS-MLP is also the first MLP-based architecture to be applied to the downstream tasks (*e.g.*, object detection and semantic segmentation). The experimental results are also impressive. Our proposed AS-MLP obtains 51.5 mAP on the COCO validation set and 49.5 MS mIoU on the ADE20K dataset, which is competitive compared to the transformer-based architectures. Our AS-MLP establishes a strong baseline of MLP-based architecture.

1 INTRODUCTION

In the past decade, Convolutional Neural Networks (CNNs) (Krizhevsky et al., 2012; He et al., 2015) have received widespread attention and have become the de-facto standard for computer vision. Furthermore, with the in-depth exploration and research on self-attention, transformer-based architectures have also gradually emerged, and have surpassed CNN-based architectures in natural language processing (*e.g.*, Bert (Devlin et al., 2018)) and vision understanding (*e.g.*, ViT (Dosovitskiy et al., 2020), DeiT (Touvron et al., 2020)) with amounts of training data. Recently, Tolstikhin et al. (2021) firstly propose MLP-based architecture, where almost all network parameters are learned from MLP (linear layer) and achieve amazing results, which is comparable with CNN-like models.

Such promising results drive our exploration of MLP-based architecture. In the MLP-Mixer (Tolstikhin et al., 2021), the model obtains the global receptive field through matrix transposition and token-mixing projection such that the long-range dependencies are covered. However, this rarely makes full use of the local information, which is very important in CNN-like architecture (Simonyan & Zisserman, 2014; He et al., 2015) because not all pixels need long-range dependencies, and the local information more focuses on extracting the low-level feature. In the transformer-based architectures, Swin Transformer (Liu et al., 2021b) computes the self-attention in a window (7×7) instead of the global receptive field, which is similar to directly using a convolution layer with a large kernel size (7×7) to cover the local receptive field. Some other papers have also already emphasized the advantages of local receptive fields, and introduced local information in the transformer, such as Localvit (Li et al., 2021), NesT (Zhang et al., 2021), *etc*. Driven by these ideas, we mainly explore the influence of locality on MLP-based architectures.

In order to introduce locality into the MLP-based architecture, one of the simplest and most intuitive ideas is to add a window to the MLP-Mixer, and then perform a token-mixing projection of the local information on the features within the window, just as done in Swin Transformer (Liu et al.,

2021b) and LINMAPPER (Fang et al., 2021). However, for the MLP-based architecture, if we divide the window (*e.g.*, 7×7) and perform the token-mixing projection in the window, then only the 49×49 shared[1] linear layer is employed, which greatly limits the model capacity and thus affects the learning of parameters and the final results. Therefore, we propose an axial shift strategy for MLP-based architecture, where we spatially shift features in both horizontal and vertical directions. The axial shift can arrange the features of *different* spatial positions in the *same* position. After that, a channel-mixing MLP is used to combine these features, which is simple yet effective. Such a method enables the model to obtain more local dependencies, thus improve the performance. It also enables us to design MLP structure as the same as the convolutional kernel, for instance, to design the kernel size and dilation rate.

Based on the axial shift strategy, we design Axial Shifted MLP architecture, named AS-MLP. Our AS-MLP obtains 83.3% Top-1 accuracy with 88M parameters and 15.2 GFLOPs in the ImageNet-1K dataset without any extra training data. Such a simple yet effective method outperforms all MLP-based architectures and achieves competitive performance compared to the transformer-based architectures. The AS-MLP architecture can also be transferred to downstream tasks (*e.g.*, object detection). As far as we know, it is also the first work to apply MLP-based architecture to the downstream task[2]. With the pre-trained model in the ImageNet-1K dataset, AS-MLP obtains 51.5 mAP on the COCO validation set and 49.5 MS mIoU on the ADE20K dataset, which is competitive compared to the transformer-based architectures.

2 Related Work

CNN-based Architectures. Since AlexNet (Krizhevsky et al., 2012) won the ImageNet competition in 2012, the CNN-based architectures have gradually been utilized to automatically extract image features instead of hand-crafted features. Subsequently, the VGG network (Simonyan & Zisserman, 2014) is proposed, which purely uses a series of 3×3 convolution and fully connected layers, and obtains outstanding performance in image classification. Furthermore, ResNet (He et al., 2015) is proposed, which utilizes the residual connection to transfer features in different layers, thereby alleviating the problem of gradient vanishing and obtaining superior performance. After that, the residual module becomes an important component of the network design and is also employed in subsequent transformer-based architectures and MLP-based architectures. Some papers have made further improvements to the convolution operation in CNN-based architecture, such as dilated convolution (Yu & Koltun, 2015) and deformable convolution (Dai et al., 2017). These architectures build the CNN family and are used extensively in computer vision tasks.

Transformer-based Architectures. Transformer is firstly proposed in (Vaswani et al., 2017), where the attention mechanism is utilized to model the relationship between features from the different spatial positions. Subsequently, the popularity of BERT (Devlin et al., 2018) in NLP also promotes the research on transformer in the field of vision. ViT (Dosovitskiy et al., 2020) uses a pure transformer framework to extract visual features, where an image is divided into 16×16 patches and the convolution layer is completely abandoned. It shows that the transformer-based architecture can perform well in large-scale datasets (*e.g.*, JFT-300M). After that, DeiT (Touvron et al., 2020) carefully designs training strategies and data augmentation to further improve performance on the small datasets (*e.g.*, ImageNet-1K). DeepViT (Zhou et al., 2021) and CaiT (Touvron et al., 2021b) consider the optimization problem when the network deepens, and train a deeper transformer network. CrossViT (Chen et al., 2021) combines the local patch and global patch using two vision transformers. CPVT (Chu et al., 2021b) uses a conditional position encoding to effectively encodes the spatial positions of patches. LeViT (Graham et al., 2021) improves ViT from many aspects, including the convolution embedding, extra non-linear projection and batch normalization, *etc*. Transformer-LS (Zhu et al., 2021) proposes a long-range attention and a short-term attention to model long sequences for both language and vision tasks. Some papers also design hierarchical backbone to extract spatial features at different scales, such as PVT (Wang et al., 2021), Swin Transformer (Liu et al., 2021b), Twins (Chu et al., 2021a) and NesT (Zhang et al., 2021), which can be applied to downstream tasks (*e.g.*, object detection and semantic segmentation).

[1] If the linear layer is not shared, the model will not be applied to downstream tasks due to the fixed MLP dimension.

[2] It is worth noting that the MLP-Mixer cannot be transferred to the object detection because the token-mixing MLP has the fixed dimension.

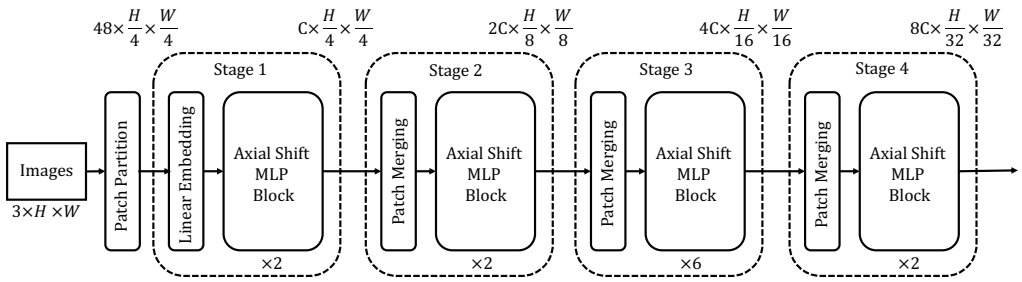


Figure 1: A tiny version of the overall Axial Shifted MLP (AS-MLP) architecture. The final output is used for image classification.

MLP-based Architectures. MLP-Mixer (Tolstikhin et al., 2021) has designed a very concise framework that utilizes matrix transposition and MLP to transmit information between spatial features. Resort to MLP, skip connection between layers and normalization layer, MLP-Mixer obtains promising experimental results. The concurrent work FF (Melas-Kyriazi, 2021) also applies a similar network architecture and reaches similar conclusions. Such experimental results are surprising, which shows that the MLP-based architecture also achieves comparable performance with CNN-based architectures and transformer-based architectures. Subsequently, Res-MLP (Touvron et al., 2021a) is proposed, which also obtains impressive performance with residual MLP only trained on ImageNet-1K. gMLP (Liu et al., 2021a) and EA (Guo et al., 2021) introduce Spatial Gating Unit (SGU) and the external attention to improve the performance of the pure MLP-based architecture, respectively. Recently, Container (Gao et al., 2021) proposes a general network that unifies convolution, transformer, and MLP-Mixer. The closest concurrent methods to our work are S^2-MLP (Yu et al., 2021) and ViP (Hou et al., 2021). S^2-MLP uses spatial-shift MLP for feature exchange. ViP proposes a Permute-MLP layer for spatial information encoding to capture long-range dependencies. Different from their work, we focus on capturing the local dependencies with axially shifting features in the spatial dimension, which obtains better performance and can be applied to the downstream tasks.

3 THE AS-MLP ARCHITECTURE

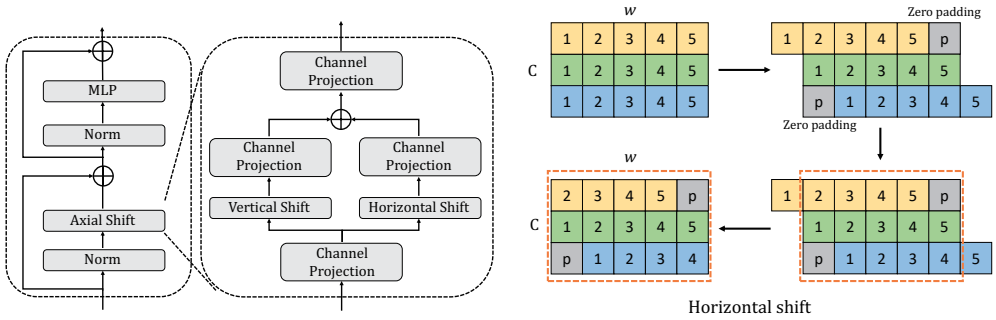
3.1 OVERALL ARCHITECTURE

Figure 1 shows our Axial Shifted MLP (AS-MLP) architecture. Given an RGB image $I \in \mathbb{R}^{3\times H\times W}$, where H and W are the height and width of the image, respectively. AS-MLP firstly performs the patch partition operation, which splits the original image into multiple patch tokens with the patch size of 4×4, thus the combination of all tokens has the size of $48 \times \frac{H}{4} \times \frac{W}{4}$. AS-MLP has four stages in total and there are different numbers of AS-MLP blocks in different stages. Figure 1 only shows the tiny version of AS-MLP, and other variants will be discussed in Sec. 3.4. All the tokens in the previous step will go through these four stages, and the final output feature will be used for image classification. In Stage 1, a linear embedding and the AS-MLP blocks are adopted for each token. The output has the dimension of $C \times \frac{H}{4} \times \frac{W}{4}$, where C is the number of channels. Stage 2 firstly performs patch merging on the features outputted from the previous stage, which groups the neighbor 2×2 patches to obtain a feature with the size of $4C \times \frac{H}{8} \times \frac{W}{8}$ and then a linear layer is adopted to warp the feature size to $2C \times \frac{H}{8} \times \frac{W}{8}$, followed by the cascaded AS-MLP blocks. Stage 3 and Stage 4 have similar structures to Stage 2, and the hierarchical representations will be generated in these stages.

3.2 AS-MLP BLOCK

The core operation of AS-MLP architecture is the AS-MLP block, which is illustrated in Figure 2a. It mainly consists of the Norm layer, Axial Shift operation, MLP, and residual connection. In the Axial Shift operation, we utilize the channel projection, vertical shift, and horizontal shift to extract features, where the channel projection maps the feature with a linear layer. Vertical shift and horizontal shift are responsible for the feature translation along the spatial directions.

As shown in Figure 2b, we take the horizontal shift as an example. The input has the dimension of $C \times h \times w$, for convenience, we omit h and assume $C = 3$, $w = 5$ in this figure. When the shift size is 3, the input feature is split into three parts and each part is shifted by $\{$-1, 0, 1$\}$ units along the



(a) (b)

Figure 2: (a) shows the structure of the AS-MLP block; (b) shows the horizontal shift, where the arrows indicate the steps, and the number in each box is the index of the feature.

horizontal direction, respectively. In this operation, zero padding is performed (indicated by gray blocks), and we also discuss the experimental results of using other padding methods in Sec. 4. After that, the features in the dashed box will be taken out and used for the next channel projection. The same operation is also performed in the vertical shift. In the process of horizontal shift and vertical shift, since the feature performs different shift units in both directions, when the features are recombined (the feature in the dashed box), the information from different spatial positions can be combined together. In the next channel projection operation, information from different spatial locations can fully flow and interact. The code of AS-MLP block is listed in Alg. 1.

Complexity. In the transformer-based architecture, the multi-head self-attention (MSA) is usually adopted, where the attention between tokens is computed. Swin Transformer (Liu et al., 2021b) introduces a window to partition the image and propose window multi-head self-attention (W-MSA), which only considers the computation within this window. It significantly reduces the computation complexity. However, in the AS-MLP block, without the concept of the window, we only Axially Shift (AS) the feature from the previous layer, which does not require any multiplication and addition operations. Further, the time cost of Axial Shift is very low and almost irrelevant to the shift size. Given a feature map (is usually named patches in transformer) with the dimension of $C \times h \times w$, each Axial shift operation in Figure 2a only has four channel projection operations, which has the computation complexity $4hwC^2$. If the window size in Swin Transformer (Liu et al., 2021b) is M, the complexities of MSA, W-MSA and AS are as follows:

$$\begin{cases} \Omega(\text{MSA}) = 4hwC^2 + 2(hw)^2C, \\ \Omega(\text{W-MSA}) = 4hwC^2 + 2M^2hwC, \\ \Omega(\text{AS}) = 4hwC^2. \end{cases} \tag{1}$$

Therefore, the AS-MLP architecture has slightly less complexity than Swin Transformer. The specific complexity calculation of each layer is shown in Appendix A.2.

3.3 Comparisons between AS-MLP, Convolution, Transformer and MLP-Mixer

In this section, we show the comparisons between AS-MLP and the recent distinct building blocks used in computer vision, *e.g.*, the standard convolution, Swin Transformer, and MLP-Mixer. Although these modules are explored in completely different routes, from the perspective of calculation, they are all based on a given output location point, and its value depends on the weighted sum of different sampling location features (multiplication and addition operation). These sampling location features include local dependencies (*e.g.*, convolution) and long-range dependencies (*e.g.*, MLP-Mixer). Figure 3 shows the main differences of these modules in the sampling location.

Convolution. Given a feature map X, a sliding kernel with the shape of $k \times k$ is performed on X to obtain the output Y_{conv}:

$$Y_{\text{conv}} = \sum_{i=0}^{h-1} \sum_{j=0}^{w-1} W^{k \times k} X_{i,j}^{k \times k}, \tag{2}$$

where W is the learnable weight. h and w are the height and width of X, respectively. As shown in Figure 3, the convolution operation has a local receptive field, thus it is more suitable at extracting features with the local dependencies.

Algorithm 1 Code of AS-MLP Block in a PyTorch-like style.

```
# norm: normalization layer
# proj: channel projection
# actn: activation layer

import torch
import torch.nn.functional as F

def shift(x, dim):
   x = F.pad(x, "constant", 0)
   x = torch.chunk(x, shift_size, 1)
   x = [ torch.roll(x_c, shift, dim) for x_s,
        shift in zip(x, range(-pad, pad+1))]
   x = torch.cat(x, 1)
   return x[:, :, pad:-pad, pad:-pad]

def as_mlp_block(x):
   shortcut = x
   x = norm(x)
   x = actn(norm(proj(x)))
   x_lr = actn(proj(shift(x, 3)))
   x_td = actn(proj(shift(x, 2)))
   x = x_lr + x_td
   x = proj(norm(x))
   return x + shortcut
```

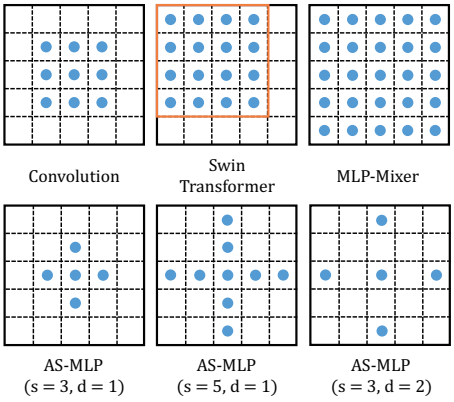


Figure 3: The different sampling locations of convolution, Swin Transformer, MLP-Mixer, and AS-MLP. *e.g.*, AS-MLP (s = 3, d = 1) shows the sampling locations when the shift size is 3 and dilation is 1.

Swin Transformer. Swin Transformer introduces a window into the transformer-based architecture to cover the local attention. The input X from a window is embedded to obtain Q, K, V matrix, and the output Y_{swin} is the attention combination of features within the window:

$$Y_{\text{swin}} = \text{Softmax}(Q(X)K(X)^T/\sqrt{d})V(X). \tag{3}$$

The introduction of locality further improves the performance of the transformer-based architecture and reduce the computational complexity.

MLP-Mixer. MLP-Mixer abandons the attention operation. It firstly transposes the input X, and then a token-mixing MLP is appended to obtain the output Y_{mixer}:

$$Y_{\text{mixer}} = (X^T W_t)^T, \tag{4}$$

where $W_t \in R^{hw \times hw}$ is the learnable weight in token-mixing MLP. MLP-Mixer perceives the global information only with matrix transposition and MLP.

AS-MLP. AS-MLP axially shifts the feature map as shown in Figure 2b. Given the input X and the shift size s, X is firstly divided into s splits in the horizontal and vertical direction, which is represented to Concat$(X_1^h, \cdots, X_s^h)$ and Concat$(X_1^v, \cdots, X_s^v)$. After the axial shift in Figure 2b, the output Y_{as} is:

$$Y_{\text{as}} = X^h W_c^h + X^v W_c^v, \tag{5}$$

where W_c^h and W_c^v are the learnable weights of channel projection in the horizontal and vertical directions, $X^h = \text{Concat}(\overrightarrow{X_1^h}, \cdots, \overrightarrow{X_s^h})$ and $X^v = \text{Concat}(\overrightarrow{X_1^v}, \cdots, \overrightarrow{X_s^v})$, which shows the axial shift. Unlike MLP-Mixer, we pay more attention to the local dependencies through axial shift of features and channel projection. Such an operation is closely related to Shift (Wu et al., 2018) and TSM (Lin et al., 2019). However, our method has the following characteristics: i) we use axial shift in the horizontal and vertical directions, which focuses more on the information exchange in two directions; ii) the proposed network is built upon Swin transformer and pure MLP-based architecture, where only linear layer is used and BN is replaced by LN; iii) as will be shown in Sec. 4, our network achieves superior performance, which shows the effectiveness of our method.

3.4 Variants of AS-MLP architecture

Figure 1 only shows the tiny version of our AS-MLP architecture. Following DeiT (Touvron et al., 2020) and Swin Transformer (Liu et al., 2021b), we also stack different number of AS-MLP blocks (the number of blocks in four stages) and expand the channel dimension (C in Figure 1) to obtain variants of the AS-MLP architecture of different model sizes, which are AS-MLP-Tiny (AS-MLP-T), AS-MLP-Small (AS-MLP-S) and AS-MLP-Base (AS-MLP-B), respectively. The specific configuration is as follows:

- AS-MLP-T: C = 96, the number of blocks in four stages = {2, 2, 6, 2};

Network	Input Resolution	Top-1 (%)	Params	FLOPs	Throughput (image / s)
CNN-based					
RegNetY-8GF (Radosavovic et al., 2020)	224×224	81.7	39M	8.0G	591.6
RegNetY-16GF (Radosavovic et al., 2020)	224×224	82.9	84M	15.9G	334.7
EfficientNet-B5 (Tan & Le, 2019)	456×456	83.6	30M	9.9G	169.1
Transformer-based					
ViT-B/16 (Dosovitskiy et al., 2020)	384×384	77.9	86M	55.5G	85.9
DeiT-B/16 (Touvron et al., 2020)	224×224	81.8	86M	17.6G	292.3
PVT-Large (Wang et al., 2021)	224×224	82.3	61M	9.8G	-
Swin-T (Liu et al., 2021b)	224×224	81.3	29M	4.5G	755.2
Swin-S (Liu et al., 2021b)	224×224	83.0	50M	8.7G	436.9
Swin-B (Liu et al., 2021b)	224×224	83.3	88M	15.4G	278.1
Swin-B (Liu et al., 2021b)	384×384	84.2	88M	47.0G	84.7
MLP-based					
gMLP-S (Liu et al., 2021a)	224×224	79.4	20M	4.5G	-
ViP-Small/14 (Hou et al., 2021)	224×224	80.5	30M	-	789.0
ViP-Small/7 (Hou et al., 2021)	224×224	**81.5**	25M	-	719.0
AS-MLP-T **(ours)**	224×224	81.3	28M	4.4G	1047.7
Mixer-B/16 (Tolstikhin et al., 2021)	224×224	76.4	59M	11.7G	-
FF (Melas-Kyriazi, 2021)	224×224	74.9	62M	11.4G	-
ResMLP-36 (Touvron et al., 2021a)	224×224	79.7	45M	8.9G	478.7
S^2-MLP-wide (Yu et al., 2021)	224×224	80.0	68M	13.0G	-
S^2-MLP-deep (Yu et al., 2021)	224×224	80.7	51M	9.7G	-
ViP-Medium/7 (Hou et al., 2021)	224×224	82.7	55M	-	418.0
AS-MLP-S **(ours)**	224×224	**83.1**	50M	8.5G	619.5
gMLP-B (Liu et al., 2021a)	224×224	81.6	73M	15.8G	-
ViP-Large/7 (Hou et al., 2021)	224×224	83.2	88M	-	298.0
AS-MLP-B **(ours)**	224×224	**83.3**	88M	15.2G	455.2
AS-MLP-B **(ours)**	384×384	**84.3**	88M	44.6G	179.2

Table 1: The experimental results of different networks on ImageNet-1K. Throughput is measured with the batch size of 64 on a single V100 GPU (32GB). The more complete accuracy and throughput comparisons are listed in Appendix B.2.

- AS-MLP-S: C = 96, the number of blocks in four stages = $\{2, 2, 18, 2\}$;
- AS-MLP-B: C = 128, the number of blocks in four stages = $\{2, 2, 18, 2\}$.

The detailed configurations can be found in Appendix A.1. Table 1 in Sec. 4 shows Top-1 accuracy, model size (Params), computation complexity (FLOPs) and throughput of different variants.

4 EXPERIMENTS

4.1 IMAGE CLASSIFICATION ON THE IMAGENET-1K DATASET

Settings. To evaluate the effectiveness of our AS-MLP, we conduct experiments of the image classification on the ImageNet-1K benchmark, which is collected in (Deng et al., 2009). It contains 1.28M training images and 20K validation images from a total of 1000 classes. We report the experimental results with single-crop Top-1 accuracy. Such a dataset is also utilized to conduct the ablation study experiments. Following the training strategy of Swin Transformer (Liu et al., 2021b), we use an initial learning rate of 0.001 with cosine decay and 20 epochs of linear warm-up. The AdamW (Loshchilov & Hutter, 2017) optimizer is employed to train the whole model for 300 epochs with a batch size of 1024. We also use label smoothing (Szegedy et al., 2016) with a smooth ratio of 0.1 and DropPath (Huang et al., 2016) strategy.

Results. All image classification results are shown in Table 1. We divide all network architectures into CNN-based, Transformer-based and MLP-based architectures. The input resolution is 224×224. Our proposed AS-MLP outperforms other MLP-based architectures when keeping similar parameters and FLOPs. *e.g.*, AS-MLP-S obtains higher top-1 accuracy (83.1%) with less parameters than Mixer-B/16 (Tolstikhin et al., 2021) (76.4%) and ViP-Medium/7 (Hou et al., 2021) (82.7%). Furthermore, it achieves competitive performance compared with transformer-based architectures, *e.g.*, AS-MLP-B (83.3%) *vs*. Swin-B (Liu et al., 2021b) (83.3%), which shows the effectiveness of our AS-MLP architecture.

Results of Mobile Setting. In addition to standard experiments, we also compare the results of AS-MLP in the mobile setting, which is shown in Table 2. We build the Swin (mobile) model and AS-MLP (mobile) model with similar parameters (about 10M). The specific network details can be found in Appendix A.3. The experimental results show that our model significantly exceeds Swin Transformer (Liu et al., 2021b) in the mobile setting (76.05% *vs*. 75.11%).

Method	Top-1 (%)	Top-5 (%)	Params
Swin (mobile)	75.11	92.50	11.2M
AS-MLP (mobile)	76.05	92.81	9.6M

Table 2: The result comparisons of the mobile setting.

4.2 Ablation Study

The core component in the AS-MLP block is the axial shift. We perform the ablation studies for the axial shift operation, including different configurations[3] of the AS-MLP block and its connection types. All ablations are conducted based on the AS-MLP-T, as shown in the setting of Sec. 3.4.

Shift size	Padding method	d.r.	Top-1 (%)	Top-5 (%)
(1, 1)	N/A	1	74.17	91.13
(3, 3)	No / Circular padding	1	81.04	95.37
(3, 3)	Zero padding	1	81.26	95.48
(3, 3)	Reflect padding	1	81.14	95.37
(3, 3)	Replicate padding	1	81.14	95.42
(3, 3)	Zero padding	2	80.50	95.12
(5, 5)	Zero padding	2	80.57	95.12
(5, 5)	Zero padding	1	81.34	95.56
(7, 7)	Zero padding	1	81.32	95.55
(9, 9)	Zero padding	1	81.16	95.45

(a) The impacts of the different configurations of the AS-MLP architecture. d.r. means dilation rate.

Connection type	Structure	Top-1 (%)	Top-5 (%)
Serial	(1, 1) $\rightarrow$ (1, 1)	74.32	91.46
	(3, 3) $\rightarrow$ (3, 3)	81.21	95.42
	(5, 5) $\rightarrow$ (5, 5)	81.28	95.58
	(7, 7) $\rightarrow$ (7, 7)	81.17	95.54
Parallel	(1, 1) + (1, 1)	74.17	91.13
	(3, 3) + (3, 3)	81.26	95.48
	(5, 5) + (5, 5)	81.34	95.56
	(7, 7) + (7, 7)	81.32	95.55

(b) The impacts of the different connection types. '$\rightarrow$' means serial and '+' means parallel.

Table 3: Ablations. Note that (1, 1) in (a) means the shift size = 1, which is equivalent to the model where only channel-mixing MLP is used.

Different Configurations of AS-MLP Block. In order to encourage the information flow from different channels in the spatial dimension, the features from the horizontal shift and the vertical shift are aggregated together in Figure 2. We evaluate the influence of different configurations of AS-MLP block, including shift size, padding method, and dilation rate, which are similar to the configuration of a convolution kernel. All experiments of different configurations are shown in Table 3a. We have three findings as follows: i) 'Zero padding' is more suitable for the design of AS-MLP block than other padding methods[4]; ii) increasing the dilation rate slightly reduces the performance of AS-MLP, which is consistent with CNN-based architecture. Dilation is usually used for semantic segmentation rather than image classification; iii) when expanding the shift size, the accuracy will increase first and then decrease. A possible reason is that the receptive field is enlarged (shift size = 5 or 7) such that AS-MLP pays attention to the global dependencies, but when shift size is 9, the network pays too much attention to the global dependencies, thus neglecting the extraction of local features, which leads to lower accuracy. Therefore, we use the configuration (shift size = 5, zero padding, dilation rate = 1) in all experiments, including object detection and semantic segmentation.

Connection Type. We also compare the different connection types of AS-MLP block, such as serial connection and parallel connection, and the results are shown in Table 3b. Parallel connection consistently outperforms serial connection in terms of different shift sizes, which shows the effectiveness of the parallel connection. When the shift size is 1, the serial connection is better but it is not representative because only channel-mixing MLP is used.

4.3 Object Detection on COCO

Settings. For the object detection and instance segmentation, we employ mmdetection (Chen et al., 2019) as the framework and COCO (Lin et al., 2014) as the evaluation dataset, which consists of 118K training data and 5K validation data. We compare the performance of our AS-MLP with other

[3]There exist amounts of configurations of the block, we cannot enumerate all configurations. Here we only list some general configurations instead.

[4]Since we use circular shift, thus 'No padding' and 'Circular padding' are equivalent.

Backbone	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}	Params	FLOPs
Mask R-CNN (3×)								
ResNet50 (He et al., 2015)	41.0	61.7	44.9	37.1	58.4	40.1	44M	260G
PVT-Small (Wang et al., 2021)	43.0	65.3	46.9	39.9	62.5	42.8	44M	245G
Swin-T (Liu et al., 2021b)	46.0	68.2	50.2	41.6	65.1	44.8	48M	267G
AS-MLP-T **(ours)**	46.0	67.5	50.7	41.5	64.6	44.5	48M	260G
ResNet101 (He et al., 2015)	42.8	63.2	47.1	38.5	60.1	41.3	63M	336G
PVT-Medium (Wang et al., 2021)	44.2	66.0	48.2	40.5	63.1	43.5	64M	302G
Swin-S (Liu et al., 2021b)	48.5	70.2	53.5	43.3	67.3	46.6	69M	359G
AS-MLP-S **(ours)**	47.8	68.9	52.5	42.9	66.4	46.3	69M	346G
Cascade Mask R-CNN (3×)								
DeiT-S (Touvron et al., 2020)	48.0	67.2	51.7	41.4	64.2	44.3	80M	889G
ResNet50 (He et al., 2015)	46.3	64.3	50.5	40.1	61.7	43.4	82M	739G
Swin-T (Liu et al., 2021b)	50.5	69.3	54.9	43.7	66.6	47.1	86M	745G
AS-MLP-T **(ours)**	50.1	68.8	54.3	43.5	66.3	46.9	86M	739G
ResNext101-32 (Xie et al., 2017)	48.1	66.5	52.4	41.6	63.9	45.2	101M	819G
Swin-S (Liu et al., 2021b)	51.8	70.4	56.3	44.7	67.9	48.5	107M	838G
AS-MLP-S **(ours)**	51.1	69.8	55.6	44.2	67.3	48.1	107M	824G
ResNext101-64 (Xie et al., 2017)	48.3	66.4	52.3	41.7	64.0	45.1	140M	972G
Swin-B (Liu et al., 2021b)	51.9	70.9	56.5	45.0	68.4	48.7	145M	982G
AS-MLP-B **(ours)**	51.5	70.0	56.0	44.7	67.8	48.4	145M	961G

Table 4: The object detection and instance segmentation results of different backbones with 3x schedule on the COCO val2017 dataset. The results with 1x schedule (12 epochs) are listed in Appendix B.1.

backbones on COCO. Following Swin Transformer (Liu et al., 2021b), we consider two typical object detection frameworks: Mask R-CNN (He et al., 2017) and Cascade R-CNN (Cai & Vasconcelos, 2018). The training strategies are as follows: optimizer (AdamW), learning rate (0.0001), weight decay (0.05), and batch size (2 imgs/per GPU$\times$8 GPUs). We utilize the typical multi-scale training strategy (Carion et al., 2020; Sun et al., 2021) (the shorter side is between 480 and 800 and the longer side is at most 1333). All backbones are initialized with weights pre-trained on ImageNet-1K and all models are trained with 3x schedule (36 epochs).

Results. The results are shown in Table 4. It is worth noting that we do not compare our method with MLP-Mixer (Tolstikhin et al., 2021) because it uses a fixed spatial dimension for token-mixing MLP, which cannot be transferred to the object detection. As far as we know, we are the first work to apply MLP-based architecture to object detection. Our AS-MLP achieves comparable performance with Swin Transformer in the similar resource limitation. To be specific, Cascade Mask R-CNN + Swin-B achieves 51.9 AP^b with 145M parameters and 982 GFLOPs, and Cascade Mask R-CNN + AS-MLP-B obtains 51.5 AP^b with 145M parameters and 961 GFLOPs. The visualizations of object detection are shown in Appendix C.

4.4 Semantic Segmentation on ADE20K

Settings. Following Swin Transformer (Liu et al., 2021b), we conduct experiments of AS-MLP on the challenging semantic segmentation dataset, ADE20K, which contains 20,210 training images and 2,000 validation images. We utilize UperNet (Xiao et al., 2018) and AS-MLP backbone as our main experimental results. The framework is based on mmsegmentation (Contributors, 2020). The training strategies are as follows: optimizer (AdamW), learning rate (6×10^{-5}), weight decay (0.01), and batch size (2 imgs/per GPU$\times$8 GPUs). We utilize random horizontal flipping, random re-scaling within ratio range [0.5, 2.0] and random photometric distortion as data augmentation. The input image resolution is 512×512, the stochastic depth ratio is set as 0.3 and all models are initialized with weights pre-trained on ImageNet-1K and are trained 160K iterations.

Results. Table 5 shows the performance of our AS-MLP on the ADE20K dataset. Note that we are also the first to apply the MLP-based architecture to semantic segmentation. The results are impressive. With slightly fewer FLOPs, AS-MLP-T achieves better result than Swin-T (46.5 *vs.* 45.8 MS mIoU). For the large model, UperNet + Swin-B has 49.7 MS mIoU with 121M parameters and 1188 GFLOPs, and UperNet + AS-MLP-B has 49.5 MS mIoU with 121M parameters and 1166 GFLOPs, which also shows the effectiveness of our AS-MLP architecture in processing the downstream task. The visualizations of semantic segmentation are shown in Appendix C.

Method	Backbone	val MS mIoU	Params	FLOPs
DANet (Fu et al., 2019a)	ResNet-101	45.2	69M	1119G
DeepLabv3+ (Chen et al., 2018)	ResNet-101	44.1	63M	1021G
ACNet (Fu et al., 2019b)	ResNet-101	45.9	-	-
DNL (Yin et al., 2020)	ResNet-101	46.0	69M	1249G
OCRNet (Yuan et al., 2020)	ResNet-101	45.3	56M	923G
UperNet (Xiao et al., 2018)	ResNet-101	44.9	86M	1029G
OCRNet (Yuan et al., 2020)	HRNet-w48	45.7	71M	664G
DeepLabv3+ (Chen et al., 2018)	ResNeSt-101	46.9	66M	1051G
DeepLabv3+ (Chen et al., 2018)	ResNeSt-200	48.4	88M	1381G
UperNet (Xiao et al., 2018)	Swin-T (Liu et al., 2021b)	45.8	60M	945G
	AS-MLP-T **(ours)**	46.5	60M	937G
UperNet (Xiao et al., 2018)	Swin-S (Liu et al., 2021b)	49.5	81M	1038G
	AS-MLP-S **(ours)**	49.2	81M	1024G
UperNet (Xiao et al., 2018)	Swin-B (Liu et al., 2021b)	49.7	121M	1188G
	AS-MLP-B **(ours)**	49.5	121M	1166G

Table 5: The semantic segmentation results of different backbones on the ADE20K validation set.

4.5 Visualization

We visualize the heatmap of learned features from Swin Transformer and AS-MLP in Figure 4, where the first column shows the image from ImageNet, and the second column shows the activation heatmap of the last layer of Swin transformer (Swin-B). The third, fourth, and fifth columns respectively indicate the response after the horizontal shift (AS-MLP (h)), the vertical shift (AS-MLP (v)) and the combination of both in the last layer of AS-MLP (AS-MLP-B). From Figure 4, one can see that i) AS-MLP can better focus on object regions compared to Swin transformer; ii) AS-MLP (h) can better focus on the vertical part of objects (as shown in the second row) because it shifts feature in the horizontal direction. It is more reasonable because the shift in the horizontal direction can cover the edge of the vertical part, which is more helpful for recognizing the object. Similarly, AS-MLP (v) can better focus on the horizontal part of objects (as shown in the fourth row). These phenomena further show the superiority of AS-MLP.

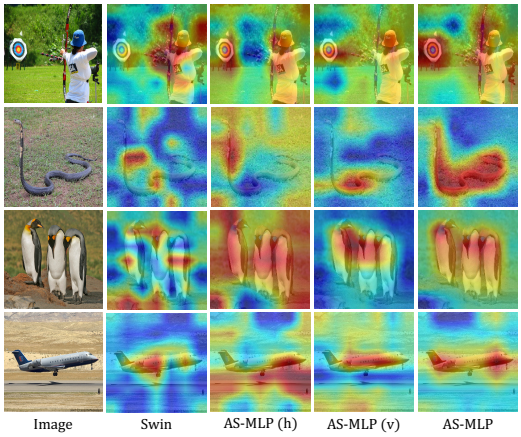


Figure 4: The visualization of features from Swin Transformer and our AS-MLP.

5 Conclusion and Future Work

In this paper, we propose an axial shifted MLP architecture, named AS-MLP, for vision. Compared with MLP-Mixer, we pay more attention to the local features extraction and make full use of the channel interaction between different spatial positions through a simple feature axial shift. With the proposed AS-MLP, we further improve the performance of MLP-based architecture and the experimental results are impressive. Our model obtains 83.3% Top-1 accuracy with 88M parameters and 15.2 GFLOPs on the ImageNet-1K dataset. Such a simple yet effective method outperforms all MLP-based architectures and achieves competitive performance compared to the transformer-based architectures even with slightly lower FLOPs. We are also the first work to apply AS-MLP to the downstream tasks (*e.g.*, object detection and semantic segmentation). The results are also competitive or even better compared to transformer-based architectures, which shows the ability of MLP-based architectures in handling downstream tasks.

For future work, we will investigate the effectiveness of AS-MLP in natural language processing, and further explore the performance of AS-MLP on downstream tasks.

REFERENCES

Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6154–6162, 2018.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision (ECCV)*, pp. 213–229. Springer, 2020.

Chun-Fu Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. *arXiv preprint arXiv:2103.14899*, 2021.

Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 801–818, 2018.

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*, 2021a.

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*, 2021b.

MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. `https://github.com/open-mmlab/mmsegmentation`, 2020.

Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 764–773, 2017.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Yuxin Fang, Xinggang Wang, Rui Wu, Jianwei Niu, and Wenyu Liu. What makes for hierarchical vision transformer? *arXiv preprint arXiv:2107.02174*, 2021.

Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3146–3154, 2019a.

Jun Fu, Jing Liu, Yuhang Wang, Yong Li, Yongjun Bao, Jinhui Tang, and Hanqing Lu. Adaptive context network for scene parsing. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 6748–6757, 2019b.

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

Ben Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet's clothing for faster inference. *arXiv preprint arXiv:2104.01136*, 2021.

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.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2961–2969, 2017.

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.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision (ECCV)*, pp. 646–661. Springer, 2016.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

Yawei Li, Kai Zhang, Jiezhang Cao, Radu Timofte, and Luc Van Gool. Localvit: Bringing locality to vision transformers. *arXiv preprint arXiv:2104.05707*, 2021.

Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7083–7093, 2019.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pp. 740–755. Springer, 2014.

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

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*, 2021b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

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.

Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al. Sparse r-cnn: End-to-end object detection with learnable proposals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14454–14463, 2021.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning (ICML)*, pp. 6105–6114, 2019.

Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, et al. Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021.

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.

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*, 2021a.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

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.

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*, pp. 9127–9135, 2018.

Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 418–434, 2018.

Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Minghao Yin, Zhuliang Yao, Yue Cao, Xiu Li, Zheng Zhang, Stephen Lin, and Han Hu. Disentangled non-local neural networks. In *European Conference on Computer Vision (ECCV)*, pp. 191–207. Springer, 2020.

Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

Tan Yu, Xu Li, Yunfeng Cai, Mingming Sun, and Ping Li. S^2-mlp: Spatial-shift mlp architecture for vision. *arXiv preprint arXiv:2106.07477*, 2021.

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.

Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pp. 173–190. Springer, 2020.

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

Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Zihang Jiang, Qibin Hou, and Jiashi Feng. Deepvit: Towards deeper vision transformer. *arXiv preprint arXiv:2103.11886*, 2021.

Chen Zhu, Wei Ping, Chaowei Xiao, Mohammad Shoeybi, Tom Goldstein, Anima Anandkumar, and Bryan Catanzaro. Long-short transformer: Efficient transformers for language and vision. *arXiv preprint arXiv:2107.02192*, 2021.

A THE ARCHITECTURE DETAILS

A.1 THE DETAILED CONFIGURATIONS OF DIFFERENT ARCHITECTURES

We show the detailed configurations of different architectures in Table 6, where we assume the size of the input image is 224×224. The second column shows the output size of the image after each stage. Following Swin Transformer (Liu et al., 2021b), we use "Concat $n \times n$" to indicate a concatenation of $n \times n$ neighboring features in a patch. "shift size (5, 5)" means that the shift size in the horizontal and vertical directions is 5.

	downsp. rate (output size)	AS-MLP-T	AS-MLP-S	AS-MLP-B
stage 1	4× (56×56)	concat 4×4, 96-d, LN	concat 4×4, 96-d, LN	concat 4×4, 128-d, LN
		[shift size (5, 5), dim 96] × 2	[shift size (5, 5), dim 96] × 2	[shift size (5, 5), dim 128] × 2
stage 2	8× (28×28)	concat 2×2, 192-d , LN	concat 2×2, 192-d , LN	concat 2×2, 256-d , LN
		[shift size (5, 5), dim 192] × 2	[shift size (5, 5), dim 192] × 2	[shift size (5, 5), dim 256] × 2
stage 3	16× (14×14)	concat 2×2, 384-d , LN	concat 2×2, 384-d , LN	concat 2×2, 512-d , LN
		[shift size (5, 5), dim 384] × 6	[shift size (5, 5), dim 384] × 18	[shift size (5, 5), dim 512] × 18
stage 4	32× (7×7)	concat 2×2, 768-d , LN	concat 2×2, 768-d , LN	concat 2×2, 1024-d , LN
		[shift size (5, 5), dim 768] × 2	[shift size (5, 5), dim 768] × 2	[shift size (5, 5), dim 1024] × 2
Params		28M	50M	88M
FLOPs		4.4G	8.5G	15.2G

Table 6: The detailed configurations of different architectures.

A.2 THE COMPUTATIONAL COMPLEXITY OF AS-MLP ARCHITECTURE

In this section, we show the specific computational complexity in each layer of AS-MLP architecture. The symbol definition is firstly given as follows. An input image: $I \in \mathbb{R}^{3\times H\times W}$; patch size (p, p); the number of blocks in four stages: $\{n_1, n_2, n_3, n_4\}$; Channel dimension C; MLP ratio: r. The specific computational complexity is shown in Table 7, where only convolution operation is computed.

	Stage 1		Stage 2	
	Linear embedding	AS-MLP block	Patch merging	AS-MLP block
Params	$3Cp^2$	$(4+2r)C^2n_1$	$8C^2$	$(4+2r)4C^2n_2$
FLOPs	$3Cp^2\frac{H}{p}\frac{W}{p}$	$(4+2r)C^2\frac{H}{p}\frac{W}{p}n_1$	$8C^2\frac{H}{2p}\frac{W}{2p}$	$(4+2r)4C^2\frac{H}{2p}\frac{W}{2p}n_2$
	Stage 3		Stage 4	
	Patch merging	AS-MLP block	Patch merging	AS-MLP block
Params	$32C^2$	$(4+2r)16C^2n_3$	$128C^2$	$(4+2r)64C^2n_4$
FLOPs	$32C^2\frac{H}{4p}\frac{W}{4p}$	$(4+2r)16C^2\frac{H}{4p}\frac{W}{4p}n_3$	$128C^2\frac{H}{8p}\frac{W}{8p}$	$(4+2r)64C^2\frac{H}{8p}\frac{W}{8p}n_4$

Table 7: The computational complexity of the AS-MLP Architecture.

Backbone	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}	Params	FLOPs
Mask R-CNN (1×)								
ResNet50 (He et al., 2015)	38.0	58.6	41.4	34.4	55.1	36.7	44M	260G
PVT-Small (Wang et al., 2021)	40.4	62.9	43.8	37.8	60.1	40.3	44M	245G
Swin-T (Liu et al., 2021b)	43.7	**66.6**	47.7	39.8	**63.3**	42.7	48M	267G
AS-MLP-T **(ours)**	**44.0**	66.0	**48.5**	**40.0**	62.8	**43.1**	48M	260G
ResNet101 (He et al., 2015)	40.4	61.1	44.2	36.4	57.7	38.8	63M	336G
PVT-Medium (Wang et al., 2021)	42.0	64.4	45.6	39.0	61.6	42.1	64M	302G
AS-MLP-S **(ours)**	**46.7**	**68.8**	**51.4**	**42.0**	**65.6**	**45.2**	69M	346G
Cascade Mask R-CNN (1×)								
ResNet50 (He et al., 2015)	46.3	64.3	50.5	40.1	61.7	43.4	82M	739G
Swin-T (Liu et al., 2021b)	48.1	67.1	52.2	41.7	64.4	45.0	86M	745G
AS-MLP-T **(ours)**	**48.4**	**67.1**	**52.6**	**42.0**	**64.5**	**45.3**	86M	739G
AS-MLP-S **(ours)**	**50.5**	**69.4**	**54.7**	**43.7**	**66.9**	**47.3**	107M	824G
AS-MLP-B **(ours)**	**51.1**	**70.0**	**55.6**	**44.2**	**67.4**	**47.8**	145M	961G

Table 8: The object detection and instance segmentation results of different backbones with 1x schedule on the COCO val2017 dataset. Mask R-CNN and Cascade Mask R-CNN frameworks are employed.

A.3 The Network Details in the Mobile Setting

In addition to AS-MLP-T, AS-MLP-S, and AS-MLP-B, we also design AS-MLP in the mobile setting. For a fair comparison, we modify the Swin Transformer correspondingly to adopt to the mobile setting. The configurations are as follow:

- Swin (mobile): C = 64, the number of blocks in four stages = {2, 2, 2, 2}, the number of heads = {2, 4, 8, 16};
- AS-MLP (mobile): C = 64, the number of blocks in four stages = {2, 2, 2, 2};

B More Experiments

B.1 More Experimental Results on COCO

Table 4 lists the object detection and instance segmentation results of different backbones with 3x schedule (36 epochs). For a complete comparison, we also conduct the experiments with 1x schedule (12 epochs). The results are shown in Table 8. Our AS-MLP-T outperforms Swin-T (Liu et al., 2021b) under the Mask R-CNN (44.0 vs. 43.7 AP^b) and Cascade Mask R-CNN (48.4 vs. 48.1 AP^b) frameworks.

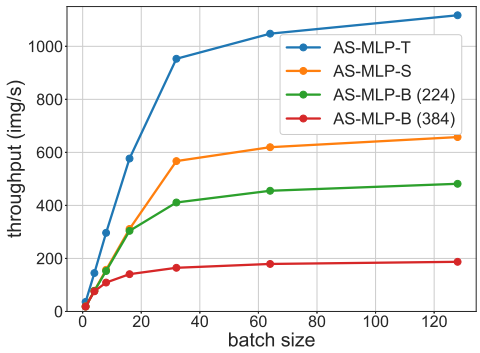


Figure 5: The throughput curve when the batch size is 1, 4, 8, 16, 32, 64, 128, respectively.

Network	Input Resolution	Top-1 (%)	Params	FLOPs	Throughput (image / s)
CNN-based					
RegNetY-8GF (Radosavovic et al., 2020)	224×224	81.7	39M	8.0G	591.6
RegNetY-16GF (Radosavovic et al., 2020)	224×224	82.9	84M	15.9G	334.7
EfficientNet-B3 (Tan & Le, 2019)	300×300	81.6	12M	1.8G	732.1
EfficientNet-B5 (Tan & Le, 2019)	456×456	83.6	30M	9.9G	169.1
Transformer-based					
ViT-B/16 (Dosovitskiy et al., 2020)	384×384	77.9	86M	55.5G	85.9
DeiT-B/16 (Touvron et al., 2020)	224×224	81.8	86M	17.6G	292.3
PVT-Large (Wang et al., 2021)	224×224	82.3	61M	9.8G	-
CPVT-B (Chu et al., 2021b)	224×224	82.3	88M	17.6G	285.5
TNT-B (Han et al., 2021)	224×224	82.8	66M	14.1G	-
T2T-ViT$_t$-24 (Yuan et al., 2021)	224×224	82.6	65M	15.0G	-
CaiT-S36 (Touvron et al., 2021b)	224×224	83.3	68M	13.9G	-
Swin-T (Liu et al., 2021b)	224×224	81.3	29M	4.5G	755.2
Swin-S (Liu et al., 2021b)	224×224	83.0	50M	8.7G	436.9
Swin-B (Liu et al., 2021b)	224×224	83.3	88M	15.4G	278.1
Nest-B (Zhang et al., 2021)	224×224	83.8	68M	17.9G	235.8
Container (Gao et al., 2021)	224×224	82.7	22M	8.1G	347.8
Swin-B (Liu et al., 2021b)	384×384	84.2	88M	47.0G	84.7
MLP-based					
gMLP-S (Liu et al., 2021a)	224×224	79.4	20M	4.5G	-
ViP-Small/14 (Hou et al., 2021)	224×224	80.5	30M	-	789.0
ViP-Small/7 (Hou et al., 2021)	224×224	**81.5**	25M	-	719.0
AS-MLP-T **(ours)**	224×224	81.3	28M	4.4G	1047.7
Mixer-B/16 (Tolstikhin et al., 2021)	224×224	76.4	59M	11.7G	-
FF (Melas-Kyriazi, 2021)	224×224	74.9	62M	11.4G	-
ResMLP-36 (Touvron et al., 2021a)	224×224	79.7	45M	8.9G	478.7
S^2-MLP-wide (Yu et al., 2021)	224×224	80.0	68M	13.0G	-
S^2-MLP-deep (Yu et al., 2021)	224×224	80.7	51M	9.7G	-
ViP-Medium/7 (Hou et al., 2021)	224×224	82.7	55M	-	418.0
AS-MLP-S **(ours)**	224×224	**83.1**	50M	8.5G	619.5
gMLP-B (Liu et al., 2021a)	224×224	81.6	73M	15.8G	-
ViP-Large/7 (Hou et al., 2021)	224×224	83.2	88M	-	298.0
AS-MLP-B **(ours)**	224×224	**83.3**	88M	15.2G	455.2
AS-MLP-B **(ours)**	384×384	**84.3**	88M	44.6G	179.2

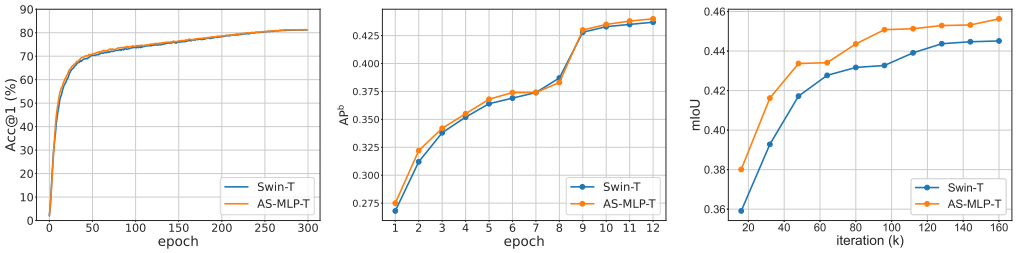
Table 9: The complete experimental results of different networks on ImageNet-1K. Throughput is measured with the batch size of 64 on a single V100 GPU (32GB).

B.2 The Complete Classification Accuracy and Throughput Comparison

Table 9 shows the complete accuracy comparison with other state-of-the-art architectures on the ImageNet-1K dataset. In addition, Table 1 shows the throughput results of the AS-MLP architecture measured with the batch size 64 on a single V100 GPU (32GB). In order to make a fair comparison with other papers, we also conduct a thorough evaluation of throughput. The results are shown in Figure 5, where we list the throughputs when the batch size is 1, 4, 8, 16, 32, 64, 128, respectively.

B.3 Evaluation Accuracy

In Figure 6, we visualize the evaluation accuracy of AS-MLP and Swin transformer on the ImageNet, COCO and ADE20K datasets during training. For image classification on ImageNet, AS-MLP-T keeps pace with Swin-T in each epoch and they finally converge to the similar accuracy (81.3 vs. 81.3). For object detection and semantic segmentation on COCO and ADE20K, we can see that AS-MLP-T achieves better performance than Swin-T in the early stage, and keeps winning during the training process.



(a) Accuracy of AS-MLP. (b) AP^b of AS-MLP. (c) mIoU of AS-MLP.

Figure 6: The evaluation accuracy of AS-MLP and Swin transformer on the ImageNet, COCO and ADE20K datasets during training.

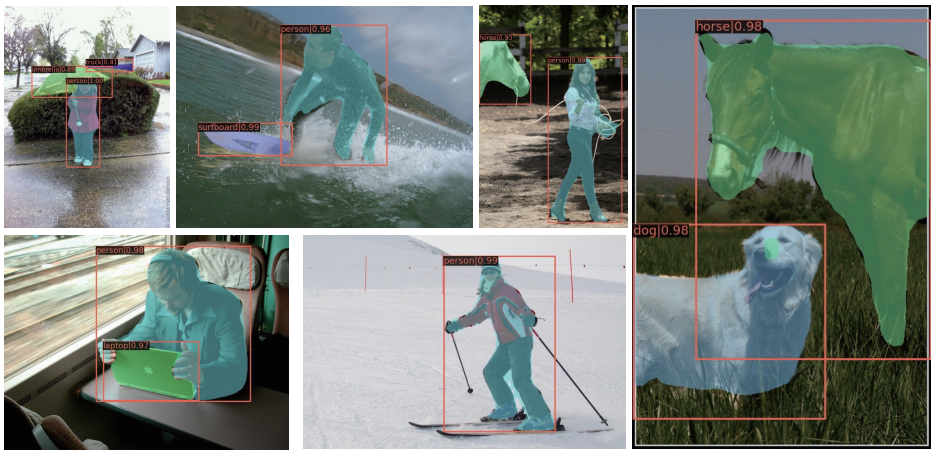


Figure 7: The object detection and instance segmentation results on the COCO dataset.

C THE VISUALIZATION OF RESULTS ON COCO AND ADE20K

We visualize the object detection and instance segmentation results on the COCO dataset in Figure 7, where the Cascade Mask R-CNN model with the AS-MLP-T backbone is used. We also visualize the semantic segmentation results on the ADE20K dataset in Figure 8, where we utilize the UperNet model with the AS-MLP-T backbone. The object can be detected and segmented correctly.

Figure 8: The semantic segmentation results on the ADE20K dataset.