An Image Patch is a Wave: Phase-Aware Vision MLP

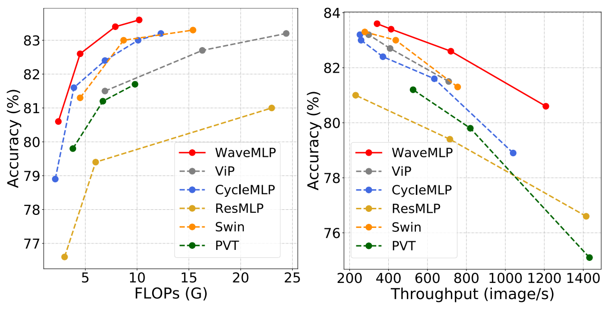
Yehui Tang[1,2], Kai Han[2], Jianyuan Guo[2,3], Chang Xu[3],
Yanxi Li[2,3], Chao Xu[1], Yunhe Wang[2]
[1]Key Lab of Machine Perception (MOE), Dept. of Machine Intelligence, Peking University.
[2]Huawei Noah's Ark Lab.
[3]School of Computer Science, Faculty of Engineering, University of Sydney.
yhtang@pku.edu.cn, {kai.han, yunhe.wang, an.xiao, yiping.deng}@huawei.com,
c.xu@sydney.edu.au, xuchao@cis.pku.edu.cn.

Abstract

Different from traditional convolutional neural network (CNN) and vision transformer, the multilayer perceptron (MLP) is a new kind of vision model with extremely simple architecture that only stacked by fully-connected layers. An input image of vision MLP is usually split into multiple tokens (patches), while the existing MLP models directly aggregate them with fixed weights, neglecting the varying semantic information of tokens from different images. To dynamically aggregate tokens, we propose to represent each token as a wave function with two parts, amplitude and phase. Amplitude is the original feature and the phase term is a complex value changing according to the semantic contents of input images. Introducing the phase term can dynamically modulate the relationship between tokens and fixed weights in MLP. Based on the wave-like token representation, we establish a novel Wave-MLP architecture for vision tasks. Extensive experiments demonstrate that the proposed Wave-MLP is superior to the state-of-the-art MLP architectures on various vision tasks such as image classification, object detection and semantic segmentation.

1. Introduction

In computer vision, convolutional neural networks (CNNs) have been the mainstream architectures for a long time [15, 21, 30]. It is challenged by the recent works [9, 27, 39], in which a standard Transformer [38] model can also work well on various computer vision tasks, such as image classification, object detection and semantic segmentation. Considering the high complexity of self-attention modules in the vision transformer, more simple architectures (*e.g.*, MLP-Mixer [35], ResMLP [36]) stacking only multi-layer perceptrons (MLPs) have attracted much attention. Compared with CNNs and Transformers, these vision MLP architectures involve less inductive bias and have potential to be applied on more diverse tasks.



(a) Accuracy w.r.t. FLOPs. (b) Accuracy w.r.t. throughput.

Figure 1. Performance comparison between the proposed Wave-MLP and existing architectures. Top-1 accuracies on ImageNet are reported.

Taking a sequence of image patches (tokens) as input, MLP-like models [35, 36] mainly contain two separable blocks, *i.e.*, channel-mixing MLP and token-mixing MLP, both composing of full-connected layers and activation functions. The channel-mixing MLP transforms feature of each token and the token-mixing MLP tries to aggregate information from different tokens. By stacking these two types of MLP block alternatively, the simple MLP architecture could have sufficient capacity to extract features and achieve good performance on vision tasks.

However, the performance of MLP architecture is still inferior to that of SOTA Transformer and CNN architectures. We point out that one of the bottlenecks for vision MLP lies in its manner of aggregating different tokens, *i.e.*, mixing different tokens with fixed weights of fully-connected layers. Recall that Transformer [9, 38] aggregates tokens with weights dynamically adjusted by the attention mechanism. The inner products between different tokens are calculated and tokens with higher similarities tend to have larger weights in the aggregation process of each other. However, the existing vision MLP models aggregate different tokens with fixed weights. The same weights are used for tokens from different input images, neglecting differences in semantic information of various tokens, which may not aggregate tokens well for all the input images.

Different from Transformer that delicately designs the attention mechanism, we aim to improve the representation way of tokens for dynamically aggregating them according to their semantic contents. Actually, in the physical world especially quantum mechanics, an entity (*e.g.*, electron, photon) is usually represented by a wave function containing both amplitude and phase [1, 11, 16]. The amplitude part measures the maximum intensity of a wave and the phase part modulates the intensity by indicating the location of a point in the wave period. Inspired by them, we describe each token as a wave to realize the dynamic aggregation procedure of tokens.

In this paper, we present a novel vision MLP architecture (dubbed as Wave-MLP), which takes each token as a wave with both amplitude and phase. The amplitude is the real-value feature representing the content of each token, while the phase term is a unit complex value modulating the relationship between tokens and fixed weights in MLP. The phase difference between these wave-like tokens affects their aggregated output and tokens with close phases tend to enhance each other. Considering that tokens from different input images contain diverse semantic contents, we use a simple module to dynamically estimate the phase for each token. With tokens equipped with amplitude and phase information, we introduce a phase-aware token mixing module (PATM in Figure 2) to aggregate these tokens. The whole Wave-MLP architecture is constructed by stacking the PATM module and channel-mixing MLP, alternately.

The proposed Wave-MLP architecture shows a large superiority to the existing architectures (shown in Figure 1). For example, the proposed Wave-MLP-S model achieves 82.6% top-1 accuracy on ImageNet with 4.5G FLOPs, which significantly surpasses Swin-T [27] with 81.3% accuracy and 4.5G FLOPs. Besides, Wave-MLP also achieves strong performance on the dense prediction tasks such as object detection and semantic segmentation.

The paper is organized as follows: Section 2 briefly reviews the existing works about designing model architectures, and Section 3 discusses the proposed Wave-MLP architecture detailedly. In Section 4, we empirically investigate the method's effectiveness on multiple vision tasks and make conclusions in Section 5.

2. Related Work

CNN-based Architectures. Convolutional neural networks (CNNs) have been the mainstream in computer vision for a long time. The prototype of CNN model is presented in [22] for the document recognition task, where convolution is the core operation. Beginning with great success of AlexNet [21] in ILSVRC 2012, various architectures such as GoogleNet [34], VGGNet [32], ResNet [15], RegNet [30] are developed. Though the model architectures become more complex for pursuing high performance, the core operations have always the convolution and its variants. The occurrence of new computing paradigm such as vision Transformer [38], vision MLP [35] bring new blood to the area of architecture design in computer vision.

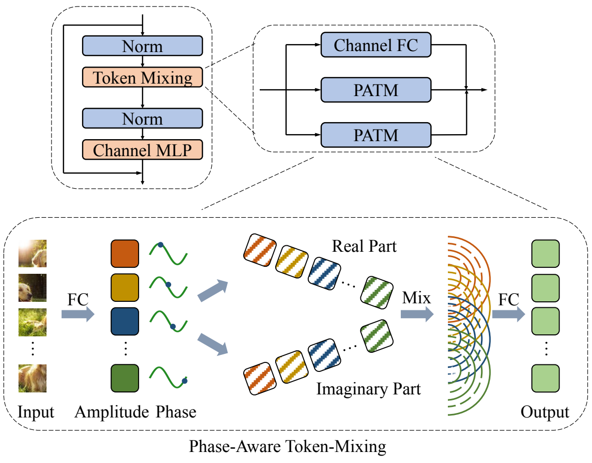


Figure 2. The diagram of a block in the Wave-MLP architecture.

Transformer-based Architectures. Transformer [38] is originally proposed for the natural language processing (NLP) tasks such as language modeling and machine translation. Dosovitskiy *et al.* [9] introduce it to computer vision and achieve excellent performance on image classification tasks especially when training data are extremely sufficient. Touvron *et al.* [37] refine the training recipe and present a teacher-student strategy specific to transformers, which produce competitive transformer models trained on ImageNet from scratch. Then many works explore the architecture design of vision transformers [5, 13, 40, 41]. For example, Han *et al.* [13] present a nested transformer architecture to capture global and local information simultaneously. To be compatible with the dense prediction task such as object detection and semantic segmentation, hierarchical architectures are adopted in [9, 17, 39], which splits the whole architecture into multiple stages and reduce the spatial resolution stage-wisely. Swin Transformer [27] extract representation with shifted windows and limit the self-attention in local regions. Compared with the self-attention in [9] connecting all the tokens in a layer, the shifted window operation is more efficient.

MLP-based Architectures. Recently, MLP-like architectures composing of fully connected layers and non-linear activation functions have been paid much attention. Though they have more simple architectures and introduce less inductive bias, their performances are still comparable with SOTA models. The MLP-Mixer model [35] uses two type of MLP layers, *i.e.*, channel-mixing MLP and token-mixing MLPs. The channel-MLP extract features for each tokens while the token-mixing MLPs capture the spatial information. Touvron *et al.* [36] present a similar architecture and replace the Layer Normalization [2] with the simpler affine transformation. Liu *et al.* [26] empirically validate that MLP architectures with gating can achieve similar performance with Transformers in both language and vision tasks. To preserve the positional information of input images, Hou *et al.* [18] keep the 2D shape of the input image and extract features by permuting them along width and height, respectively. Based on MLP-Mixer, Yu *et al.* [42] replace the token-mixing MLP with a spatial shift operation for capturing the local spatial information, which is also computationally efficient. Currently, Lian *et al.* [23] propose to shift tokens along two orthogonal directions to obtain an axial receptive field. Chen *et al.* [6] propose a cycle fully-connected layer, which mixes information along the spatial and channel dimensions simultaneously and can cope with variable input image scales. Different from them, we explore how to represent the tokens in vision MLP and take each token as a wave with both amplitude and phase. Empirically, we find that our Wave-MLP architecture achieves a better trade-off between accuracy and computational cost compared with the existing architectures.

3. Method

In this section, we discuss the proposed Wave-MLP models detailedly. After introducing the vision MLP architecture briefly, we present the phase-aware token mixing module (PATM), which represents each token as a wave and aggregate them by considering amplitude and phase simultaneously. At last, we describe the blocks in Wave-MLP and architecture variants with different computational costs.

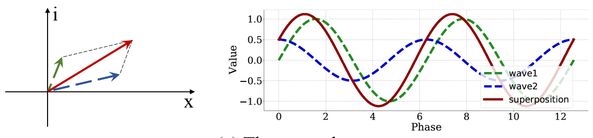
3.1. Preliminaries

A MLP-like model is a neural architecture mainly composed of full-connected layers and non-linear activation functions. For the vision MLP, it first splits an image into multiple patches (also referred to as tokens) and then extract their features with two components, *i.e.*, channel-FC and token-FC described as following.

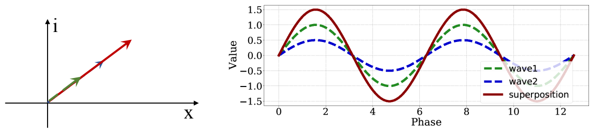
Denote the intermediate feature containing n tokens as $Z = [\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_n]$, where each token $\boldsymbol{z}_j$ is a d-dimension vector. The channel-FC is formulated as:

$$\text{Channel-FC}(\boldsymbol{z}_j, W^c) = W^c \boldsymbol{z}_j, j = 1, 2, \cdots n, \quad (1)$$

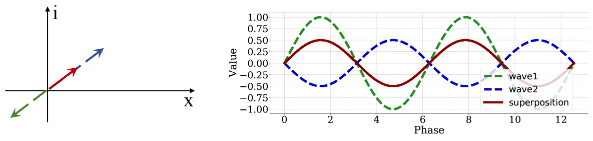
where W^c is the weight with learnable parameters. The channel-FC operates on each token independently to extract



(a) The general case.



(b) Two waves have the same phase.



(c) Two waves have the opposite phase.

Figure 3. The interaction between two waves with different phase. The left is the superposition of two waves in the complex-value domain, while the right shows how their projections along the real axis varies w.r.t. the phase. The dashed lines denote two waves with different initial phase, and the solid line is their superposed wave.

their features. To enhance the transformation ability, multiple channel-FC layers are usually stacked together with the non-linear activation function, which constructs a channel-mixing MLP.

To aggregate information from different tokens, the token-FC operation is required, *i.e.*,

$$\text{Token-FC}(Z, W^t)_j = \sum_k W^t_{jk} \odot \boldsymbol{z}_k, j = 1, 2, \cdots n, \quad (2)$$

where W^t is the token-mixing weight, $\odot$ denotes element-wise multiplication and the subscript j indicates the j-th output token. The token-FC operation tries to capture the spatial information by mixing features from different tokens. In the existing MLP-like models such as MLP-Mixer [35], ResMLP [36], a token-mixing MLP is also constructed by stacking the token-FC layers and activation functions. Such a simple token-mixing MLP with fixed weights neglects the varying semantic contents of tokens from different input images, which is a bottleneck restricting the representation ability of MLP-like architectures.

3.2. Phase-Aware Token Mixing

To dynamically modulate the relationship between tokens and fixed weights in MLP for aggregating tokens more properly, we take each token as a wave with both amplitude and phase. We firstly discuss the wave-like representation of a token and then present the phase-aware token mixing module (PATM) for aggregating tokens.

Wave-like representation. In Wave-MLP, a token is represented as a wave $\tilde{\boldsymbol{z}}_j$ with both amplitude and phase information, *i.e.*,

$$\tilde{z}_j = |\boldsymbol{z}_j| \odot e^{i\boldsymbol{\theta}_j}, j = 1, 2, \cdots, n, \quad (3)$$

where i is the imaginary unit satisfying $i^2 = -1$. $|\cdot|$ denotes the absolute value operation and $\odot$ is element-wise multiplication. The amplitude $|\boldsymbol{z}_j|$ is a real-value feature representing the content of each token. $e^{i\boldsymbol{\theta}_j}$ is a periodic function whose elements always have the unit norm. $\boldsymbol{\theta}_j$ indicates the phase, which is the current location of token within a wave period. With both amplitude and phase, each token $\tilde{z}_j$ is represented in the complex-value domain.

When aggregating different tokens, the phase term $\boldsymbol{\theta}_j$ modulates their superposition modes. Supposing $\tilde{z}_r = \tilde{z}_1 + \tilde{z}_2$ is the aggregated results of wave-like token $\tilde{z}_1$, $\tilde{z}_2$ [1], its amplitude $|\boldsymbol{z}_r|$ and phase $\boldsymbol{\theta}_r$ can be calculated as following:

$$|\boldsymbol{z}_r| = \sqrt{|\boldsymbol{z}_i|^2 + |\boldsymbol{z}_j|^2 + 2|\boldsymbol{z}_i| \odot |\boldsymbol{z}_j| \odot \cos(\boldsymbol{\theta}_j - \boldsymbol{\theta}_i)}, \quad (4)$$

$$\boldsymbol{\theta}_r = \boldsymbol{\theta}_i + \operatorname{atan2}(|\boldsymbol{z}_j| \odot \sin(\boldsymbol{\theta}_j - \boldsymbol{\theta}_i), \\ |\boldsymbol{z}_i| + |\boldsymbol{z}_j| \odot \sin(\boldsymbol{\theta}_j - \boldsymbol{\theta}_i)), \quad (5)$$

where $\operatorname{atan2}(x, y)$ is the two-argument arctangent function. As shown in the above equations, the phase difference $|\boldsymbol{\theta}_j - \boldsymbol{\theta}_i|$ between two tokens has a large impact on the amplitude of aggregated result $\boldsymbol{z}_r$. An intuitive diagram is shown in Figure 3. The left is the superposition of two waves in the complex-value domain, while the right shows how their projections along the real axis varies w.r.t. the phase. When two tokens have the same phase ($\boldsymbol{\theta}_j = \boldsymbol{\theta}_i + 2\boldsymbol{\pi} * m, m \in [0, \pm 2, \pm 4, \cdots]$), they will be enhanced by each other, *i.e.*, $|\boldsymbol{z}_r| = |\boldsymbol{z}_i| + |\boldsymbol{z}_j|$ (Figure 3 (b)). For the opposite phase ($\boldsymbol{\theta}_j = \boldsymbol{\theta}_i + \boldsymbol{\pi} * m, m \in [\pm 1, \pm 3, \cdots]$), the resultant wave will be weakened ($|\boldsymbol{z}_r| = ||\boldsymbol{z}_i| - |\boldsymbol{z}_j||$). In other cases, their interaction is more complex but whether they will be enhanced or weakened also depends on the phase difference (Figure 3 (a)). Note that the classical representation strategy with only real-value feature is a special case of Eq 3, whose phase $\boldsymbol{\theta}_j$ is only the integer multiple of $\boldsymbol{\pi}$.

Amplitude. To get the wave-like tokens in Eq. 3, both amplitude and phase information are required. The amplitude $|\boldsymbol{z}_i|$ is similar to the real-value feature in the traditional model, expect for an absolute operation. Actually, the element-wisely absolute operation can be absorbed into the phase term, *i.e.*, $|z_{j,t}|e^{i\theta_{j,t}} = z_{j,t}e^{i\theta_{j,t}}$ if $z_{j,t} > 0$, and $|z_{j,t}|e^{i\theta_{j,t}} = z_{j,t}e^{i(\theta_{j,t}+\pi)}$ otherwise, where $z_{j,t}$ and $\theta_{j,t}$ denote the t-th element in $\boldsymbol{z}_j$ and $\boldsymbol{\theta}_j$. Thus we remove the absolute operation in practical implementation for simplicity. Denoting $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]$ as the input of a block, we get the token's amplitude $\boldsymbol{z}_j$ by a plain channel-FC operation, *i.e.*,

$$\boldsymbol{z}_j = \text{Channel-FC}(\boldsymbol{x}_j, W^c), j = 1, 2, \cdots, n. \quad (6)$$

[1]Without affecting the conclusion, the aggregating weights are set to 1 for simplicity.

Phase. Recalling that the phase indicates the current location of token in a period of wave, we discuss different strategy to generate phases as following. The simplest strategy ('static phase') is to represent the phase $\boldsymbol{\theta}_j$ of each tokens with fixed parameters, which can be learned in the training process. Though the static phase can distinguish different tokens, it neglects the diversity of different input images either.

To capture the particular attributes for each input respectively, we use an estimation module Θ to generate the phase information according to input features $\boldsymbol{x}_j$, *i.e.*, $\boldsymbol{\theta}_j = \Theta(\boldsymbol{x}_j, W^\theta)$, where W^θ denotes the learnable parameters. Considering that simplicity is an important characteristic of MLP-like architectures, complex operations are undesirable. Thus we also adopt the simple channel-FC in Eq. 1 as the phase estimation module. The estimation module can also be constructed with other formulations, whose impact on the model performance is empirically investigated in Table 6 of Section 4.4.

Token aggregation. In Eq. 3, the wave-like tokens are represented in the complex domain. To embed it in a general MLP-like architecture, we unfold it with Euler's formula and represent it with real part and imaginary part, *i.e.*,

$$\tilde{z}_j = |\boldsymbol{z}_j| \odot \cos \boldsymbol{\theta}_j + i|\boldsymbol{z}_j| \odot \sin \boldsymbol{\theta}_j, j = 1, 2, \cdots, n. \quad (7)$$

In the above equation, a complex-value token is represented as two real-value vectors, indicating the real and imaginary parts, respectively. Different tokens $\tilde{z}_j$ are then aggregated with the token-FC operation (Eq. 2), *i.e.*,

$$\tilde{\boldsymbol{o}}_j = \text{Token-FC}(\tilde{Z}, W^t)_j, j = 1, 2, \cdots, n, \quad (8)$$

where $\tilde{Z} = [\tilde{z}_1, \tilde{z}_2, \cdots, \tilde{z}_n]$ denotes all the wave-like tokens in a layer. In Eq. 8, different tokens interact with each other considering both the amplitude and phase information. The output $\tilde{\boldsymbol{o}}_j$ is the complex-value representation of the aggregated feature. Following the common quantum measurement methods [3, 19] which project a quantum state with complex-value representation to the real-value observable, we get the real-value output $\boldsymbol{o}_j$ by summing the real and imaginary part of $\tilde{\boldsymbol{o}}_j$ with weights. Combined with Eq. 8, the output $\boldsymbol{o}_j$ can be obtained as:

$$\boldsymbol{o}_j = \sum_k W^t_{jk} \boldsymbol{z}_k \odot \cos \boldsymbol{\theta}_k + W^i_{jk} \boldsymbol{z}_k \odot \sin \boldsymbol{\theta}_k, \\ j = 1, 2, \cdots, n, \quad (9)$$

where W^t, W^i are both learnable weights. In the above equation, the phase $\boldsymbol{\theta}_k$ adjusts dynamically according to the semantic content of input data. Besides the fixed weights, the phases also modulate the aggregating process of different tokens.

In vision MLP, we construct a phase-aware token mixing module (PATM) to conduct the above token aggregating procedure, which is shown in Figure 2. Given the input feature $\boldsymbol{x}_j$, the amplitude $\boldsymbol{z}_j$ and phase $\boldsymbol{\theta}_j$ are generated with the channel-FC and phase estimation module, respectively. Then the wave-like token $\tilde{\boldsymbol{z}}_j$ is unfolded with Eq. 7 and aggregated to get output feature $\boldsymbol{o}_j$ (Eq. 9). The final module output is obtained by transforming $\boldsymbol{o}_j$ with another channel-FC to enhance the representation capacity.

3.3. Wave-MLP Block

A basic unit in the proposed Wave-MLP mainly contains two blocks, channel-mixing MLP and phase-aware token-mixing block (Figure 2). The channel-mixing MLP is stack by two channel-FC layers (Eq. 1) and non-linear activation functions, which extracts features for each token. The token-mixing block composes of the proposed PATM modules, aggregating different tokens by considering both amplitude and phase information.

To be more compatible with computer vision tasks, we preserve the 2D spatial shape of input image by using feature maps with shape $H \times W \times C$, which H, W, C are the height, width and channel's number, respectively. This is a successful practice widely used in recent vision transformer architectures (*e.g.*, PVT [39], Swin-Transformer [27]). There are two parallel PATM modules, which aggregate spatial information along high and width dimensions, respectively. In the traditional MLP-Mixer [35], each token-FC layer connects all tokens together, whose dimension depends on specific input size. Thus it is not compatible with the dense prediction tasks (*e.g.*, object detection and semantic segmentation) with varying sizes of input images. To address this issue, we use a simple strategy that restricts the FC layers only connect tokens within a local window. The empirical investigation of the window size is shown in Table 7 of Section 4.4. Besides the PATM modules, another channel-FC connecting the input and output directly is also used to preserve the original information. The final output of the block is the summation of these three branches.

The whole model is constructed by stacking phase-aware token-mixing blocks, channel-mixing MLPs, and normalization layers, alternately. To produce hierarchical features, we split the architecture into 4 stages, which reduces the size of feature maps and increases the number of channels stage-wisely. By varying the width and depth of model, we develop 4 models with different parameters and computational costs, denoted as Wave-MLP-T, Wave-MLP-S, Wave-MLP-M, Wave-MLP-B, sequentially. The detailed configures of these models can be found in the supplemental material.

4. Experiments

In this section, we empirically investigate the proposed Wave-MLP architecture on multiple tasks, containing image classification, object detection and semantic segmentation. Wave-MLP is firstly compared with the existing vision MLPs, vision Transformers and CNNs on ImageNet [8] for image classification. Then it is used as the backbone of two detectors (RetinaNet [24] and Mask R-CNN [14]) for object detection and instance segmentation on COCO dataset [25]. As for semantic segmentation, the widely used semantic FPN [20] on ADE20K [46] is adopted. Finally, ablation studies are conducted to verify the effectiveness of each component.

Table 1. Comparison of the proposed Wave-MLP architecture with existing vision MLP models on ImageNet.

Model	Params.	FLOPs	Throughput (image / s)	Top-1 acc. (%)
EAMLP-14 [12]	30M	-	771	78.9
EAMLP-19 [12]	55M	-	464	79.4
Mixer-B/16 [35]	59M	12.7G	-	76.4
ResMLP-S12 [36]	15M	3.0G	1415	76.6
ResMLP-S24 [36]	30M	6.0G	715	79.4
ResMLP-B24 [36]	116M	23.0G	231	81.0
gMLP-S [26]	20M	4.5G	-	79.6
gMLP-B [26]	73M	15.8G	-	81.6
S^2-MLP-wide [42]	71M	14.0G	-	80.0
S^2-MLP-deep [42]	51M	10.5G	-	80.7
ViP-Small/7 [18]	25M	6.9G	719	81.5
ViP-Medium/7 [18]	55M	16.3G	418	82.7
ViP-Large/7 [18]	88M	24.4G	298	83.2
AS-MLP-T [23]	28M	4.4G	862	81.3
AS-MLP-S [23]	50M	8.5G	473	83.1
AS-MLP-B [23]	88M	15.2G	308	83.3
CycleMLP-B1 [6]	15M	2.1G	1040	78.9
CycleMLP-B2 [6]	27M	3.9G	635	81.6
CycleMLP-B3 [6]	38M	6.9G	371	82.4
CycleMLP-B4 [6]	52M	10.1G	259	83.0
CycleMLP-B5 [6]	76M	12.3G	253	83.2
Wave-MLP-T* (ours)	15M	2.1G	1257	**80.1**
Wave-MLP-T (ours)	17M	2.4G	1208	**80.6**
Wave-MLP-S (ours)	30M	4.5G	720	**82.6**
Wave-MLP-M (ours)	44M	7.9G	413	**83.4**
Wave-MLP-B (ours)	63M	10.2G	341	**83.6**

4.1. Image Classification on ImageNet

Settings. We conduct image classification experiments on the benchmark dataset ImageNet [8], which contains 1.28M training images and 50k validation images from 1000 classes. For a fair comparison, we use the same training strategy as [37]. Specially, the model is trained for 300 epochs with AdamW [28] optimizer, whose learning rate is initialized as 0.001 and declines with a cosine decay strategy. The batchsize and weight decay are set to 1024 and 0.05, respectively. We use the common data augmentation strategies following [37], containing Mixup [45], CutMix [44] and Rand-Augment [7]. At the inference phase,

Table 2. Comparison of the proposed Wave-MLP architecture with SOTA models on ImageNet.

Model	Family	Params.	FLOPs	Throughput (image / s)	Top-1 acc. (%)
ResNet18 [15]	CNN	12M	1.8G	-	69.8
ResNet50 [15]	CNN	26M	4.1G	-	78.5
ResNet101 [15]	CNN	45M	7.9G	-	79.8
RegNetY-4G [30]	CNN	21M	4.0G	1157	80.0
RegNetY-8G [30]	CNN	39M	8.0G	592	81.7
RegNetY-16G [30]	CNN	84M	16.0G	335	82.9
GFNet-H-S [31]	FFT	32M	4.5G	-	81.5
GFNet-H-B [31]	FFT	54M	8.4G	-	82.9
BoT-S1-50 [33]	Hybrid	21M	4.3G	-	79.1
BoT-S1-59 [33]	Hybrid	34M	7.3G	-	81.7
DeiT-S [37]	Trans	22M	4.6G	940	79.8
DeiT-B [37]	Trans	86M	17.5G	292	81.8
PVT-Small [39]	Trans	25M	3.8G	820	79.8
PVT-Medium [39]	Trans	44M	6.7G	526	81.2
PVT-Large [39]	Trans	61M	9.8G	367	81.7
T2T-ViT-14 [43]	Trans	22M	5.2G	764	81.5
T2T-ViT-19 [43]	Trans	39M	8.9G	464	81.9
T2T-ViT-24 [43]	Trans	64M	14.1G	312	82.3
TNT-S [13]	Trans	24M	5.2G	428	81.5
TNT-B [13]	Trans	66M	14.1G	246	82.9
iRPE-K [41]	Trans	87M	17.7G	-	82.4
iRPE-QKV [41]	Trans	22M	4.9G	-	81.4
GLiT-Small [4]	Trans	25M	4.4G	-	80.5
GLiT-Base [4]	Trans	96M	17.0G	-	82.3
Swin-T [27]	Trans	29M	4.5G	755	81.3
Swin-S [27]	Trans	50M	8.7G	437	83.0
Swin-B [27]	Trans	88M	15.4G	278	83.3
Wave-MLP-T*	MLP	15M	2.1G	1257	**80.1**
Wave-MLP-T	MLP	17M	2.4G	1208	**80.6**
Wave-MLP-S	MLP	30M	4.5G	720	**82.6**
Wave-MLP-M	MLP	44M	7.9G	413	**83.4**
Wave-MLP-B	MLP	63M	10.2G	341	**83.6**

the top-1 accuracy on a single crop is reported. To be compatible with the downstream tasks, we use a local window for token-FC and set the window size to 7 empirically. By adjusting the architecture configures, four models (T, S ,M, B) with different parameters and computational costs are developed. Besides, by replacing the FC layer of phase estimation module with a depth-wise convolution, an more efficient architecture is developed and denoted as Wave-MLP-T*. All the experiments are conducted with PyTorch [29] on NVIDIA V100 GPUs.

Comparison with the existing MLP-like architectures. Table 1 compares the proposed Wave-MLP with existing vision MLP models proposed recently or currently. Throughput is measured on a V100 GPU following [27, 37][2]. The family of Wave-MLP achieves a better trade-off between the computational cost and accuracies than the existing methods. For example, our Wave-MLP-M model achieves 83.4% top-1 accuracy with only 7.9G FLOPs, which shows a large superiority to ResMLP-B24 [36] (81.0% accuracy with 23.0G FLOPs). Compared with the SOTA MLP architecture CycleMLP [6], Wave-MLP also achieves higher accuracies with similar parameters and FLOPs, *e.g.*, Wave-MLP-T achieve a accuracy of 80.6%, much higher than that of CycleMLP-B1 with 78.9% accuracy. It shows that equipping each token with the phase information can well capture the relationship between varying tokens and fixed weights to improve the performance of MLP architecture.

Comparison with SOTA models on ImageNet. We further compare the proposed Wave-MLP with typical CNN and transformer architectures on ImageNet in Table 2. Compared with Swin Transformer [27], our Wave-MLP achieves higher performance with fewer parameters and computational costs. For example, with 4.5G FLOPs, Wave-MLP-S achieves 82.6% top-1 accuracy, wihch significantly superior to Swin-T with 81.3% accuracy. Its trade-off between computational cost and accuracy also suppresses the typical CNN architectures such as RegNetY and ResNet18. The superiority of Wave-MLP implies that the simple MLP architecture has a large potential and modulating the token aggregating process with phase term can exploit it adequately.

4.2. Object Detection on COCO

Settings. We further investigate the proposed Wave-MLP architecture on the object detection and instance segmentation tasks. The experiments are conducted on the COCO 2017 dataset [25], which contains 118k training images and 5k validation images. Wave-MLP is used as the backbone and embedded into two prevalent detectors, RetinaNet [24] and Mask R-CNN [14]. For a fair comparison, we follow the training recipe in [39] and train the model with AdamW [28] optimizer for 12 epochs (1× training scheduler). The batchsize is set to 16 and initial learning to 0.0001. The backbones are initialized with the pre-trained weights on ImageNet while other layers are initialized with Xavier [10].

Results. Table 3 compares the object detection results with different architectures as the backbone. For both RetinaNet and Mask R-CNN, the proposed Wave-MLP achieves obviously higher performance compared with the existing models. For example, With RetinaNet 1×, Wave-MLP-T achieves 40.4% AP with only 25.3M parameters and 196.3G FlOPs, which is higher than CycleMLP-B1 (38.6 AP) with similar model size by 1.8 AP. When using Mask R-CNN as the detector, the performance improvements are also significant. Compared with Swin-T of 42.2 box AP and 39.1 mask AP with 47.8M parameters and 264.0G FLOPs, our Wave-MLP-S achieves significantly higher performance (44.0 box AP and 40.0 mask AP) with fewer pa-

[2]Note that AS-MLP [23] reports throughput under the mixed precision mode (mixed FP16 and FP32). For a fair comparison with the existing models, we remeasure it with pure FP32 following [27, 37]

Table 3. Results of object detection and instance segmentation on COCO val2017.

Backbone	RetinaNet 1×							Mask R-CNN 1×						
	Params. / FLOPs	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	Params. / FLOPs	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
ResNet18 [15]	21.3M / 188.7G	31.8	49.6	33.6	16.3	34.3	43.2	31.2M / 207.3G	34.0	54.0	36.7	31.2	51.0	32.7
PVT-Tiny [39]	23.0M / 189.5G	36.7	56.9	38.9	22.6	38.8	50.0	32.9M / 208.1G	36.7	59.2	39.3	35.1	56.7	37.3
CycleMLP-B1 [6]	24.9M / 195.0G	38.6	59.1	40.8	21.9	41.8	50.7	34.8M / 213.6G	39.4	61.4	43.0	36.8	58.6	39.1
Wave-MLP-T	25.3M / 196.3G	**40.4**	61.0	43.4	24.9	43.7	51.7	35.2M / 214.6G	**41.5**	63.7	45.4	38.2	60.9	40.7
ResNet50 [15]	37.7M / 239.3G	36.3	55.3	38.6	19.3	40.0	48.8	44.2M / 260.1G	38.0	58.6	41.4	34.4	55.1	36.7
Swin-T [27]	38.5M / 244.8G	41.5	62.1	44.2	25.1	44.9	**55.5**	47.8M / 264.0G	42.2	64.6	46.2	39.1	61.6	42.0
PVT-Small [39]	34.2M /226.5G	40.4	61.3	43.0	25.0	42.9	55.7	44.1M / 245.1G	40.4	62.9	43.8	37.8	60.1	40.3
CycleMLP-B2 [6]	36.5M / 230.9G	40.9	61.8	43.4	23.4	44.7	53.4	46.5M /249.5G	41.7	63.6	45.8	38.2	60.4	41.0
Wave-MLP-S	37.1M / 231.3G	**43.4**	64.4	46.5	26.6	47.1	57.1	47.0M /250.3G	**44.0**	65.8	48.2	40.0	63.1	42.9
ResNet101 [15]	56.7M / 315.4G	38.5	57.8	41.2	21.4	42.6	51.1	63.2M / 336.4G	40.4	61.1	44.2	36.4	57.7	38.8
Swin-S [27]	59.8M / 334.8G	44.5	65.7	47.5	27.4	48.0	59.9	69.1M / 353.8G	44.8	66.6	48.9	40.9	63.4	44.2
PVT-Medium [39]	53.9M / 283.1G	41.9	63.1	44.3	25.0	44.9	57.6	63.9M / 301.7G	42.0	64.4	45.6	39.0	61.6	42.1
CycleMLP-B3 [6]	48.1M / 291.3G	42.5	63.2	45.3	25.2	45.5	56.2	58.0M / 309.9G	43.4	65.0	47.7	39.5	62.0	42.4
Wave-MLP-M	49.4M / 291.3G	**44.8**	65.8	47.8	28.0	48.2	59.1	59.6M / 311.5G	**45.3**	67.0	49.5	41.0	64.1	44.1
PVT-Large [39]	71.1M / 345.7G	42.6	63.7	45.4	25.8	46.0	58.4	81.0M / 364.3G	42.9	65.0	46.6	39.5	61.9	42.5
CycleMLP-B4 [6]	61.5M / 356.6G	43.2	63.9	46.2	26.6	46.5	57.4	71.5M / 375.2G	44.1	65.7	48.1	40.2	62.7	43.5
CycleMLP-B5 [6]	85.9M / 402.2G	42.7	63.3	45.3	24.1	46.3	57.4	95.3M / 421.1G	44.1	65.5	48.4	40.1	62.8	43.0
Wave-MLP-B	66.1M / 333.9G	**44.2**	65.1	47.1	27.1	47.8	58.9	75.1M / 353.2G	**45.7**	67.5	50.1	27.8	49.2	59.7

Table 4. The semantic segmentation results of different backbones on the ADE20K validation set. † Results are from GFNet [31].

Backbone	Semantic FPN		
	Params.	FLOPs	mIoU (%)
ResNet18 [15]	15.5M	127G	32.9
PVT-Tiny [39]	17.0M	123G	35.7
CycleMLP-B1 [6]	18.9M	130G	39.5
Wave-MLP-T (ours)	19.3M	131G	**41.2**
ResNet50 [15]	28.5M	183G	36.7
PVT-Small [39]	28.2M	163G	39.8
Swin-S† [27]	31.9M	182G	41.5
GFNet-H-Ti [31]	26.6M	126G	41.0
CycleMLP-B2 [6]	30.6M	167G	42.4
Wave-MLP-S (ours)	31.2M	168G	**44.4**
ResNet101 [15]	47.5M	260G	38.8
PVT-Medium [39]	48.0M	219G	41.6
GFNet-H-S [31]	47.5M	179G	42.5
Swin-B† [27]	53.2M	274G	45.2
GFNet-H-B [31]	74.7M	261G	44.8
CycleMLP-B3 [6]	42.1M	229G	44.5
CycleMLP-B4 [6]	55.6M	296G	45.1
CycleMLP-B5 [6]	79.4M	343G	45.6
Wave-MLP-M (ours)	43.3M	231G	**46.8**

rameters (47.0M) and lower computational cost (250.3G).

4.3. Semantic Segmentation on ADE20K

Settings. The experiments for the semantic segmentation task are conducted on the challenging ADE20K dataset [46], which contains 25k images from 150 semantic categories, 20k for training, 2k for validation and 3k for testing. Following [39], we combine the proposed Wave-MLP architecture with the widely used Semantic FPN [20] approach. With the pre-trained weights on ImageNet, the model is fine-tuned for 40k iterations with AdamW [28] optimizer and the batchsize is set to 32. The initial learning rate is 0.0001 and decays with the polynomial schedule (a power of 0.9). The images are randomly resized and cropped to 512 × 512 for training and rescaled to have a shorter side of 512 for testing. The FLOPs are tested with 2048×512 input.

Results. The results of different models for semantic segmentation are shown in Table 4. Under different configures of parameters and computational costs, Wave-MLP outperforms the existing models consistently. Compared with the transformer-based model such as PVT, the model show a

Table 5. The effectiveness of phase information.

Mode	Params.	FLOPs	Top-1 accuracy (%)
No phase	15M	2.1G	78.8
Static phase	15M	2.1G	79.3
Dynamic phase	15M	2.1G	80.1

Table 6. The formulation of phase estimation module.

size	Params.	FLOPs	Top-1 accuracy (%)
Baseline	15M	2.1G	78.8
Identity	15M	2.1G	79.3
Depth-wise	15M	2.1G	80.1
Channel-FC	17M	2.4G	80.6

Table 7. The size of window for aggregating tokens.

Size	Params.	FLOPs	Top-1 accuracy (%)
3	15M	2.1G	79.7
5	15M	2.1G	79.8
7	15M	2.1G	80.1
All	16M	2.3G	80.0

large superiority, *e.g.*, 4.6% mIoU gap between Wave-MLP-S (44.4% mIoU) and PVT-Tiny (39.8% mIoU). It also suppress the CycleMLP-B2 model with 42.4% mIoU and Swin-S with 41.5%. We infer that modulating the aggregating process of different tokens with the phase term can capture more detailed information and thus enhance the semantic segmentation results.

4.4. Ablation Studies

For better understanding the proposed method, we investigate the effectiveness of each component via ablation studies. The experiments are conducted on ImageNet with the Wave-MLP-T* model.

The effectiveness of phase information. The phase plays a vital role in aggregating the information of different tokens, whose effectiveness is investigated in Table 5. Without the phase information ('No phase'), the model's performance is obviously inferior compared with others, with only 78.8% top-1 accuracy. It shows that the fixed wights in token-FC cannot aggregate tokens well without the dynamical modulation of the phase term. 'Static phase' uses fixed parameters to represent the token's phases from all the inputs, whose performance is poor as the diversity of different input images is neglected. The proposed 'dynamic phase' flexibly generates phases and modulate the aggregating process for each input instance, which achieves much better performance (*e.g.*, 80.1% top-1 accuracy).

The formulation of phase estimation module. The phase estimation module generates phases for different inputs, which can be implemented with different formulations. We investigate three simple formulations, depth-wise convolution, channel-FC and identity projection. The identity projection directly copies the input feature instead of estimating the phase, incurring poor performance (*i.e.*, 79.3%). The depth-wise convolution and channel-FC can achieve high accuracy improvement compared with the baseline (*e.g.*, 1.3% and 1.8%), implying they can capture the phase information well for aggregating tokens. Using channel-FC achieves higher performance than the depth-wise convolution, but also increase the computational cost slightly.

The size of window for aggregating tokens. To be compatible with dense prediction tasks (*e.g.*, object detection and semantic segmentation) with varying sizes of input images, we restrict that the token-FC only aggregates features within a local window, and Table 7 investigates the impact of window size. Changing window size from 3 to 7, the top-1 accuracies increase accordingly. 'All' denotes that the token-FC connects all the tokens in a layer, which achieves similar performance with window size 7. However, its parameter configure is corrected to the size of input image, and thus is infeasible in dense prediction tasks such as object detection and semantic segmentation.

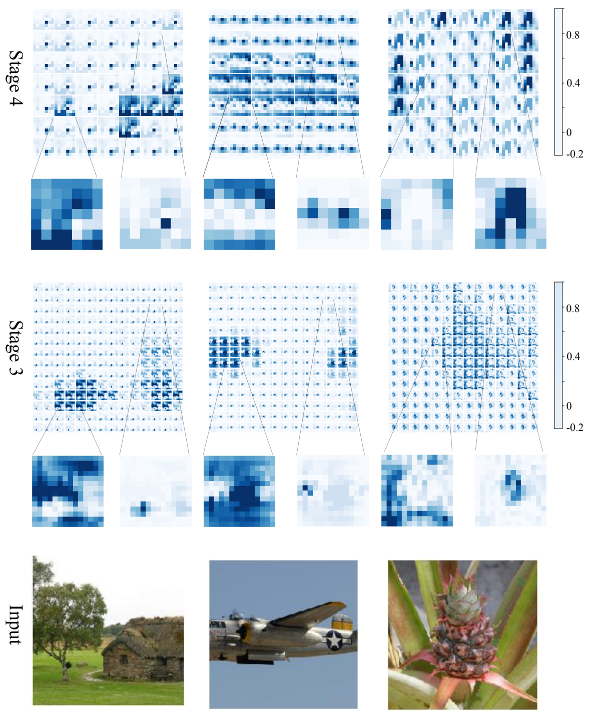


Figure 4. Visualization of the phase difference between tokens.

Visualization. The phase difference between two tokens ($|\boldsymbol{\theta}_j - \boldsymbol{\theta}_i|$) directly affects the aggregating process as analyzed in Section 3.2 (Eq. 4, 5). In order to have an intuitive understanding, we show the cosine value of phase difference of the 3rd and 4th stages in Figure 4. Take the visualized figure of the 1st image and the 4th stage for example, the 7×7 values in the (i, j)-th patch denote the phase differences between the (i, j)-th token and all the 7×7 tokens. From the figure, we can see that tokens with similar contents tend to have close phases and then enhanced by each other. For example, in the first image, a token describing the 'house' has a closer phase with another token of the 'house' than that of the sky (magnifying parts in the figure). The phase difference of different tokens also varies w.r.t. different input images depending on the image contents.

5. Conclusion

This paper proposes a Wave-MLP architecture for vision tasks, which takes each token as a wave with both amplitude and phase information. Amplitude is the original real-value feature and the phase modulates relationship between the varying tokens and fixed weights in MLP. With the dynamically produced phase, the tokens are aggregated according to their varying contents from different input images. Extensive experiments show that the proposed Wave-MLP suppresses the existing MLP-like architectures and can also be used as a strong backbone for the dense prediction tasks such as object detection and semantic segmentation. In the future, we will further explore the potential of MLP-like architectures on more diverse tasks.

References

[1] Markus Arndt, Olaf Nairz, Julian Vos-Andreae, Claudia Keller, Gerbrand Van der Zouw, and Anton Zeilinger. Wave–particle duality of c 60 molecules. *nature*, 401(6754):680–682, 1999. 2

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 3

[3] Vladimir B Braginsky and Braginskiĭ. *Quantum measurement*. 4

[4] Boyu Chen, Peixia Li, Chuming Li, Baopu Li, Lei Bai, Chen Lin, Ming Sun, Junjie Yan, and Wanli Ouyang. Glit: Neural architecture search for global and local image transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12–21, 2021. 6

[5] Chun-Fu Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. *arXiv preprint arXiv:2103.14899*, 2021. 2

[6] 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. 3, 5, 6, 7

[7] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020. 5

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 5

[9] 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. 1, 2

[10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010. 6

[11] David J Griffiths and Darrell F Schroeter. *Introduction to quantum mechanics*. Cambridge University Press, 2018. 2

[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. 5

[13] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021. 2, 6

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 5, 6, 11

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 2, 6, 7, 11

[16] EJ Heller, MF Crommie, CP Lutz, and DM Eigler. Scattering and absorption of surface electron waves in quantum corrals. *Nature*, 369(6480):464–466, 1994. 2

[17] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. *arXiv preprint arXiv:2103.16302*, 2021. 2

[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. 3, 5

[19] Kurt Jacobs and Daniel A Steck. A straightforward introduction to continuous quantum measurement. *Contemporary Physics*, 47(5):279–303, 2006. 4

[20] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019. 5, 7

[21] 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. 1, 2

[22] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2

[23] Dongze Lian, Zehao Yu, Xing Sun, and Shenghua Gao. As-mlp: An axial shifted mlp architecture for vision. *arXiv preprint arXiv:2107.08391*, 2021. 3, 5, 6

[24] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 5, 6

[25] 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*, pages 740–755. Springer, 2014. 5, 6, 11

[26] Hanxiao Liu, Zihang Dai, David R So, and Quoc V Le. Pay attention to mlps. *arXiv preprint arXiv:2105.08050*, 2021. 3, 5

[27] 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. *International Conference on Computer Vision (ICCV)*, 2021. 1, 2, 5, 6, 7

[28] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5, 6, 7

[29] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 6

[30] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollar. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020. 1, 2, 6

[31] Yongming Rao, Wenliang Zhao, Zheng Zhu, Jiwen Lu, and Jie Zhou. Global filter networks for image classification. *arXiv preprint arXiv:2107.00645*, 2021. 6, 7

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[33] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16519–16529, 2021. 6

[34] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 2

[35] 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. 1, 2, 3, 5

[36] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, and Herve Jegou. Resmlp: Feedforward networks for image classification with data-efficient training. *arXiv preprint arXiv:2105.03404*, 2021. 1, 3, 5, 6

[37] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 2, 5, 6

[38] 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*, pages 5998–6008, 2017. 1, 2

[39] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *IEEE ICCV*, 2021. 1, 2, 5, 6, 7, 11

[40] Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Zhicheng Yan, Masayoshi Tomizuka, Joseph Gonzalez, Kurt Keutzer, and Peter Vajda. Visual transformers: Token-based image representation and processing for computer vision. *arXiv preprint arXiv:2006.03677*, 2020. 2

[41] Kan Wu, Houwen Peng, Minghao Chen, Jianlong Fu, and Hongyang Chao. Rethinking and improving relative position encoding for vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10033–10041, 2021. 2, 6

[42] Tan Yu, Xu Li, Yunfeng Cai, Mingming Sun, and Ping Li. S2-mlp: Spatial-shift mlp architecture for vision. *arXiv preprint arXiv:2106.07477*, 2021. 3, 5

[43] 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. 6

[44] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019. 5

[45] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 5

[46] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127(3):302–321, 2019. 5, 7

Table 8. Detailed architecture specifications of Wave-MLP. 'Dimension' and 'expansion' denote the dimension of feature and expand ratio, respectively. H and W are the height and width of input image. FLOPs is calculated with input size of 224×224.

	Output size	Wave-MLP-T	Wave-MLP-S	Wave-MLP-M	Wave-MLP-B
stage 1	$\frac{H}{4} \times \frac{W}{4}$	$\begin{bmatrix}\text{dimension} = 64\\ \text{expansion} = 4\end{bmatrix} \times 2$	$\begin{bmatrix}\text{dimension} = 64\\ \text{expansion} = 4\end{bmatrix} \times 2$	$\begin{bmatrix}\text{dimension} = 64\\ \text{expansion} = 8\end{bmatrix} \times 3$	$\begin{bmatrix}\text{dimension} = 96\\ \text{expansion} = 4\end{bmatrix} \times 2$
stage 2	$\frac{H}{8} \times \frac{W}{8}$	$\begin{bmatrix}\text{dimension} = 128\\ \text{expansion} = 4\end{bmatrix} \times 2$	$\begin{bmatrix}\text{dimension} = 128\\ \text{expansion} = 4\end{bmatrix} \times 3$	$\begin{bmatrix}\text{dimension} = 128\\ \text{expansion} = 8\end{bmatrix} \times 4$	$\begin{bmatrix}\text{dimension} = 192\\ \text{expansion} = 4\end{bmatrix} \times 2$
stage 3	$\frac{H}{16} \times \frac{W}{16}$	$\begin{bmatrix}\text{dimension} = 320\\ \text{expansion} = 4\end{bmatrix} \times 4$	$\begin{bmatrix}\text{dimension} = 320\\ \text{expansion} = 4\end{bmatrix} \times 10$	$\begin{bmatrix}\text{dimension} = 320\\ \text{dexpansion} = 4\end{bmatrix} \times 18$	$\begin{bmatrix}\text{dimension} = 384\\ \text{expansion} = 4\end{bmatrix} \times 18$
stage 4	$\frac{H}{32} \times \frac{W}{32}$	$\begin{bmatrix}\text{dimension} = 512\\ \text{expansion} = 4\end{bmatrix} \times 2$	$\begin{bmatrix}\text{dimension} = 512\\ \text{expansion} = 4\end{bmatrix} \times 3$	$\begin{bmatrix}\text{dimension} = 512\\ \text{dexpansion} = 4\end{bmatrix} \times 3$	$\begin{bmatrix}\text{dimension} = 768\\ \text{expansion} = 4\end{bmatrix} \times 2$
# Parameters		17M	30M	44M	63M
FLOPs		2.4G	4.5G	7.9G	10.2G

Table 9. Results of object detection and instance segmentation on COCO val2017. The Mask R-CNN model trained with 3× schedule and multi-scale training strategy [14] is used as the detector.

Backbone	Params. / FLOPs	AP^b	AP^b_{50}	AP^b_{75}	AP^b_S	AP^b_M	AP^b_L	AP^m	AP^m_{50}	AP^m_{75}	AP^m_S	AP^m_M	AP^m_L
ResNet18 [15]	31.2M / 207.3G	36.9	57.1	40.0	-	-	-	33.6	53.9	35.7	-	-	-
PVT-Tiny [39]	32.9M / 208.1G	39.8	62.2	43.0	-	-	-	37.4	59.3	39.9	-	-	-
Wave-MLP-T	25.3M / 196.3G	**44.1**	66.0	48.2	28.4	47.6	55.9	**40.1**	63.1	43.2	24.3	43.5	53.2
ResNet50 [15]	44.2M / 260.1G	41.0	61.7	44.9	-	-	-	37.1	58.4	40.1	-	-	-
PVT-Small [39]	44.1M / 245.1G	43.0	65.3	46.9	-	-	-	39.9	62.5	42.8	-	-	-
Wave-MLP-S	37.1M / 231.3G	**45.5**	66.9	49.3	29.4	48.7	58.7	**41.0**	64.2	44.0	25.0	44.2	54.7
ResNet101 [15]	63.2M / 336.4G	42.8	63.2	-	-	-	47.1	38.5	60.1	41.3	-	-	-
PVT-Medium	63.9M / 301.7G	44.2	66.0	48.2	-	-	-	40.5	63.1	43.5	-	-	-
PVT-Large	71.1M / 345.7G	44.5	66.0	48.3	-	-	-	40.7	63.4	43.7	-	-	-
Wave-MLP-M	49.4M / 291.3G	**46.3**	67.8	50.3	29.5	49.3	60.3	**41.5**	65.2	44.1	24.9	44.7	55.6

A. Detailed Architectures

Table 8 shows the detailed specifications of the proposed Wave-MLP architecture. To get hierarchical features, we split the whole model into four stages, and reduce the size of feature map stage-wisely. The Wave-MLP family contains four models with different parameters and computational costs by adjusting the depths and widths of architecture specifications, which are denoted as Wave-MLP-T, Wave-MLP-S, Wave-MLP-M, and Wave-MLP-B, sequentially. From Wave-MLP-T to Wave-MLP-B, the number of parameters varies from 17M to 63M, and FLOPs varies from 2.4G to 10.2G.

B. More Experiments

For the object detection and instance segmentation tasks on COCO [25], we further train Mask R-CNN models with 3× schedule and multi-scale training strategy [14]. The results of different backbone are shown in Table 9. Compared with other backbones, the proposed Wave-MLP achieves much higher performance. For example, our Wave-MLP-T achieves 44.1 box AP and 40.1 mask AP with 25.3M parameters and 196.3G FLOPs, which is significantly superior to the PVT-Tiny model with 39.8 box AP, 37.4 mask AP, 25.3M parameters and 196.3G FLOPs.