Review

Are we ready for a new paradigm shift?
A survey on visual deep MLP

Ruiyang Liu,[1,3] Yinghui Li,[2,3] Linmi Tao,[1,*] Dun Liang,[1] and Hai-Tao Zheng[2]

[1]Department of Computer Science and Technology, BNRist, Tsinghua University & Key Lab of Pervasive Computing, Ministry of Education of China, Beijing 100084, China
[2]Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China
[3]These authors contributed equally
*Correspondence: linmi@tsinghua.edu.cn
https://doi.org/10.1016/j.patter.2022.100520

THE BIGGER PICTURE The last decade has been called the third spring of deep learning, when extraordinary progress in both theory and applications has been made. This has attracted the enthusiasm of academia and an investment of resources from the whole society. During this period, the scale of networks has grown tremendously, and paradigms have emerged constantly. People wonder how long this trend will continue and what the next network will be.

We point out that existing paradigms blossom in a riot of color on the tree rooted in weighted summation with GPU computing. In terms of computation and power, its applications are still promising, but simply scaling the network with more GPUs is not sustainable. We recognize that the brilliance of this generation of networks is based on the switch of computing hardware from CPU to GPU and, similarly, we expect that the next paradigm will be brought about by the emergence of a new physical system-based computing hardware with a non-weighted summation network, such as the Boltzmann machine.

SUMMARY

Recently, the proposed deep multilayer perceptron (MLP) models have stirred up a lot of interest in the vision community. Historically, the availability of larger datasets combined with increased computing capacity led to paradigm shifts. This review provides detailed discussions on whether MLPs can be a new paradigm for computer vision. We compare the intrinsic connections and differences between convolution, self-attention mechanism, and token-mixing MLP in detail. Advantages and limitations of token-mixing MLP are provided, followed by careful analysis of recent MLP-like variants, from module design to network architecture, and their applications. In the graphics processing unit era, the locally and globally weighted summations are the current mainstreams, represented by the convolution and self-attention mechanism, as well as MLPs. We suggest the further development of the paradigm to be considered alongside the next-generation computing devices.

INTRODUCTION

In computer vision, the ambition to create a system that imitates how the brain perceives and understands visual information fueled the initial development of neural networks.[1,2] Subsequently, convolutional neural networks (CNNs),[3–5] multilayer perceptrons (MLPs),[6] and Boltzmann machine[7,8] were proposed, and achieved fruitful results in theoretical research[9–14] in the last century. CNNs stood out due to their computational efficiency over MLPs and deep Boltzmann machines in the contest to replace hand-crafted features, and topped the list for vast visual tasks in the 2010s. From 2020, the Transformer-based models introduced from the natural language processing field to the visual field have once again reached a new peak. With the introduction of MLP-Mixer[15] in 2021, the hot topic in the vision community comes: *Will MLP*

become a new paradigm and push computer vision to a new height? This survey aims to provide opinions on this issue.

From a historical perspective, the availability of larger datasets combined with the transition from CPU-based training to graphics processing unit (GPU)-based training leads to paradigm shifts and a gradual reduction in human intervention. The locally weighted summation represented by convolution and globally weighted summation represented by self-attention are the current mainstreams. The token-mixing MLP[15] in MLP-Mixer further abandons the artificially designed self-attention mechanism and allows the model learn the global weights matrix autonomously from the raw data, seemingly in line with the laws of historical development.

We review MLP-Mixer in detail, and compare the intrinsic connections and differences between convolution, self-attention mechanism, and token-mixing MLP. We observe that the

token-mixing MLP is an enlarged and weights-shared-between-channel version of depthwise convolution,[16] which faces challenges, such as high computational complexity and resolution sensitivity. Exhaustive analysis reveals that, not only the recent MLP-like variant designs are gradually approaching the direction of CNN, but the performance of these variants in visual tasks still lags behind CNN- and Transformer-based designs. At this moment, MLP is not a new paradigm that can push computer vision to new heights. In fact, computing paradigm and computing hardware are cooperative. The current weighted-sum paradigms have driven the booming of GPU-based computing and deep learning itself, while we believe the next paradigm or Boltzmann-like will also grow up with a new generation of computing hardware.

The rest of the paper is organized as follows. Preliminary reviews MLP, CNN, and Transformer, as well as their corresponding paradigms from a historical perspective. Pioneering model and new paradigm reviews the design of the latest MLP pioneering models, describes the differences and connections between token-mixing MLP, convolution, and self-attention mechanism, and presents the bottlenecks and challenges faced by the seemingly new paradigm. Block of MLP variants and architecture of MLP variants discuss the block evolution and network architecture of MLP-like variants. Applications of MLP variants sheds light on applications of MLP-like variants. Summary and outlook gives our summary and discusses potential future research directions.

PRELIMINARY

For completeness and to provide helpful insight into the visual deep MLP presented in the subsequent sections, we briefly introduce MLP, CNN, and Transformer, including their brief histories and corresponding paradigms.

Multilayer perceptron and Boltzmann machine

The original "Perceptron" model was developed by Frank Rosenblatt in 1958,[2] and can be viewed as a fully connected layer with only one output element. In 1969, a famous book entitled *Perceptrons* by Marvin Minsky and Seymour Papert[17] critically analyzed perceptron and pointed out several critical weaknesses of perceptron, including that perceptron was unable to learn an XOR function. For a while, interest in perceptron waned.

Interest in perceptron revived in 1985 when Hinton and co-workers[6] recognized that a feedforward neural network with two or more layers had a greater fitting ability than a single-layered perceptron. Indeed Hinton and co-workers proposed the MLP, a network composed of multiple layers of perceptrons and activation function, to solve the XOR problem. And they provided a reasonably effective training algorithm called *backpropagation* for neural networks. As shown in 1989 by Cybenko,[9] Hornik et al.,[10] and Funahashi,[11] MLPs are universal function approximators and can be used to construct mathematical models for classification and regression analysis.

In MLP, the fully connected layer can be viewed as a paradigm for extracting features. As the name implies, the main feature lies in the full connectivity, i.e., all the neurons from one layer are connected to every neuron in the following layer (Figure 1D). One problem with fully connected layers is the input resolution sensitivity, where the number of neurons is related to the input size.

Another significant problem with full connectivity is the enormous parameter cardinality and computational cost, growing quadratically with the image resolution.

The Boltzmann machine[7] proposed in 1985 is more theoretically intriguing because of the analogy of its dynamics to simple physical processes; and a restricted Boltzmann machine (RBM)[8] comprises a layer of visible neurons and a layer of hidden neurons with only visible-hidden connections between the two layers. RBM is stacked[18] and driven by recovering to a minimum energy state computed by a predetermined energy function. Contrarily speaking, stacked RBM is more computationally expensive than MLP, especially during inference. Computing power has been the main factor limiting the development of both MLP and RBM.

Convolution neural network

CNN was first proposed by Fukushima[3,19,20] in an architecture called Neocognitron, which involved multiple pooling and convolutional layers, and inspired later CNNs. In 1989, LeCun et al.[4] proposed a multilayered CNN for handwritten zip code recognition, and the prototype of the architecture later called LeNet. After years of research, LeCun[5] proposed LeNet-5, which outperformed all other models on handwritten character recognition. In the CPU era, it was widely accepted that the backpropagation algorithm was ineffective, considering the limited computational power of CPUs, in converging to the global minima of the error surface, and hand-crafted features were generally better than that of CNN-based extractors.[21]

In 2007, NVIDIA developed the CUDA programming platform;[22,23] and, in 2009, ImageNet,[24] a large image dataset, was proposed to provide the raw material for networks to learn image features autonomously. Three years later, AlexNet[25] won the ImageNet competition, a symbolically significant event of the first paradigm shift. CNN-based architectures have gradually been utilized to extract image features automatically instead of hand-crafted features. In traditional computer vision algorithms, features such as gradient, texture, and color are extracted locally. Hence, the inductive biases inherent to CNNs, such as local connectivity and translation invariance, significantly help image feature extraction. The development of self-supervision[26–29] and training strategies[30–35] further assisted the continuous improvement of CNNs. In addition to classification, CNNs outperform traditional algorithms for almost all computer vision tasks, such as object detection,[36,37] segmentation,[38,39] demosaicing,[40] super-resolution,[41–43] and deblurring.[44] CNN is the de facto standard for computer vision.

A CNN architecture typically comprises alternate convolutional and pooling layers with several fully connected layers behind, where the standard local-connected convolutional layer is the paradigm (Figure 1A); and depthwise convolution is a variant of convolution, where it applies different convolutional filters to different single channel (Figure 1B). In the convolutional operation, sliding kernels with the same set of weights can extract full sets of features within an image, making the convolutional layer more parameter efficient than the fully connected layer.

Vision Transformer

Keeping pace with Moore's law,[45] the computing capability has increased steadily with each new generation of chips. In 2020, visual researchers noted the application and success of the
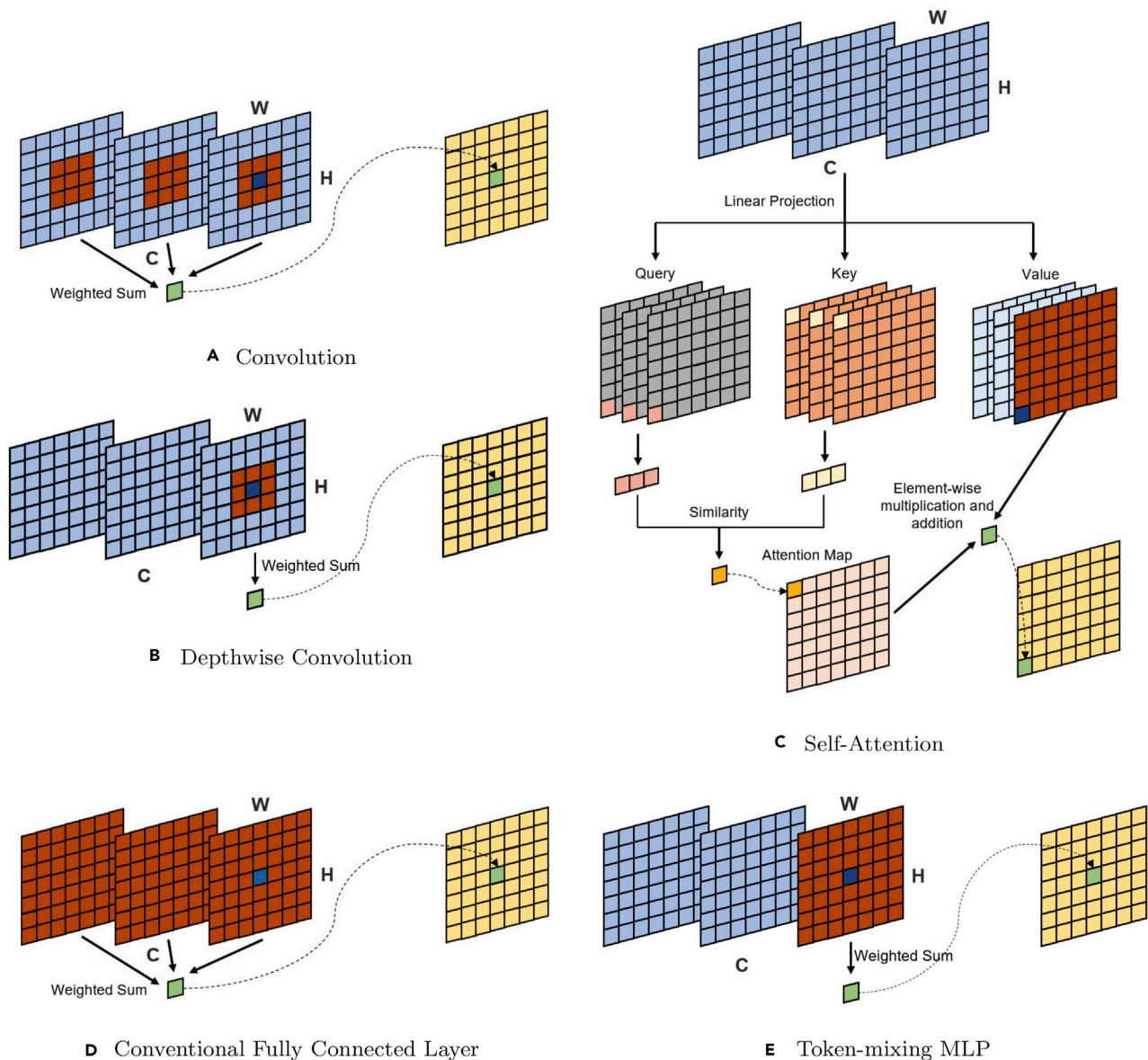
Figure 1. Illustrative shift between different weighted-sum paradigms

Illustrative shift between different weighted-sum paradigms in CNN (A and B), Transformer (C), and MLP (D and E). The input feature map is $H \times W \times C$, where H, W, and C are the feature map's height, width, and channel numbers, respectively. The light blue part highlights the input features, and the yellow part is the output features. The dark blue dot represents the position of interest, the dark orange denotes other features used in the calculation process, and the green dot represents the corresponding output feature. The token-mixing MLP is reduced to one fully connected layer to facilitate understanding. Linear projection performs a 1×1 convolution along the channel dimension, and weighted sum means the elements are multiplied by the weights and then summed.

Transformer[46] in natural language processing and suggested moving beyond the limits of local connectivity and toward global connectivity. Vision Transformer (ViT[47]) is the first work promoting research on Transformer in the field of vision. It uses stacked transformer blocks of the same architecture to extract visual features, where each transformer block comprises two parts: a multi-head self-attention layer and a two-layer MLP, in which layer normalization and residual path are also added. Since then, the Transformer-based architecture has been widely accepted in the vision community, outperforming CNN-based architectures in tasks such as objection detection[48,49] and seg-

mentation,[49–51] and achieving state-of-the-art performance on denoising,[52] deraining,[52] and super-resolution.[52–55] Furthermore, several works[56–59] demonstrate that the Transformer-based architecture is more robust than CNN-based methods. All the developments over the last two years indicate that *Transformer has become another de facto standard for computer vision*.

The paradigm in Transformer can be boiled down to the self-attention layer, where each input vector is linearly mapped into three different vectors: called *query* **q**, *key* **k** and *value* **v**. Then the *query*, *key*, and *value* vectors come from different input
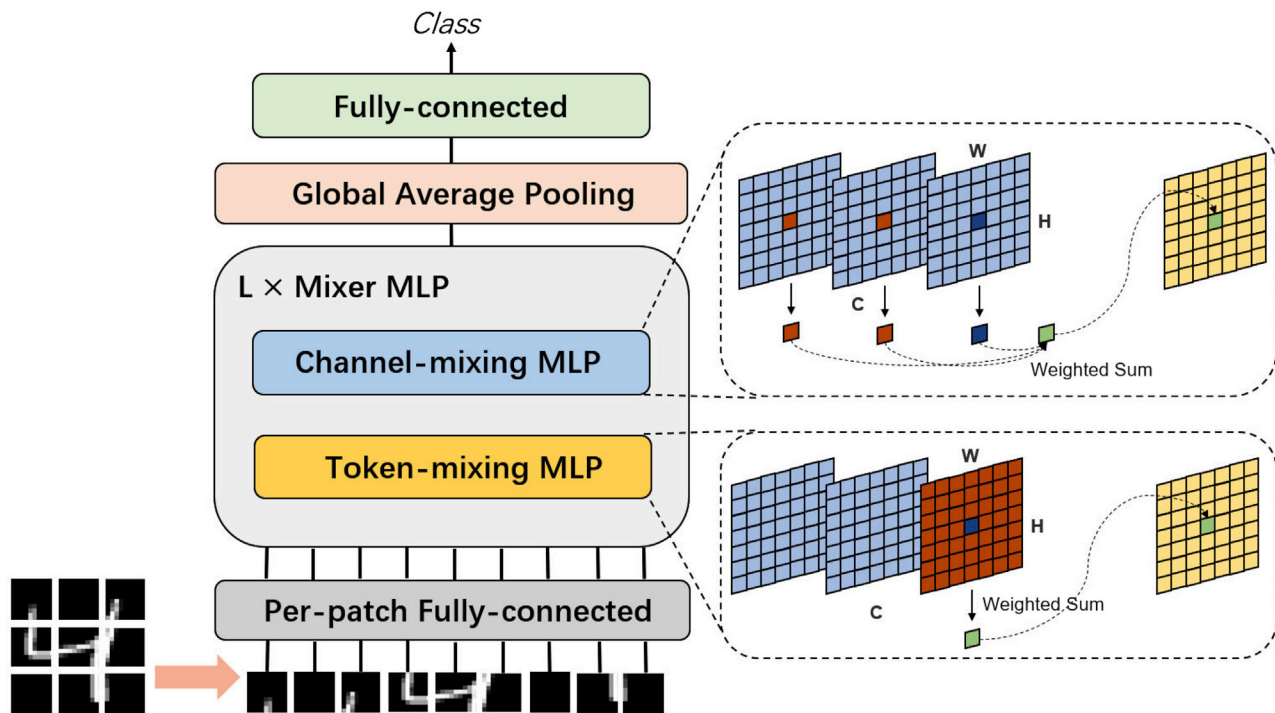
Figure 2. MLP-Mixer structure
We omit the layer normalization, non-linear activation function, and the residual path to improve readability. The token-mixing and channel-mixing MLP are reduced to one fully connected layer to ease understanding. The expression in the dashed box is consistent with Figure 1.

vectors are aggregated into three corresponding matrices, denoted, **Q**, **K**, and **V** (Figure 1C). The self-attention mainly computes the cosine similarity between each query vector and all key vectors, and the *SoftMax* is applied to normalize the similarity and obtain the attention matrix. Output features then become the weighted sum of all value vectors in the entire space, where the attention matrix gives the weights. Compared with the convolutional layer, which focuses only on local characteristics, the self-attention layer can capture long-distance characteristics and easily derive global information.

PIONEERING MODEL AND NEW PARADIGM

The success of ViT marks the paradigm shift to the era of the global receptive field in computer vision, placing the consecutive question: Can we further abandon the artificially designed self-attention mechanism and allow the model learn the global weights matrix autonomously from the raw data? This motivation reminds researchers of the long-dusted simplest structure, MLP. After a long period of slumber, MLP finally reappears in May 2021, when the first deep MLP, called MLP-Mixer,[15] is launched.

This section reviews in detail the structure of the latest so-called pioneering MLP model, MLP-Mixer,[15] followed by a brief review of the contemporaneous ResMLP[60] as well as FeedForward.[61] After that, we strip the new paradigm, token-mixing MLP, from the network and elaborate its differences and connections with convolution and self-attention mechanisms. Finally, we explore the bottlenecks of token-mixing MLP and lay the foundation for introducing subsequent variants.

Structure of pioneering model
MLP-Mixer[15] is the first proposed visual deep MLP network identified by the vision community as the pioneering MLP model. Compared with conventional MLP, it gets deeper and involves several differences. In detail, the MLP-Mixer comprises three modules, a per-patch fully connected layer for patch embedding, a stack of *L* mixer layers for feature extraction, and a classification header for classification (Figure 2).
Patch embedding
The patch embedding is inherited from ViT,[47] comprising three steps: (1) cut an image into non-overlap patches, (2) flatten the patches, and (3) linearly project these flattened patches. Specifically, an input image of the size $H\times W\times 3$ is split into S non-overlapping $p\times p\times 3$ patches, where H and W are the image's height and width, respectively, S donates patch number, and p represents the patch size (typically 14 or 16). The patch is also called a *token*. Each patch is then unfolded into a vector $\mathbf{p}\in\mathbb{R}^{3p^2}$. In total, we obtain a set of flattened patches $\mathbf{P}=\{\mathbf{p}_1,\cdots,\mathbf{p}_S\}$, which are input of the MLP-Mixer. For each $\mathbf{p}_i\in\mathbf{P}$, the per-patch fully connected layer maps it into a C-dimensional embedding vector:

$$\mathbf{x}_i=\mathbf{W}\mathbf{p}_i^T,\quad \mathbf{W}\in\mathbb{R}^{C\times 3p^2},\mathbf{p}_i\in\mathbb{R}^{3p^2},\text{ for }i=1\ldots S,$$

(Equation 1)

where $\mathbf{x}_i\in\mathbb{R}^C$ is the embedding vector of $\mathbf{p}_i$ and $\mathbf{W}$ represents weights of the per-patch fully connected layer. In practice, it is possible to combine three steps presented above into a single step using a 2D convolution operation, where the convolutional kernel size and stride are equal to patch size.

Table 1. Comparison between convolution, self-attention, and token-mixing MLP

Operation	Information aggregation	Receptive field	Resolution sensitive	Spatial	Channel	Params	FLOPs
Convolution	static	local	false	agnostic	specific	$\mathcal{O}(k^2C^2)$	$\mathcal{O}(HWC^2)$
Depthwise convolution	static	local	false	agnostic	specific	$\mathcal{O}(k^2C)$	$\mathcal{O}(HWC)$
Self-attention[47]	dynamic	global	false	agnostic	specific	$\mathcal{O}(3C^2)$	$\mathcal{O}(H^2W^2C)$
Token-mixing MLP[15]	static	global	True	specific	agnostic	$\mathcal{O}(H^2W^2)$	$\mathcal{O}(H^2W^2C)$

H, W, and C are the height, width, and channel numbers of the feature map, respectively. k is the convolutional kernel size. "Information aggregation" refers to whether the weights are fixed or dynamically generated based on the input during inference. "Resolution sensitive" refers to whether the operation is sensitive to input resolution. "Spatial" refers whether feature extraction is sensitive to the spatial location of objects, "specific" means true, while "agnostic" means false. "Channel specific" means no weights are shared between channels, "Channel agnostic" means weights are shared between channels.

Mixer layers

The MLP-Mixer stacks L mixer layers of the same architecture, where a single mixer layer essentially consists of a token-mixing MLP and a channel-mixing MLP. Let the patch features at the input of each mixer layer be $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_S] \in \mathbb{R}^{C \times S}$, where S is the number of patches, and C is the dimension of each patch feature, i.e., the number of channels. The token-mixing MLP works on each column of $\mathbf{X}$, maps $\mathbb{R}^S \mapsto \mathbb{R}^S$, and the weights are shared among all columns. The channel-mixing MLP works on each row of $\mathbf{X}$, maps $\mathbb{R}^C \mapsto \mathbb{R}^C$, and the weights are shared among all rows. Both token-mixing and channel-mixing MLPs comprise two fully connected layers, and there is a non-linear activation function between the two layers. Thus, a mixer layer can be written as follows (omitting layer indices):

$$\mathbf{U}_{*,i} = \mathbf{X}_{*,i} + \mathbf{W}_2\, \sigma \left(\mathbf{W}_1\, \text{Layer Norm}(\mathbf{X})_{*,i} \right), \quad \text{for } i = 1 \ldots C,$$

$$\mathbf{Y}_{j,*} = \mathbf{U}_{j,*} + \mathbf{W}_4\, \sigma \left(\mathbf{W}_3\, \text{Layer Norm}(\mathbf{U})_{j,*} \right), \quad \text{for } j = 1 \ldots S,$$

(Equation 2)

where σ is the GELU[62] activation function, and LayerNorm($\cdot$) denotes the layer normalization[63] widely used in Transformer-based models. $\mathbf{W}$ represents the weights of a fully connected layer, where $\mathbf{W}_1 \in \mathbb{R}^{rS \times S}$, $\mathbf{W}_2 \in \mathbb{R}^{S \times rS}$, $\mathbf{W}_3 \in \mathbb{R}^{rC \times C}$, and $\mathbf{W}_4 \in \mathbb{R}^{C \times rC}$, and $r > 1$ is the expansion ratio (commonly $r = 4$). It is worth mentioning that each mixer layer takes an input of the same size, which is most similar to Transformers or deep recurrent neural networks in other domains. However, it opposes most CNNs, which have a *pyramidal* structure: deeper layers have a lower resolution input but more channels.

Classification header

After processing with L stacking Mixer layers, S patch features are generated. Then, a global vector is calculated from the features through the average pooling scheme, which is forwarded into a fully connected layer for classification.

Compared with MLP-Mixer, FeedForward[61] and ResMLP[60] were proposed a few days later. FeedForward[61] adopts essentially the same structure as the MLP-Mixer, but swaps the channel-mixing MLP and token-mixing MLP order. As another contemporary work, ResMLP[60] simplifies the token-mixing MLP in the MLP-Mixer from two fully connected layers to one. Meanwhile, ResMLP proposes an affine element-wise transformation to replace the layer normalization in the MLP-Mixer and stabilize training.

Experimentally, these MLP pioneering models achieve comparable performance to CNN and ViT for image classification (Image classification). These empirical results significantly break past perceptions, challenge the necessity for the convolutional layer and the self-attention layer, and prompt the community to rethink whether the convolutional layer or the self-attention layer is necessary. The latter induces us to explore whether a pure MLP stack will become the new paradigm for computer vision.

### Token-mixing MLP, a new paradigm?

To find out whether pure MLPs stacked into a mixer layer will become a new paradigm, it is necessary first to reveal the difference between it, convolution, and self-attention mechanisms. There is an indisputable fact that both the channel-mixing MLP in MLP-Mixer and the MLP in ViT are just a 1×1 convolution commonly used in CNNs, allowing communication between different channels. The core points to compare come naturally to token-mixing MLP, self-attention, and convolution, which allow communication between different spatial locations. A detailed comparison between the three is reported in Table 1.

Convolution usually performs the aggregation computation of spatial information in a local region, but poorly models long-term dependencies. The token-mixing MLP (Figure 1E) can be viewed as an unusual convolution type whose convolutional kernel covers the entire space. To enhance efficiency, it aggregates spatial information from a single channel and shares weights for all channels. This is very close to the depthwise convolution[16] (Figure 1B) used in CNN, which independently applies convolutions to each channel. However, the convolutional kernels in depthwise convolution are usually small and not shared between each channel. The self-attention mechanism considers all patches as well, where the weights are dynamically generated based on the input, while weights in token-mixing MLP and the convolutional layer are fixed and input independent.

We now shift to another important metric, namely complexity. Without loss of generality, we assume that the input feature map size is $HW \times C$ (or $H \times W \times C$ in CNNs), where H and W are the spatial resolutions, and C is the number of channels. Intuitively, local computation has minimal computational complexity, i.e., $\mathcal{O}(HWC)$ in depthwise convolution and $\mathcal{O}(HWC^2)$ in dense convolution. However, both self-attention and token-mixing MLP involving a global receptive field have greater complexity, $\mathcal{O}(H^2W^2C)$. Fortunately, token-mixing MLP is less computationally intensive than the self-attention layer due to the lack of
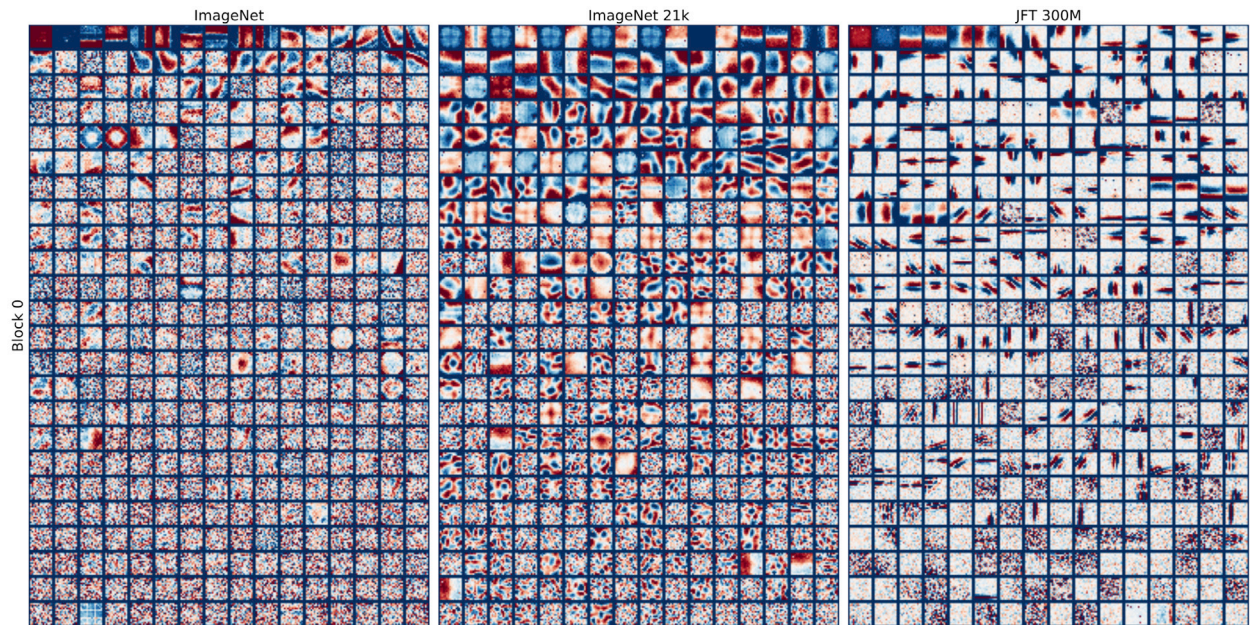
A MLP-Mixer
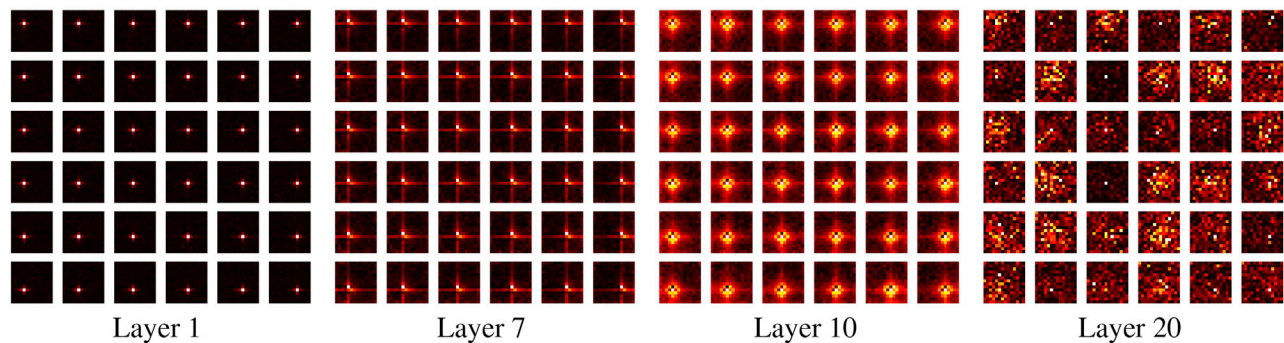
B ResMLP

Figure 3. Visualizing the weights of the fully connected layers in MLP-Mixer and ResMLP
Visualizing the weights of the fully connected layers in MLP-Mixer (A) and ResMLP (B). Each weight matrix is resized to 14×14 pixel images. In MLP-Mixer, white denotes that weight is 0, red means positive weights, blue means negative weights, and the brighter, the greater the weight. In ResMLP, black indicates that weight is 0, and the brighter, the greater the weight's absolute value. The results are from Tolstikhin et al.[15] and Touvron et al.,[60] respectively.

calculations, such as the attention matrix. As for the parameter cardinality, the parameter complexity is $\mathcal{O}(H^2W^2)$ in token-mixing MLP, strongly correlated with the image resolution. So, once the network is fixed, the corresponding image resolution is also fixed. In comparison, other paradigms have more advantages in parameters. The newly proposed MLP-like variants are optimized in complexity, see Complexity analysis.

The above comparative analysis reveals that token-mixing MLP is seemly a new paradigm. But what does the new paradigm bring to the learning weights? Figure 3 visualizes the weights of fully connected layers (FC kernels) of MLP-Mixer and ResMLP trained on ImageNet[24] or JFT-300M.[64] ImageNet-1k contains 1.2 million labeled images, ImageNet-21k involves 14 million images, and JFT-300M has 300 million

images. As the amount of training data increases, the number of FC kernels for locality computation in MLP-Mixer increases. ResMLP's shallow FC kernels also present some local connectivity properties. These fully connected layers actually still perform local computation, which convolutional layers can replace. Thus, we conclude that: *the shallow fully connected layers of deep MLP implement a convolution-like scheme*. As the number of layers increases, i.e., the network becomes deeper, the effective range of receptive field increases, and the weights become disorganized. However, it is unclear if this is due to a lack of training data or if it should be so. Notably, while MLP-Mixer and ResMLP share a highly similar structure (except for the normalization layer), their learning weights are vastly different. This questions whether MLP is
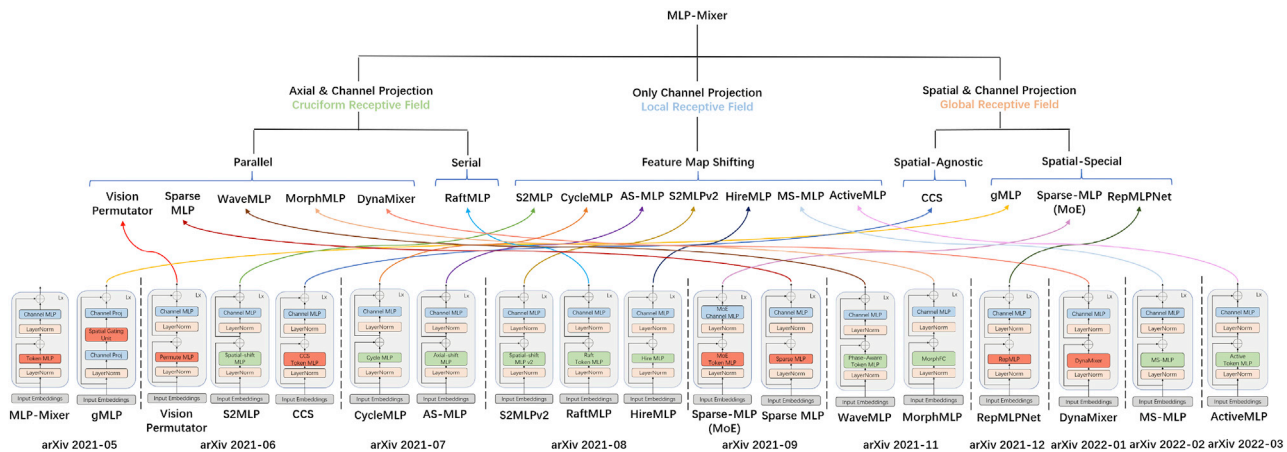
Figure 4. Block comparison of MLP-Mixer variants
Sorted from left to right by model release month ("arXiv 2021-05" denotes "May, 2021"). Red-orange, token-mixing modules are resolution sensitive; green, token-mixing modules are resolution insensitive.

learning generic visual features. Moreover, MLP's interpretability stands far behind.

Bottlenecks

Based on the above analysis and comparison, it is evident that the seemingly new paradigm still faces several bottlenecks:

1. Without the inductive biases of the local connectivity and the self-attention, the token-mixing MLP is more flexible and has a stronger fitting capability. However, it has a greater risk of over-fitting than the convolutional and self-attention layers. Thus, the large-scale dataset is needed to shorten the classification accuracy gap between MLP-Mixer, ViT, and CNN.
2. The complexity of the token-mixing MLP is quadratic to image size, which, for the current computing capability, makes it intractable for existing MLP-like models on high-resolution images.
3. The token-mixing MLP is resolution sensitive, and the network cannot deal with a flexible input resolution once the number of neurons in the fully connected layer is set. However, some tasks adopt a multi-scale training strategy[48] and have different input resolutions between training and evaluation stages.[65,66] In these cases, MLP models are non-transferable and impractical.

After several explorations and practices to address these challenges, the vision community has developed many MLP-like variants. Their main contributions are modifications to the token-mixing MLP, including reducing computational effort, removing resolution sensitivity, and reintroducing local receptive fields. These variants will be described in detail in the subsequent sections.

BLOCK OF MLP VARIANTS

To overcome the challenges faced by the MLP-Mixer, the vision community has made several attempts and proposed many MLP-like variants. The improvements focus on redesigning the

network's interior parts, i.e., the block module. The lower part of Figure 4 illustrates the block designs of the latest MLP-like variants, highlighting that they are primarily modified for token-mixing MLP. Except for the gMLP,[67] the remaining blocks retain the tandem spatial MLP and the channel MLP. Moreover, most of the improvements reduce the spatial MLP's sensitivity to image resolution (green rectangle).

In this work, we reproduce most variants of MLP-like models in Jittor[68,69] and Pytorch.[70] Moreover, this section first details the redesigned blocks of the latest MLP-like variants, then compares their properties and receptive field, and finally discusses the findings.

MLP block variants

We divide the MLP block variants into three categories: (1) mappings employing both the axial direction and channel dimensions, (2) mappings considering only channel dimensions, and (3) mappings utilizing the entire spatial and channel dimensions. The upper part of Figure 4 categorizes the network variants. Since the channel-mixing MLP is basically the same (except for the gMLP), we review and describe the changes to the token-mixing MLP.

Axial and channel projection blocks

The global receptive field of the initial token-mixing MLP is heavily parameterized and computationally very complex. Some researchers have proposed orthogonally decomposing the spatial projections and maintaining long-range dependence while no longer encoding spatial information along a flat spatial dimension.

Hou et al.[71] present the Vision Permutator (ViP), which separately encodes the feature representations along the height and width dimensions with linear projections. This design allows ViP to capture long-range dependencies along one spatial direction while preserving precise positional information along the other direction. As illustrated in Figure 5A, Permute-MLP comprises three branches responsible for encoding information along the height, width, or channel dimension. Specifically, it first splits the feature map into *g* segments along the channel
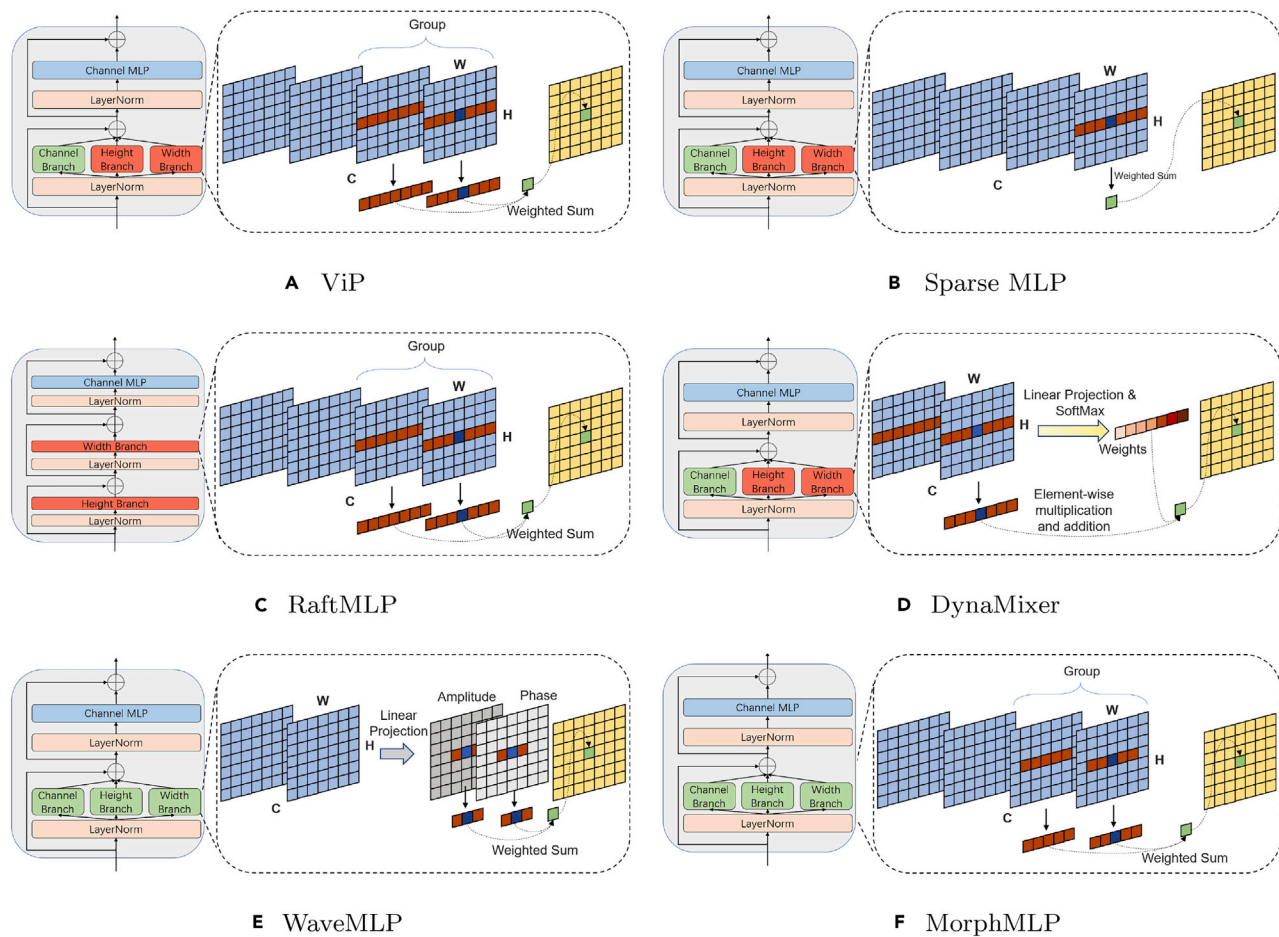
Figure 5. Visualizing the module diagram of the MLP-like variants
Visualizing the module diagram of the MLP-like variants, including ViP (A), Sparse MLP (B), RaftMLP (C), DynaMixer (D), WaveMLP (E), and MorphMLP (F). Except for RaftMLP, the remaining block designs adopt three-branch parallelism. Red-orange branch, image resolution sensitive; green branch, resolution insensitive. The channel branch and linear projection both perform a linear projection along the channel dimension. Group means that the channel is split into different groups. The expression in the dashed box is consistent with Figure 1.

dimension, where $g = C/H$ and the segments are then concatenated along the height dimension. Thus, it maps $\mathbb{R}^{gH} \mapsto \mathbb{R}^{gH}$ and is shared across the width and part channels. After mapping, the feature map is recovered to the original shape. The branch treatment in the width direction is the same as in the height direction. The third path is a simple mapping process along the channel dimension, which can also be regarded as a 1×1 convolution. Finally, the outputs from all three branches are fused by exploiting the split attention.[72]

Tang et al.[73] adopt a strategy consistent with ViP for spatial information extraction and build an attention-free network called Sparse MLP. As illustrated in Figure 5B, the block contains three parallel branches. The difference from ViP is no longer splitting the feature map along the channel dimension, but directly mapping $\mathbb{R}^H \mapsto \mathbb{R}^H$ and $\mathbb{R}^W \mapsto \mathbb{R}^W$ along the height and width dimension. Without the split attention, Sparse MLP's fusion strategy involves concatenating the three tributary outputs by channel and then passing them through a fully connected layer for dimensionality reduction. There is also a minor modification where Sparse MLP places a depthwise convolution in front of each block.

RaftMLP[74] employs serial mappings in high and wide dimensions to form a raft-token-mixing block (Figure 5C), which is different from the parallel branches of ViP and Sparse MLP. In the specific implementation, the raft-token-mixing block also splits the feature map along the channel dimension, consistent with ViP.

DynaMixer[75] generates mixing matrices dynamically for each set of tokens to be mixed by considering their contents. DynaMixer adopts the parallel strategy for a computational speedup and mixes tokens in a row-wise and column-wise way (Figure 5D). The proposed DynaMixer operation performs dimensionality reduction and flattening first and then utilizes a linear function to estimate a $H \times H$ or $W \times W$ mixing matrix. SoftMax is performed on each row of the mixing matrix to obtain the mixing weights. The output equals the product of the mix weights and the input.

WaveMLP[76] considers the dynamic aggregation tokens and image resolution issues. It considers each token a wave with both amplitude and phase information (Figure 5E). The tokens are aggregated according to their varying contents from different

input images with the dynamically produced phase. There are two parallel paths to aggregate spatial information along the horizontal and vertical directions, respectively. To address the issue of sensitivity to image resolution, WaveMLP adopts a simple strategy that restricts the fully connected layers only to connect tokens within a local window.[49] However, the local window limits long-range dependencies.

MorphMLP[77] considers long-range and short-range dependencies while continuing the static aggregation strategy (Figure 5F). It focuses on local details in the early stages of the network and gradually changes to long-term modeling in the later stages. The local window is used to solve the image resolution sensitivity problem, and the window size increases as the number of layers increases. The authors find that such a feature extraction model is beneficial for images and videos.

Discussion. ViP, Sparse MLP, RaftMLP, and DynaMixer encode spatial information along the axial direction instead of the entire plane, preserving the long-range dependence to a certain extent and reducing the complexity of parameters and the computational cost. However, they are still image resolution sensitive. WaveMLP and MorphMLP adopt a local window strategy but discard long-range dependencies. Furthermore, all those variants cannot mix tokens both globally and locally.

Channel-only projection blocks

The mainstream method adopts Swin's proposal[49] and uses a local window to achieve resolution insensitivity. Another approach replaces all the spatial fully connected layers with channel projection, i.e., 1×1 convolution. However, it causes the tokens to no longer interact with each other, and the concept of the receptive field disappears. To reintroduce the receptive field, many works align features at different spatial locations to the same channel by shifting (or moving) the feature maps and then interacting with spatial information through channel projection. Such an operation enables an MLP-like architecture to achieve the same local receptive field as a CNN-like architecture.

Yu et al.[78] propose a spatial-shift MLP-like architecture for vision, called S2MLP. The actual practice is quite simple. As shown in Figure 6A, the proposed spatial-shift module groups C channels into several groups, shifting different channel groups in different directions. The feature map after the shift aligns different token features to the same channel, and then the interaction of the spatial information can be realized after channel projection. Given the simplicity of this approach, Yu et al.[79] exploit the idea of ViP to extend the spatial-shift module into three parallel branches and then fuse the branch features by a split attention module to further improve the network's performance (Figure 6B). This newly proposed network is called S2MLPv2.

Unlike grouping and then performing the same shift operation for each group, the Axial Shifted MLP (AS-MLP)[80] performs different operations within each group that contains a few channels, e.g., every three feature maps in the channel direction are left-shifted, no-shifted, right-shifted, and so on (Figure 6C). In addition, AS-MLP uses two parallel branches for horizontal and vertical shifting, where the outputs are added element-wise and projected along the channel dimension for information integration. It is worth mentioning that AS-MLP is also extended with different shifting strategies, allowing the receptive field to be similar to the dilated convolution (atrous convolution).[81,82]

CycleMLP[83] was published three days after AS-MLP. Although CycleMLP does not directly shift feature maps, it integrates features at different spatial locations along the channel direction by employing deformable convolution,[84] an equivalent approach to shifting the feature map. As illustrated in Figure 6D, CycleMLP and AS-MLP slightly differ as CycleMLP has three branches and AS-MLP has only two branches. Furthermore, CycleMLP relies on split attention to fuse the branched features.

ActiveMLP[85] dynamically estimates the offset, rather than manually setting it like AS-MLP and CycleMLP do (Figure 6G). It first predicts the spatial locations of helpful contextual features along each direction at the channel level, and then finds and fuses them. This is equivalent to a 1×1 deformable convolution.[84]

HireMLP[86] adopts inner-region and cross-region rearrangements before channel projection to communicate spatial information. The inner-region rearrangement expands the feature map along the height or width direction, and the cross-region rearrangement moves the feature map cyclically along the width or height direction. The HireMLP block still comprises three parallel branches, and the output feature is obtained by adding the branched features (Figure 6E).

The six models mentioned above can communicate only localized information through feature map movement. MS-MLP[87] effectively expands the range of receptive fields by mixing neighboring and distant tokens from fine- to coarse-grained levels and then gathering them via a shifting operation. From the implementation aspect, MS-MLP performs depthwise convolution of different sizes before channel alignment (Figure 6F). Compared with the global receptive field or local window, there is still some vacancy in the receptive field of MS-MLP.

Discussion. After shifting the feature maps, channel projection is equivalent to sampling features at different locations in different channels for aggregation. In other words, this strategy is an artificially designed deformable convolution. Thus, it may be far better to call these models CNN-like, as only local feature extraction can be performed.

Spatial and channel projection blocks

Some variants still retain full space and channel projection. Their module designs are not short of sparkle and enhance performance. Nevertheless, these methods are resolution sensitive, prohibiting them from being a general vision backbone.

gMLP[67] is the first proposed MLP-Mixer variant. gMLP was developed by Liu et al., who experimented with several design choices for token-mixing architecture and found that spatial projections work well when they are linear and paired with multiplicative gating. In detail, the spatial gating unit first linearly projects the input X, $f_{W,b}(X) = WX + b$. Then the output of the spatial gating unit is $X \odot f_{W,b}(X)$, where $\odot$ denotes the element-wise product. The authors found it effective to split X into two independent parts (X_1, X_2) along the channel dimension for the gating function and the multiplicative bypass: $X_{out} = X_1 \odot f_{W,b}(X_2)$. Note that the gMLP block has a channel projection before and after the spatial gating unit. However, there is no longer a channel-mixing MLP. Pleasingly, gMLP achieves good performance in both computer vision and natural language processing tasks.
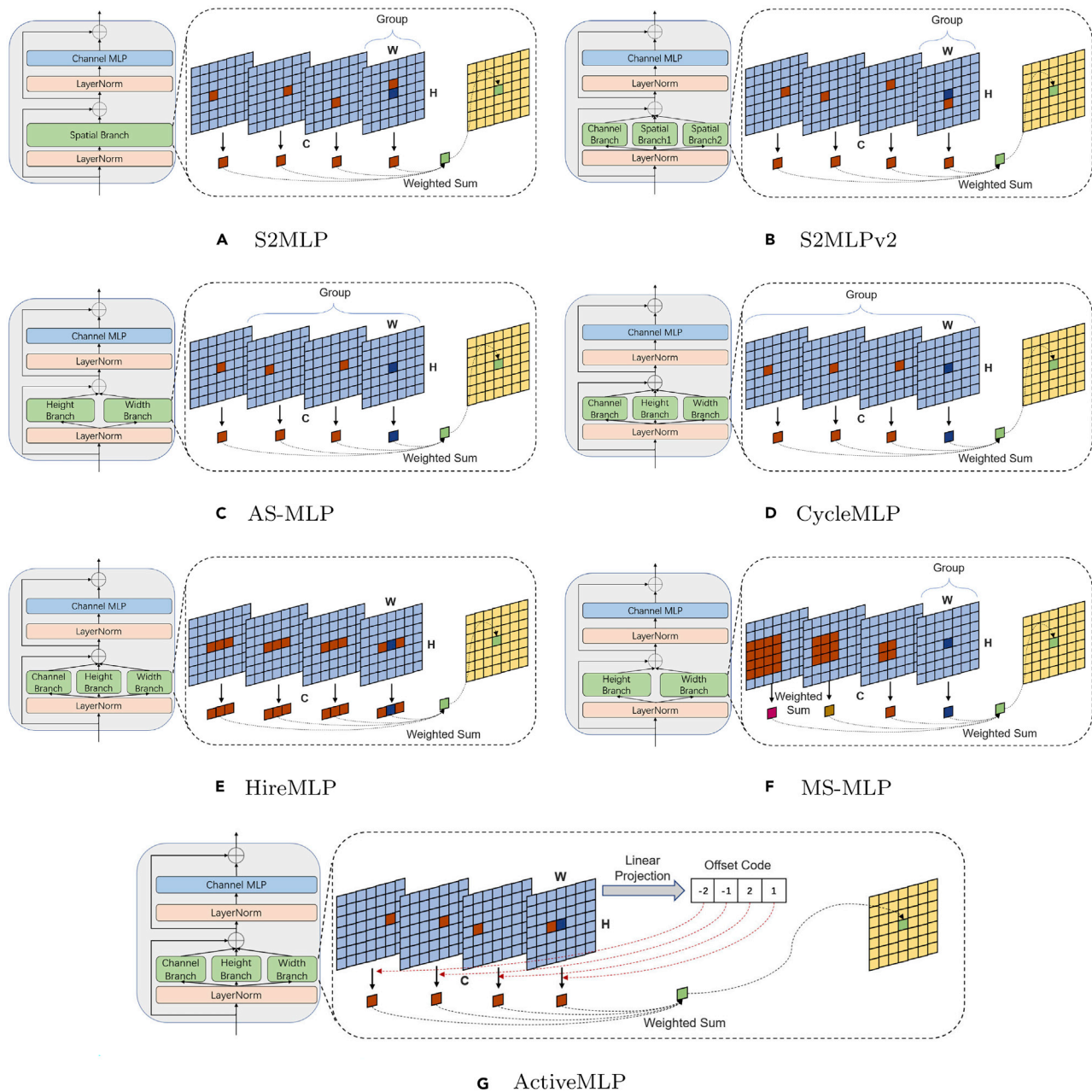
Figure 6. Visualizing the module diagram of the MLP-like variants
Visualizing the module diagram of the MLP-like variants, including S2MLP (A), S2MLPv2 (B), AS-MLP (C), CycleMLP (D), HireMLP (E), MS-MLP (F), and ActiveMLP (G). Green branch, resolution insensitive. The channel branch and linear projection both perform a linear projection along the channel dimension. Group means that the channel is split into different groups. The expression in the dashed box is consistent with Figure 1.

Lou et al.[88] consider how to scale the MLP-Mixer to more parameters with comparable computational cost, making the model more computationally efficient and better performing. Specifically, they introduce the mixture-of-experts (MoE)[89] scheme into the MLP-Mixer and propose Sparse-MLP(MoE). Carlos et al.[90] had already applied MoE on the MLP of the Transformer block, i.e., channel-mixing MLP in MLP-Mixer, a few months earlier. As a continuation of Carlos' work, Lou et al. expand MoE from a channel-mixing MLP to a token-mixing MLP and achieve some performance gains compared with the primitive MLP-Mixer. More details about MoE can be found in Shazeer and co-workers.[89,90]

Yu et al.[91] impose a circulant-structure constraint on the token-mixing MLP weights, reducing the spatial projection's sensitivity to spatial translation while preserving the global receptive field. It should be noted that the model is still resolution sensitive. The authors reduce the number of parameters from $\mathcal{O}(H^2W^2)$ to $\mathcal{O}(HW)$, but do not reduce the computation cost. Therefore, the

Table 2. Comparison of different spatial information fusion modules

Spatial operation	Block	Spatial	Channel	Resolution sensitive	Params	FLOPs
Spatial projection	MLP-Mixer[15]	specific	agnostic	true	$\mathcal{O}(H^2W^2)$	$\mathcal{O}(H^2W^2C)$
	ResMLP[60]	specific	agnostic	true	$\mathcal{O}(H^2W^2)$	$\mathcal{O}(H^2W^2C)$
	FeedForward[61]	specific	agnostic	true	$\mathcal{O}(H^2W^2)$	$\mathcal{O}(H^2W^2C)$
	gMLP[67]	specific	agnostic	true	$\mathcal{O}(H^2W^2)$	$\mathcal{O}(H^2W^2C)$
	Sparse-MLP (MoE)[88]	specific	agnostic	true	$\mathcal{O}(H^2W^2)$	$\mathcal{O}(H^2W^2C)$
	CCS[91]	agnostic	group specific	true	$\mathcal{O}(HW)$	$\mathcal{O}(HW\log(HW)C)$
	RepMLPNet[92]	specific	group specific	true	$\mathcal{O}(H^2W^2 + C^2)$	$\mathcal{O}(H^2W^2C)$
Axial projection	RaftMLP[74]	specific	agnostic	true	$\mathcal{O}(H^2 + W^2)$	$\mathcal{O}(HWC(H + W))$
	ViP[71]	specific	specific	true	$\mathcal{O}(H^2 + W^2 + C^2)$	$\mathcal{O}(HWC(H + W + C))$
	Sparse MLP[73]	specific	specific	true	$\mathcal{O}(H^2 + W^2 + C^2)$	$\mathcal{O}(HWC(H + W + C))$
	DynaMixer[75]	specific	specific	true	$\mathcal{O}(H^3 + W^3 + C^2)$	$\mathcal{O}(HWC(H^2 + W^2 + C))$
	WaveMLP[76]	specific	specific	false	$\mathcal{O}(2L^2 + C^2)$	$\mathcal{O}(HWC(L + L + C))$
	MorphMLP[77]	specific	specific	false	$\mathcal{O}(2L^2 + C^2)$	$\mathcal{O}(HWC(L + L + C))$
Shifting & channel projection	S2MLP[78]	agnostic	specific	false	$\mathcal{O}(C^2)$	$\mathcal{O}(HWC^2)$
	S2MLPv2[79]	agnostic	specific	false	$\mathcal{O}(C^2)$	$\mathcal{O}(HWC^2)$
	AS-MLP[80]	agnostic	specific	false	$\mathcal{O}(C^2)$	$\mathcal{O}(HWC^2)$
	CycleMLP[83]	agnostic	specific	false	$\mathcal{O}(C^2)$	$\mathcal{O}(HWC^2)$
	HireMLP[86]	agnostic	specific	false	$\mathcal{O}(C^2)$	$\mathcal{O}(HWC^2)$
	MS-MLP[87]	agnostic	specific	false	$\mathcal{O}(C^2)$	$\mathcal{O}(HWC^2)$
	ActiveMLP[85]	agnostic	specific	false	$\mathcal{O}(C^2)$	$\mathcal{O}(HWC^2)$

H, W, and C are the feature map's height, width, and channel numbers, respectively. L is the local window size. "Spatial" refers to whether feature extraction is sensitive to the spatial location of objects, "specific" means true, while "agnostic" means false. "Channel" refers whether weights are shared between channels, "agnostic" shares weights between all channels, "group specific" shares weights between groups, and "specific" does not share. "Resolution sensitive" refers to whether the module is resolution sensitive.

authors employ a fast Fourier transform to reduce the FLOPs and enhance computational efficiency.

Finally, and the most ingeniously, Ding et al.[92] propose a novel structural re-parameterization technique to merge the convolutional layers into the fully connected layers. Therefore, during training, the proposed RepMLPNet can learn parallel fully connected layers (global receptive field) and convolutional layers (local receptive field) and combine the two via transforming the parameters. Compared with the weight values of the fully connected layers before merging, the re-parameterization resultant weight has larger values around a specific position, suggesting that the model focuses more on the neighborhood. Although large images can be sliced and input to RepMLPNet for feature extraction, the resolution sensitivity makes it not a general vision backbone.

Others
ConvMLP[93] is another special variant, which is lightweight, stage-wise, and comprises a co-design of convolutional layers. ConvMLP replaces the token-mixing MLP with a 3× 3 depthwise convolution, and therefore we consider ConvMLP as a pure CNN model rather than an MLP-like model.

In addition, LIT[94] replaces the self-attention layers with MLP in the first two stages of a pyramid ViT. UniNet[95] jointly searches the optimal combination of convolution, self-attention, and MLP to build a series of all-operator network architectures with

high performances on visual tasks. However, these methods are beyond the scope of Vision MLP.

Receptive field and complexity analysis
The main novelty of MLP is allowing the model to *autonomously learn the global receptive field from raw data*. However, do these so-called MLP-like variants still hold to the original intent? Immediately following the module design, we compare and analyze the receptive field of these blocks to provide a more in-depth presentation for those MLP-like variants. Following that, we compare and analyze the complexity of different modules, with the corresponding results reported in Table 2. A comparison of the block's spatial sensitivity, channel sensitivity, and resolution sensitivity is also provided.

Receptive field
Receptive field was first used by Sherrington[96] in 1906 to describe the skin area that could trigger a scratch reflex in a dog. Nowadays, the term *receptive field* is also used in describing artificial neural networks, which is deemed as the size of the region in the input that produces the output value. It measures the relevance of an output feature to the input region. Different information aggregation methods will generate different receptive fields, which we divide into three categories, global, cruciform, and local. CNNs have local receptive fields, while vision Transformers own global receptive fields. Swin
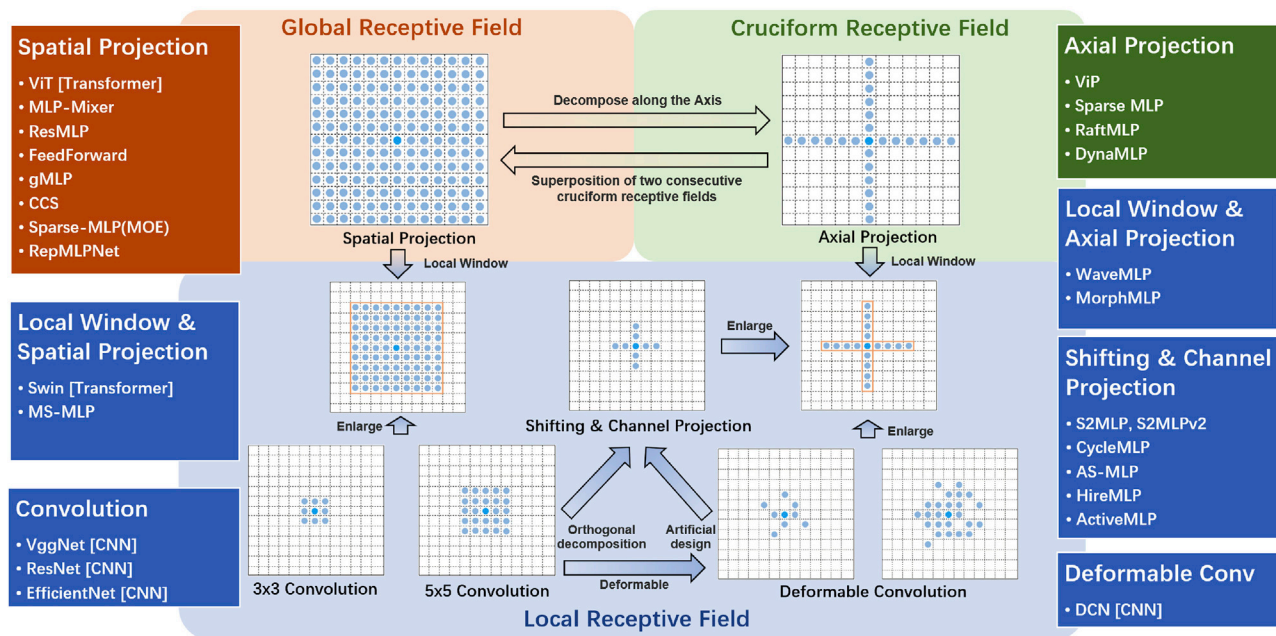
Figure 7. Schematic diagram of different receptive fields, including global, cruciform, and local
The bright blue dot indicates the position of the encoded token, and the light blue dots are other locations involved in the calculation. The range of the blue dots constitutes the receptive field. The orange rectangle is the local window, which covers a larger region than the commonly used convolutional kernels. The transition between different receptive fields is marked next to the arrow. The corresponding networks are listed on the left and right sides. Although MS-MLP[87] uses shifting and channel projection, its receptive field is most similar to the local window and spatial projection due to the depthwise convolution before shifting feature maps. The local receptive fields formed by MLP-like variants can be realized by convolution, causing them to lack an essential difference from CNNs.

Transformer[49] introduces the concept of local window, reducing the global receptive field to a fixed region independent of the image resolution, which is still more extensive than the standard convolutional layer. Figure 7 displays the schematic diagrams of different receptive fields and the corresponding MLP-like variants in detail.

Similar to MLP-Mixer,[15] the full spatial projection in gMLP,[67] Sparse-MLP(MoE),[88] CCS,[91] and RepMLPNet[92] still retain the global receptive field, i.e., the encoded features of each token are the weighted sum of all input token features. It must be acknowledged that this global receptive field is at the patch level, not at the pixel level in the traditional sense. In other words, the globalness is approximated by patch partition, similar to the Transformer-based methods. The size of the patch affects the final result and the network's computational complexity. Notably, the patch partition is a strong artificial assumption that is always ignored.

To balance long-distance dependence and computational cost, axial projection decomposes the full spatial projection orthogonally, i.e., along with horizontal and vertical directions. The projection on both axes is made serially (RaftMLP[74]) or parallel (ViP,[71] Sparse MLP,[73] and DynaMixer[75]). Thus, the token encoded only interacts with horizontal or vertical tokens in a single projection and forms a cruciform receptive field, which retains horizontal and vertical long-range dependence. However, if the token of interest and the current token are not at the same horizontal or vertical area, the two tokens cannot interact.

The global and cruciform receptive fields are required to cover the entire height and width of the space, resulting in a one-to-one correspondence between the neurons' number in the fully con-

nected layer and the image resolution, further restricting the model to utilize images of a specific size. To eliminate resolution sensitivity, many MLP-like variants (blue box in Figure 7) choose to use local receptive fields. Mainly, two approaches are adopted: local window and channel mapping after shifting feature maps. However, these operations can be achieved by expanding the convolutional kernel size and using deformable convolution, making these variants not fundamentally different from CNNs. A concern is that these MLP-like variants abandon the global receptive field, a significant MLP feature.

Virtually, the cruciform and local receptive field is a particular case of the global receptive field, e.g., the weight value is approximately 0 except for a specific area. Therefore, these two receptive field types are equivalent to learning only a small part of the weights of the global receptive field and setting the other weights to be constantly 0. This is also an artificial inductive bias, similar to the locality introduced by the convolutional layer.

Complexity analysis
Since all network blocks mentioned above contain the channel-mixing MLP module, where the number of parameters is $\mathcal{O}(C^2)$, and the FLOPs are $\mathcal{O}(HWC^2)$, we ignore this item in the later analysis and focus mainly on the module used for spatial information fusion, i.e., token-mixing MLP and its variants. Table 2 lists the comparison results of the attribute and complexity analysis of different spatial information fusion modules, where the network names are used to name each module for ease of understanding. The complexity is referenced from the analysis in the original papers.

The full spatial projection in the MLP-Mixer[15] and its variants contains $\mathcal{O}(H^2W^2)$ parameters and has $\mathcal{O}(H^2W^2C)$ FLOPs,

both quadratic with the image resolution. Theoretically, it is difficult to apply the network to large-resolution images with the current computational power. Hence, a compromise to enhance computational efficiency significantly increases the patch size, e.g., 14×14 or 16 × 16, and the information extracted is too coarse to discriminate small objects. The only difference is that CCS[91] constructs an $HW \times HW$ weight matrix using weight vectors of length HW and uses a fast Fourier transform to reduce the computational cost.

In contrast and similar to RaftMLP,[74] the axial projection, the orthogonal decomposition of the full spatial projection, reduces the parameter cardinality from $\mathcal{O}(H^2W^2)$ to $\mathcal{O}(H^2 + W^2)$, and the FLOPs from $\mathcal{O}(H^2W^2C)$ to $\mathcal{O}(HWC(H + W))$. If three parallel branches are used, the number of parameters and the FLOPs are $\mathcal{O}(H^2 + W^2 + C^2)$ and $\mathcal{O}(HWC(H + W + C))$, respectively. Notice that fusing information from multiple branches does not increase the computational complexity, such as splitting attention and dimensionality reduction after channel concatenation. If there is a local window, the number of parameters and FLOPs are further reduced to $\mathcal{O}(2L^2 + C^2)$ and $\mathcal{O}(HWC(L + L + C))$, where L is the window size, supposing a channel branch exists. DynaMixer[75] requires dynamic estimation of the weight matrix, which leads to higher complexity.

The approach based on shifting the feature map and channel projection further reduce computational complexity, i.e., the number of parameters is $\mathcal{O}(C^2)$ with $\mathcal{O}(HWC^2)$ FLOPs. MS-MLP[87] adds some depthwise convolutions, ActiveMLP[85] adds some channel projections, but both do not affect the overall complexity. Moreover, the number of weights is decoupled from the image resolution to no longer constrain these variants.

It is worth noting that computational complexity is only one of the determinants of inference time, as reshaping and shifting feature maps is also time-consuming. In addition, reducing complexity does not mean that the proposed network has fewer parameters. Conversely, the various networks retain a comparable number of parameters (Table 4). This allows networks of lower complexity to have more layers and more channels.

Discussion on block

The bottleneck of token-mixing MLP (Bottlenecks) induces researchers to redesign the block. Recently released MLP-like variants reduce the model's computational complexity, dynamic aggregation information, and resolve image resolution sensitivity. Specifically, researchers decompose the full spatial projection orthogonally, restrict interaction within a local window, perform channel projection after shifting feature maps, and make other artificial designs. These careful and clever designs demonstrate that researchers have noticed that the current amount of data and computational power is insufficient for pure MLPs. Comparing the computational complexity has a theoretical significance, but it is not the only determinant of inference time and final model efficiency. Analysis of the receptive field shows that the new paradigm is instead moving toward the old paradigm. To put it more bluntly, the development of MLP heads back to the way of CNNs. Hence, we still need to make efforts to balance long-distance dependence and image resolution sensitivity.

ARCHITECTURE OF MLP VARIANTS

Multiple blocks are stacked to form an architecture via selecting a network structure. According to our investigation, the traditional structures for classification models are also applicable to MLP-like architectures that can be divided into three categories (Figure 8): (1) a single-stage structure inherited from the ViT,[47] (2) a two-stage structure with smaller patches in the early stage and larger patches in the later stage, and (3) a CNN-like pyramid structure; and, in each stage, there are multiple identical blocks. Table 3 illustrates the stacking structures of MLP-Mixer and its variants.
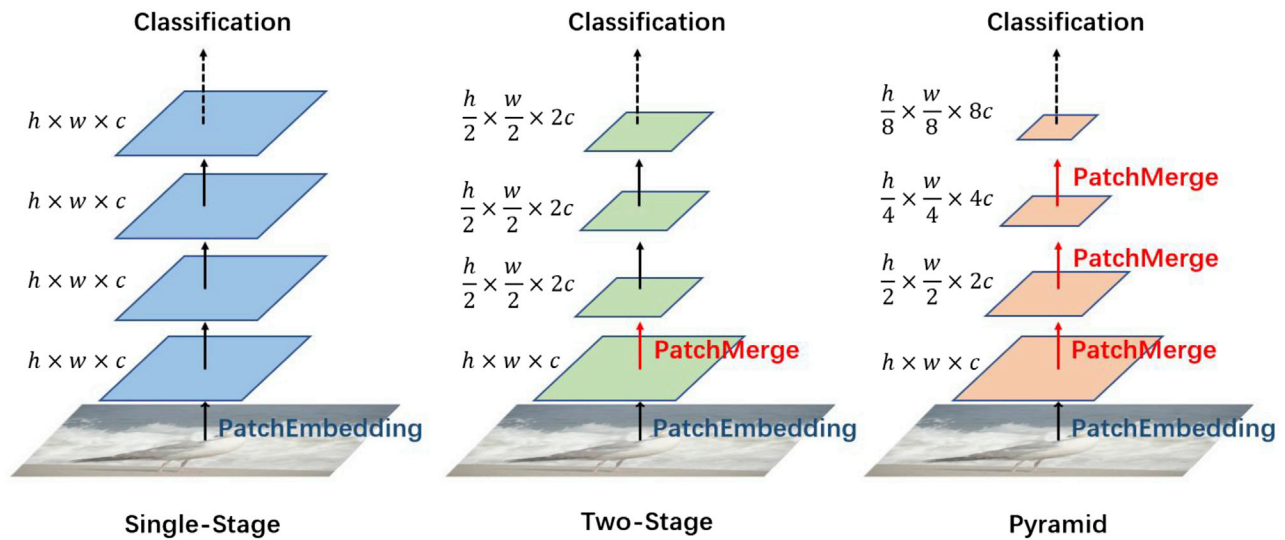
From single stage to pyramid

MLP-Mixer[15] inherits the "isotropic" design of ViT,[47] i.e., after the patch embedding each block does not change the size of the feature map. This is called a *single-stage* structure. Models of this design include FeedForward,[61] ResMLP,[60] gMLP,[67] S2MLP,[78] CCS,[91] RaftMLP,[74] and Sparse-MLP(MoE).[88] Due to the limited computing resources, the patch partition during patch embedding of the single-stage model is usually large, e.g., 16×16 or 14 × 14, with the coarse-grained patch partition limiting the subsequent feature fineness. Although the impact is not significant for single-object classification, it impacts many downstream tasks, such as object detection and instance segmentation, especially for small targets.

Intuitively, smaller patches are beneficial in modeling fine-grained details in the images and tend to achieve higher recognition accuracy. ViP[71] further proposes a *two-staged* configuration. Specifically, the network considers 7×7 patch slices in the initial patch embedding and performs a 2×2 patch merge after a few layers. During patch merging, the height and width of the feature map halve while the channels double. Compared with the 14×14 patch embedding, encoding fine-level patch representations brings a slight top 1 accuracy improvement on ImageNet-1k (from 80.6% to 81.5%). S2MLPv2[79] follows ViP and achieves a similar top 1 accuracy improvement on ImageNet-1k (from 80.9% to 82.0%), while DynaMixer[75] also adopts two-staged configurations.

If the initial patch size is further reduced, e.g., 4 × 4, more patch merging is required subsequently to reduce the number of patches (or tokens), promoting the network to adopt a pyramid structure. Specifically, the entire structure contains four stages, where the feature resolution reduces from $H/4 \times W/4$ to $H/32 \times W/32$, and the output dimension increases accordingly. Almost all the recently proposed MLP-like variants adopt the pyramid structure (right side of Table 3). Worth mentioning, a convolutional layer can equivalently achieve patch embedding if its kernel size and stride are equal to the patch size. In the pyramid structure, the researchers find that using an overlapping patch embedding provides better results, that is, convolution of 7×7 with stride = 4 instead of 4 × 4, which is similar to ResNet (the initial embedding layers of ResNet is a 7×7 convolutional layer with stride = 2 followed by a 2×2 max-pooling layer).[83,97]

Discussion on architecture

The architecture of MLP-like variants gradually evolves from single stage to pyramid, with smaller and smaller patch size and higher feature fineness. We believe this development trend is not only to cater to downstream task frameworks, such as

Figure 8. Comparison of different hierarchical architectures for classification models
After patch embedding, the feature map size is $h \times w \times c$, where h, w, and c are the height, width, and channel numbers. There is a patch merging operation between every two stages, usually 2×2 patches are merged, and the number of channels doubles. The resolutions of the feature maps are different, usually $h = H/16$ in single stage, $h = H/7$ in two stage, and $h = H/4$ in pyramid, where H and W are the height and width of the input image.

FPN,[98] but also to balance the original intention and computing power. When the patch size decreases, the number of tokens increases accordingly, so token interactions are limited to a small range to control the amount of computation. With a small range of token interactions, the global receptive field can only be preserved by reducing the size of the feature map, and the pyramid structure appears. This is consistent with the CNN concept, where the alternating use of convolutional and pooling layers has been around since 1979![3]

We would like to point out that it is unfair to compare single-stage and pyramid models directly based on current configurations (the bottom half of Figure 8). What if a 4×4 patch partition is used in a single-stage model? Will it be worse than the pyramid model? Currently, these are unknown. What is known is that the cost of calculations will increase significantly and is constrained by the current computing devices.

APPLICATIONS OF MLP VARIANTS

This section reviews the applications of MLP-like variants in computer vision, including image classification, object detection and semantic segmentation, low-level vision, video analysis, and

Table 3. Stacking structures for MLP-Mixer and its variants

Spatial operation	Single stage	Two stage	Pyramid
Spatial projection	MLP-Mixer,[15] ResMLP,[60] gMLP,[67] FeedForward,[61] CCS,[91] Sparse-MLP(MoE)[88]	–	RepMLPNet[92]
Axial projection	RaftMLP[74]	ViP,[71] DynaMixer[75]	Sparse MLP,[73] WaveMLP,[76] MorphMLP[77]
Shifting and channel projection	S2MLP[78]	S2MLPv2[79]	CycleMLP,[83] AS-MLP,[80] HireMLP,[86] MS-MLP,[87] ActiveMLP[85]

Table 4. Image classification results of MLP-like models on ImageNet-1K benchmark without extra data

Model	Date	Structure	Top 1 (%)	Params (M)	FLOPs (G)	Open source code
Small models						
Sparse-MLP(MoE)-S[88]	2021.09	single stage	71.3	21	–	false
RepMLPNet-T224[92]	2021.12	pyramid	76.4	15.2	2.8	true
ResMLP-12[60]	2021.05	single stage	76.6	15	3.0	true
Hire-MLP-Ti[86]	2021.08	pyramid	78.9	17	2.1	false[b]
gMLP-S[67]	2021.05	single stage	79.4	20	4.5	true
AS-MLP-T[80]	2021.07	pyramid	81.3	28	4.4	true
ViP-small/7[71]	2021.06	two stage	81.5	25	6.9	true
CycleMLP-B2[83]	2021.07	pyramid	81.6	27	3.9	true
MorphMLP-T[77]	2021.11	pyramid	81.6	23	3.9	false
Sparse MLP-T[73]	2021.09	pyramid	81.9	24.1	5.0	false
ActiveMLP-T[85]	2022.03	pyramid	82.0	27	4.0	false
S2-MLPv2-small/7[79]	2021.08	two stage	82.0	25	6.9	false
MS-MLP-T[87]	2022.02	pyramid	82.1	28	4.9	true
WaveMLP-S[76]	2021.11	pyramid	82.6	30.0	4.5	false[b]
DynaMixer-S[75]	2022.01	two stage	82.7*	26	7.3	false
Medium models						
FeedForward[61]	2021.05	single stage	74.9	62	11.4	true
Mixer-B/16[15]	2021.05	single stage	76.4	59	11.7	true
Sparse-MLP(MoE)-B[88]	2021.09	single stage	77.9	69	–	false
RaftMLP-12[74]	2021.08	single stage	78.0	58	12.0	false
ResMLP-36[60]	2021.05	single stage	79.7	45	8.9	true
Mixer-B/16 + CCS[91]	2021.06	single stage	79.8	57	11	false
RepMLPNet-B224[92]	2021.12	pyramid	80.1	68.2	6.7	true
S2-MLP-deep[78]	2021.06	single stage	80.7	51	9.7	false
ViP-medium/7[71]	2021.06	two stage	82.7	55	16.3	true
CycleMLP-B4[83]	2021.07	pyramid	83.0	52	10.1	true
AS-MLP-S[80]	2021.07	pyramid	83.1	50	8.5	true
Hire-MLP-B[86]	2021.08	pyramid	83.1	58	8.1	false[b]
MorphMLP-B[77]	2021.11	pyramid	83.2	58	10.2	false
Sparse MLP-B[73]	2021.09	pyramid	83.4	65.9	14.0	false
MS-MLP-S[87]	2022.02	pyramid	83.4	50	9.0	true
ActiveMLP-B[85]	2022.03	pyramid	83.5	52	10.1	false
S2-MLPv2-medium/7[79]	2021.08	two stage	83.6	55	16.3	false
WaveMLP-B[76]	2021.11	pyramid	83.6	63.0	10.2	false[b]
DynaMixer-M[75]	2022.01	two stage	83.7*	57	17.0	false
Large Models						
Sparse-MLP(MoE)-L[88]	2021.09	single stage	79.2	130	–	false
S2-MLP-wide[78]	2021.06	single stage	80.0	71	14.0	false
gMLP-B[67]	2021.05	single stage	81.6	73	15.8	true
RepMLPNet-L256[a,92]	2021.12	pyramid	81.8	117.7	11.5	true
ViP-large/7[71]	2021.06	two stage	83.2	88	24.3	true
CycleMLP-B5[83]	2021.07	pyramid	83.2	76	12.3	true
AS-MLP-B[80]	2021.07	pyramid	83.3	88	15.2	true
Hire-MLP-L[86]	2021.08	pyramid	83.4	96	13.5	false[b]
MorphMLP-L[77]	2021.11	pyramid	83.4	76	12.5	false
ActiveMLP-L[85]	2022.03	pyramid	83.6	76	12.3	false

Table 4. *Continued*

Model	Date	Structure	Top 1 (%)	Params (M)	FLOPs (G)	Open source code
MS-MLP-B[87]	2022.02	pyramid	83.8	88	16.1	true
DynaMixer-L[75]	2022.01	two stage	84.3*	97	27.4	false

The training and testing size is 224 × 224. "Date" means the initial release date on arXiv, where 2021.05 denotes May, 2021. "Open source code" refers to whether there is officially open source code.
[a]The training and testing size is 256 × 256.
[b]Unofficial code and weights are open sourced at https://github.com/sithu31296/image-classification.
*The best performance.

point cloud. Due to the short development time of MLP, we focus on the first two aspects and give an intuitive comparison of MLP, CNN, and Transformer-based models. We are limited to some brief introduction for the latter three aspects, as only a few works are currently available.

Image classification

ImageNet[24] is a large vision dataset designed for visual object classification. Since its release, it has been used as a benchmark for evaluating models in computer vision. Classification performance on ImageNet is often regarded as a reflection of the network's ability to extract visual features. After training on ImageNet, the model can be well transferred to other datasets and downstream tasks, e.g., object detection and segmentation, where the transferred part is usually called *vision backbone*.

Table 4 compares the performance of current Vision MLP models on ImageNet-1k, including top 1 accuracy, parameters, and FLOPs, where all results are derived from the cited papers. We further divide the MLP models into three configuration types based on the number of parameters, and the rows are sorted by top 1 accuracy. The results highlight that, under the same training configuration, the recently proposed variants bring good performance gains. Table 5 provides more detailed and comprehensive information. Compared with the latest CNN and Transformer models, MLP-like variants still pose a performance gap. Without the support of extra training data, both CNN and Transformer exceed 87% top 1 accuracy, while the MLP-like variant currently achieves only 84.3%. High performance may benefit from better architecture-specific training strategies, e.g., PeCo,[99] but we do not yet have a training mode specific to MLP. The gap between MLP and other networks is further widened with additional data support.

From Tables 4 and 5, it is concluded that: (1) MLP-like models can achieve competitive performance compared with CNN-based and Transformer-based architectures with the same training strategy and data volume, (2) the performance gains brought by increasing data volume and architecture-specific training strategies may be greater than the module redesign, (3) the visual community is encouraged to build self-supervised methods and appropriate training strategies for pure MLPs.

Object detection and semantic segmentation

Some MLP-like variants[76,77,80,83,85–87] pre-trained on ImageNet are transferred to downstream tasks, such as object detection and semantic segmentation. Such tasks are more challenging than classification due to involving multiple objects of interest in one input image. However, we currently do not have a pure MLP framework for object detection and segmentation. These MLP variants are used as backbone networks to traditional CNN-based frameworks, such as Mask R-CNN[38] and UperNet,[123] requiring the variant to have a pyramidal structure and resolution insensitivity.

Table 6 reports object detection and semantic segmentation results of different backbones on the COCO val2017 dataset.[65] As we limit the training strategy to Mask R-CNN 1x,[38] the results are not state-of-the-art on the COCO dataset. Table 7 reports semantic segmentation results of different backbones on the ADE20K[124] validation set, employing the Semantic FPN[125] and UperNet[123] frameworks. Empirical results show that the performance of MLP-like variants on object detection and semantic segmentation is still weaker than the most advanced CNN and Transformer-based backbones.

Currently, an optimal backbone choice is Transformer based, followed by CNN. Due to the resolution sensitivity, pure MLPs have not been used for downstream tasks. Recently, Transformer-based frameworks, e.g., DETR[48] have been proposed. Thus, we expect the proposal of a pure MLP framework. To this end, MLPs still need to be further explored in these fields.

Low-level vision

Research on applying MLPs to the low-level vision domains, such as image generation and processing, is just beginning. These tasks output images instead of labels or boxes, making them more challenging than high-level vision tasks, such as image classification, object detection, and semantic segmentation.

Cazenavette and Guevara[128] propose MixerGAN for unpaired image-to-image translation. Specially, MixerGAN adopts the framework of CycleGAN,[129] but replaces the convolution-based residual block with the mixer layer of MLP-Mixer. Their experiments show that the MLP-Mixer succeeds at generative objectives and, although being an initial exploration, it is promising in extending the MLP-based architecture to image composition tasks further.

Tu et al.[130] propose MAXIM, a UNet-shaped hierarchical structure that supports long-range interactions enabled by spatially gated MLPs. MAXIM contains two MLP-based building blocks: a multi-axis-gated MLP and a cross-gating block, both are variants of the gMLP block.[67] By applying gMLP to low-level vision tasks to gain global information, the MAXIM family has achieved state-of-the-art performance in multiple image processing tasks with moderate complexity, including image dehazing, deblurring, denoising, deraining, and enhancement.

Table 5. Image classification results of representative CNN, ViT, and MLP-like models on ImageNet-1K benchmark

Model	Pre-trained dataset	Top 1 (%)	Params (M)	FLOPs (G)
CNN based				
VGG-16[100]	–	71.5	134	15.5
Xception[16]	–	79.0	22.9	–
Inception-ResNet-V2[101]	–	80.1	–	–
ResNet-50[97,102]	–	80.4	25.6	4.1
ResNet-152[97,102]	–	82.0	60.2	11.5
RegNetY-8GF[102,103]	–	82.2	39	8.0
RegNetY-16GF[103]	–	82.9	84	15.9
ConvNeXt-B[104]	–	83.8	89.0	15.4
VAN-Huge[105]	–	84.2	60.3	12.2
EfficientNetV2-M[106]	–	85.1	54	24.0
EfficientNetV2-L[106]	–	85.7	120	53.0
PolyLoss(EfficientNetV2-L)[107]	–	87.2[1]	–	–
EfficientNetV2-XL[106]	ImageNet-21k	87.3	208	94.0
RepLKNet-XL[108]	ImageNet-21k	87.8[2]	335	128.7
Meta pseudo labels (EfficientNet-L2)[109]	JFT-300M	90.2[3]	480	–
Transformer based				
ViT-B/16[47]	–	77.9	86	55.5
DeiT-B/16[110]	–	81.8	86	17.6
T2T-ViT-24[111]	–	82.3	64.1	13.8
PVT-large[112]	–	82.3	61	9.8
Swin-B[49]	–	83.5	88	15.4
Nest-B[113]	–	83.8	68	17.9
PyramidTNT-B[114]	–	84.1	157	16.0
CSWin-B[115]	–	84.2	78	15.0
CaiT-M-48-448[116]	–	86.5	356	330
PeCo(ViT-H)[99]	–	88.3[1]	635	–
ViT-L/16[47]	ImageNet-21k	85.3	307	–
SwinV1-L[49]	ImageNet-21k	87.3	197	103.9
SwinV2-G[117]	ImageNet-21k	90.2[2]	3000	–
V-MoE[90]	JFT-300M	90.4	14,700	–
ViT-G/14[47]	JFT-300M	90.5[3]	1843	–
CNN + Transformer				
Twins-SVT-B[118]	–	83.2	56	8.6
Shuffle-B[119]	–	84.0	88	15.6
CMT-B[120]	–	84.5	45.7	9.3
CoAtNet-3[121]	–	84.5	168	34.7
VOLO-D3[122]	–	85.4	86	20.6
VOLO-D5[122]	–	87.1[1]	296	69.0
CoAtNet-4[121]	ImageNet-21k	88.1[2]	275	360.9
CoAtNet-7[121]	JFT-300M	90.9[3]	2440	–
MLP based				
DynaMixer-L[75]	–	84.3[1]	97	27.4
ResMLP-B24/8[60]	ImageNet-21k	84.4[2]	129.1	100.2
Mixer-H/14[15]	JFT-300M	86.3[3]	431	–

Pre-trained dataset column provides extra data information. PloyLoss, PeCo, and meta pseudo labels are different training strategies, where the used model is in the bracket

[1]The best performance on ImageNet-1k without pre-trained dataset.
[2]The best performance on ImageNet-1k with ImageNet-21k pre-training.
[3]The best performance on ImageNet-1k with JFT-300M pre-training.

Table 6. Object detection and instance segmentation results of different backbones on the COCO val2017 dataset

Backbone	Mask R-CNN 1 × [38]						Params	FLOPs
	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}		
CNN based								
ResNet101[97]	40.4	61.1	44.2	36.4	57.7	38.8	63.2M	336G
ResNeXt101[126]	42.8	63.8	47.3	38.4	60.6	41.3	101.9M	493G
VAN-large[105]	47.1	67.9	51.9	42.2	65.4	45.5	64.4M	–
Transformer based								
PVT-large[112]	42.9	65.0	46.6	39.5	61.9	42.5	81M	364G
Swin-B[49]	46.9	–	–	42.3	–	–	107M	496G
CSWin-B[115]	48.7*	70.4*	53.9*	43.9*	67.8*	47.3	97M	526G
MLP based								
CycleMLP-B5[83]	44.1	65.5	48.4	40.1	62.8	43.0	95.3M	421G
WaveMLP-B[76]	45.7	67.5	50.1	27.8	49.2	59.7*	75.1M	353G
HireMLP-L[86]	45.9	67.2	50.4	41.7	64.7	45.3	115.2M	443G
MS-MLP-B[87]	46.4	67.2	50.7	42.4	63.6	46.4	107.5M	557G
ActiveMLP-L[85]	47.4	69.9	52.0	43.2	67.3	46.5	96.0M	–

Employing the Mask R-CNN,[38] where "1x" means that a single-scale training schedule is used.
*The best performance.

Video analysis

Several works extend MLPs to temporal modeling and video analysis. MorphMLP[77] achieves competitive performance with recent state-of-the-art methods on Kinetics-400[131] dataset, demonstrating that the MLP-like backbone is also suitable for video recognition. Skating-Mixer[132] extends the MLP-Mixer-based framework to learn from video. It was used to score figure skating at the Beijing 2022 Winter Olympic Games and demonstrated satisfactory performance. However, compared with other methods, the number of frames in a single input has not been increased. Therefore, their advantage may be the larger spatial receptive field, instead of capturing long-term temporal information.

Point cloud

Point cloud analysis can be considered a special vision task, which is increasingly used in real-time by robots and self-driving vehicles to understand their environment and navigate through it. Unlike images, point clouds are inherently sparse, unordered, and irregular. The unordered nature is one of the biggest challenges for CNNs based on local receptive fields, because input adjacent does not imply spatial adjacent. In contrast, MLPs are naturally invariant to permutation, which perfectly fits the characteristic of point cloud,[133] making classical frameworks, such as MLP-based PointNet[134] and PointNet++.[135]

Choe et al.[136] design PointMixer, which embeds geometric relations between point features into the MLP-Mixer's framework. The relative position vector is utilized for processing unstructured 3D points, and token-mixing MLP is replaced with a softmax function. Ma et al.[133] construct a pure residual MLP network, called PointMLP. It introduces a lightweight geometric affine module to transform the local points to a normal distribution. It then employs simple residual MLPs to represent local points, as they are permutation invariant and straightforward. PointMLP achieves the new state-of-the-art performance on multiple datasets. In addition, recently proposed Transformer-based networks[137,138] show competitive performance, where

self-attention is permutation invariant for processing a sequence of points, making it well suited for point cloud learning.

From the above analysis, it is evident that Transformer and MLP are appealing solutions for unordered data, where disorder makes it challenging to design artificial inductive biases.

Discussion on application

MLP-like variants have been applied for diverse vision tasks, such as image classification, image generation, object detection, semantic segmentation, and video analysis, achieving outstanding performance due to the artificial redesigning of the MLP block. Nevertheless, constructing MLP frameworks and employing MLP-specific training strategies may improve performance further. In addition, pure MLPs have already demonstrated their advantages in point cloud analysis, encouraging the application of MLPs to visual tasks with unordered data.

SUMMARY AND OUTLOOK

As the history of computer vision attests, the availability of larger datasets along with the increase in computational power often triggers a paradigm shift; and, within these paradigm shifts, there is a gradual reduction in human intervention, i.e., removing hand-crafted inductive biases and allowing the model to further freely learn from the raw data.[15] The MLP and Boltzmann machines proposed in the last century exceeded the computational conditions at the time and were not widely used. In contrast, computationally efficient CNNs are more popular and replace manual feature extraction. From CNNs to Transformer, we have seen the models' receptive field expand step by step, and the spatial range considered when encoding features is getting larger and larger. From Transformer to deep MLP, we no longer use similarity as the weight matrix, but allow the model to learn the weights from the raw data. The latest MLP works all seem to suggest that

Table 7. Semantic segmentation results of different backbones on the ADE20K validation set

Backbone	Semantic FPN[125]			UperNet[123]		
	Params	FLOPs	mIoU (%)	Params	FLOPs	mIoU (%)
CNN based						
ResNet101[97]	47.5M	260G	38.8	86M	1029G	44.9
ResNeXt101[126]	86.4M	–	40.2	–	–	–
VAN-large[105]	49.0M	–	48.1	75M	–	50.1
ConvNeXt-XL[104]	–	–	–	391M	3335G	54.0
RepLKNet-XL[108]	–	–	–	374M	3431G	56.0
Transformer based						
PVT-medium[112]	48.0M	219G	41.6	–	–	–
Swin-B[49]	53.2M	274G	45.2	121M	1188G	49.7
CSWin-B[115]	81.2M	464G	49.9*	109.2M	1222G	52.2
BEiT-L[127]	–	–	–	–	–	57.0
SwinV2-G[117]	–	–	–	–	–	59.9*
MLP based						
MorphMLP-B[77]	59.3M	–	45.9	–	–	–
CycleMLP-B5[83]	79.4M	343G	45.6	–	–	–
Wave-MLP-M[76]	43.3M	231G	46.8	–	–	–
AS-MLP-B[80]	–	–	–	121M	1166G	49.5
HireMLP-L[86]	–	–	–	127M	1125G	49.9
MS-MLP-B[87]	–	–	–	122M	1172G	49.9
ActiveMLP-L[85]	79.8M	–	48.1	108M	1106G	51.1

Semantic FPN[125] and UperNet[123] frameworks are employed.
*The best performance.

deep MLPs are making a strong comeback as the new paradigm. In the latest MLP development, we see compromises, such as:

1. The latest proposed deep MLP-based models use patch partition instead of flattening the entire input to compromise computational cost. This allows the full connectivity and global receptive field to be approximated at the patch level. The patch partition forms a two-dimensional matrix $hw \times Cp^2$, instead of a one-dimensional vector $1 \times HWC$ as in the entire input flattening case, where $H = ph, W = pw$ are the input resolution, p is the patch size, and C is the input channel. Subsequently, the fully connected projections are performed alternately on space and channels. This is an orthogonal decomposition of traditional fully connected projections, just as the full space projection is further orthogonalized into horizontal and vertical directions.

2. At the module design level, there are two main improvement routes. One MLP-like variant type focuses on reducing computational complexity, which compromises computational power, and this reduction comes at the expense of decoupling the full spatial connectivity. Another type of variant addresses the resolution sensitivity problem, making it possible to transfer pre-trained models to downstream tasks. These works adopt CNN-like improvements, but the full connectivity and global receptive field as in MLPs are eroded. The receptive field evolves in the opposite direction in these models, becoming smaller and smaller and backing to the CNN ways.

3. At the architecture level, the traditional block-stacking patterns are also applicable to MLPs, and it seems that the pyramid structure is still the best choice, with the initial smaller patch size helping to obtain finer features. Note that this comparison is unfair because the initial patch of the current single-stage model is larger (16 × 16), and the initial patch of the pyramid model is smaller (4 × 4). The pyramid structure compromises small patches and low computational costs to some extent. What if a 4×4 patch partition is used in a single-stage model? Will it be worse than the pyramid model? These are unknown. What is known is that the cost of calculations will increase significantly, and it strains the current computing devices.

The results of our research suggest that the current amount of data and computational capability are still not enough to support pure Vision MLP models to learn effectively. Moreover, human intervention still occupies an important place. Based on this conclusion, we elaborate on potential future research directions.

Vision-tailored designs

With the current amount of data and computation, human guidance remains important, and it seems natural to combine the advantages of other architectures.[95,139] Currently, most MLP-like variants remain an either/or choice for short- and long-range dependencies and need further intuitions to enhance their efficiency on visual inputs. RepMLPNet[92] has made a viable attempt. We believe that, in the future, the community should focus on how to combine short-range dependencies and long-range

dependencies rather than keeping only one or the other. This is consistent with human intuition as local details are beneficial for understanding individual objects and the interactions across the entire visual field remain significant. Note that image resolution insensitivity is important, ensuring the network to be a universal vision backbone. To sum up, we encourage the community to rethink tailored visual designs further, i.e., to integrate the global receptive field (long-range dependencies) and local receptive field (short-range dependencies) while maintaining resolution insensitivity.

Scaling-up/down techniques

It has long been recognized that larger vision models generally perform better on vision tasks,[97,100] but the size of most vision networks is only a few million to over a hundred million. Furthermore, various configurations of an MLP-like variant offer limited gains despite the increased number of parameters. Recently, the visual community has conducted some scaling-up research on the vision Transformers with self-supervised pre-training, including V-MoE,[90] Swinv2,[117] and ViTAEv2,[140] which afford a considerable performance boost. Nevertheless, scaling-up techniques specific to MLPs need further exploration. Moving from the lab to life, MLPs can have intensive power and computation requirements, hindering their deployment on edge devices and resource-constrained environments, such as mobile phone platforms. Such hardware efficient designs are currently lacking for the vision MLPs to enable their seamless deployment in resource-constrained devices. How do MLPs perform with low precision training and inference? How do MLPs perform knowledge distillation? How about using neural architecture search[141] to design more efficient and lightweight MLP models? It will be interesting to see how these questions are answered.

Dedicated pre-training and optimizing method

MLPs enjoy greater freedom and larger solution space with less inductive biases than CNNs and Transformer. We currently have difficulties finding optimal solutions for MLPs, which is commonly attributed to computation power and data volume constraints. Pre-training helps generalization, and the very limited information in the labels is used only to slightly adjust the weights found by pre-training. Many self-supervised frameworks, such as MoCo,[28,29,142] SimCLR,[27,143] SimMIM,[144] MAE,[145] and MaskFeat,[146] have already provided a big boost to CNNs and Transformer. Are these self-supervised learning methods still effective for MLPs? Can a better self-supervised training method be designed for MLPs? What's more, is this related to the optimizer used? We know that SGD[147] is a good optimizer for CNNs and that AdamW[148] performs well for Transformer. What is the best choice for MLPs? A recent work[149] has conducted a preliminary exploration and investigates the MLP-Mixer from the lens of loss landscape geometry. GGA-MLP[150] proposes a greedy genetic algorithm to optimize weights and biases in MLP. We believe that dedicated pre-training and optimizing methods will be an excellent boost to accelerate the development of deep MLP models.

Interpretability

Another optional direction is toward a more in-depth analysis and comparison of the filters learned by the network and the re-

sulting feature maps. MLPs continue the long-term trend of removing hand-crafted inductive biases and allowing models to further freely learn from the raw data. What follows is that the interpretability of the model is getting lower and lower. Both mathematical explanations and visual analysis are possibly helpful to understand what neural networks can freely learn from massive amounts of raw data with fewer priors. This can assist in proving whether some of the past artificial priors are correct or incorrect and potentially guide the design choices of future networks. In addition, the theoretical understanding of why networks might be vulnerable is also a key topic.

Beyond MLP

With the further improvement of the data volume and computing power, we should be beyond the horizon of present knowledge, the weighted-sum paradigms, and reconsider more theoretical paradigms from mathematic and physic systems, such as Boltzmann machines. The weighted-sum paradigms have driven the booming of GPU-based computing and deep learning itself, while we believe the Boltzmann-like paradigm will also grow up with a new generation of computing hardware.

Data and code availability

We reproduce most variants of MLP-like models in Jittor and Pytorch. Code is available at https://github.com/liuruiyang98/Jittor-MLP. This review did not generate new data.

AUTHOR CONTRIBUTIONS

L.T. designed the roadmap of the article. L.T. and R.L. initiated the project and supervised the team in collaboration with Y.L., D.L., H.-T.Z., and L.T. R.L. and Y.L. drafted the manuscript. R.L. and D.L. reproduced and open sourced the code of MLP-like variants in Jittor. All co-authors participated in discussions and revised the manuscript iteratively.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

1. McCulloch, W.S., and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys. *5*, 115–133. https://doi.org/10.1007/bf02478259.

2. Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. Psychol. Rev. *65*, 386–408. https://doi.org/10.1037/h0042519.

3. Fukushima, K. (1979). Neural network model for a mechanism of pattern recognition unaffected by shift in position-neocognitron. IEICE Technical Report. A *62*, 658–665.

4. LeCun, Y., Boser, B.E., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W.E., and Jackel, L.D. (1989). Backpropagation applied to

handwritten zip code recognition. Neural Comput. *1*, 541–551. https://doi.org/10.1162/neco.1989.1.4.541.

5. LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. Proc. IEEE *86*, 2278–2324. https://doi.org/10.1109/5.726791.

6. Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1985). Learning internal representations by error propagation. In Tech. Rep. (California Univ San Diego La Jolla Inst for Cognitive Science).

7. Ackley, D.H., Hinton, G.E., and Sejnowski, T.J. (1985). A learning algorithm for Boltzmann machines. Cogn. Sci. *9*, 147–169. https://doi.org/10.1207/s15516709cog0901_7.

8. Smolensky, P. (1986). Information processing in dynamical systems: foundations of harmony theory. In Tech. Rep. (Colorado Univ at Boulder Dept of Computer Science).

9. Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. Math. Control, Signals, Syst. MCSS *2*, 303–314. https://doi.org/10.1007/BF02551274.

10. Hornik, K., Stinchcombe, M.B., and White, H. (1989). Multilayer feedforward networks are universal approximators. Neural Network. *2*, 359–366. https://doi.org/10.1016/0893-6080(89)90020-8.

11. Funahashi, K.I. (1989). On the approximate realization of continuous mappings by neural networks. Neural Network. *2*, 183–192. https://doi.org/10.1016/0893-6080(89)90003-8.

12. Hinton, G.E., and Sejnowski, T.J. (1986). Learning and relearning in Boltzmann machines. Parallel distributed processing: Explorations in the microstructure of cognition *1*, *2*.

13. Hinton, G.E. (1989). Deterministic Boltzmann learning performs steepest descent in weight-space. Neural Comput. *1*, 143–150. https://doi.org/10.1162/neco.1989.1.1.143.

14. Tanaka, T. (1998). Mean-field theory of Boltzmann machine learning. Phys. Rev. *58*, 2302–2310. https://doi.org/10.1103/physreve.58.2302.

15. Tolstikhin, I.O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., et al. (2021). Mlp-mixer: an all-mlp architecture for vision. Preprint at arXiv. https://doi.org/10.48550/arXiv.2105.01601.

16. Chollet, F. (2017). Xception: deep learning with depthwise separable convolutions. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017 (IEEE Computer Society), pp. 1800–1807. https://doi.org/10.1109/CVPR.2017.195.

17. Minsky, M., and Papert, S. (1969). Perceptrons.

18. Hinton, G.E., and Salakhutdinov, R.R. (2006). Reducing the dimensionality of data with neural networks. Science *313*, 504–507. https://doi.org/10.1126/science.1127647.

19. Fukushima, K., and Miyake, S. (1982). Neocognitron: a new algorithm for pattern recognition tolerant of deformations and shifts in position. Pattern Recognit. *15*, 455–469. https://doi.org/10.1016/0031-3203(82)90024-3.

20. Fukushima, K., Miyake, S., and Ito, T. (1983). Neocognitron: a neural network model for a mechanism of visual pattern recognition. IEEE Trans. Syst. Man Cybern. *13*, 826–834. https://doi.org/10.1109/TSMC.1983.6313076.

21. Schmidhuber, J. (2007). New millennium AI and the convergence of history. In Challenges for Computational Intelligence, *63*, W. Duch and J. Mandziuk, eds. (Springer), pp. 15–35. https://doi.org/10.1007/978-3-540-71984-7_2.

22. Nickolls, J., Buck, I., Garland, M., and Skadron, K. (2008). Scalable parallel programming with cuda: is cuda the parallel programming model that application developers have been waiting for? Queue *6*, 40–53. https://doi.org/10.1145/1365490.1365500.

23. Lindholm, E., Nickolls, J., Oberman, S.F., and Montrym, J. (2008). NVIDIA tesla: a unified graphics and computing architecture. IEEE Micro. *28*, 39–55. https://doi.org/10.1109/MM.2008.31.

24. Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Fei-Fei, L. (2009). Imagenet: a large-scale hierarchical image database. In 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA (IEEE Computer Society), pp. 248–255. https://doi.org/10.1109/CVPR.2009.5206848.

25. Krizhevsky, A., Sutskever, I., and Hinton, G.E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States, P.L. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, eds., pp. 1106–1114.

26. Zhang, J., Yu, J., and Tao, D. (2018). Local deep-feature alignment for unsupervised dimension reduction. IEEE Trans. Image Process. *27*, 2420–2432. https://doi.org/10.1109/TIP.2018.2804218.

27. Chen, T., Kornblith, S., Norouzi, M., and Hinton, G.E. (2020a). A simple framework for contrastive learning of visual representations. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, *119* (PMLR), pp. 1597–1607.

28. He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R.B. (2020). Momentum contrast for unsupervised visual representation learning. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020 (Computer Vision Foundation/IEEE), pp. 9726–9735. https://doi.org/10.1109/CVPR42600.2020.00975.

29. Chen, X., Fan, H., Girshick, R.B., and He, K. (2020b). Improved baselines with momentum contrastive learning. Preprint at arXiv. https://doi.org/10.48550/arXiv.2003.04297.

30. He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., and Li, M. (2019). Bag of tricks for image classification with convolutional neural networks. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019 (Computer Vision Foundation/IEEE), pp. 558–567. https://doi.org/10.1109/CVPR.2019.00065.

31. Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. (2020). Big transfer (bit): general visual representation learning. In Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part V, *12350*, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, eds. (Springer), pp. 491–507. https://doi.org/10.1007/978-3-030-58558-7_29.

32. Wightman, R. (2019). Pytorch Image Models. https://doi.org/10.5281/zenodo.4414861. https://github.com/rwightman/pytorch-image-models.

33. Contributors, M. (2020). MMSegmentation: Openmmlab Semantic Segmentation Toolbox and Benchmark. https://github.com/open-mmlab/mmsegmentation.

34. Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., et al. (2019). Mmdetection: open mmlab detection toolbox and benchmark. Preprint at arXiv. https://doi.org/10.48550/arXiv.1906.07155.

35. Yu, J., Tan, M., Zhang, H., Rui, Y., and Tao, D. (2022). Hierarchical deep click feature prediction for fine-grained image recognition. IEEE Trans. Pattern Anal. Mach. Intell. *44*, 563–578. https://doi.org/10.1109/TPAMI.2019.2932058.

36. Ren, S., He, K., Girshick, R.B., Sun, J., and Faster, R.-C.N.N. (2015). Towards real-time object detection with region proposal networks. In Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, and R. Garnett, eds., pp. 91–99.

37. Redmon, J., and Farhadi, A. (2018). Yolov3: an incremental improvement. Preprint at arXiv. https://doi.org/10.48550/arXiv.1804.02767.

38. He, K., Gkioxari, G., Dollár, P., Girshick, R.B., and Mask, R.-C.N.N. (2017). In IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017 (IEEE Computer Society), pp. 2980–2988. https://doi.org/10.1109/ICCV.2017.322.

39. Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: convolutional networks for biomedical image segmentation. In Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th

International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III, *9351*, N. Navab, J. Hornegger, W.M.W. III, and A.F. Frangi, eds. (Springer), pp. 234–241. https://doi.org/10.1007/978-3-319-24574-4_28.

40. Syu, N., Chen, Y., and Chuang, Y. (2018). Learning deep convolutional networks for demosaicing. Preprint at arXiv. https://doi.org/10.48550/arXiv.1802.03769.

41. Dong, C., Loy, C.C., He, K., and Tang, X. (2014). Learning a deep convolutional network for image super-resolution. In Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part IV, *8692*, D.J. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds. (Springer), pp. 184–199. https://doi.org/10.1007/978-3-319-10593-2_13.

42. Dong, C., Loy, C.C., and Tang, X. (2016). Accelerating the super-resolution convolutional neural network. In Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II, *9906*, B. Leibe, J. Matas, N. Sebe, and M. Welling, eds. (Springer), pp. 391–407. https://doi.org/10.1007/978-3-319-46475-6_25.

43. Yu, J., Fan, Y., Yang, J., Xu, N., Wang, Z., Wang, X., and Huang, T. (2018). Wide activation for efficient and accurate image super-resolution. Preprint at arXiv. https://doi.org/10.48550/arXiv.1808.08718.

44. Nah, S., Kim, T.H., and Lee, K.M. (2017). Deep multi-scale convolutional neural network for dynamic scene deblurring. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017 (IEEE Computer Society), pp. 257–265. https://doi.org/10.1109/CVPR.2017.35.

45. Moore, G.E. (1998). Cramming more components onto integrated circuits. Proc. IEEE *86*, 82–85. https://doi.org/10.1109/jproc.1998.658762.

46. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, I. Guyon, U. von Luxburg, S. Bengio, H.M. Wallach, R. Fergus, and S.V.N. Vishwanathan, et al., eds., pp. 5998–6008.

47. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., and Uszkoreit, J. (2020). An image is worth 16x16 words: transformers for image recognition at scale. Preprint at arXiv. https://doi.org/10.48550/arXiv.2010.11929.

48. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I, *12346*, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, eds. (Springer), pp. 213–229. https://doi.org/10.1007/978-3-030-58452-8_13.

49. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin transformer: hierarchical vision transformer using shifted windows. In 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021 (IEEE), pp. 9992–10002. https://doi.org/10.1109/ICCV48922.2021.00986.

50. Hatamizadeh, A., Tang, Y., Nath, V., Yang, D., Myronenko, A., Landman, B.A., Roth, H.R., and Xu, D. (2022). UNETR: transformers for 3d medical image segmentation. In IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2022, Waikoloa, HI, USA, January 3-8, 2022 (IEEE), pp. 1748–1758. https://doi.org/10.1109/WACV51458.2022.00181.

51. Valanarasu, J.M.J., Oza, P., Hacihaliloglu, I., and Patel, V.M. (2021). Medical transformer: gated axial-attention for medical image segmentation. In Medical Image Computing and Computer Assisted Intervention - MICCAI 2021 - 24th International Conference, Strasbourg, France, September 27 - October 1, 2021, Proceedings, Part I, *12901*, M. de Bruijne, P.C. Cattin, S. Cotin, N. Padoy, S. Speidel, and Y. Zheng, et al., eds. (Springer), pp. 36–46. https://doi.org/10.1007/978-3-030-87193-2_4.

52. Chen, H., Wang, Y., Guo, T., Xu, C., Deng, Y., Liu, Z., Ma, S., Xu, C., Xu, C., and Gao, W. (2021). Pre-trained image processing transformer. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR

2021, virtual, June 19-25, 2021 (Computer Vision Foundation/IEEE), pp. 12299–12310.

53. Cao, J., Li, Y., Zhang, K., and Gool, L.V. (2021). Video super-resolution transformer. Preprint at arXiv. https://doi.org/10.48550/arXiv.2106.06847.

54. Liang, J., Cao, J., Sun, G., Zhang, K., Gool, L.V., and Timofte, R. (2021). Swinir: image restoration using swin transformer. In IEEE/CVF International Conference on Computer Vision Workshops, ICCVW 2021, Montreal, BC, Canada, October 11-17, 2021 (IEEE), pp. 1833–1844. https://doi.org/10.1109/ICCVW54120.2021.00210.

55. Wang, Z., Cun, X., Bao, J., and Liu, J. (2021). Uformer: A general u-shaped transformer for image restoration. Preprint at arXiv. https://doi.org/10.48550/arXiv.2106.03106.

56. Aldahdooh, A., Hamidouche, W., and Déforges, O. (2021). Reveal of vision transformers robustness against adversarial attacks. Preprint at arXiv. https://doi.org/10.48550/arXiv.2106.03734.

57. Bhojanapalli, S., Chakrabarti, A., Glasner, D., Li, D., Unterthiner, T., and Veit, A. (2021). Understanding robustness of transformers for image classification. In 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021 (IEEE), pp. 10211–10221. https://doi.org/10.1109/ICCV48922.2021.01007.

58. Mahmood, K., Mahmood, R., and van Dijk, M. (2021). On the robustness of vision transformers to adversarial examples. In 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021 (IEEE), pp. 7818–7827. https://doi.org/10.1109/ICCV48922.2021.00774.

59. Naseer, M., Ranasinghe, K., Khan, S.H., Hayat, M., Khan, F.S., and Yang, M. (2021). Intriguing Properties of Vision Transformers. Preprint at arXiv. https://doi.org/10.48550/arXiv.2105.10497.

60. Touvron, H., Bojanowski, P., Caron, M., Cord, M., El-Nouby, A., Grave, E., Izacard, G., Joulin, A., Synnaeve, G., Verbeek, J., and Jégou, H. (2021). Resmlp: feedforward networks for image classification with data-efficient training. Preprint at arXiv. https://doi.org/10.48550/arXiv.2105.03404.

61. Melas-Kyriazi, L. (2021). Do you even need attention? A stack of feed-forward layers does surprisingly well on imagenet. Preprint at arXiv. https://doi.org/10.48550/arXiv.2105.02723.

62. Hendrycks, D., and Gimpel, K. (2016). Bridging nonlinearities and stochastic regularizers with Gaussian error linear units. Preprint at arXiv. https://doi.org/10.48550/arXiv.1606.08415.

63. Ba, L.J., Kiros, J.R., and Hinton, G.E. (2016). Layer normalization. Preprint at arXiv. https://doi.org/10.48550/arXiv.1607.06450.

64. Sun, C., Shrivastava, A., Singh, S., and Gupta, A. (2017). Revisiting unreasonable effectiveness of data in deep learning era. In IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017 (IEEE Computer Society), pp. 843–852. https://doi.org/10.1109/ICCV.2017.97.

65. Lin, T., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C.L. (2014). Microsoft COCO: common objects in context. In Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V, *8693*, D.J. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds. (Springer), pp. 740–755. https://doi.org/10.1007/978-3-319-10602-1_48.

66. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016 (IEEE Computer Society), pp. 3213–3223. https://doi.org/10.1109/CVPR.2016.350.

67. Liu, H., Dai, Z., So, D.R., and Le, Q.V. (2021). Pay attention to mlps. Preprint at arXiv. https://doi.org/10.48550/arXiv.2105.08050.

68. Hu, S.M., Liang, D., Yang, G.Y., Yang, G.W., and Zhou, W.Y. (2020). Jittor: a novel deep learning framework with meta-operators and unified graph execution. Sci. China Inf. Sci. *63*, 222103–222121. https://doi.org/10.1007/s11432-020-3097-4.

69. Liu, R. (2021). Jittor-mlp. https://github.com/liuruiyang98/Jittor-MLP.

70. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., and Desmaison, A. (2019). Pytorch: an imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, H.M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E.B. Fox, and R. Garnett, eds., pp. 8024–8035.

71. Hou, Q., Jiang, Z., Yuan, L., Cheng, M., Yan, S., and Feng, J. (2021). Vision permutator: a permutable mlp-like architecture for visual recognition. Preprint at arXiv. https://doi.org/10.48550/arXiv.2106.12368.

72. Zhang, H., Wu, C., Zhang, Z., Zhu, Y., Zhang, Z., Lin, H., Zhang, Z., Sun, Y., He, T., Mueller, J., et al. (2020). Resnest: split-attention networks. Preprint at arXiv. https://doi.org/10.48550/arXiv.2004.08955.

73. Tang, C., Zhao, Y., Wang, G., Luo, C., Xie, W., and Zeng, W. (2021). Sparse MLP for image recognition: is self-attention really necessary?. Preprint at arXiv. https://doi.org/10.48550/arXiv.2109.05422.

74. Tatsunami, Y., and Taki, M. (2021). Raftmlp: do mlp-based models dream of winning over computer vision?. Preprint at arXiv. https://doi.org/10.48550/arXiv.2108.04384.

75. Wang, Z., Jiang, W., Zhu, Y., Yuan, L., Song, Y., and Liu, W. (2022). Dynamixer: a vision MLP architecture with dynamic mixing. Preprint at arXiv. https://doi.org/10.48550/arXiv.2201.12083.

76. Tang, Y., Han, K., Guo, J., Xu, C., Li, Y., Xu, C., and Wang, Y. (2021). An image patch is a wave: phase-aware vision MLP. Preprint at arXiv. https://doi.org/10.48550/arXiv.2111.12294.

77. Zhang, D.J., Li, K., Chen, Y., Wang, Y., Chandra, S., Qiao, Y., Liu, L., and Shou, M.Z. (2021). Morphmlp: a self-attention free, mlp-like backbone for image and video. Preprint at arXiv. https://doi.org/10.48550/arXiv.2111.12527.

78. Yu, T., Li, X., Cai, Y., Sun, M., and Li, P. (2022). S^2-mlp: spatial-shift MLP architecture for vision. In IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2022, Waikoloa, HI, USA, January 3-8, 2022 (IEEE), pp. 3615–3624. https://doi.org/10.1109/WACV51458.2022.00367.

79. Yu, T., Li, X., Cai, Y., Sun, M., and Li, P. (2021). S^2-mlpv2: improved spatial-shift MLP architecture for vision. Preprint at arXiv. https://doi.org/10.48550/arXiv.2108.01072.

80. Lian, D., Yu, Z., Sun, X., and Gao, S. (2021). AS-MLP: An axial shifted MLP architecture for vision. Preprint at arXiv. https://doi.org/10.48550/arXiv.2107.08391.

81. Yu, F., and Koltun, V. (2016). Multi-scale context aggregation by dilated convolutions. In 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Y. Bengio and Y. LeCun, eds. (Conference Track Proceedings).

82. Wang, P., Chen, P., Yuan, Y., Liu, D., Huang, Z., Hou, X., and Cottrell, G. (2018). Understanding convolution for semantic segmentation. In 2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018, Lake Tahoe, NV, USA, March 12-15, 2018 (IEEE Computer Society), pp. 1451–1460. https://doi.org/10.1109/WACV.2018.00163.

83. Chen, S., Xie, E., GE, C., Chen, R., Liang, D., and Luo, P. (2022). CycleMLP: a MLP-like architecture for dense prediction. In International Conference on Learning Representations. URL: https://openreview.net/forum?id=NMEceG4v69Y

84. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., and Wei, Y. (2017). Deformable convolutional networks. In IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017 (IEEE Computer Society), pp. 764–773. https://doi.org/10.1109/ICCV.2017.89.

85. Wei, G., Zhang, Z., Lan, C., Lu, Y., and Chen, Z. (2022). Activemlp: an mlp-like architecture with active token mixer. Preprint at arXiv. https://doi.org/10.48550/arXiv.2203.06108.

86. Guo, J., Tang, Y., Han, K., Chen, X., Wu, H., Xu, C., Xu, C., and Wang, Y. (2021). Hire-mlp: vision MLP via hierarchical rearrangement. Preprint at arXiv. https://doi.org/10.48550/arXiv.2108.13341.

87. Zheng, H., He, P., Chen, W., and Zhou, M. (2022). Mixing and shifting: exploiting global and local dependencies in vision mlps. Preprint at arXiv. https://doi.org/10.48550/arXiv.2202.06510.

88. Lou, Y., Xue, F., Zheng, Z., and You, Y. (2021). Sparse-mlp: a fully-mlp architecture with conditional computation. Preprint at arXiv. https://doi.org/10.48550/arXiv.2109.0200.

89. Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q.V., Hinton, G.E., and Dean, J. (2017). Outrageously large neural networks: the sparsely-gated mixture-of-experts layer. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings (OpenReview.net).

90. Riquelme, C., Puigcerver, J., Mustafa, B., Neumann, M., Jenatton, R., Pinto, A.S., Keysers, D., and Houlsby, N. (2021). Scaling vision with sparse mixture of experts. Preprint at arXiv. https://doi.org/10.48550/arXiv.2106.05974.

91. Yu, T., Li, X., Cai, Y., Sun, M., and Li, P. (2021). Rethinking token-mixing MLP for mlp-based vision backbone. Preprint at arXiv. https://doi.org/10.48550/arXiv.2106.14882.

92. Ding, X., Chen, H., Zhang, X., Han, J., and Ding, G. (2021). Repmlpnet: hierarchical vision MLP with re-parameterized locality. Preprint at arXiv. https://doi.org/10.48550/arXiv.2112.11081.

93. Li, J., Hassani, A., Walton, S., and Shi, H. (2021). Convmlp: hierarchical convolutional mlps for vision. Preprint at arXiv. https://doi.org/10.48550/arXiv.2109.04454.

94. Pan, Z., Zhuang, B., He, H., Liu, J., and Cai, J. (2021). Less is more: Pay less attention in vision transformers. Preprint at arXiv. https://doi.org/10.48550/arXiv.2105.14217.

95. Liu, J., Li, H., Song, G., Huang, X., and Liu, Y. (2021). Uninet: unified architecture search with convolution, transformer, and MLP. Preprint at arXiv. https://doi.org/10.48550/arXiv.2110.04035.

96. Sherrington, C.S. (1906). Observations on the scratch-reflex in the spinal dog. J. Physiol. *34*, 1–50. https://doi.org/10.1113/jphysiol.1906.sp001139.

97. He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016 (IEEE Computer Society), pp. 770–778. https://doi.org/10.1109/CVPR.2016.90.

98. Lin, T., Dollár, P., Girshick, R.B., He, K., Hariharan, B., and Belongie, S.J. (2017). Feature pyramid networks for object detection. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017 (IEEE Computer Society), pp. 936–944. https://doi.org/10.1109/CVPR.2017.106.

99. Dong, X., Bao, J., Zhang, T., Chen, D., Zhang, W., Yuan, L., Chen, D., Wen, F., and Yu, N. (2021). Peco: perceptual codebook for BERT pre-training of vision transformers. Preprint at arXiv. https://doi.org/10.48550/arXiv.2111.12710.

100. Simonyan, K., and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Y. Bengio and Y. LeCun, eds. (Conference Track Proceedings). https://doi.org/10.48550/arXiv.1409.1556.

101. Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A.A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA, S.P. Singh and S. Markovitch, eds. (AAAI Press), pp. 4278–4284. URL: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14806.

102. Wightman, R., Touvron, H., and Jégou, H. (2021). Resnet strikes back: an improved training procedure in timm. Preprint at arXiv. https://doi.org/10.48550/arXiv.2110.00476.

103. Radosavovic, I., Kosaraju, R.P., Girshick, R.B., He, K., and Dollár, P. (2020). Designing network design spaces. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020 (Computer Vision Foundation/IEEE), pp. 10425–10433. https://doi.org/10.1109/CVPR42600.2020.01044.

104. Liu, Z., Mao, H., Wu, C., Feichtenhofer, C., Darrell, T., Xie, S.. A Convnet for the 2020s. CoRR 2022;Preprint at arXiv. https://doi.org/10.48550/arXiv.2201.03545.

105. Guo, M., Lu, C., Liu, Z., Cheng, M., and Hu, S. (2022). Visual attention network. Preprint at arXiv. https://doi.org/10.48550/arXiv.2202.09741.

106. Tan, M., and Le, Q.V. (2021). Efficientnetv2: smaller models and faster training. In Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event, *139*, M. Meila and T. Zhang, eds. (PMLR), pp. 10096–10106.

107. Leng, Z., Tan, M., Liu, C., Cubuk, E.D., Shi, J., Cheng, S., and Anguelov, D. (2021). Polyloss: a polynomial expansion perspective of classification loss functions. In International Conference on Learning Representations.

108. Ding, X., Zhang, X., Zhou, Y., Han, J., Ding, G., Sun, J.. Scaling up your kernels to 31x31: revisiting large kernel design in cnns. Preprint at arXiv. https://doi.org/10.48550/arXiv.2203.06717.

109. Pham, H., Dai, Z., Xie, Q., and Le, Q.V. (2021). Meta pseudo labels. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021 (Computer Vision Foundation/IEEE), pp. 11557–11568.

110. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. (2021). Training data-efficient image transformers & distillation through attention. In Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event, *139*, M. Meila and T. Zhang, eds. (PMLR), pp. 10347–10357.

111. Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z., Tay, F.E., Feng, J., and Yan, S. (2021). Tokens-to-token vit: training vision transformers from scratch on imagenet. In 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021 (IEEE), pp. 538–547. https://doi.org/10.1109/ICCV48922.2021.00060.

112. Wang, W., Xie, E., Li, X., Fan, D., Song, K., Liang, D., Lu, T., Luo, P., and Shao, L. (2021). Pyramid vision transformer: a versatile backbone for dense prediction without convolutions. In 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021 (IEEE), pp. 548–558. https://doi.org/10.1109/ICCV48922.2021.00061.

113. Zhang, Z., Zhang, H., Zhao, L., Chen, T., and Pfister, T. (2021). Aggregating nested transformers. Preprint at arXiv. https://doi.org/10.48550/arXiv.2105.12723.

114. Han, K., Guo, J., Tang, Y., and Wang, Y. (2022). Pyramidtnt: improved transformer-in-transformer baselines with pyramid architecture. Preprint at arXiv. https://doi.org/10.48550/arXiv.2201.00978.

115. Dong, X., Bao, J., Chen, D., Zhang, W., Yu, N., Yuan, L., Chen, D., and Guo, B. (2021). Cswin transformer: a general vision transformer backbone with cross-shaped windows. Preprint at arXiv. https://doi.org/10.48550/arXiv.2107.00652.

116. Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., and Jégou, H. (2021). Going deeper with image transformers. In 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021 (IEEE), pp. 32–42. https://doi.org/10.1109/ICCV48922.2021.00010.

117. Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., and Wei, F. (2021). Swin Transformer V2: Scaling up Capacity and Resolution. Preprint at arXiv. https://doi.org/10.48550/arXiv.2111.09883.

118. Chu, X., Tian, Z., Wang, Y., Zhang, B., Ren, H., Wei, X., Xia, H., and Shen, C. (2021). Twins: revisiting the design of spatial attention in vision transformers. Adv. Neural Inf. Process. Syst. *34*. https://doi.org/10.48550/arXiv.2104.13840.

119. Huang, Z., Ben, Y., Luo, G., Cheng, P., Yu, G., and Fu, B. (2021). Shuffle transformer: rethinking spatial shuffle for vision transformer. Preprint at arXiv. https://doi.org/10.48550/arXiv.2106.03650.

120. Guo, J., Han, K., Wu, H., Xu, C., Tang, Y., Xu, C., and Wang, Y. (2021). CMT: convolutional neural networks meet vision transformers. Preprint at arXiv. https://doi.org/10.48550/arXiv.2107.06263.

121. Dai, Z., Liu, H., Le, Q.V., and Tan, M. (2021). Coatnet: marrying convolution and attention for all data sizes. Adv. Neural Inf. Process. Syst. *34*, 3965–3977.

122. Yuan, L., Hou, Q., Jiang, Z., Feng, J., and Yan, S. (2021). VOLO: Vision Outlooker for Visual Recognition. Preprint at arXiv. https://doi.org/10.48550/arXiv.2106.13112.

123. Xiao, T., Liu, Y., Zhou, B., Jiang, Y., and Sun, J. (2018). Unified perceptual parsing for scene understanding. In Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part V, *11209*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds. (Springer), pp. 432–448. https://doi.org/10.1007/978-3-030-01228-1_26.

124. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., and Torralba, A. (2017). Scene parsing through ADE20K dataset. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017 (IEEE Computer Society), pp. 5122–5130. https://doi.org/10.1109/CVPR.2017.544.

125. Kirillov, A., Girshick, R.B., He, K., and Dollár, P. (2019). Panoptic feature pyramid networks. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019 (Computer Vision Foundation/IEEE), pp. 6399–6408. https://doi.org/10.1109/CVPR.2019.00656.

126. Xie, S., Girshick, R.B., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017 (IEEE Computer Society), pp. 5987–5995. https://doi.org/10.1109/CVPR.2017.634.

127. Bao, H., Dong, L., and Wei, F. (2021). Beit: BERT pre-training of image transformers. Preprint at arXiv. https://doi.org/10.48550/arXiv.2106.08254.

128. Cazenavette, G., and Guevara, M.L.D. (2021). Mixergan: an mlp-based architecture for unpaired image-to-image translation. Preprint at arXiv. https://doi.org/10.48550/arXiv.2105.14110.

129. Zhu, J., Park, T., Isola, P., and Efros, A.A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017 (IEEE Computer Society), pp. 2242–2251. https://doi.org/10.1109/ICCV.2017.244.

130. Tu, Z., Talebi, H., Zhang, H., Yang, F., Milanfar, P., Bovik, A., and Li, Y. (2022). MAXIM: multi-axis MLP for image processing. Preprint at arXiv. https://doi.org/10.48550/arXiv.2201.02973.

131. Carreira, J., and Zisserman, A. (2017). Quo vadis, action recognition? A new model and the kinetics dataset. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017 (IEEE Computer Society), pp. 4724–4733. https://doi.org/10.1109/CVPR.2017.502.

132. Xia, J., Zhuge, M., Geng, T., Fan, S., Wei, Y., He, Z., and Zheng, F. (2022). Skating-mixer: multimodal mlp for scoring figure skating. Preprint at arXiv. https://doi.org/10.48550/arXiv.220303990.

133. Ma, X., Qin, C., You, H., Ran, H., and Fu, Y. (2022). Rethinking network design and local geometry in point cloud: a simple residual MLP framework. Preprint at arXiv. https://doi.org/10.48550/arXiv.2202.07123.

134. Qi, C.R., Su, H., Mo, K., and Guibas, L.J. (2017). Pointnet: deep learning on point sets for 3d classification and segmentation. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017 (IEEE Computer Society), pp. 77–85. https://doi.org/10.1109/CVPR.2017.16.

135. Qi, C.R., Yi, L., Su, H., and Guibas, L.J. (2017). Pointnet++: deep hierarchical feature learning on point sets in a metric space. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, I. Guyon, U. von Luxburg, S. Bengio, H.M. Wallach, R. Fergus, and S.V.N. Vishwanathan, et al., eds., pp. 5099–5108.

136. Choe, J., Park, C., Rameau, F., Park, J., and Kweon, I.S. (2021). Pointmixer: mlp-mixer for point cloud understanding. Preprint at arXiv. https://doi.org/10.48550/arXiv.2111.11187.

137. Guo, M.H., Cai, J.X., Liu, Z.N., Mu, T.J., Martin, R.R., and Hu, S.M. (2021). PCT: point cloud transformer. Comput Vis Media *7*, 187–199. https://doi.org/10.1007/s41095-021-0229-5.

138. Zhao, H., Jiang, L., Jia, J., Torr, P.H.S., and Koltun, V. (2021). Point transformer. In 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021 (IEEE), pp. 16239–16248. https://doi.org/10.1109/ICCV48922.2021.01595.

139. Valanarasu, J.M.J., and Patel, V.M. (2022). Unext: mlp-based rapid medical image segmentation network. Preprint at arXiv. arXiv:2203.04967. https://doi.org/10.48550/arXiv.2203.04967.

140. Zhang, Q., Xu, Y., Zhang, J., and Tao, D. (2022). Vitaev2: vision transformer advanced by exploring inductive bias for image recognition and beyond. Preprint at arXiv. https://doi.org/10.48550/arXiv.2202.10108.

141. Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L., Fei-Fei, L., Yuille, A., Huang, J., and Murphy, K. (2018). Progressive neural architecture search. In Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part I, *11205*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds. (Springer), pp. 19–35. https://doi.org/10.1007/978-3-030-01246-5_2.

142. Chen, X., Xie, S., and He, K. (2021). An empirical study of training self-supervised vision transformers. Preprint at arXiv. https://doi.org/10.48550/arXiv.2104.02057.

143. Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G.E. (2020). Big self-supervised models are strong semi-supervised learners. In Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, *33*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds., pp. 22243–22255. URL: https://proceedings.neurips.cc/paper/2020/hash/fcbc95ccdd551da181207c0c1400c655-Abstract.html

144. Xie, Z., Zhang, Z., Cao, Y., Lin, Y., Bao, J., Yao, Z., Dai, Q., and Hu, H. (2021). Simmim: a simple framework for masked image modeling. Preprint at arXiv. https://doi.org/10.48550/arXiv.2111.09886.

145. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R.B. (2021). Masked autoencoders are scalable vision learners. Preprint at arXiv. https://doi.org/10.48550/arXiv.2111.06377.

146. Wei, C., Fan, H., Xie, S., Wu, C., Yuille, A.L., and Feichtenhofer, C. (2021). Masked feature prediction for self-supervised visual pre-training. Preprint at arXiv. https://doi.org/10.48550/arXiv.2112.09133.

147. Bottou, L. (2012). Stochastic gradient descent tricks. In Neural Network.: Tricks of the Trade, Second Edition, *7700*, G. Montavon, G.B. Orr, and K. Müller, eds. (Springer), pp. 421–436. https://doi.org/10.1007/978-3-642-35289-8_25.

148. Loshchilov, I., and Hutter, F. (2017). Fixing weight decay regularization in Adam. Preprint at arXiv. https://doi.org/10.48550/arXiv.1711.05101.

149. Chen, X., Hsieh, C., and Gong, B. (2021). When vision transformers outperform resnets without pretraining or strong data augmentations. Preprint at arXiv. https://doi.org/10.48550/arXiv.2106.01548.

150. Bansal, P., Lamba, R., Jain, V., Jain, T., Shokeen, S., Kumar, S., Singh, P.K., and Khan, B. (2022). Gga-mlp: A Greedy Genetic Algorithm to Optimize Weights and Biases in Multilayer Perceptron (Contrast Media & Molecular Imaging), p. 2022.