



# CMNet: a novel model and design rationale based on comparison studies and synergy of CNN and MetaFormer

Haowen Yu<sup>1</sup> · Liming Chen<sup>2</sup>

Received: 8 December 2022 / Revised: 3 May 2023 / Accepted: 10 August 2023 / Published online: 22 September 2023  
© The Author(s) 2023

## Abstract

Convolutional- and Transformer-based backbone architecture are two dominant, widely accepted, models in computer vision. Nevertheless, it is still a challenge, thus a focus of research, to decide which backbone architecture performs better, and under which circumstances. In this paper, we conduct an in-depth investigation into the differences of the macroscopic backbone design of the CNN and Transformer models with the ultimate purpose of developing new models to combine the strengths of both types of architectures for effective image classification. Specifically, we first analyze the model structures of both models and identified four main differences, then we design four sets of ablation experiments using the ImageNet-1K dataset with an image classification problem as an example to study the impacts of these four differences on model performance. Based on the experimental results, we derive four observations as rules of thumb for designing a vision model backbone architecture. Informed by the experiment findings, we then conceive a novel model called CMNet which marries the experiment-proved best design practices of CNN and Transformer architectures. Finally, we carry out extensive experiments on CMNet using the same dataset against baseline classifiers. Initial results prove CMNet achieves the highest top-1 accuracy of 80.08% on the ImageNet-1K validation set, this is a very competitive value compared to previous classical models with similar computational complexity. Details of the implementation, algorithms and codes, are publicly available on Github: <https://github.com/Arwin-Yu/CMNet>.

**Keywords** Transformer · MetaFormer · Attention mechanism · Convolutional neural network

## 1 Introduction

Convolution neural networks (CNN) have several built-in inductive biases that give them a natural advantage in feature extraction on images. For example, the convolutional kernel computes only local information of an image at a time, which allows CNN to better extract texture information; meanwhile, the parameter sharing and sliding traversal strategies make CNN an efficient way to process image data. Specifically, CNN can save a large number of parameters compared to the computation image of neural networks,

which facilitates the training of the network. As a result, since AlexNet [1] obtained impressive accuracy the in ILSVRC-2012 image classification competition, a large number of studies have been undertaken to improve the effectiveness of CNN processing image data from different perspectives. For example, ZFNet [2] demonstrated that CNNs can extract texture features well in the shallow layer of the model, while semantic features can also be extracted by stacking convolution layers. Vgg [3] proposed that increasing the depth of the model by stacking convolutional layers with a small kernel is particularly critical for image processing, and this model design guideline has been used in a lot of work until now. GoogLeNet [4] proposed an inception structure, which can obtain multiple sets of feature maps by designing branches with different computational receptive fields, and concatenate these feature maps in the channel dimension to fuse features of different scales. ResNet [5] proposed a simple residual structure that can help the network to perform better gradient propagation and learn some constant mapping information to a certain extent. DenseNet [6] further improved the

✉ Liming Chen  
l.chen@ulster.ac.uk

Haowen Yu  
haowen.yu@postgrad.manchester.ac.uk

<sup>1</sup> Faculty of Biology, Medicine and Health, The University of Manchester, Oxford Rd, Manchester M13 9PL, UK

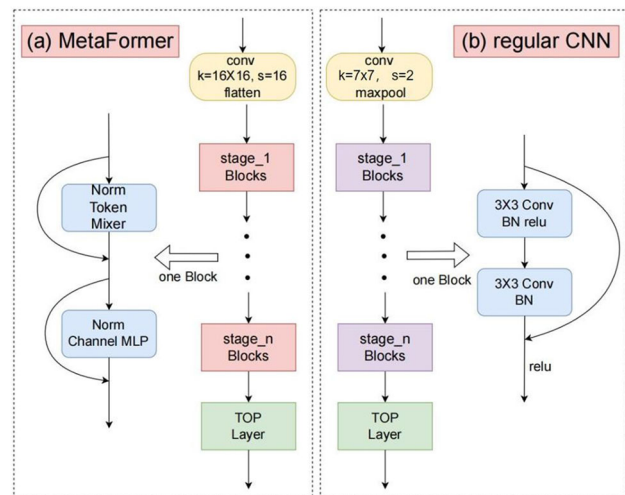
<sup>2</sup> School of Computing, Ulster University, Cromore Road, Belfast BT52 1SA, UK

performance of the network by feature maps reuse. In addition, some lightweight models have also been designed to help CNN achieve industrial strengths, such as MobileNet [7], ShuffleNet [8] and ghostNet [9]. CNNs have long been the dominant model in the field of computer vision (CV).

Around the same time, Transformer [10] is rapidly gaining a dominant position in the field of natural language processing (NLP) because its built-in inductive bias can focus on global information in every computation and redistribute the global information with importance through the self-attention mechanism. This is very useful for processing textual information. Considering the dominance of Transformer in NLP, ViT [11] introduced the self-attention mechanism and Transformer blocks into the field of CV and achieved experimental results that were comparable with CNN. A large number of studies [12–15] has verified that the Transformer has shown huge potential in various vision tasks such as image classification [16], detection [17], video processing [18] and segmentation [19], and can even achieve better performance than regular CNN backbone architectures with the support of large amounts of training data. Some work has investigated what operations make the Transformer backbone architecture effective, which has been long attributed to the attention-based Token Mixer [10]. However, a recent work [20], which attempts to replace the self-attention mechanism with the most basic multi-layer perceptron in Token Mixer, achieved a good performance. This finding indicates that self-attention is not a necessary operation in the Transformer for vision task. Informed by this finding, MetaFormer [21] more radically replaces the self-attention operation with a pool operation with no trainable parameters, which also achieves good performance. MetaFormer attributes the success of Transformer to the structure of backbone architecture. A reasonable question is whether we can use the Transformer backbone architecture summarized by MetaFormer to improve the performance of CNNs?

To compare the similarities and differences of the two architectures, we depict the backbone architecture of regular CNN-based models and MetaFormer-based models in Fig. 1a, b, respectively.

In general, deep learning models for CV have a hierarchical structure by designing different stages, each of them is stacked with many repeated identical blocks, and each stage maintains the same dimension of the input information when it is computed. There may be operations between stages to downsample the spatial dimension of the input information and transfer it to the channel dimension, especially in regular CNN backbone architectures. As for the block, it is a stack of many neural network layers with a large degree of freedom, and thus is an innovative point for many classical model designs. For example, Fig. 1a shows a MetaFormer block [21], which is composed of a Token Mixer, a Channel MLP, two norm layers, and two residual connections.



**Fig. 1** The architectures of CNN-based models and MetaFormer-based models

Figure 1b shows a basic block of ResNet18 [5], which is composed of two convolutional layers, two norm layers, two ReLU functions, and a residual connection. In this paper, we use ResNet18 as the baseline model.

As is observed in Fig. 1 there are four main differences between the two backbone architectures. Firstly, the Token Mixer only computes information in the spatial dimension while channel information is handled by Channel MLP. This means the MetaFormer block handles spatial information and channel information separately, though spatial information and channel information can be processed at the same time in regular CNN-based models. Secondly, the residual connections of the two architectures are different. MetaFormer's residual connections are more detailed, which exist after each spatial information processing or channel information processing. Thirdly, the two backbone architecture blocks have different computational receptive fields. As the self-attention mechanism can compute global information, the receptive field for each spatial information computation operation is also global. On the other hand, the computational receptive field for convolution is local, mainly related to the size of the convolutional kernel and the stride during kernel sliding. Fourthly, the operations of the two backbone architectures are different when they process an image for the first time. For the convenience of description, we will call the first-time operation of processing an image as 'stem layers.' The stem layers of ResNet18 is using a convolutional layer (kernel size = 7, stride = 2) and a max-pool layer (kernel size = 3, stride = 2) to do downsampling. In Metaformers, a more aggressive strategy is used as the stem layers, which corresponds to a large kernel size and non-overlapping convolution (kernel size = 16, stride = 16).

To investigate the impact of the above four differences on the performance of the two architectures, we designed

ablation experiments on ImageNet-1k dataset [35] to explore which operation is more effective for processing image data using the image classification task as an example. Based on the experimental results, we derive four observations for designing a model back-bone architecture for vision. These four sets of ablation experiments and observations are described in Sect. 3.

In addition to extensive comparison and analysis, inspired by the difference in stem layers between regular CNN-based models and MetaFormer-based models, we design three kinds of sophisticated stem layers, which can be easily embedded in the backbone architectures of CNN-based models and MetaFormer-based models. These stem layers allow the model to extract rich multi-scale information from the original picture from the beginning and improve the performance of the model. Built upon the experimental observations and our proposed stem layers, we propose a novel and simple network model called CMNet by marrying the advantages of CNN and MetaFormer, which is detailed in Sect. 4. To explore the performance of CMNet, we take the classical picture classification problem as an example. And compared to previous classical models with similar computational complexity, CMNet achieves sufficiently competitive classification accuracy.

Our contributions are mainly the following three points. Firstly, we compare and summarize the four main differences between regular CNN-based models and MetaFormer-based models, and proposed four observations through the results of ablation experiments, which have a lot of reference value for the design of visual models. Secondly, we propose three kinds of sophisticated stem layers that can extract rich multi-scale features, which is beneficial to model performance, notably these stem layers can be easily embedded into existing mainstream vision models. Finally, we designed a novel and simple model (referred to as CMNet) based on the results of four sets of ablation experiments, which combines the advantages of CNN and MetaFormer.

In Sect. 2, we present related works, including the current development of CNN and Transformer, classical CNN-based model ResNets, and classical TransFormer-based model MetaFormers. In Sect. 3, according to the four main differences between CNN-based models and MetaFormer-based models, we design the corresponding ablation experiments and analyze the experimental results, which provide reference experiences for future visual model design. In Sect. 4, we describe our novel model CMNet in detail, including the design rationale, model composition, data computation processes and scalability issues. In Sect. 5, we compare CMNet with previous classical models, and demonstrate their image classification accuracy on the ImageNet-1k dataset and the specific training scheme of the model, we also discuss the

future research directions of CMNet. In Sect. 6, we summarize the CMNet as well as the ablation experiments of CNN-based models and MetaFormer-based models.

## 2 Related work

### 2.1 Status of CNN and transformer

Since the ViT [11] model demonstrated the potential of Transformer [10] in processing vision tasks, the long-standing dominance of CNN in the computer vision field was shaken for the first time. After that, the two network backbone architectures have always learned from each other and developed together. The following will introduce the current state of model development of the two backbone architectures.

Because some transform-based models outperform the regular CNN-based models currently, especially swin-transformer [15], many researchers believe that Transformer is the future mainstream of vision model backbone architecture. However, Liu et al. [22] pointed out that by mimicking the model design and training strategy of swin-transformer, pure regular CNN models can get higher performance than swin-transformer, and this work made researchers start thinking again about the capability of convolution. Unlike this work, we no longer compare a single model (swin-transformer) but the whole back-bone architecture (MetaFormer), and in addition, we propose three kinds of sophisticated stem layers and a novel model: CMNet.

Due to the long-standing dominance of regular CNN-based models in processing image tasks, many works have attempted to use convolution operations to improve transform-based models. On the other hand, the functions of Token Mixer and Channel MLP in Transformer can be easily replaced by variants of convolution, specifically, depth-separable convolution [24] can easy instead of Token Mixer in MetaFormer to compute spatial information, and point-wise convolution [25] can easy instead of Channel MLP in MetaFormer to compute channel information, which allows convolutional operations to be easily embedded in the Transformer architecture. For example, ConvMixer [23] uses depth-separable convolution instead of self-attention, and point-wise convolution instead of Channel MLP. This simple pure convolutional network can achieve good performance as well. In order to mimic the property that the self-attention mechanism can acquire global computational receptive fields, the LKA mechanism in VAN [26] increases the receptive field by using large convolution kernels, depth-wise separable convolutions, and dilated convolutions during computation, and obtain very good performance. These works prove that CNN-based models and MetaFormer-based

models can learn from each other, and in general, models that incorporate both architectures will get better performance.

## 2.2 ResNet

The proposal of Deep Residual Network (ResNet [5]) is a milestone event in the history of CV. ResNet achieved the first place in five competitions such as image classification, detection, etc. in the year of publication in 2015. Until today, residual connections are still frequently seen in various state-of-the-art models. In addition, there are many variants based on ResNet, among which the well-known model are ResNeXt [27], SENet [33], SKNet [28], CBAM [29], etc. As a classical model in regular CNN-based models, ResNet has been used as a baseline model in many works. By adjusting the depth and width of the network, there are many scaled versions of ResNet, and in this paper, we use ResNet18 as the baseline model, which is most similar to our proposed model CMNet in terms of computational complexity.

## 2.3 MetaFormer

MetaFormer [21] uses the backbone structure as a key factor for the Transformer to be effective, as shown in Fig. 1a. Most of the Transformer-based models follow this structure, only the operations in the token Mixer are different, for example, in Transformer [10], it is a self-attention operation; in MetaFormer [21], it is a pool operation; in VAN [26], it is a LKA operation; in MLP-mixer [20], it is a multi-layer perception and so on. Whatever the operation is, the spatial information is computed in the Token Mixer. And Channel MLP mainly does computation on channel information.

## 3 Comparison studies between CNN-based models and MetaFormer-based models

We designed four ablation experiments using ImageNet-1k to demonstrate which differences between regular CNN-based models and MetaFormer-based models are more beneficial to improving the accuracy of image classification. For this purpose, we ensure that the models in each set of experiments have similar computational complexity by adjusting the depth of networks. Moreover, we use the same simple training strategy which is the cosine learning rate (starting learning rate of 0.01, final learning rate of 0.0001, and 150 epochs of training rounds) in experiments. As for four ablation experiments, first, we investigate which is better, the way to compute spatial and channel information together in a regular CNN or the way to compute spatial and channel

information separately in a Transformer. Second, we investigate the effect of the residual connection method in regular CNN and the more detailed residual connection method in Transformer on the model. Third, the computation of regular CNN is local and the computation of Transformer is global. We simulate the global receptive fields in Transformer by increasing the size of convolutional kernel to investigate which is more suitable for image processing. Fourth, observing that regular CNN can extract more information than Transformer in the stem layer, we try to investigate the impact on network classification results by designing a new stem layer to extract rich multi-scale features in the shallow layer of the model. We analyze these experiments and draw four observations, which are presented and discussed in detail below.

### 3.1 Separate computing of spatial information and channel information is more effective

In regular CNN-based models, a convolution operation can compute both spatial information and channel information, however, in MetaFormer, spatial information is computed by Token Mixer while channel information is computed by Channel MLP. Therefore, this set of ablation experiments was designed to explore which is the more reasonable way to calculate. In the convolution operation, a combination of depth-wise convolution and point-wise convolution can be easily implemented to calculate spatial information and channel information separately, which is used in many works [7, 30]. But they did not discuss which combination is the most reasonable, so we designed a set of ablation experiments with resnet18 [5] as the baseline, using different combinations of depth-wise convolution and point-wise convolution to replace the regular convolution blocks in resnet18, as shown in Fig. 2.

Figure 2a depict a basic block of baseline model resnet18 [5]. In order to replace the two regular convolutions in the baseline block, Fig. 2b first goes through two depth-wise convolutions and then two point-wise convolutions. Figure 2c goes through two serial combinations of depth-wise convolution and point-wise convolution. Figure 2d goes through two parallel combinations of depth-wise convolution and point-wise convolution. Using these four blocks to perform ablation experiments while ensuring that the rest of the model is identical. The experimental results are shown in Table 1.

The results prove that under the premise that the parametric quantities of the models are similar, higher accuracy can be obtained by computing spatial information and channel information separately, where block(d) can achieve the highest accuracy.

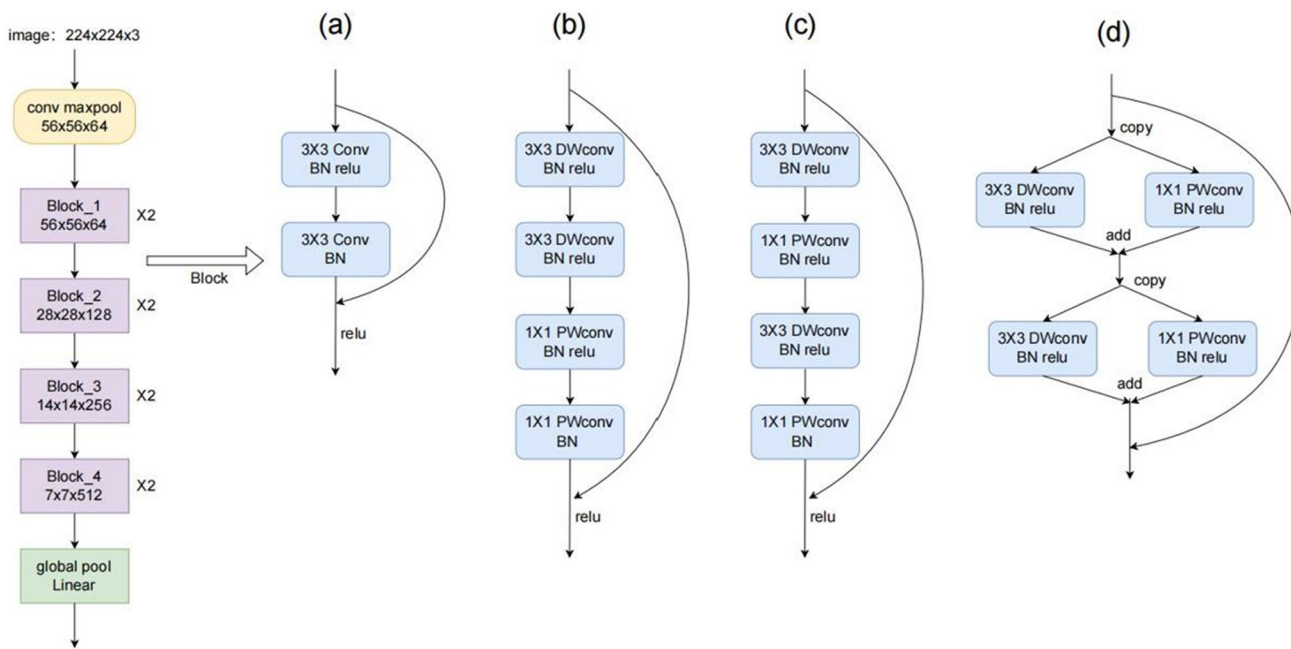


Fig. 2 Four ways to replace CNNs with depth-wise convolutions and point-wise convolutions

Table 1 Top1 accuracy of the models in Fig. 2 on ImageNet-1K

	(a)	(b)	(c)	(d)
Train accuracy	0.744	0.777	0.778	0.794
Val accuracy	0.677	0.696	0.699	<b>0.711</b>

### 3.2 Residual connection method of MetaFormer-based models are sensitive to learning rate

MetaFormer-based models is connected by residuals in every computation of spatial and channel information, which is more detailed than the residuals of regular CNN-based models. Therefore, we designed ablation experiments to investigate which residual connection is the best way, and the experimental design is as Fig. 3.

Figure 3a represents a block of ResNet18 where a residual connection contains two regular CNN operations in between. Figure 3b represents a block of MetaFormer where residual connections exist each time spatial information and channel information are computed. Figure 3c represents a block using depth-wise convolution as the operation to compute spatial information in Token Mixer and point-wise convolution as the operation to compute channel information in Channel MLP. When training the model with different residual connection methods (Fig. 3a, c), we found that the model using residual Fig. 3c is more sensitive to the learning rate compared with Fig. 3a, and it is easy to cause training failure when the learning rate is too large, therefore, if the residual Fig. 3c is used in a model, it should be matched with a smaller

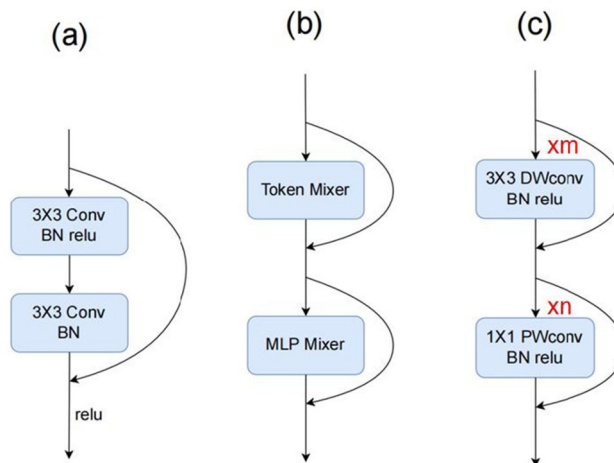


Fig. 3 Different ways of connecting residuals

training learning rate. In addition, we explore the impact on the model when adding the number of depth-wise convolution or point-wise convolution is m or n. The experimental results are shown in Fig. 4.

As is seen in Fig. 4 the residual connection of regular CNN-based models can obtain higher accuracy. However, a (ResNet18) and c(m = 2, n = 2) are compared, and the

	Train accuracy	Val accuracy
a(ResNet18)	0.744	0.677
c(m=2, n=2)	0.746	0.676
c(m=2, n=3)	0.667	0.002
c(m=3, n=2)	0.692	0.001
c(m=3, n=3)	0.691	0.002

Fig. 4 Top1 accuracy of the models in Fig. 3 on ImageNet-1K

Table 2 Top1 accuracy of the models in Fig. 3c with different size of convolutional kernel on ImageNet-1K

		Train accuracy	Val accuracy
Fig. 3c	(m = 2, n = 2, k = 3)	0.746	0.676
Fig. 3c	(m = 2, n = 2, k = 5)	0.746	0.677
Fig. 3c	(m = 2, n = 2, k = 7)	0.748	0.680
Fig. 3c	(m = 2, n = 2, k = 9)	0.746	0.679
Fig. 3c	(m = 2, n = 2, k = 11)	0.747	0.681

difference in the accuracy of image classification that can be obtained by the models corresponding to the two residual connection methods is not significant.

Noteworthy, when using the residual connection of MetaFormer, if there are more network layers (more than twice) between a residual connection, it can easily lead to model training failure. For example, the training accuracy and validation accuracy of c(m = 2, n = 3), c(m = 3, n = 2) and c(m = 3, n = 3) are vastly different.

### 3.3 It is effective to use large convolution kernels to simulate the self-attention mechanism

In regular CNN-based models, since VGG [3] advocated the use of small size convolutional kernels(3\*3), a large amount of work has followed this guideline to choose the use of small convolutional kernels when designing models, and few have used large-sized convolutional kernels. The Token Mixer in ViT [11] has recently implemented a self-attention mechanism, which included a property to compute global information. Some work [23, 31] has simulated the property by designing a large convolutional kernel to obtain a large computational receptive field, and has obtained good results. Informed by this finding, in this part of the exploration, we experiment to increase the size of the depth-wise convolution kernel in model Fig. 3c and obtained the results in Table 2.

Experimental results demonstrate that better accuracy can be obtained using large convolutional kernels than small convolutional kernels. Our explanation for this is that small convolution kernels extract texture information due to the small computational receptive field, which has been demonstrated in ZFNet [2]. Large convolutional kernels can extract semantic information better due to the larger computational

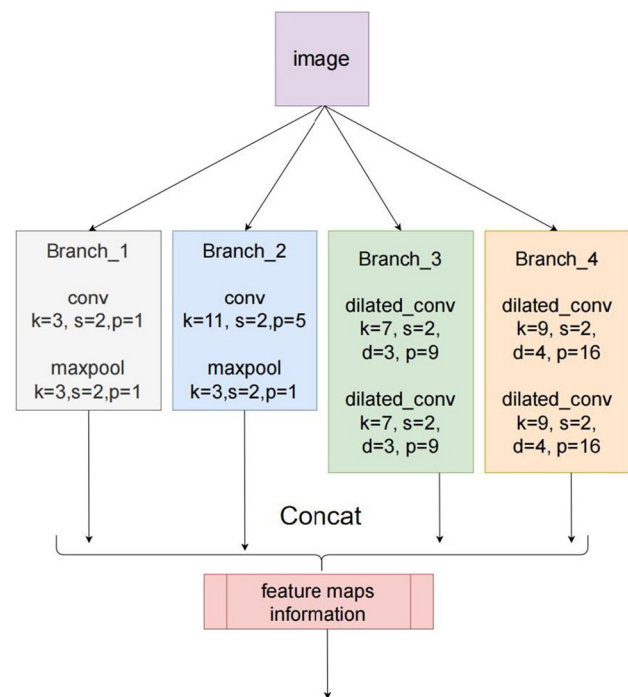


Fig. 5 The stem a layer in CMNet

receptive field, which is more useful for certain tasks that rely on semantic information. Therefore, the size of the convolutional kernel cannot be generalized, and it is dependent on whether the task to be handled by the model relies more on semantic information or on texture information.

### 3.4 Extracting more detailed features in the stem layers is important

A major difference between Metaformer-based models and regular CNN-based models is that the processing of the Metaformer stem layers is more aggressive. This means that MetaFormer easily loses a lot of information in the stem layers, which is detrimental to the subsequent feature extraction. Some works try to mimic regular CNN-based models in the stem layers of Metaformer-based models. For example, Xiao et al. [32] replaced the stem layers in MetaFormer with 5 regular convolutions(k = 3, s = 2 or 1). This attempt not only helped the model to obtain higher accuracy, but also helped the model to be trained more stably. Informed by this finding, we designed three kinds of more sophisticated stem layers, which can incorporate the attention mechanism into the stem layers and can extract the features of different receptive fields, as shown in Fig. 5.

As can be seen from the initial version of the stem layer in CMNet(referred as *stem a*), four copies of the original image data are fed into each of the four branches, and each branch is combined with different kinds of layers to guarantee

step 1:	$t1 = \text{branches}(x)$
step 2:	$t2 = V(t1)$
step 3:	$\text{attention\_tensor} = \text{sigmoid}(fc2(fc1(t2)))$
step 4:	$\text{output} = \text{attention\_vector} * t1$

Fig. 6 Pseudocode for stem b layer

different sizes of computational receptive field and the same downsampling multiplicity. Specifically, Branch 1 consists of a convolutional layer ( $k = 3, s = 2, p = 1$ ) and a maxpooling ( $k = 3, s = 2, p = 1$ ). Branch 2 has a similar structure as Branch 1, except that the size of the kernel in the convolution is replaced by 11 and the padding by 5. Branch 3 is created by stacking two dilated convolutions ( $k = 7, s = 2, p = 9, d = 3$ ) to obtain a larger computational receptive field. Branch 4 is similar in structure to Branch 3, except that the parameters in the dilated convolution are replaced with ( $k = 9, s = 2, d = 4, p = 16$ ). The computational receptive field of the four branches are  $7 * 7, 15 * 15, 55 * 55$  and  $97 * 97$ , respectively.

By this combination, the branches with small computational receptive fields can better extract texture information, and the branches with large computational receptive fields can better extract semantic information. Finally, the computational results of the four branches are concatenated in channel dimension to obtain a set of feature maps as the input of the downstream structure of the model.

In order to integrate attention mechanism into stem layers to improve model performance, we change the contribution of the four branches from the same to variable and introduce the attention mechanism in the channel dimension. This leads to an improved version of the *stem a* which is referred to as *stem b* as shown in Fig. 5.

Consider that texture information and semantic information are in different proportions in the image, and in the steam layer, the branch with smaller computational receptive fields is better at extracting texture information, and the branch with larger computational receptive fields is better at extracting semantic information, so it is reasonable to let the model learn by itself which branch is more important. Further, each feature map channel can also redistribute the importance by multiplying an attention tensor. Specifically, as shown in Fig. 6, we define a vector  $V$  of four learnable parameters  $v1, v2, v3, v4$ , with the same initial values of  $1e-2$ , which can be updated iteratively by network training in order to learn which branch is more important. After introducing these parameters for adaptive learning, we can describe the operation of the algorithm in Fig. 6 as follows.  $x$  denotes an image. The four sets of feature maps obtained after feeding  $x$  into each of the four branches are collectively referred to as  $t1$ . The four sets of feature maps represented by  $t1$  are multiplied by  $V(v1, v2, v3, v4)$ , respectively, to redistribute the branch importance, and obtain new feature maps ( $t2$ ). As for redistributing channel importance,  $t2$  will be sent to two fully connected layers( $fc1, fc2$ ) and a sigmoid function in order to

Table 3 Pseudocode for stem c layer

Step 1:	$t1 = \text{branches}(x)$
Step 2:	$t2 = \text{GlobalAvgPool}(t1)$
Step 3:	$t3 = t2.\text{transpose}(0, 2)$
Step 4:	$t4 = \text{softmax}(fc2(fc1(x))).\text{transpose}(0,2)$
Step 5:	$\text{attention vector} = \text{sigmoid}(fc4(fc3(t4)))$
Step 6:	$\text{value} = (\text{attention tensor} * t1).\text{transpose}(0, 1).\text{reshape}(B, 96, 56, 56)$

compute the attention tensor, which is similar to SENet [33]. Finally, attention tensor multiplied by  $t1$  to obtain an output.

Furthermore, in addition to initializing the vector  $V$ , we can also calculate the importance of branches by using the neural network layer, we designed another more sophisticated stem layers that can take into account the attention mechanism in both the branch and channel dimensions which is referred to as *stem c*. *stem c* can help the network learn important features faster and better and eventually improve the performance of the model. The pseudocode is as shown in Table 3.

In Table 3,  $x$  denotes an image with shape( $B = b, C = 3, H = 224, W = 224$ ), where  $B$  denotes the batch size for each batch gradient descent operation,  $C$  the channel dimension,  $H$  the image height, and  $W$  the image width. Four copies of  $x$  are fed into each of the four branches to obtain four sets of feature maps of shape  $(b, 32, 56, 56)$ , and they are stacked into a set of tensor  $t1$  of shape  $(4, b, 32, 56, 56)$ . Then,  $t1$  undergoes global average pooling in spatial dimension to obtain tensor  $t2$  of shape  $(4, b, 32)$ . Next, transpose the first and third dimensions of  $t2$  to obtain tensor  $t3$  with shape  $(32, b, 4)$ , and feed  $t3$  into two fully connected neural network layers ( $fc1, fc2$ ) and a softmax function to learn the importance of the four branches, and then transpose back to tensor  $t4$  with shape  $(4, b, 32)$ . Next, feed  $t4$  into two other fully connected layers ( $fc3, fc4$ ) and a sigmoid function to learn the importance of each channel to obtain an attention tensor, which has computed the attention of both branch and channel dimensions, and then this attention tensor is multiplied with  $t1$  and transposed to the first and second dimensions to obtain a tensor of shape  $(b, 4, 32, 56, 56)$ , and finally reshape it into the value of  $(b, 96, 56, 56)$  as the output result.

In order to prove the effectiveness of our designed three kinds of stem layers, we take resnet18 as the baseline model and replace the stem layers of resnet18 with stem a, stem b and stem c, respectively, the experimental results are shown in Table 4.

As shown in Table 4, we designed three kinds of stem layers to achieve higher verification accuracy compared to ResNet18, with stem c achieving the highest verification accuracy (68.6%), an improvement of 0.9% compared to

**Table 4** The accuracy of ResNet18 with different stem layers

	Resnet18	Stem a	Stem b	Stem c
Train accuracy	0.744	0.732	0.733	0.733
Val accuracy	0.677	0.680	0.685	0.686

ResNet18. This result illustrates that extracting rich multi-scale information in the stem layers of the network is beneficial to the model for better processing of image information. The presence of branches with a large computational receptive field in the stem layers of our design can help the network to extract semantic information early, instead of stacking network layers to extract semantic information in the deeper layers of the model as in the case of regular CNNs. In terms of human visual habits, when people observe image data, they first obtain some semantic information at the object level, and only when they look further carefully will they consider some texture, color, structure and other textural information. This is actually the opposite order of extracting image features by regular CNNs. And our proposed method of extracting image features by stem layers is more like the human visual habit.

## 4 The architecture of CMNet

Based on the experimental findings described in Section 3, we marry the strengths of both CNN and MetaFormer models to conceive a novel, yet simple backbone architecture dubbed as CMNet. This model adopts Fig. 2d as the block of the model, as such it can process images more efficiently compared to a regular CNN block. Considering the effect on sensitivity to model training hyperparameters, in our designed model CMNet, the residual connection of Fig. 3a will be used in the block of the model. As for convolution kernel size, because the improvement of large convolution kernels in image classification is not obvious, and considering that large convolution kernels will increase the complexity of the model, we still use small convolution kernels(3\*3) in the model CMNet. This model adopts stem b as the stem layer of the model, although the experimental results of stem c are a little bit better than stem b, considering the computational complexity, stem c is not used in CMNet, as shown in Fig. 7.

Specifically, in the stem layers, we use *stem b* to extract more detailed features and perform quadruple downsampling of the spatial dimension while increasing the dimension of the channel. On the one hand, *stem b* could extract texture features and semantic features at the same time by different computational receptive field in branches. On the other hand, *stem b* could learn the branch importance by parameters  $v$ , and learn the channel importance by two fully connected layers and sigmoid function. This means that *stem b* integrates

attention mechanisms into both spatial and channel dimensions.

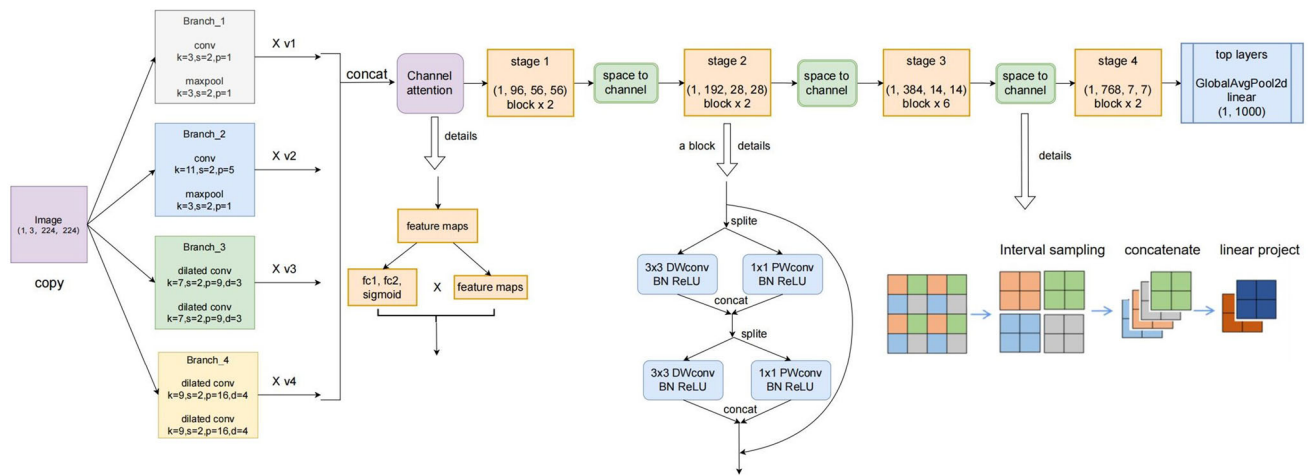
In term of S2C (space to channel), the operation purpose is to transfer the feature information from spatial dimensional to channel dimensional. Specifically, the four elements in the upper-left corner of the feature map are used as four starting points, and interval sampling to obtain four sub-features, and then the four sub-features are stack in the channel dimension, then through a point-wise convolution to obtain the final output. In terms of feature map shape, the result after S2C operation is two times downsampled in the spatial dimension and two times upsampled in the channel dimension than the input feature map.

As for stage, we designed the model with a total of four stages, each of which consists of many blocks (Fig. 2d), and during the computation, the size of the features map is not changed in a stage, but perform two-fold downsampling of the spatial dimension by an S2C (space to channel) structure between each stage. In a block, there exists a residual connection that contains two calculations of spatial and channel information within the connection, where DWconv (depth-wise convolution) responses for compute spatial information of input, and PWConv (point-wise convolution) responses for compute channel information. It is worth noting that each calculation of spatial information and channel information is performed separately and in parallel. The result after each convolution operation is subjected to BN (batch normalization) and fed to the activation function (ReLU).

In the top layer, we use a global average pooling operation to downscale the feature maps computed by the upper layer network into feature vectors, and then compute the feature vectors through a fully connected layer and output the final classification results.

In term of scaled models, the entire model can be scaled by adjusting the number of blocks and channels in each stage, and the model can be adapted to other vision tasks by modifying the top layer design. The following Table 5 shows one of our recommended model scaling strategies. To represent the computational complexity of the model, we computed the number of parameters and floating point operations (FLOPs) of the model, where the unit M of parameters is  $1 \times 10^6$  and the unit G of FLOPs is  $1 \times 10^9$ . The specific parameters and FLOPs calculation formula for regular CNNs and fully





**Fig. 7** CMNet: the novel and simple backbone architecture for vision task.  $k$  denotes the kernel size of convolution,  $s$  the stride of convolution,

$p$  the padding before convolution,  $d$  the dilate ratio of dilated convolution,  $v$  the parameters to learn branches importance, ( $batch, channel, height, weight$ ) the shape of the input tensor

**Table 5** The scaled version of CMNet:  $b$  denotes the number of blocks in a stage,  $s$  denotes the shape of the output, and the values in the shape denote the channels, feature map height and feature map width, respectively

	CMNet-tiny	CMNet-small	CMNet-base	CMNet-large	CMNet-huge
Stage 1	$b = 1$ $s = (64,56,56)$	$b = 2$ $s = (96,56,56)$	$b = 2$ $s = (128,56,56)$	$b = 4$ $s = (128,56,56)$	$b = 4$ $s = (160,56,56)$
Stage 2	$b = 1$ $s = (128,28,28)$	$b = 2$ $s = (192,28,28)$	$b = 2$ $s = (256,28,28)$	$b = 4$ $s = (256,28,28)$	$b = 12$ $s = (320,28,28)$
Stage 3	$b = 1$ $s = (256,14,14)$	$b = 2$ $s = (384,14,14)$	$b = 6$ $s = (512,14,14)$	$b = 24$ $s = (512,14,14)$	$b = 36$ $s = (640,14,14)$
Stage 4	$b = 1$ $s = (512,7,7)$	$b = 2$ $s = (768,7,7)$	$b = 2$ $s = (1024,7,7)$	$b = 4$ $s = (1024,7,7)$	$b = 4$ $s = (1280,7,7)$
Parameters (M)	1.9	5.6	11.7	26.0	52.0
FLOPs (G)	0.3	0.9	2.0	4.7	10.5

connected layers is as follows:

$$\text{param}_{\text{conv}} = (k_w * k_h * c_{in}) * c_{out} + c_{out}$$

$$\text{param}_{\text{fc}} = (n_{in} * n_{out}) + n_{out}$$

$$\text{FLOPs}_{\text{conv}} = [2 * (k_w * k_h * c_{in}) * c_{out} + c_{out}] * H * W$$

$$\text{FLOPs}_{\text{fc}} = 2 * (n_{in} * n_{out}) + n_{out}$$

In above formula,  $W$  and  $H$  denote the weight and height of input feature maps, respectively,  $k_w$  and  $k_h$  the convolutional kernel weight and height, respectively,  $c_{in}$  and  $c_{out}$  the number of channels of the input feature map before convolution and the number of channels of the output feature

map after convolution, respectively.  $n_{in}$  and  $n_{out}$  the number of input features and output features of the fully connected layer, respectively.

## 5 Experiments and discussions

To verify the performance of CMNet, we designed a set of comparison experiments using the ImageNet-1k dataset for image classification task with some classical models with similar computational complexity. The training strategy and experimental results are as follows.

**Table 6** Experimental results of CMNet and classical models on imagenet-1k

Model	Params. (M)	FLOPs. (G)	Top-1 Acc (%)	Top-5 Acc (%)
ResNet18	11.69	1.8	71.21	90.82
DenseNet169	14.30	3.4	76.25	93.27
Xception	22.86	8.4	79.00	94.55
EfficientNet-B2	9.2	9.2	80.15	94.93
RegNetY-1.6GF	11.2	1.6	78.73	92.75
ConViT-Ti +	10	2.0	76.75	90.17
PoolFormer-S12	11.9	2.0	77.27	91.69
CMNet-tiny	4.9	0.7	76.11	88.99
CMNet-small	9.1	1.3	78.29	90.21
CMNet-base	11.7	2.0	80.08	93.70
CMNet-big	33.9	6.3	81.09	94.65
CMNet-huge	50.4	9.1	83.77	95.51

## 5.1 Dataset

ImageNet [34] is a computer vision system recognition project built by computer scientists to simulate the human recognition system. It provides datasets for several classical computer vision tasks such as image classification and target detection. ImageNet-1k is a subset of the ImageNet database, commonly referred to as ISLVR 2012 (ImageNet Large Scale Visual Recognition Challenge) [35], and it is one of the most well-known datasets for image classification tasks and has been used as a standard dataset in many works for model performance comparisons. The dataset has 1281167 labeled training images, 50,000 validation images and 10,000 training images, with a total of 1000 categories.

## 5.2 Training setting

The model training setting we used mostly follows [36], which mainly includes dataset augmentation and training scheme. As for augment the training data, we adopt cutmix [37], mixup [38], repeated augmentation [39], label-smoothing [40], random erasing [41] random clipping and random horizontal flipping. In term of training scheme, we use *Adam* optimizer, the batch size is set to 1024, the number of training epochs is 600, the learning rate uses Cosine descent strategy [42] and Warm-Up mechanism [43], where the initial learning rate is 0.01 and the number of warm-up epochs is 5, Exponential moving average (EMA) [44] was also used during training. As for the training device, we used eight Tesla V100 GPUs and trained CMNet for seven days, finally, we report the top-1 accuracy in the ImageNet-1k validation dataset.

## 5.3 Experiment results

As shown in Table 6, we compare classical models with similar computational complexity, such as ResNet18 [5], DenseNet169 [6], xception [30], EfficientNet-B2 [45], RegNetY-1.6GF [46], ConViT-Ti+ [12] and PoolFormer-S12 [21]. The experimental results prove that CMNet achieves the best validation accuracy compared to these seven classical models with similar computational complexity.

From the table, we can find that CMNet can obtain the highest classification accuracy compared with the previous classical models with similar computational complexity. First, because of the presence of the stem b layer, CMNet can extract rich multi-scale features at the early stage of the network than other classical models, which is important for the subsequent computation. Second, compared with regular CNN, depth-wise convolution can save a large number of parameters, and parallel execution with point-wise convolution can perform feature extraction more efficiently while preserving the inductive bias of convolution. Finally, inspired by the attention mechanism in Transformer, we incorporate the attention mechanism into the branch and channel of the stem b layer, which is helpful for selecting important features. Combining the advantages of CNN and TransFormer, the effectiveness of CMNet can be expected.

To further study the generalization performance of the model, we verified it on the CIFAR-100 data set and compared it with other models. We found that the performance of CMNet is still excellent, as shown in Table 7. Compared with the previous SOTA model, CMNet can get better performance under the premise of similar parameters and computational complexity. Especially CMNet-tiny, the number of parameters is less than half of ResNet18, but the accuracy is higher. Compared with the previous SOTA model,

**Table 7** Experimental results of CMNet and classical models on CIFAR-100

Model	Params. (M)	FLOPs. (G)	Top-1 Acc (%)	Top-5 Acc (%)
ResNet18	11.69	1.8	77.69	93.49
DenseNet169	14.30	3.4	80.48	93.27
Xception	22.86	8.4	80.00	94.02
CMNet-tiny	4.9	0.7	78.12	93.94
CMNet-small	9.1	1.3	83.20	94.60
CMNet-base	11.7	2.0	84.08	96.72
CMNet-big	33.9	6.3	88.04	97.68
CMNet-huge	50.4	9.1	90.77	97.89

CMNet can get better performance under the premise of similar parameters and computational complexity. Especially CMNet-tiny, the number of parameters is less than half of ResNet18, but the accuracy is higher.

## 5.4 Future work

In future, we will continue to explore the performance of CMNet from the following aspects.

### 5.4.1 Testing the impact of scaling models on performance

Due to the limited training resources, the computational complexity of our experimental CMNet-base model is smaller than that of the existing SOTA models, the number of parameters is only about 0.2 times that of the SOTA models [47–50]. Therefore, a future experimental direction is to increase the computational complexity of CMNet by increasing the number of blocks and channels in each stage of the model, and to test the performance of the scaled-up model on imagenet-1k in order to explore the highest accuracy that can be achieved by scaling CMNet.

### 5.4.2 Exploring the performance of CMNet as a backbone on other visual tasks

Specifically, we use the part of CMNet with the top layer removed as a feature extractor for images, and use the extracted features due to other classical computer vision tasks, such as object detection of images and semantic segmentation. In this way, we test the robustness of CMNet as a backbone for vision tasks. We look forward to seeing CMNets becoming a general model. By the way, trying to improve model performance by pre-training CMNet with an additional large dataset before the training task is also a direction worth exploring.

### 5.4.3 Continue to explore the stem layer

In CMNet, we design three kinds of sophisticated stem layers (*stem a*, *stem b* and *stem c*) to help the model extract rich features. In fact, the stem layers can be made deeper. For example, in CMNet, each branch contains only two layers, we can make the stem layer deeper by increasing the number of layers in each branch. In addition, the stem layer in CMNet implements quadruple downsampling, and it also makes sense to explore more downsampling in the stem layer. In addition, for multi-object detection, a task that is sensitive to multi-scale features, we have reasons to believe that the stem layers we designed are more helpful for the model to improve detection accuracy.

## 6 Conclusion

In this paper, we analyzed and identified four main differences between regular CNN-based models and MetaFormer-based models in the design of backbone architecture. We designed four sets of ablation experiments to evaluate which is more beneficial to processing images and subsequently draw four observations based on the experimental results, which can be serve references for designing the vision model backbone architecture. They are: (1) Separate computing of spatial information and channel information is more effective. (2) MetaFormer's residual connection method is more sensitive to learning rate.

(3) It is effective to use large convolution kernels to simulate the self-attention mechanism. (4) Extracting more detailed features in the stem layers is important. Notably, to satisfy the principle (4) we design three kinds of sophisticated stem layers, which can be easily embedded into the existing classical model and help the model to extract multi-scale features. Finally, based on these ablation experiments observations, we design a novel and simple model: CMNet, which exhibits sufficiently competitive experimental results compared to previous classical models of similar computational

complexity. We also discussed in details potential research topics and directions to inform and inspire future research.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* 25 (2012)
- Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: *European Conference on Computer Vision*, pp. 818–833. Springer, Cham (2014)
- Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Rabinovich, A.: Going deeper with convolutions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9 (2015)
- He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778 (2016)
- Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700–4708 (2017)
- Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Adam, H.: Mobilenets: efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017)
- Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: an extremely efficient convolutional neural network for mobile devices. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6848–6856 (2018)
- Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., Xu, C.: Ghostnet: more features from cheap operations. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1580–1589 (2020)
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Polosukhin, I. (2017). Attention is all you need. *Adv. Neural Inf. Process. Syst.* 30
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Houselby, N.: An image is worth 16x16 words: transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020)
- d'Ascoli, S., Touvron, H., Leavitt, M.L., Morcos, A. S., Biroli, G., Sagun, L.: Convit: improving vision transformers with soft convolutional inductive biases. In: *International Conference on Machine Learning*, pp. 2286–2296. PMLR (2021)
- Han, K., Xiao, A., Wu, E., Guo, J., Xu, C., Wang, Y.: Transformer in transformer. *Adv. Neural. Inf. Process. Syst.* 34, 15908–15919 (2021)
- Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Shao, L.: Pyramid vision transformer: a versatile backbone for dense prediction without convolutions. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 568–578 (2021)
- Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022 (2021)
- Bello, I., Zoph, B., Vaswani, A., Shlens, J., Le, Q.V.: Attention augmented convolutional networks. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3286–3295 (2019)
- Hu, H., Gu, J., Zhang, Z., Dai, J., Wei, Y.: Relation networks for object detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3588–3597 (2018)
- Sun, C., Myers, A., Vondrick, C., Murphy, K., Schmid, C.: Videobert: a joint model for video and language representation learning. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7464–7473 (2019)
- Strudel, R., Garcia, R., Laptev, I., Schmid, C.: Segmenter: transformer for semantic segmentation. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7262–7272 (2021)
- Tolstikhin, I.O., Houselby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., et al.: Mlp-mixer: an all-mlp architecture for vision. *Adv. Neural Inf. Process. Syst.* 34, 24261–24272 (2021)
- Yu, W., Luo, M., Zhou, P., Si, C., Zhou, Y., Wang, X., Yan, S.: Metaformer is actually what you need for vision. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10819–10829 (2022)
- Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986 (2022).
- Trockman, A., Kolter, J.Z. Patches are all you need?. *arXiv preprint arXiv:2201.09792* (2022).
- Sifre, L., Mallat, S.: Rigid-motion scattering for texture classification. *arXiv preprint arXiv:1403.1687* (2014).
- Lin, M., Chen, Q., Yan, S.: Network in network. *arXiv preprint arXiv:1312.4400* (2013).
- Guo, M.H., Lu, C.Z., Liu, Z.N., Cheng, M.M., Hu, S.M.: Visual attention network. *arXiv preprint arXiv:2202.09741* (2022).
- Xie, S., Girshick, R., Doll'ar, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1492–1500 (2017)
- Li, X., Wang, W., Hu, X., Yang, J.: Selective kernel networks. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 510–519 (2019)
- Woo, S., Park, J., Lee, J.Y., Kweon, I.S.: Cbam: convolutional block attention module. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3–19 (2018)
- Chollet, F.: Xception: deep learning with depthwise separable convolutions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1251–1258 (2017)
- Ding, X., Zhang, X., Han, J., Ding, G.: Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11963–11975 (2022)
- Xiao, T., Singh, M., Mintun, E., Darrell, T., Dollar, P., Girshick, R.: Early convolutions help transformers see better. *Adv. Neural Inf. Process. Syst.* 34, 30392–30400 (2021)

33. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7132–7141 (2018)
34. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. IEEE (2009)
35. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Fei-Fei, L.: Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.* **115**(3), 211–252 (2015)
36. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jegou, H.: Training data-efficient image transformers & distillation through attention. In: International Conference on Machine Learning, pp. 10347–10357. PMLR (2021)
37. Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.: Cutmix: regularization strategy to train strong classifiers with localizable features. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6023–6032 (2019)
38. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412* (2017).
39. Hoffer, E., Ben-Nun, T., Hubara, I., Giladi, N., Hoefler, T., Soudry, D.: Augment your batch: better training with larger batches. *arXiv preprint arXiv:1901.09335* (2019).
40. Müller, R., Kornblith, S., Hinton, G.E.: When does label smoothing help?. In: *Adv. Neural. Inf. Process. Syst.* **32** (2019).
41. Zhong, Z., Zheng, L., Kang, G., Li, S., Yang, Y.: Random erasing data augmentation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34(07), pp. 13001–13008 (2020)
42. Loshchilov, I., Hutter, F.: SGDR: stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983* (2016).
43. Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., He, K.: Accurate, large minibatch SGD: training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677* (2017).
44. Polyak, B.T., Juditsky, A.B.: Acceleration of stochastic approximation by averaging. *SIAM J. Control. Optim.* **30**(4), 838–855 (1992)
45. Tan, M., Le, Q.: Efficientnet: rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning, pp. 6105–6114. PMLR (2019)
46. Radosavovic, I., Kosaraju, R.P., Girshick, R., He, K., Dollár, P.: Designing network design spaces. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10428–10436 (2020)
47. Yu, J., Wang, Z., Vasudevan, V., Yeung, L., Seyedhosseini, M., Wu, Y.: Coca: contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917* (2022)
48. Wortsman, M., Ilharco, G., Gadre, S.Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A.S., et al.: Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In: International Conference on Machine Learning, pp. 23965–23998. PMLR (2022)
49. Dai, Z., Liu, H., Le, Q.V., Tan, M.: Coatnet: marrying convolution and attention for all data sizes. *Adv. Neural. Inf. Process. Syst.* **34**, 3965–3977 (2021)
50. Zhai, X., Kolesnikov, A., Houlsby, N., Beyer, L.: Scaling vision transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12104–12113 (2022)

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Haowen Yu** received his B.Sc in Computer Science and Technology from Shandong University of Science and Technology, China, in 2020, and M.Sc in Artificial Intelligence from University of Southampton in the UK in 2021. He is currently studying at the University of Manchester, UK, where he will receive his Master's degree in November 2023 in Bioinformatics and Biosystems. His research interests are in data analytics, machine learning, artificial intelligence and applications of the above algorithms to computer vision and bioinformatics. He has published one conference article and two books in the above fields.

**Liming Chen** received his B.Eng and M.Eng degrees from Beijing Institute of Technology, China, in 1985 and 1988 respectively, and his Ph.D. degree from De Montfort University, UK, in 2003. He is currently Professor of Data Analytics and Research Director for the School of Computing, Ulster University, UK. His research interests include pervasive computing, data analytics, artificial intelligence, user-centred intelligent systems and their applications in smart health-care and cyber security. He has over 300 publications in the aforementioned areas. Liming is an IET Fellow and a Senior Member of IEEE.