Knowledge-Based Systems 258 (2022) 109792

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Multi-Scale MLP-Mixer for image classification

Hong Zhang [a,b,*], ZhiXiang Dong [a,b], Bo Li [a,b], Siyuan He [c]

[a] College of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan, 430065, Hubei, China
[b] Hubei Province Key Laboratory of Intelligent Information Processing and Real-Time Industrial System, Wuhan University of Science and Technology, Wuhan, 430065, Hubei, China
[c] Gulf Coast Research and Education Center, University of Florida, Wimauma, 33598, FL, USA

ARTICLE INFO

Article history:
Received 25 May 2022
Received in revised form 23 August 2022
Accepted 24 August 2022
Available online 5 September 2022

Keywords:
MLP-Mixer
Theoretical calculation cost
Multi-scale
Actual reasoning speed

ABSTRACT

MLP-Mixer is a vision architecture that solely relies on multilayer perceptrons (MLPs), which despite their simple architecture, they achieve a slightly inferior accuracy to the state-of-the-art models on ImageNet. Given that the MLP-Mixer segments each input image into a fixed number of patches, small-scale MLP-Mixers are preferred due to attaining better accuracy because the image is segmented into more patches. However, this strategy significantly increases the computational burden. Nevertheless, this paper argues that even in the same dataset, each image has a different recognition difficulty due to its characteristics. Therefore, in the ideal case, choosing an independently scaled MLP-Mixer for each image is the most economical computational approach. Hence, this paper experimentally verifies the objective existence of this phenomenon, which inspires us to propose the Multi-Scale MLP-Mixer (MSMLP) that utilizes a suitably scaled MLP-Mixer for each input image. MSMLP comprises several MLP-Mixers of different scales. During testing, these MLP-Mixers are activated in order of scale from large to small (increasing number of patches and decreasing patch size). In addition, to reduce redundant computations, a feature reuse mechanism is designed between neighboring MLP-Mixers so that the small-scale MLP-Mixer downstream can reuse the features learned by the larger-scale MLP-Mixer upstream. Finally, extensive experiments on the public dataset CIFAR10/100 reveal that our method's theoretically estimated computational cost and actual inference speed are significantly higher than those of MLP-Mixer.

1. Introduction

The MLP-Mixer [1] is a more concise visual architecture than Vision Transformers (ViT), while its inference speed is higher than ViT. An MLP-Mixer solely relies on multi-layer perceptrons (MLPs) to achieve a slightly weaker accuracy than the state-of-the-art (SOTA) models on ImageNet. Since the MLP-Mixer was proposed, various MLP-Mixer-based vision architectures have emerged, including RepMLP, ResMLP, and gMLP. These models segment each image into a fixed number of patches and then fuse their spatial and channel domain, respectively. In general, segmenting the input image into more patches using a smaller patch size helps to improve accuracy but is computationally more expensive.

However, in this paper, we argue that it is not optimal to use the same scale of MLP-Mixers for all samples, as even in the same data set, each image has a different recognition difficulty due to its characteristics. Therefore, using the most suitable scale of MLP-Mixers for images with different recognition difficulties to segment the images into the correct number of patches significantly reduces the computational effort. For the MLP-Mixer, smaller scales, i.e., using smaller patch sizes, tend to afford higher accuracy and greater computational effort. This conclusion is verified on CIFAR10, with the corresponding results reported in Table 1, which highlights that splitting the image into 64 patches afford correctly identifying more samples but increases the GFLOPs. Moreover, splitting the image into four patches can also correctly identify 70.6% of the samples, revealing that 70.6% of the samples in the dataset only need to be split into four patches to be correctly identified. However, if this part of the sample had been split into 64 patches, the computational effort would also have increased.

Based on this observation, we propose a Multi-Scale MLP-Mixer (MSMLP) framework, which aims to improve the computational efficiency using MLP-Mixers of different scales for images with different recognition difficulties. Specifically, MSMLP comprises multiple MLP-Mixers at different scales. During testing, these MLP-Mixers are activated in order of scale from large to small (increasing number of patches and decreasing patch size). Once the prediction reaches an appropriate level or the

* Corresponding author at: College of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan, 430065, Hubei, China.
E-mail address: zhanghong_wust@163.com (H. Zhang).

https://doi.org/10.1016/j.knosys.2022.109792
0950-7051/

Table 1
Accuracy and GFLOPs of MLP-Mixer vs. numbers of patches on the CIFFA-10 dataset.

Patches (patch size)	GFLOPs	Accuracy
4 (16 × 16)	0.07	66.53%
16 (8 × 8)	0.30	74.65%
64 (4 × 4)	1.21	81.87%

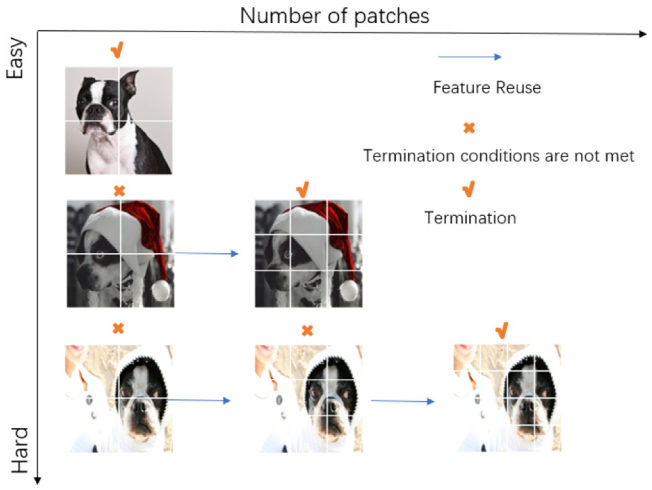


Fig. 1. Three images containing a tortoise. The vertical axis represents the difficulty of correctly predicting the image. As the recognition difficulty increases, the number of patches required to predict the image correctly also increases. The horizontal axis represents the number of patches allocated by MSMLP to the current image. If the prediction does not reach the confidence level, MSMLP will continue to increase the number of patches allocated until it reaches the output of the last layer.

final layer's output, the model terminates the inference and outputs the result. In this way, the computational resources can be allocated to the samples more efficiently, resulting in a significant increase in computational efficiency. In addition, we design a feature reuse mechanism to reduce redundant computations. This strategy allows the downstream small-scale MLP-Mixer to reuse the features learned by the upstream large-scale MLP-Mixer (Fig. 1).

Compared with the original model MLP-Mixer the proposed method has the following advantages:

- MSMLP can dynamically allocate the appropriate computing resources to reduce the waste of computing resources.
- We further add a feature reuse mechanism that allows the downstream small-scale MLP-Mixer to reuse the features learned by the upstream large-scale MLP-Mixer. This strategy further reduces the computational effort downstream.

The performance of the proposed MSMLP is evaluated on the public dataset CIFAR10/100 [2], with the experimental results demonstrating that MSMLP significantly reduces the computing resources requirements. Furthermore, our trials reveal that the actual inference speed on the NVIDIA 1070 GPU is also improved.

2. Related work

This section reviews the related work on MLP-mixers, efficient deep networks, and dynamic models, which are the governing methods inspiring our network architecture.

2.1. MLP-Mixer

The MLP-Mixer is a visual architecture proposed by Google that does not require convolution and attention modules but solely comprises MLP. Despite its simple structure, it achieves extremely competitive results. Indeed, when pre-trained on large datasets (≥100M images), it achieves an accuracy/cost trade-off nearly the state-of-the-art performance claimed by CNNs and Transformers [1].

Some follow-up works such as ResMLP [3] and gMLP [4] focused on improving the MLP-Mixer architecture. An alternative research direction proposed adding a local prior to the MLP, e.g., RepMLP [5], which re-parameterized Convolutions into Fully-connected Layers (FC). Additionally, MLP-Mixers have also been used in GNNs for graph neural networks [6–9]. Overall, current works pave the way for applying the MLP-Mixer model to different research fields. Nevertheless, to the best of our knowledge, this work is the first to consider optimizing the MLP-Mixer model by constructing a multi-scale MLP-Mixer.

2.2. Efficient deep networks

Although the computational power is increasing, applying deep networks in real applications improves their computational efficiency. To date, many works have investigated reducing the computational cost of CNNs [10–16], with new ideas considering reducing the computational cost of Vision Transformer [17–21]. Meanwhile, other computer vision areas are constantly exploring ways to utilize computational resources efficiently [22]. However, designing an efficient MLP-Mixer is still an under-explored topic, partially because the MLP-Mixer was developed not to use self-attention, a mechanism that requires more computing resources. Current approaches include RepMLP, which re-parameterized the convolution into a fully connected layer so that the network significantly increases its parameters without affecting the inference speed. Compared to the models with fixed computational maps, the proposed MSMLP framework is less computationally intensive by adaptively varying the scale of the MLP-Mixer on a per-sample basis.

2.3. Multi-scale models

Designing multi-scale models is an effective tool used in image classification to reduce the amount of deep learning computations [23]. Since the introduction of the multi-classifier Multi-Scale DenseNets (MSDnet) [24] in 2017, various variants of multi-classifier architectures have also emerged [25–28], which terminate the model's inference early. In addition, several multi-scale models exist that reduce the computational burden by skipping redundant layers [29–31] or channels [32] that are conditional on the input. However, all these multi-scale models are based on CNNs. Some multi-scale model studies are also based on Transformer [33]. For example, the Dynamic Vision Transformer (DVT) [27] is a Vision Transformer-based on a multiclassifier architecture, while ConvMAE [34] proposes a simple chunking mask strategy to ensure a multiscale Vision Transformer [35] that is computationally efficient. The suggested multi-scale model is based on the MLP-Mixer.

3. Multi-Scale MLP-Mixer

An MLP-Mixer segments each 2D image into a sequence of patches based on the patch size and then uses these patches as input. The mixer uses two MLP layer types: channel-mixing MLPs and token-mixing MLPs, with the former type allowing the communication between different channels and the latter acting on the columns of each patch. The experiments in [1] revealed that a smaller scale MLP-Mixer (more patches) affords better accuracy, which also implies more computational effort. However, Table 1 reports that on the same data set, each image imposes a

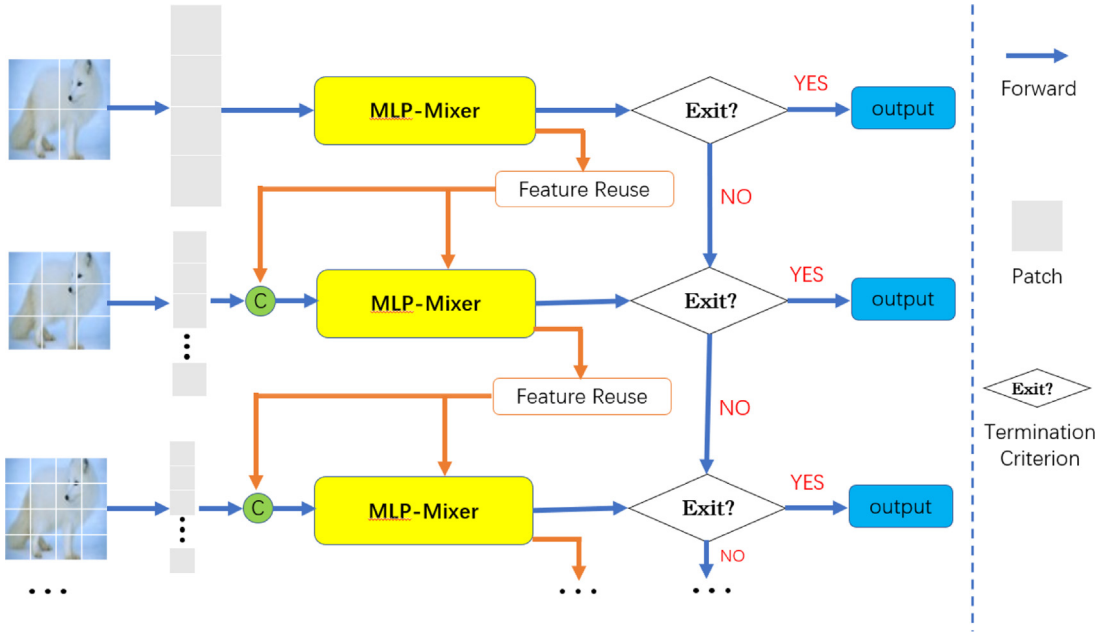


Fig. 2. An overview of the Multi-Scale MLP-Mixer (MSMLP). We cascade multiple MLP-Mixers with decreasing patch size and add a feature reuse mechanism between them to reduce the computing resources consumption. During testing, the entire model is activated from top to bottom until a sufficient degree of confidence is reached, or the final model is inferred.

different recognition difficulty originating from its characteristics, where images of lower recognition difficulty can typically be correctly recognized with fewer patches (large scale MLP-Mixer). Spurred by this finding, we propose a Multi-Scale MLP-Mixer (MSMLP) that improves the MLP-Mixer's computational efficiency by adaptively selecting the appropriate scale of the MLP-Mixer for each input image.

3.1. Overview

Inference procedure. The Inference procedure of MSMLP is illustrated in Fig. 2. For each test sample, first, we split the image into a sequence of patches according to the patch size, and the input shape is patches $\times$ channels. This is achieved directly through a convolutional layer. Then we infer an MLP-Mixer that utilizes these image patches to obtain a quick prediction. Since the computational cost of MLP-Mixer increases as the number of patches increases, the related processing efficiency is very high when the number of patches is small. Then, MSMLP uses a certain standard to evaluate the predicted value to determine whether the result is reliable and if it can be retrieved immediately. If it is considered reliable, the process terminates.

Unless the termination condition is met, MSMLP will proceed to the downstream computation, i.e., the small-scale MLP-Mixer is used to process the image. The original image will be split into more patches to obtain more accurate but computationally expensive inferences. However, the number of channels is constant, i.e., the number of channels in the input shape is unchanged, and we only change the number of image patches. After splitting the input image into more patches, we activate an additional MLP-mixer with the same structure as the previous one but at a different scale. In theory, some "difficult" test samples are calculated at this stage. To improve efficiency and not waste the computations of the upstream large-scale MLP-Mixer, we design a feature reuse mechanism that allows the downstream small-scale MLP-Mixer to reuse the features learned by the upstream model. Similarly, after obtaining a new prediction, the MSMLP continues to judge whether it meets the criteria for early termination. If the criteria are not met, the procedure is repeated until the sample exits, or the final MLP-Mixer is reached.

Training. For our model, all outputs are predictions of the same type. Therefore, we only need to train the model's output during training. Formally, the optimization objective is:

$$Minimize \frac{1}{\|D_{train}\|} \sum_{(x,y)\in D_{train}} \left[\sum_{i} L_{CE}(p_i, y) \right]. \tag{1}$$

where D_{train} represents the training set and (x,y) denotes a sample in the training set D_{train} and its corresponding label. We adopt the standard cross-entropy loss function (CELoss) $L_{CE}(\cdot)$, with P_i denoting the softmax prediction probability output by the i^th exit. We found that this training strategy affords better results in practice.

3.2. Feature reuse

In order to utilize computing resources more effectively, we promote the reuse of computing. Specifically, when the upstream MLP-Mixer's forecast does not meet the conditions for early termination, discarding the computations performed by previous models is an inefficient approach. This is because although the upstream model did not meet the conditions for early termination, it has the same training goal as the downstream model and has extracted valuable information for fulfilling this task. Therefore, we propose a feature reuse mechanism that reduces additional computer costs and improves the test accuracy by using the upstream model's features.

For ease of introduction, we first revisit the basic formulation of MLP-Mixer, which contains multi-layer perceptron (MLP) blocks and Mixer blocks. The MLP blocks comprise two fully connected (FC) layers, and an activation function GELU [36]. Accordingly, the Mixer blocks comprise two MLP block types: channel-mixing and patch-mixing MLPs. The former type allows the communication between different channels [1] and the latter type allows communication between different spatial locations (patches). The layer normalization (LN) [37] and Skip-connection [38] are applied before and after each channel-mixing

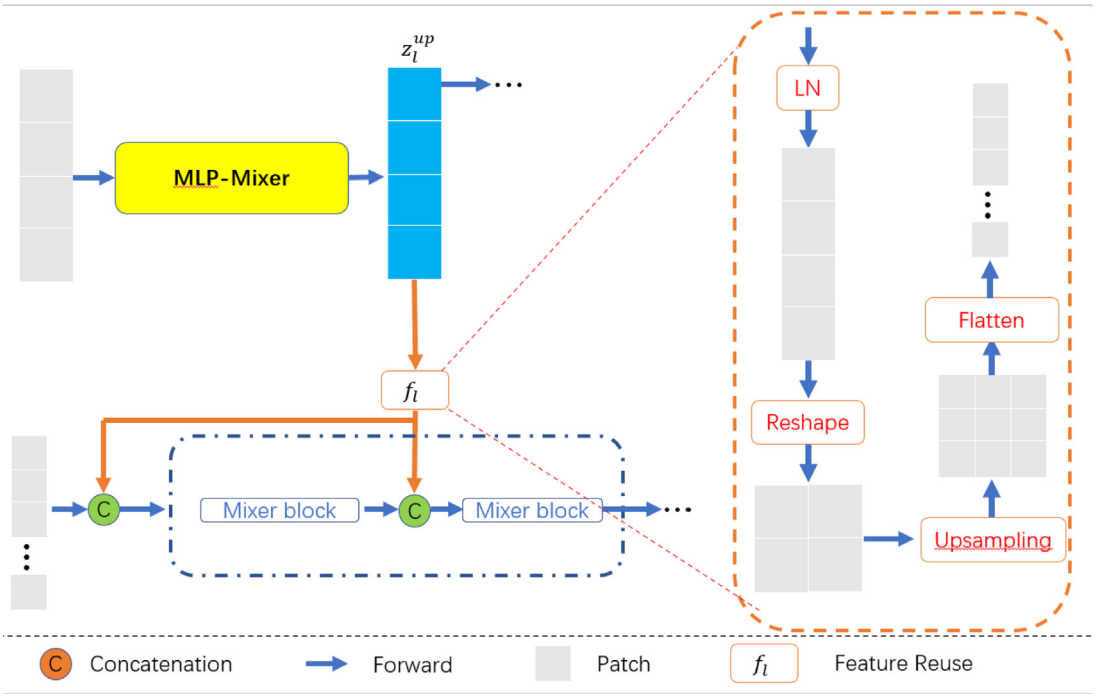


Fig. 3. Illustration of the feature reuse mechanism. First, LN processes the upstream model's output Z_l^{up}. Then the one-dimensional data is reshaped into two-dimensional data, and in the third step, the two-dimensional data is transformed through up-sampling to match the scale of the downstream model. Finally, the data is flattened, used as input to the downstream model, and integrated into the Mixer block of each downstream MLP-Mixer layer [1].

MLP and patch-mixing MLP, respectively. Let $X \in R^{(S\times C)}$ denote the input of the Mixer block, where S is the number of patches, and C is the number of channels. If the original input image and each patch have a resolution of (P, P) and (H, W), respectively, then the number of patches is S = HW/P*P [1]. Formally, the Mixer block can be written as follows:

$$M = FC(GELU(FC(x))) \tag{2}$$

$$MC = M(LN(X^{S\times C})) \tag{3}$$

$$MP = M(LN(X^{C\times S})) \tag{4}$$

$$Z_l' = Z_{l-1} + MP(Z_{l-1}) \qquad l \in \{1, \ldots, L\} \tag{5}$$

$$Z_l = Z_l' + MC(Z_l') \qquad l \in \{1, \ldots, L\} \tag{6}$$

where L is the total number of Mixer blocks in an MLP-Mixer and M denotes an MLP block, which returns the output of the same dimension as the input by specifying the hidden layer dimension. MC denotes a Channel-mixing MLP, which accepts input with a resolution of S×C, and similarly, MP is a patch-mixing MLP that accepts input images of C×S resolution. Moreover, Z_l is input into an LN followed by a Global average pooling layer (GAP) [39] and a fully-connected layer for the final prediction. The feature reuse mechanism aims to reuse the upper layer Z_l.

Reuse of upstream large scale MLP-Mixer features. All Mixer blocks in the MLP-Mixer share the same goal of clearly separating the per-location (channel-mixing) and cross-location (patch-mixing) operations for accurate recognition. Therefore, the downstream model learns based on the features obtained during the upstream process instead of extracting features from scratch, which is more efficient because the computational resources consumed in the upstream are also utilized in the downstream. To realize this idea, we propose a feature reuse mechanism (see: Fig. 3). Specifically, we use the output Z_l^{up} of the upstream large-scale Mixer block to learning the layer-wise embedding E_l of the downstream model:

$$E_l = f_l(Z_l^{up}) \in \mathbb{R}^{S^{up}\times C} \tag{7}$$

Herein, f_l: $R^{(S^{up}\times C)} \rightarrow R^{(S^{down}\times C)}$ comprises a sequence of operations starting with an LN. Then the image patches are reshaped to the corresponding locations in the original image, which are upsampled and flatted to match the number of patches of the downstream model (S^{down}).

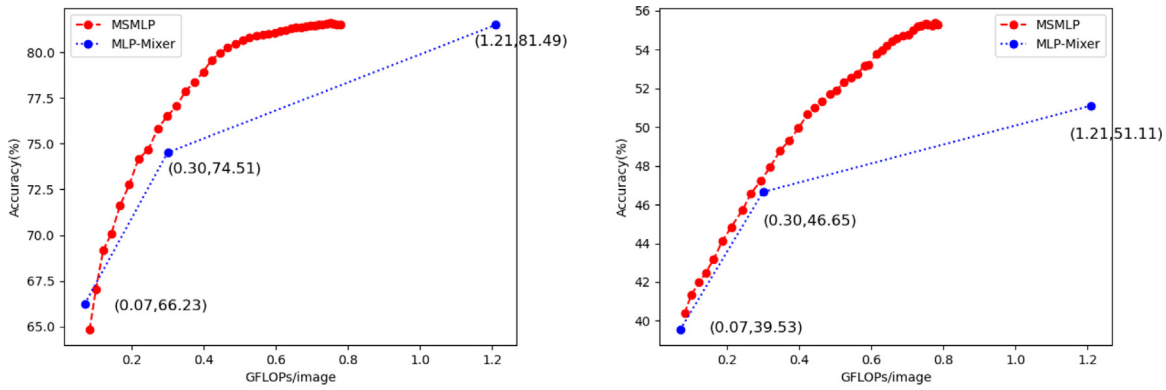
After obtaining the embedding E_l of the upstream large-scale MLP-Mixer, we concatenate it with the input $Z_{(l-1)}$ of Mixer blocks in the downstream process. Formally, we replace (5) and (6) by:

$$Z_l' = MLP(LN(Concat(Z_l, E_l))) \qquad l \in \{1, \ldots, L\} \tag{8}$$

where E_l is concatenated with $Z_{(l-1)}$. To change the dimension from $S^down + S^up$ to S^down, we add LN and the first layer of MLP since E_l is based on the upstream outputs Z_l^{up} and the downstream inputs $Z_{(l-1)}$ have more patches than Z_l^{up}. Therefore, Z_l' concludes the context information of the input image for different patches. So, E_l is considered embedding the prior information into the following. According to our experiments, E_l can be used multiple times in the downstream model, further enhancing our schemes' effectiveness. Since the goal is to minimize the final loss (1), the feature reuse mechanism implicitly increases the model's depth.

3.3. Adaptive scale

As mentioned above, our proposed multi-scale MLP-Mixer uses a smaller-scale MLP-Mixer to process the samples until the entire model computation process is finished or terminated early. In this way, images with different recognition difficulties exploit MLP-Mixers of different scales to improve the overall efficiency. Specifically, we set the corresponding threshold η_i for the ith exit



(a) CIFAR -10 (b) CIFAR -100

Fig. 4. Top-1 accuracy v.s. GFLOPs on CIFAR -10/100. MSMLP is implemented on MLP-Mixer-s.

Table 2
The practical speed of MSMLP on CIFAR -10/100.

Models	CIFAR-10		CIFAR-100	
	Top-1 acc	Throughput	Top −1 acc	Throughput
MLP-Mixer-s	81.49%	774 img/s	51.11%	746 img/s
MSMLP	**81.58%**	1064 img/s (↑1.37x)	**55.81%**	1032 img/s (↑1.38x)
MLP-Mixer-b	81.51%	292 img/s	52.34%	292 img/s
MSMLP	**81.87%**	400 img/s (↑1.36x)	**55.26%**	406 img/s (↑1.39x)

of the ith layer. If the confidence is $\psi_i \leq \eta_i$, the inference stops, and the results are output. Otherwise, the MSMLP activates the downstream small-scale MLP-Mixer for inference. For the lowest model, the threshold is fixed to zero. Since the input of ith exit is the softmax prediction p_i, we set as confidence the largest entry of p_i [24].

In the following experiment, we first set a weight for each classifier, representing the percentage of the sample data output by the classifier. For example, consider 100 samples, and the weight of the first classifier is 0.5. Then this classifier should output the classification results of 50 samples. The weights are manually set, and we define 40 sets of weights. Then we use the validation set to determine the need to calculate all samples, and the 50th predicted probability is the threshold of this classifier. During the experiment, we use the validation set to determine the threshold of each classifier and then evaluate it on the test set. The detailed procedure is presented in Algorithm 1.

Algorithm 1 Determining the classifier's threshold value

Require: weight W: [w1, w2,...wn]; Validation set: V; Trained model: model; classifiers C: [c1,c2,...cn];
Ensure: threshold T :[t1,t2,...tn]
output=model(V),output:[classifiers,Samples,classes]
for i,classifier in enumerate(output) **do**
C[i]=classifier.max(dim=1)
C[i]=C[i].sort(dim=0)
T[i]=C[i][$W[i] * len(V)$]
end for

4. Experiment

This section validates the proposed Multi-Scale MLP on the public dataset CIFAR-10/100 [40]. The evaluation indicators employed are (1) Gillions of Floating Point Operations (GLOPs), (2) Throughput per unit time, and (3) Final accuracy. It should be mentioned that our goal is not to achieve state-of-the-art results but to attain better results than the original MLP-Mixer. Therefore, unless otherwise specified, our competitor model is the MLP-Mixer [1].

Datasets. The CIFAR-10/100 dataset contains 10/100 classes, comprising 50,000 training and 10,000 test images of size 32 × 32 pixels. We utilize 5000 images from the training set as the validation set. Finally, our data pre-processing and data augmentation policy are the same as in [24,41].

Backbones. Our experiments are based on an MLP-Mixer. Unless otherwise stated, we deploy the Multi-Scale MLP-Mixer with three outlets. When calculating FLOPs, we employ the fvcore toolkit provided by Facebook AI Research [42–44]. Additionally, the MLP-Mixer-s and MLP-Mixer-b in the experimental section represent the MLP-Mixer with 8 and 12 Mixer blocks, respectively.

Training Details. On the two CIFAR data sets, all models are trained with a batch size of 64 using the stochastic gradient descent (SGD), with a 0.9 momentum and a 10^{-4} weight decay. All models are trained for 300 epochs, with an initial learning rate of 0.03, divided by a coefficient of 10 after 150 and 225 iterations. The patch size corresponding to the three classifiers is 16 × 16, 8 × 8, and 4 × 4, respectively.

4.1. Main results

The CIFAR results are illustrated in Figs. 4(a) and 4(b), where the backbone is the MLP-Mixer-s. As mentioned above, we first set 40 sets of weights and drew their results as a red curve. By comparing against the MLP-Mixer results, we find that MSMLP always reduces computing costs. For example, MSMLP achieves an 81.49% accuracy with 1.5x fewer FLOPs than the MLP-Mixer-s. By adjusting the weights, we can also flexibly obtain all points on the MSMLP curve.

The practical speed of the Multi-Scale MLP. In this experiment, we exploited an NVIDIA 1070GPU to test the model's actual inference speed, and the batch size per model input is 16. In this trial, we use the MLP-Mixer-s and MLP-Mixer-b as the backbone model to test the actual inference speed of MSMLP. The corresponding results are reported in Table 2, highlighting that MSMLP increases the inference speed by x1.36–1.39 without reducing accuracy.

Table 3
Effects of feature reuse.

Feature reuse	1st exit (patch size=16)		2nd exit (patch size=8)		3rd exit (patch size=4)	
	Top-1 acc	GFLOPs	Top-1 acc	GFLOPs	Top-1 acc	GFLOPs
YES	37.30%	0.07	47.46%	0.24 (↑14.3%)	**55.81%**	0.79 (↑9.7%)
NO	39.46%	0.07	46.17	0.21	50.97%	0.72

Table 4
Comparisons of the embedded position of feature reuse.

Embedded position	Top-1 acc		GFLOPs
	CIFAR100	CIFAR10	
First	52.0%	81.44%	0.73
All	**55.81%**	**81.58%**	0.79 (↑8.2%)

Table 5
The effect of the LN in feature reuse mechanism.

Ablation	TOP −1 acc	
	CIFAR100	CIFAR10
Remove LN in $f_i(.)$	54.65%	80.7%
MSMLP	**55.81%**	**81.58%**

The above two experiments prove that our MSMLP model reduces the loss of computing resources in theory and practically improves the inference speed.

4.2. Ablation study

Effects of feature reuse. We verify the effectiveness of feature reuse by ablating the feature reuse mechanism. We first stop early termination, then use the validation set to calculate the accuracy of each classifier's early exit and the GFLOPs. The corresponding results are reported in Table 3. In the subsequent trial, we use three-exit MSMLP based on MLP-Mixer-s and conduct experiments on CIFAR100. The results indicate that after using the feature reuse mechanism, the accuracy of the second and third exits has significantly improved, and the calculation burden is only increased by 14.3%. However, we found that the accuracy of the first exit after using the feature reuse mechanism decreases. This is due to the change made by MSMLP to enhance the entire model's accuracy. Overall, the MSMLP's accuracy significantly improves after using the feature reuse mechanism.

The embedded position of feature reuse. We embed the output of the feature reuse process into the Mixer block, where each layer of the MLP-Mixer-s has eight Mixer blocks. Therefore, we speculate that the embedding position of the feature reuse mechanism will also affect the results. Here we compare two embedding methods: embedding only the first Mixer block and all Mixer blocks. The corresponding results are presented in Table 4, revealing that embedding the output of the feature reuse scheme into all Mixer blocks effectively improves the model's accuracy and increases the calculation burden by only 8.2%. Therefore, in our design, we embed the output of feature reuse into all Mixer blocks.

Design feature reuse mechanism. When designing the feature reuse mechanism, in addition to the necessary cropping and conversion operations applied to the image, we also add an LN based on experience [1,27]. In order to verify the effectiveness of the LN, we conduct experiments on MSMLP using the MLP-Mixer-s as the backbone. The experimental results presented in Table 5 highlight that after adding LN, the accuracy of MSMLP improves.

4.3. Visualization

In order to illustrate that our method can effectively separate "easy" and "hard" images, i.e., configure an appropriate number of patches for each picture, we present 24 pictures from four different categories from CIAFR10 (Fig. 5). These images are divided into "easy" and "hard" images, which correspond to the images that can be correctly identified by the first exit and the third exit of MSMLP, respectively. It can be observed that the "easy" image structure is relatively clear and contains the recognition object completely. On the contrary, the "hard" image is more complicated, and the background color is similar to the color of the recognized object or partially contains the recognized object.

Fig. 5. Visualizing the "easy" and "hard" samples in MSMLP. "Easy" images are exited from the first classifier of MSMLP and "Hard" are the images exited from the last classifier of MSMLP.

5. Conclusion

This paper proposes a novel resource-efficient image classification method called Multi-Scale MLP (MSMLP). Compared with the original MLP-mixer, the MSMLP dynamically allocates the appropriate computational resources to the input images, thus reducing the waste of computational resources. Specifically, MSMLP processes samples by using multiple MLP-Mixers of different scales, which are activated in decreasing order of scale size until the last classifier of the model is reached and the result is output. If the prediction value of the large-scale MLP-Mixer upstream of the model reaches the corresponding confidence level, it will be terminated early and the result will be output. We further add a feature reuse mechanism, enabling the downstream small-scale MLP-Mixer to reuse the features learned by the upstream large-scale MLP-Mixer to use the computational resources efficiently. Our final experimental results prove that our proposed MSMLP model consumes significantly less computational resources and is significantly faster in inference than the MLP-Mixer.

CRediT authorship contribution statement

Hong Zhang: Conceptualization, Methodology, Supervision, Funding acquisition, Writing – review & editing. **ZhiXiang Dong:**

Methodology, Formal analysis, Investigation, Writing – review & editing. **Bo Li:** Resources, Investigation, Writing – review & editing. **Siyuan He:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] I.O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, et al., Mlp-mixer: An all-mlp architecture for vision, Adv. Neural Inf. Process. Syst. 34 (2021).

[2] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[3] H. Touvron, P. Bojanowski, M. Caron, M. Cord, A. El-Nouby, E. Grave, G. Izacard, A. Joulin, G. Synnaeve, J. Verbeek, et al., Resmlp: Feedforward networks for image classification with data-efficient training, 2021, arXiv preprint arXiv:2105.03404.

[4] H. Liu, Z. Dai, D. So, Q.V. Le, Pay attention to mlps, Adv. Neural Inf. Process. Syst. 34 (2021) 9204–9215.

[5] X. Ding, C. Xia, X. Zhang, X. Chu, J. Han, G. Ding, Repmlp: Re-parameterizing convolutions into fully-connected layers for image recognition, 2021, arXiv preprint arXiv:2105.01883.

[6] Y. Hu, H. You, Z. Wang, Z. Wang, E. Zhou, Y. Gao, Graph-MLP: Node classification without message passing in graph, 2021, arXiv preprint arXiv:2106.04051.

[7] Y. Feng, H. You, Z. Zhang, R. Ji, Y. Gao, Hypergraph neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, no. 01, 2019, pp. 3558–3565.

[8] I. Chami, Z. Ying, C. Ré, J. Leskovec, Hyperbolic graph convolutional neural networks, Adv. Neural Inf. Process. Syst. 32 (2019).

[9] Q. Liu, M. Nickel, D. Kiela, Hyperbolic graph neural networks, Adv. Neural Inf. Process. Syst. 32 (2019).

[10] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, et al., Searching for mobilenetv3, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1314–1324.

[11] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017, arXiv preprint arXiv:1704.04861.

[12] N. Ma, X. Zhang, H.-T. Zheng, J. Sun, Shufflenet v2: Practical guidelines for efficient cnn architecture design, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 116–131.

[13] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.

[14] L. Yang, H. Jiang, R. Cai, Y. Wang, S. Song, G. Huang, Q. Tian, Condensenet v2: Sparse feature reactivation for deep networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 3569–3578.

[15] X. Zhang, X. Zhou, M. Lin, J. Sun, Shufflenet: An extremely efficient convolutional neural network for mobile devices, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 6848–6856.

[16] M. Tan, Q. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, in: International Conference on Machine Learning, PMLR, 2019, pp. 6105–6114.

[17] B. Graham, A. El-Nouby, H. Touvron, P. Stock, A. Joulin, H. Jégou, M. Douze, Levit: A vision transformer in Convnet's clothing for faster inference, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 12259–12269.

[18] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, B. Guo, Swin transformer: Hierarchical vision transformer using shifted windows, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 10012–10022.

[19] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z.-H. Jiang, F.E. Tay, J. Feng, S. Yan, Tokens-to-token VIT: Training vision transformers from scratch on imagenet, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 558–567.

[20] J. Miao, X. Wang, Y. Wu, W. Li, X. Zhang, Y. Wei, Y. Yang, Large-scale video panoptic segmentation in the wild: A benchmark, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 21033–21043.

[21] X. Wang, L. Zhu, H. Wang, Y. Yang, Interactive prototype learning for egocentric action recognition, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 8168–8177.

[22] Z. Yang, Y. Wei, Y. Yang, Collaborative video object segmentation by multi-scale foreground-background integration, IEEE Trans. Pattern Anal. Mach. Intell. (2021).

[23] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, Y. Wang, Dynamic neural networks: A survey, IEEE Trans. Pattern Anal. Mach. Intell. (2021).

[24] G. Huang, D. Chen, T. Li, F. Wu, L. Van Der Maaten, K.Q. Weinberger, Multi-scale dense networks for resource efficient image classification, 2017, arXiv preprint arXiv:1703.09844.

[25] L. Yang, Y. Han, X. Chen, S. Song, J. Dai, G. Huang, Resolution adaptive networks for efficient inference, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 2369–2378.

[26] G. Huang, Z. Liu, G. Pleiss, L. Van Der Maaten, K. Weinberger, Convolutional networks with dense connectivity, IEEE Trans. Pattern Anal. Mach. Intell. (2019).

[27] Y. Wang, R. Huang, S. Song, Z. Huang, G. Huang, Not all images are worth 16x16 words: Dynamic vision transformers with adaptive sequence length, 2021, arXiv E-Prints, arXiv–2105.

[28] H. Li, H. Zhang, X. Qi, R. Yang, G. Huang, Improved techniques for training adaptive deep networks, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1891–1900.

[29] A. Veit, S. Belongie, Convolutional networks with adaptive inference graphs, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 3–18.

[30] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, J.E. Gonzalez, Skipnet: Learning dynamic routing in convolutional networks, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 409–424.

[31] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L.S. Davis, K. Grauman, R. Feris, Blockdrop: Dynamic inference paths in residual networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8817–8826.

[32] J. Lin, Y. Rao, J. Lu, J. Zhou, Runtime neural pruning, Adv. Neural Inf. Process. Syst. 30 (2017).

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).

[34] P. Gao, T. Ma, H. Li, J. Dai, Y. Qiao, Convmae: Masked convolution meets masked autoencoders, 2022, arXiv preprint arXiv:2205.03892.

[35] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., An image is worth 16x16 words: Transformers for image recognition at scale, 2020, arXiv preprint arXiv:2010.11929.

[36] D. Hendrycks, K. Gimpel, Gaussian error linear units (gelus), 2016, arXiv preprint arXiv:1606.08415.

[37] J.L. Ba, J.R. Kiros, G.E. Hinton, Layer normalization, 2016, arXiv preprint arXiv:1607.06450.

[38] H.-C. Shin, H.R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, R.M. Summers, Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning, IEEE Trans. Med. Imaging 35 (5) (2016) 1285–1298.

[39] M. Lin, Q. Chen, S. Yan, Network in network, 2013, arXiv preprint arXiv:1312.4400.

[40] A. Krizhevsky, G. Hinton, et al., Learning Multiple Layers of Features from Tiny Images, Citeseer, 2009.

[41] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, Ieee, 2009, pp. 248–255.

[42] A. Adcock, V. Reis, M. Singh, Z. Yan, L. van der Maaten, Classy vision, 2019, https://github.com/facebookresearch/ClassyVision.

[43] H. Fan, Y. Li, W.-Y.L. Bo Xiong, C. Feichtenhofer, Pyslowfast, 2020, https://github.com/facebookresearch/slowfast.

[44] V. Pham, C. Pham, T. Dang, Road damage detection and classification with detectron2 and faster R-CNN, in: 2020 IEEE International Conference on Big Data, Big Data, IEEE, 2020, pp. 5592–5601.