Sparse-MLP: A Fully-MLP Architecture with Conditional Computation

Yuxuan Lou, Fuzhao Xue, Zangwei Zheng, Yang You

National University of Singapore, Singapore
{yuxuanlou, f.xue}@u.nus.edu, zhengzangw@gmail.com, youy@comp.nus.edu.sg

Abstract

Mixture-of-Experts (MoE) with sparse conditional computation has been proved an effective architecture for scaling attention-based models to more parameters with comparable computation cost. In this paper, we propose Sparse-MLP, scaling the recent MLP-Mixer model with sparse MoE layers, to achieve a more computation-efficient architecture. We replace a subset of dense MLP blocks in the MLP-Mixer model with Sparse blocks. In each Sparse block, we apply two stages of MoE layers: one with MLP experts mixing information within channels along image patch dimension, one with MLP experts mixing information within patches along the channel dimension. Besides, to reduce computational cost in routing and improve expert capacity, we design Re-represent layers in each Sparse block. These layers are to re-scale image representations by two simple but effective linear transformations. When pre-training on ImageNet-1k with MoCo v3 algorithm, our models can outperform dense MLP models by 2.5% on ImageNet Top-1 accuracy with fewer parameters and computational cost. On small-scale downstream image classification tasks, *i.e.,* Cifar10 and Cifar100, our Sparse-MLP can still achieve better performance than baselines.

1 Introduction

Fully MLP-based models recently achieved promising results in vision (Tolstikhin et al. 2021; Liu et al. 2021a; Lee-Thorp et al. 2021; Hou et al. 2021). However, scaling such models by simply extending model depth or hidden dimensions will lead to a rapid increase in trainable parameters and computational cost. We argue that such scaling strategy is not a necessity and there exists a potency to further improve MLP-based models with a sparse instead of dense scaling. Existing works have shown that Mixture-of-Experts (MoE) (Shazeer et al. 2017) can achieve better modeling capacity with comparable computation cost based on transformers (Lepikhin et al. 2020; Fedus, Zoph, and Shazeer 2021; Riquelme et al. 2021; Xue et al. 2021). Inspired by them, in this work, we propose Sparse-MLP to achieve a more computation-efficient architecture.

Sparse-MLP scales MLP-Mixer (Tolstikhin et al. 2021) by replacing a subset of Dense blocks (Mixer layers) with Sparse blocks. In each Sparse block, we apply MoE layers at two stages. (1) Channel-mixing MoE ($\mathrm{MoE_C}$)), mixing the information across the spatial locations of image representations. (2) Token-mixing MoE ($\mathrm{MoE_S}$), mixing the information within the representation patches. Besides, we design Re-represent layers at the beginning and the end of each Sparse block. The Re-present layer at the beginning of each Sparse block can re-scale the input representation shape, making the routing computation and experts computation at token mixing MoE more balanced. The Re-represent layer at the end of each Sparse block shapes the output of the block to the original dimension so that Sparse blocks and Dense blocks can be combined flexibly with high model capacity.

A significant contribution of our work is that we scale dense MLP models with conditional computation in two directions: both in patch dimension and channel dimension. It is also a major difference between our model and previous Transformer-MoE models (Fedus, Zoph, and Shazeer 2021; Riquelme et al. 2021; Xue et al. 2021). Previous models which apply MoE to transformer-based architecture only replace the FFN after the multi-head self-attention with sparse MoE. In our model, we have channel-mixing MoE layers function in a similar way: mixing information within the spatial location. Furthermore, we have token-mixing MoE layers function in another direction: mixing the information across the spatial locations of the representation. We prove with experiments that such a two-dimensional scaling design is effective and efficient in improving model capacity.

Finally, We apply our Sparse-MLP models to image classification tasks and obtain outstanding results. After pre-trained with the self-supervised algorithm(MoCo v3) (Chen, Xie, and He 2021) on ILSVRC2012 ImageNet-1k dataset (Russakovsky et al. 2015), our Sparse-B model reaches 77.9% ImageNet-1k top-1 validation accuracy, 2.0% higher than Mixer-B/16 model with comparable computational cost. Our Sparse-L model reaches 79.2 ImageNet-1k top-1 validation accuracy, outperforming Mixer-L/16 by 2.5% with 62.8% parameters and 85.5% pre-training cost.

The contributions of this work can be summarized as follows:

Sparse fully-MLP architecture. We extend the MLP-based model by sparse conditional computation. To our best knowledge, this is the first work focusing on expanding the MLP-like model with MoE to achieve a more computation-efficient architecture.

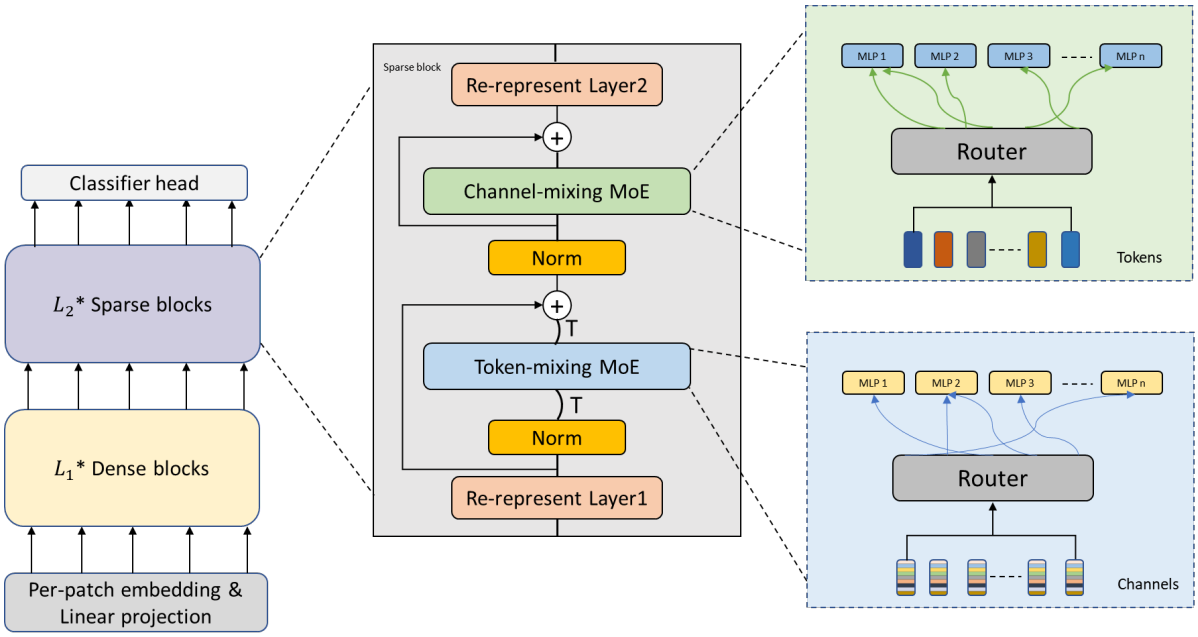

Figure 1: Sparse-MLP architecture overview

Applying MoE to new dimension. Novel model components in our Sparse-MLP architecture include the token-mixing MoE and the Re-represent layer. Token-mixing MoE is designed to focus on information across the spatial locations. With the Re-represent layer, the token mixing MoE demonstrate its capacity to improve models with high efficiency.

Performance on image classification tasks. We show that our Sparse-MLP model can strongly outperform the dense MLP-Mixer model on several image classification tasks. On all three downstream tasks, our Sparse-MLP models can reach better performance with comparable or less computational cost with same-level MLP-Mixer models.

2 Method

An overview of our Sparse-MLP is shown in Figure 1. In general, the Sparse-MLP model includes a per-patch linear embedding layer, a sequence of Dense blocks and Sparse blocks, and a classifier head.

Given an image of shape (H, W) , before going through Dense blocks and Sparse blocks, it is split into S non-overlapping patches. When patch resolution is (P, P), the number of patches $S = HW/P^2$. To obtain a higher-level representation, each patch is projected to a hidden representation $x \in \mathbb{R}^C$ by a linear embedding layer. The Dense block in Sparse-MLP follows the architecture of Mixer Layers in the MLP-Mixer model (Tolstikhin et al. 2021). Within each Dense block, there are one token-mixing MLP and one channel-mixing MLP. In Sparse block, both token-mixing MLP and channel-mixing MLP are replaced by Mixture-of-Experts (MoE) layers. Besides, we have re-represent layers added at both the beginning and the end of each Sparse block. We propose to use these re-represent layers to re-scale the representation shape and make the representation fitting the conditional computation better at the token mixing stage.

In summary, we scale dense MLP models with MoE layers in two directions. In the meantime, we keep our Sparse-MLP model entirely based on MLPs and feed-forward networks (FFN). So our model is a fully-MLP model with conditional computation.

2.1 Dense block

We first introduce the Dense block. Following Tolstikhin et al. (2021), each Dense block in our model has two stages of MLPs: token-mixing MLP and channel-mixing MLP. Each MLP consists of two fully connected layers and one GELU (Hendrycks and Gimpel 2016) non-linear activation.

$$\mathrm{MLP}(x) = W_2(\sigma_{gelu}(W_1 x + b_1)) + b_2 \qquad (1)$$

Token-mixing MLP ($\mathrm{MLP_S}$) aims at mixing the information of same channel across different spatial locations. For an input representation $x \in \mathbb{R}^{S\times C}$, it functions on every columns of x and maps:$\mathbb{R}^S \to \mathbb{R}^S$. In practice, we first transpose x to $x^T \in \mathbb{R}^{C\times S}$ and then apply token-mixing MLP and transpose x^T back to x.

Channel-mixing MLP ($\mathrm{MLP_C}$) is to mix the information within the same token along the channel dimension . For $x \in \mathbb{R}^{S\times C}$, channel mixing MLP functions on every rows of x and maps: $\mathbb{R}^C \to \mathbb{R}^C$.

Other components in a Dense block include: skip connection (He et al. 2015), Layer Normalization on channels (Ba, Kiros, and Hinton 2016). We set dropout (Srivastava et al. 2014) in MLPs=0. Given an input $x \in \mathbb{R}^{C\times S}$, we then formulate the Dense block as:

$$y_1 = x + t(\mathrm{MLP_S}(t(\mathrm{norm}(x)))) \quad (2)$$

$$y = y_1 + \mathrm{MLP_C}(\mathrm{norm}(y1)) \quad (3)$$

where norm is layer normalization, $t()$ is a transpose and y is the output of the Dense block.

2.2 Mixture-of-Experts

In this section, we formulate Mixture-of-Experts (MoE) architecture and its key components.

Conditional Computing The Mixture-of-Experts layer (MoE) is composed of a set of experts. Only a subset of them are active and engaged in the computation on a per-example basis. In our model, experts are MLPs same as the MLPs in the Dense block.

Following Shazeer et al. (2017), given $x \in \mathbb{R}^D$, the output of one MoE layer with N Experts is:

$$\mathrm{MoE(x)} = \sum_{i=1}^{N} G(x)_i E_i(x) \quad (4)$$

where $G(x) : \mathbb{R}^D \to \mathbb{R}^N$ is the gating network which compute input-conditioned routing weights for experts. $E_i(x) : \mathbb{R}^D \to \mathbb{R}^D$ is the i^{th} expert layer. In practice, we have a sparse $G(x)$, which means each input x is restricted to be assigned to k experts ($k \ll N$). If the input x is not assigned to E_i, $G(x)_i = 0$ and E_i would not be computed. This enables us to scale to outrageously large model with a comparable computation cost.

Gating Network As we introduced above, to assign token representations x to different experts, each MoE layer has a sparse gating network. We formulate it as:

$$G(x) = \mathrm{TopK}(\mathrm{softmax}(W_g(x) + \epsilon)) \quad (5)$$

where $W_g \in \mathbb{R}^{D*N}$ is a trainable matrix and $\epsilon \sim \mathcal{N}(0, \frac{1}{N^2})$ is a normal noise to explore better assignment from experts. After computing the probability of the input x routed to each Expert, we only keep the top K of them for further forward propagation. In practice, we usually select K as 1 or 2.

Load Balance Loss To encourage a balanced assignment of inputs across experts, an auxiliary loss is added to the model for every MoE layer (Shazeer et al. 2017; Lepikhin et al. 2020; Fedus, Zoph, and Shazeer 2021; Riquelme et al. 2021). Our auxiliary consists of two parts: Importance loss and Load loss.

The importance of i^{th} expert is defined as the normalized gating network weights correspond to i^{th} expert summed over the input batch X.

$$\mathrm{Imp}_i(X) = \sum_{x \in X} \mathrm{softmax}(W_g x)_i \quad (6)$$

where W_g is the gating weight matrix of the MoE layer, and the importance loss of the MoE layer over a batch of inputs X is:

$$L_{imp}(X) = (\frac{\mathrm{std(Imp}(X))}{\mathrm{mean(Imp}(X))})^2 \quad (7)$$

In addition to the importance loss for more balanced routing weights, we also have a load loss seeking balanced routing results. The load of an Expert i given a batch of inputs X is defined as the possibility of routing to Expert i summed over the batch.

$$\mathrm{Load}_i(X) = \sum_{x \in X} p_i(x) \quad (8)$$

$$p_i(x) \triangleq P(G(x)_i \geq threshold_k(G(x))) \quad (9)$$

The load loss of one MoE layer over the batch is:

$$L_{Load}(X) = (\frac{\mathrm{std(Load}(X))}{\mathrm{mean(Load}(X))})^2 \quad (10)$$

And the total auxiliary loss of one MoE layer takes the form:

$$L_{aux} = \lambda(\frac{1}{2}L_{imp} + \frac{1}{2}L_{load}) \quad (11)$$

where λ is a hyper-parameter that controls that the auxiliary loss not only encourages balanced routing across experts but also not overwhelms the original model loss. In practice, we set $\lambda = 1e-2$. According to existing MoE-based models (Riquelme et al. 2021; Xue et al. 2021), the performance is insensitive to λ.

2.3 Sparse block

In each Sparse block, two stages of MLP are replaced by token-mixing MoE ($\mathrm{MoE_S}$) and channel-mixing MoE ($\mathrm{MoE_C}$). Token-mixing MoE functions on the columns of input $X \in \mathbb{R}^{S \times C}$ and maps: $\mathbb{R}^S \to \mathbb{R}^S$. Channel-mixing MoE functions on the rows of input X and maps: $\mathbb{R}^C \to \mathbb{R}^C$.

A significant difference between Sparse-MLP and previous models which apply MoE to transformer-based architecture (Lepikhin et al. 2020; Fedus, Zoph, and Shazeer 2021; Riquelme et al. 2021; Xue et al. 2021) is our token-mixing MoE. When MoE is applied to transformer-based models, each expert of one MoE layer only functions to interact the information between different channels within the same token. Different tokens are routed to different experts. In each Sparse block, we not only have one MoE layer functioning sparsely on channels ($\mathrm{MoE_C}$) but also have one MoE layer mixing the information among tokens ($\mathrm{MoE_S}$). $\mathrm{MoE_S}$ enables splitting and grouping different columns of the input $x \in \mathbb{R}^{S \times C}$ and functioning different MLPs corresponding to groups. Instead of applying the same MLP to all columns, conditional computing extends model capacity to learn more information between spatial positions of the input and thus improve model capacity.

2.4 Re-represent Layers in Sparse block

In the original MLP-Mixer model design (Tolstikhin et al. 2021), the channels of the image representation are much more than the patches. That is, for representation $x \in \mathbb{R}^{S \times C}, C \geq 3S$. However, for each input x, the $\mathrm{MOE_S}$ in Sparse block routes C channels and the Experts dimension is S. The original representation shape leads to unbalanced computational cost between routing and Experts forwarding.

Algorithm 1: Pseudo-code for the Re-represent layers

function RE-REPRESENT$_1(x, S_1, C_1)$
    x = norm(x, axis = 'channel')
    x = Transpose(x, 1, 2)
    x = proj(x, S_1)
    x = gelu(x)
    x = Transpose(x, 1, 2)
    x = norm(x, axis = 'channel')
    x = proj(x, C_1)
    x = gelu(x)
    return x
end function
function RE-REPRESENT$_2(x, S, C)$
    x = norm(x, axis = 'channel')
    x = proj(x, C)
    x = gelu(x)
    x = norm(x, axis = 'channel')
    x = Transpose(x, 1, 2)
    x = proj(x, S)
    x = gelu(x)
    x = Transpose(x, 1, 2)
    return x
end function

In order to fix it, we added two stages of re-represent layers at the beginning and the end of each Sparse block. Re-represent layer$_1$ reduce the channels and increase patches so that the inputs can better fitting $\mathrm{MoE_S}$. Re-represent layer$_2$ functions the opposite way so that Sparse blocks and Dense blocks can be combined flexibly. Given an input $x \in \mathbb{R}^{S\times C}$, re-represent layer1 maps:$\mathbb{R}^{C\times S} \rightarrow \mathbb{R}^{C_1\times S_1}$ re-represent layer2 maps: $\mathbb{R}^{S_1\times C_1} \rightarrow \mathbb{R}^{S\times C}$. Each re-represent layer is composed of two FFN layers to transform two dimensions S and C, respectively. Such implementation can reduce routing computation and improve expert dimension, which leads to a more balanced and effective computation. In practice, we set $S_1 = 2S, C_1 = C/2$.

2.5 Sparse-MLP with MoCo v3

We find that scaling MLP models in parameters and training them from scratch with limited training data will lead to an overfitting problem. Such finding is consistent with previous work on MLP models (Tolstikhin et al. 2021) and attention-based models (Chen, Xie, and He 2021; Dosovitskiy et al. 2020; Xue et al. 2021). In order to better obtain model capacity, we adopt MoCo v3 algorithm as our default self-supervised training algorithm (He et al. 2019; Chen et al. 2020b; Chen, Xie, and He 2021), and fine-tune models on downstream image classification tasks.

We set up our MoCo v3 algorithm the same as the standard framework in (Chen, Xie, and He 2021). Two crops of each image under random data augmentation x_1, x_2 are encoded by two encoders f_q, f_k, with output q_1, q_2, k_1, k_2. Then a contrastive loss $L(q_1, q_2, k_1, k_2)$ is usded to update f_q by back propagation. f_k is updated by momentum update. In our work, the backbone in f_q and f_k are ViT models (Dosovitskiy et al. 2020), MLP-Mixer models (Tolstikhin et al. 2021) and our Sparse-MLP models.

3 Experiments

We pretrain our Sparse-MLP models with MoCo V3 on the ILSVRC-2012 Imagenet dataset (Russakovsky et al. 2015) and evaluate our model's performance on several downstream image classification tasks. We select MLP-Mixer models(Tolstikhin et al. 2021) and ViT models (Dosovitskiy et al. 2020) as our baselines and compare our models with baseline models in two quantities: (1) classification accuracy on downstream tasks, (2) computational cost of pre-training on the upstream dataset, and fine-tuning on downstream datasets. We do not aim to reach SOTA image classification accuracy but to show that our fully-MLP model with conditional computing can outperform dense MLP models or attention-based models either in accuracy or computational cost.

3.1 Experiment Settings

Pre-training details We pretrain Sparse-MLP models and baseline models with a self-supervised learning algorithm(MoCo v3) (Chen, Xie, and He 2021) on ILSVRC-2012 ImageNet dataset. (Russakovsky et al. 2015)(1.3M training samples, 1k image classes). Following the practice in (Tolstikhin et al. 2021; Chen, Xie, and He 2021), our data augmentation policy for pretraining includes random resized crop, horizontal flipping, RandAugment (Cubuk et al. 2019), color jittering, grayscale conversion (Wu et al. 2018), blurring (Chen et al. 2020a), and solarization (Grill et al. 2020). We also apply stochastic depth (Huang et al. 2016). We pretrain all models on TPU v3 clusters. We select a batch size as 4096 at the pre-training stage, LAMB optimizer (You et al. 2019) with weight decay. We pretrain all models for 300 epochs using a cosine learning rate decay with a 10k steps warm up (Loshchilov and Hutter 2016). The image resolution for pretraining is 224.

Hyper-parameter	Value
Image resolution	224
Epochs	300
Batch size	4096
Warmup steps	10k
Optimizer	LAMB
Peak learning rate	1e-3
Learning rate decay	cosine
Weight decay rate	1e-1
Global clip norm	1
MoCo t	1
MoCo m	0.99
MoCo dim	4096

Table 1: Hyper-parameters for pre-training on ImageNet-1k

Fine-tuning details We fine-tune our model on three downstream tasks: ILSVRC-2012 Imagenet dataset (1.3M training samples, 50k validation samples, 1k classes) (Russakovsky et al. 2015) (Russakovsky et al. 2015); CIFAR-10

dataset (50k training samples, 10k validation samples, 10 classes) (Krizhevsky, Nair, and Hinton); CIFAR-100 dataset (50k training samples, 10k validation samples, 100 classes). We fine-tune our models on all downstream tasks at image resolution 224. We follow the standard fine-tune settings in Chen, Xie, and He (2021). After pretraining with MoCo v3, we remove the MLP heads of the pretrained model, add a classifier head to the encoder, and train on downstream tasks. The augmentation strategies during fine-tuning stage include random resized crop, horizontal flipping, RandAugment (Cubuk et al. 2019) and Mixup (Zhang et al. 2017). We select Adam without weight decay as the optimizer. We set our learning rate as $lr * \mathrm{BatchSize}/256$, using linear weight decay with 10k warm-up steps (Loshchilov and Hutter 2016).

3.2 Model settings

We report our main results based on three models: Sparse-S, Sparse-B, Sparse-L. In Table 2, we give the specifications of these models. Each model is composed of L_1 Dense blocks and L_2 Sparse blocks. And in all three models reported in the main results, Dense blocks are in the front and followed by Sparse blocks. D_S refers to the hidden dimension of token mixing MLPs, and D_C refers to the hidden dimension of channel mixing MLPs. $D_{S'}$ is the MLP dimension of token-mixing MoE layers, and $D_{C'}$ denotes the MLP dimension of channel-mixing MoE layers. For all MLPs in Dense blocks and Sparse blocks, we set dropout as 0. For token mixing MoEs, we select top K routing as 1. And for channel mixing MoEs, we set K as 2.

Specification	Sparse-S	Sparse-B	Sparse-L
	Dense block		
blocks L_1	6	10	8
Patches S	196	196	196
Hidden size C	512	768	768
$\mathrm{MLP_S}$ dim D_S	256	384	384
$\mathrm{MLP_C}$ dim D_C	2048	3072	3072
	Sparse block		
blocks L_2	2	2	6
New patches S'	392	392	392
New hidden size C'	512	384	384
Experts in $\mathrm{MoE_S}$	4	8	16
Experts in $\mathrm{MoE_C}$	0	4	4
$\mathrm{MoE_S}$ top K	1	1	1
$\mathrm{MoE_C}$ top K	-	2	2
$\mathrm{MoE_S}$ dim $D_{S'}$	512	768	768
$\mathrm{MoE_C}$ dim $D_{C'}$	2048	1536	1536
Positions	last 2	last 2	last 6
Parameters(M)	22	69	130

Table 2: Specifications of Sparse-MLP models

3.3 Main Results

We build our Sparse-MLP models on three parameter levels in comparison with attention-based models (*e.g.,* ViT (Dosovitskiy et al. 2020)) and dense MLP models (*e.g.,* MLP-Mixer (Tolstikhin et al. 2021)). In Table 3, we report ImageNet-1k top-1 validation accuracy and corresponding pre-training cost of each model.

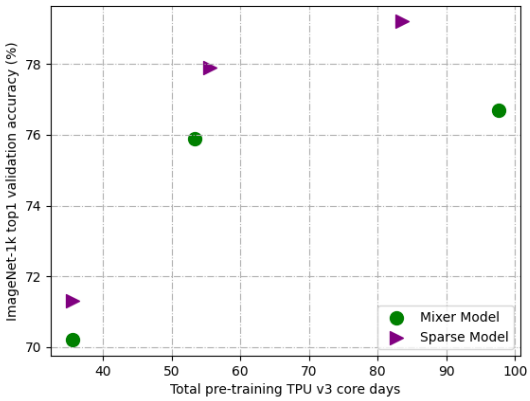

Figure 2: Comparison between Mixer models and Sparse-MLP models. With comparable or less computational cost, Sparse-MLP achieves better performance

Our Sparse-S model surpasses Mixer-S/16 on ImageNet-1k top-1 accuracy by 1.1% with comparable parameters and pre-training cost. Sparse-B model scales Mixer-B/16 with 17% (59M→69M) with comparable pre-training TPU v3 core days and outperforms Mixer-B/16 by 2.6% (75.9% → 77.9%). Our Sparse-L outperforms Mixer-L/16 by 3.3% with only 62.8% parameters and 85.5% pre-training time. Compared with ViT, our models show better performance with much fewer parameters and much less pre-training cost.

Also, we report and compare the results of Sparse-MLP models and dense MLP models (Tolstikhin et al. 2021) on two other downstream image classification tasks: Cifar10 (Krizhevsky, Nair, and Hinton), Cifar100 (Krizhevsky 2009). All models are pretrained with MoCo v3 on ImageNet-1k and then fine-tuned at downstream tasks end-to-end.

In Table 4, we can see that our Sparse-MLP models also outperform MLP-Mixer models on Cifar10 and Cifar100 image classification tasks. Also, when we scale our model to over 100M parameters, the performance of Mixer-L/16 and Sparse-L drop due to overfitting. This issue is prominent when training large MLP models on small datasets. And in such cases, our Sparse-L model still achieves higher accuracy than Mixer-L/16.

3.4 Ablation Study

In this section, we further investigate how each component of our Sparse-MLP model contributes to model capacity. All models in the ablation study are pretrained with MoCo v3 algorithm on ImageNet-1k and fine-tuned on the same dataset. We select ImageNet-1k top-1 validation accuracy and total pre-training TPU v3 core days as evaluation metrics. The ablation study is designed to answer the following questions:

Models	ImageNet Top-1(%)	Params(M)	Pre-training cost	Throughput
	attention-based			
ViT-B/16	76.7	86	67.2	861
ViT-L/16	77.6	304	195.2	268
	dense MLP-like			
Mixer-S/16	70.2	19	35.5	3986
Mixer-B/16	75.9	59	53.3	1320
Mixer-L/16	76.7	207	97.7	412
	Sparse-MLP			
Sparse-S	71.3	21	35.5	3986
Sparse-B	77.9	69	55.5	1265
Sparse-L	79.2	130	83.5	482

Table 3: ImageNet-1k results. All models are pretrained with self-supervised algorithm(MoCo v3) on ImageNet-1k and then fine-tuned. Pretrain cost is evaluated by total TPU v3 core-days used for pretraining. Throughput is evaluated by image/sec/core

Models	ImageNet top-1	Cifar10 top-1	Cifar100 top-1
Mixer-S/16	70.2	91.7	84.4
Sparse-S	71.3	91.9	84.4
Mixer-B/16	75.9	95.6	86.7
Sparse-B	77.9	**96.2**	87.2
Mixer-L/16	76.7	94.7	86.3
Sparse-L	**79.2**	95.4	**87.4**

Table 4: Results on downstream image classification tasks

- **Number of experts**: What is the impact of the number of experts in two stages MoE layers?
- **Top K routing**: Which K value(1 or 2) shall we select for $\mathrm{MoE_S}$ and $\mathrm{MoE_C}$?
- **Positions of Sparse blocks**: How shall we combine Dense blocks and Sparse blocks?
- **Re-represent layers analysis**: How do Re-present layers influence model capacity and computational cost?

Number of experts We first study the influence of the number of experts in $\mathrm{MoE_S}$ on model capacity. Different models are built based on Sparse-B. We fix all other hyper-parameters and tune the number of experts in $\mathrm{MoE_S}$ at three levels: 4, 8, 16, pretrain these models and evaluate their performance on ImageNet-1k validation top-1 accuracy.

From Figure 3, we can see that when the number of experts in $\mathrm{MoE_S}$ increases from 4 to 8, the model's performance increases a lot. However, when we scale experts to 16, the model capacity barely changes.

Similarly, for $\mathrm{MoE_C}$, we fix all other components in Sparse-B and tune the number of experts in $\mathrm{MoE_C}$ at three levels: 4,8, 16.

In Figure 3, we observe that there would be an overfitting problem when we increase the number of experts in $\mathrm{MoE_C}$. Such finding is similar to results in (Xue et al. 2021). When training data is limited, scaling the MoE layers, which mixing the information within spatial locations, will make the model easily overfit the training data.

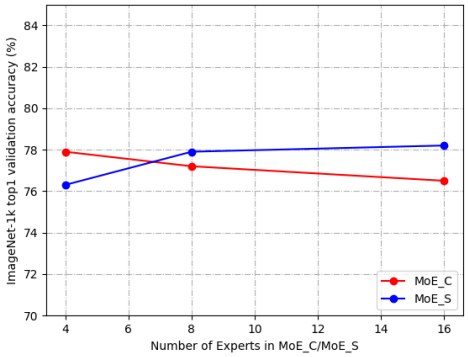

Figure 3: Influence of number of experts in $\mathrm{MoE_C}$/ $\mathrm{MoE_S}$

K	ImageNet Top-1(%)	TPU v3 core days
	$\mathrm{MoE_S}$	
1	**77.9**	**55.5**
2	77.9	57.7
	$\mathrm{MoE_C}$	
1	77.0	53.3
2	**77.9**	**55.5**

Table 5: Influence of K selecting.

The role of Top K routing Following the design in (Fedus, Zoph, and Shazeer 2021), we select K=1 or 2 for $\mathrm{MoE_S}$ and $\mathrm{MoE_C}$. We set Sparse-B as our default model(K=1 for $\mathrm{MoE_S}$, K=2 for $\mathrm{MoE_C}$). Then we report the results with K=2 for $\mathrm{MoE_S}$ and K=1 for $\mathrm{MoE_C}$ separately.

As shown in Table 5, for $\mathrm{MoE_S}$, top-1 routing and top-2 routing reach the same validation accuracy and top-1 routing cost less pre-training time. For $\mathrm{MoE_C}$, top-2 routing would

lead to prominent better performance with 4% more pre-training time.

The positions of Sparse blocks We experiment with two different placing orders of Dense blocks and Sparse blocks. (1) Dense blocks in front and Sparse blocks behind; (2) Sparse blocks as first few blocks and followed by Dense blocks. Also, we experiment with a different number of Sparse blocks while keeping the number of total Dense blocks and Sparse blocks the same. We set Sparse-B as our default model and change the orders and numbers of blocks based on Sparse-B

Positions	ImageNet Top-1(%)	Parameters(M)
N/A (Mixer-B/16)	75.9	59
Last two(Sparse-B)	77.9	69
First two	75.5	69
Last four	78.3	79

Table 6: Different combinations of Dense blocks and Sparse blocks. 'Positions' refers to the locations of Sparse blocks.

In Table 6, we can find that placing Sparse blocks in the first place and Dense blocks behind do not improve model capacity. Its ImageNet-1k validation accuracy is even lower than the original Mixer-B/16 model.

We also find that increasing the number of Sparse blocks, in the end, is an effective way to improve the model's performance. When we increase 2 Sparse blocks and keep the total number of blocks unchanged, the model's ImageNet-1k top-1 validation accuracy increased by 0.3%

The role of re-represent layers Another intuitive way to build Sparse blocks is to only replace $\mathrm{MLP_S}$ and $\mathrm{MLP_C}$ with $\mathrm{MoE_S}$ and $\mathrm{MoE_C}$ without any other changes. As stated in Section 2.4, such design with original image representations would lead to unbalanced cost between routing and experts computing in $\mathrm{MoE_S}$. Here we verify the necessity of re-represent layers by experiments. We set Sparse-B as our default model and experiment models with or without re-represent layers.

Models	ImageNet Top-1(%)	Pre-training cost
w/ r_layers	**77.9**	**55.5**
w/o r_layers	76.9	79.9

Table 7: Comparison between models with or without re-represent layers.

We can see from table 7 that re-represent layers not only reduce much pre-training cost but also improve the performance significantly.

4 Related Work

4.1 Transformer-MoE models

Mixture-of-Experts(MoE) (Jacobs et al. 1991; Jordan and Jacobs 1994; Chen, Xu, and Chi 1999; Yuksel, Wilson, and Gader 2012; Shazeer et al. 2017) has been successfully applied to many domains and tasks (Gavrila and Munder 2006; Hu, Palreddy, and Tompkins 1997; Tani and Nolfi 1999; Sminchisescu et al. 2004; Zeevi, Meir, and Adler 1997). Recently, it has been applied to transformer-based architecture to build huge models (Lepikhin et al. 2020; Fedus, Zoph, and Shazeer 2021; Riquelme et al. 2021; Xue et al. 2021) and these inspire our work. In NLP tasks, Lepikhin et al. (2020) designs an efficient distributed system to train a huge MoE model for machine translation. After that, Fedus, Zoph, and Shazeer (2021) proposes to use top-1 routing to scale transformer-based models to trillions of parameters and reach amazing model capacity. In vision tasks, (Riquelme et al. 2021) improves ViT (Dosovitskiy et al. 2020) by scaling a subset of transformer blocks with MoE. (Xue et al. 2021) applies MoE to transformer blocks with parameters sharing to improve ViT with fewer parameters. In these works, MoE layers are to replace the FFN in transformer blocks. Our model design makes a difference in that we apply MoEs in two directions and experiments demonstrate that the novel token-mixing MoE can improve model capacity effectively and efficiently.

4.2 MLP-based models

Our work is also related to MLP-base models (Tolstikhin et al. 2021; Liu et al. 2021a; Lee-Thorp et al. 2021; Hou et al. 2021) for vison. Different from CNN models (Krizhevsky, Sutskever, and Hinton 2012; Simonyan and Zisserman 2015; He et al. 2015; Tan and Le 2019) and attention-based models (Dosovitskiy et al. 2020; Touvron et al. 2020; Liu et al. 2021b), all trainable parameters in the backbones of MLP-based models are MLP-like. In MLP-Mixer (Tolstikhin et al. 2021), a token mixing MLP is to replace the multi-head self-attention (Vaswani et al. 2017) in transformer block (Tolstikhin et al. 2021). Some other MLP-like architectures (Liu et al. 2021a; Hou et al. 2021) function a similar way, mixing the information across spatial locations with MLPs or FFNs. In (Lee-Thorp et al. 2021), such work is accomplished by a Fourier transform (Bracewell and Bracewell 1986). All these MLP-lie models do have competitive results in vision tasks. Sparse-MLP scales MLP-Mixer with MoE layers and improves model capacity in a computation-efficient way.

5 Conclusions

In this work, we propose Sparse-MLP, a variant of the recent MLP-Mixer model with conditional computation. Experiments demonstrate that our two-stage MoE design and Re-represent layer design are effective in improving model capacity and reduce computational cost. Besides, we perform a comprehensive ablation study to investigate how each component contributes to the performance.

Extensions of our work could include the following topics. First, it is possible to further improve Sparse-MLP model capacity with huge pre-training datasets. Besides, we can explore the flexibility of Sparse-MLP architecture by designing different Sparse blocks in the same model. It would also be worthwhile to apply Sparse-MLP architecture to NLP or other domain tasks.

References

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer Normalization. arXiv:1607.06450.

Bracewell, R. N.; and Bracewell, R. N. 1986. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York.

Chen, K.; Xu, L.; and Chi, H. 1999. Improved Learning Algorithms for Mixture of Experts in Multiclass Classification. *Neural Netw.*, 12(9): 1229–1252.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. E. 2020a. A Simple Framework for Contrastive Learning of Visual Representations. *CoRR*, abs/2002.05709.

Chen, X.; Fan, H.; Girshick, R. B.; and He, K. 2020b. Improved Baselines with Momentum Contrastive Learning. *CoRR*, abs/2003.04297.

Chen, X.; Xie, S.; and He, K. 2021. An Empirical Study of Training Self-Supervised Vision Transformers. *CoRR*, abs/2104.02057.

Cubuk, E. D.; Zoph, B.; Shlens, J.; and Le, Q. V. 2019. RandAugment: Practical data augmentation with no separate search. *CoRR*, abs/1909.13719.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *CoRR*, abs/2010.11929.

Fedus, W.; Zoph, B.; and Shazeer, N. 2021. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *CoRR*, abs/2101.03961.

Gavrila, D.; and Munder, S. 2006. Multi-cue Pedestrian Detection and Tracking from a Moving Vehicle. *International Journal of Computer Vision*, 73: 41–59.

Grill, J.; Strub, F.; Altché, F.; Tallec, C.; Richemond, P. H.; Buchatskaya, E.; Doersch, C.; Pires, B. Á.; Guo, Z. D.; Azar, M. G.; Piot, B.; Kavukcuoglu, K.; Munos, R.; and Valko, M. 2020. Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning. *CoRR*, abs/2006.07733.

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. B. 2019. Momentum Contrast for Unsupervised Visual Representation Learning. *CoRR*, abs/1911.05722.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition. *CoRR*, abs/1512.03385.

Hendrycks, D.; and Gimpel, K. 2016. Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units. *CoRR*, abs/1606.08415.

Hou, Q.; Jiang, Z.; Yuan, L.; Cheng, M.; Yan, S.; and Feng, J. 2021. Vision Permutator: A Permutable MLP-Like Architecture for Visual Recognition. *CoRR*, abs/2106.12368.

Hu, Y. H.; Palreddy, S.; and Tompkins, W. 1997. A patient-adaptable ECG beat classifier using a mixture of experts approach. *IEEE Transactions on Biomedical Engineering*, 44(9): 891–900.

Huang, G.; Sun, Y.; Liu, Z.; Sedra, D.; and Weinberger, K. Q. 2016. Deep Networks with Stochastic Depth. *CoRR*, abs/1603.09382.

Jacobs, R. A.; Jordan, M. I.; Nowlan, S. J.; and Hinton, G. E. 1991. Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1): 79–87.

Jordan, M. I.; and Jacobs, R. A. 1994. Hierarchical Mixtures of Experts and the EM Algorithm. *Neural Comput.*, 6(2): 181–214.

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images. Technical report.

Krizhevsky, A.; Nair, V.; and Hinton, G. ???? CIFAR-10 (Canadian Institute for Advanced Research).

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F.; Burges, C. J. C.; Bottou, L.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Lee-Thorp, J.; Ainslie, J.; Eckstein, I.; and Ontañón, S. 2021. FNet: Mixing Tokens with Fourier Transforms. *CoRR*, abs/2105.03824.

Lepikhin, D.; Lee, H.; Xu, Y.; Chen, D.; Firat, O.; Huang, Y.; Krikun, M.; Shazeer, N.; and Chen, Z. 2020. GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding. *CoRR*, abs/2006.16668.

Liu, H.; Dai, Z.; So, D. R.; and Le, Q. V. 2021a. Pay Attention to MLPs. *CoRR*, abs/2105.08050.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021b. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *CoRR*, abs/2103.14030.

Loshchilov, I.; and Hutter, F. 2016. SGDR: Stochastic Gradient Descent with Restarts. *CoRR*, abs/1608.03983.

Riquelme, C.; Puigcerver, J.; Mustafa, B.; Neumann, M.; Jenatton, R.; Pinto, A. S.; Keysers, D.; and Houlsby, N. 2021. Scaling Vision with Sparse Mixture of Experts. *CoRR*, abs/2106.05974.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3): 211–252.

Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q. V.; Hinton, G. E.; and Dean, J. 2017. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. *CoRR*, abs/1701.06538.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.

Sminchisescu, C.; Kanaujia, A.; Li, Z.; and Metaxas, D. N. 2004. Learning to Reconstruct 3 D Human Motion from Bayesian Mixtures of Experts . A Probabilistic Discriminative Approach.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1): 1929–1958.

Tan, M.; and Le, Q. V. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *CoRR*, abs/1905.11946.

Tani, J.; and Nolfi, S. 1999. Learning to perceive the world as articulated: an approach for hierarchical learning in sensory-motor systems. *Neural networks : the official journal of the International Neural Network Society*, 12 7-8: 1131–1141.

Tolstikhin, I. O.; Houlsby, N.; Kolesnikov, A.; Beyer, L.; Zhai, X.; Unterthiner, T.; Yung, J.; Steiner, A.; Keysers, D.; Uszkoreit, J.; Lucic, M.; and Dosovitskiy, A. 2021. MLP-Mixer: An all-MLP Architecture for Vision. *CoRR*, abs/2105.01601.

Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; and Jégou, H. 2020. Training data-efficient image transformers & distillation through attention. *CoRR*, abs/2012.12877.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention Is All You Need. *CoRR*, abs/1706.03762.

Wu, Z.; Xiong, Y.; Yu, S. X.; and Lin, D. 2018. Unsupervised Feature Learning via Non-Parametric Instance-level Discrimination. *CoRR*, abs/1805.01978.

Xue, F.; Shi, Z.; Wei, F.; Lou, Y.; Liu, Y.; and You, Y. 2021. Go Wider Instead of Deeper. *CoRR*, abs/2107.11817.

You, Y.; Li, J.; Hseu, J.; Song, X.; Demmel, J.; and Hsieh, C.-J. 2019. Reducing BERT pre-training time from 3 days to 76 minutes. *arXiv preprint arXiv:1904.00962*.

Yuksel, S. E.; Wilson, J. N.; and Gader, P. D. 2012. Twenty Years of Mixture of Experts. *IEEE Transactions on Neural Networks and Learning Systems*, 23(8): 1177–1193.

Zeevi, A.; Meir, R.; and Adler, R. 1997. Time Series Prediction using Mixtures of Experts. In Mozer, M. C.; Jordan, M.; and Petsche, T., eds., *Advances in Neural Information Processing Systems*, volume 9. MIT Press.

Zhang, H.; Cissé, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2017. mixup: Beyond Empirical Risk Minimization. *CoRR*, abs/1710.09412.