Experts Weights Averaging: A New General Training Scheme for Vision Transformers

Yongqi Huang[1†], Peng Ye[1†], Xiaoshui Huang[2], Sheng Li[3], Tao Chen[1*], Wanli Ouyang[2]
[1]School of Information Science and Technology, Fudan University, [2]Shanghai AI Lab,
[3]Jiangxi University of Finance and Economics
{19307130163, yepeng20}@fudan.edu.cn

Abstract

Structural re-parameterization is a general training scheme for Convolutional Neural Networks (CNNs), which achieves performance improvement without increasing inference cost. As Vision Transformers (ViTs) are gradually surpassing CNNs in various visual tasks, one may question: *if a training scheme specifically for ViTs exists that can also achieve performance improvement without increasing inference cost?* Recently, Mixture-of-Experts (MoE) has attracted increasing attention, as it can efficiently scale up the capacity of Transformers at a fixed cost through sparsely activated experts. Considering that MoE can also be viewed as a multi-branch structure, *can we utilize MoE to implement a ViT training scheme similar to structural re-parameterization?* In this paper, we affirmatively answer these questions, with a new general training strategy for ViTs. Specifically, we decouple the training and inference phases of ViTs.During training, we replace some Feed-Forward Networks (FFNs) of the ViT with specially designed, more efficient MoEs that assign tokens to experts by random uniform partition, and perform Experts Weights Averaging (EWA) on these MoEs at the end of each iteration. After training, we convert each MoE into an FFN by averaging the experts, transforming the model back into original ViT for inference. We further provide a theoretical analysis to show why and how it works. Comprehensive experiments across various 2D and 3D visual tasks, ViT architectures, and datasets validate the effectiveness and generalizability of the proposed training scheme. Besides, our training scheme can also be applied to improve performance when fine-tuning ViTs. Lastly, but equally important, the proposed EWA technique can significantly improve the effectiveness of naive MoE in various 2D visual small datasets and 3D visual tasks.

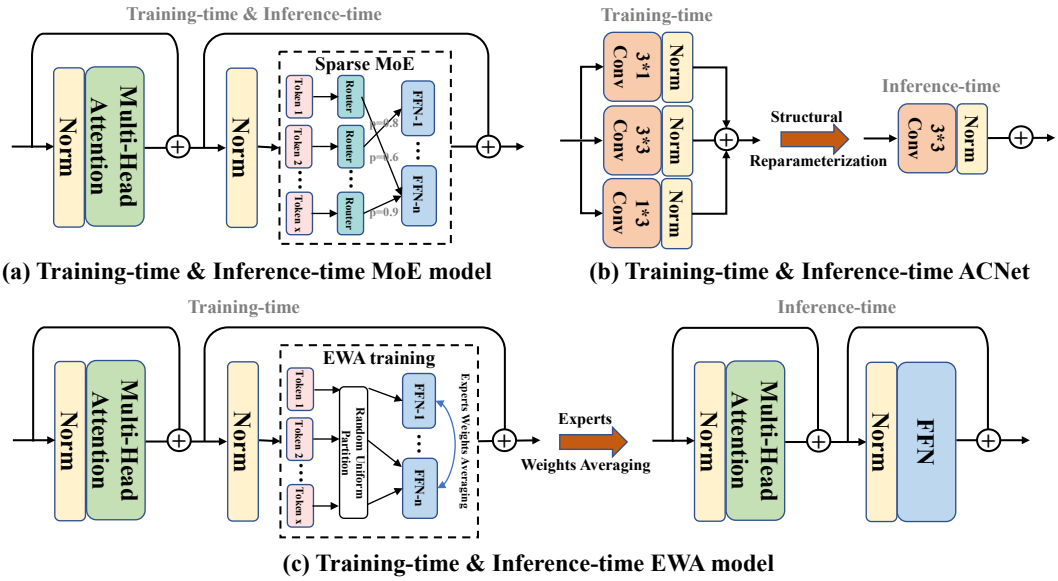
1 Introduction

In recent years, the development of general training schemes has played a critical role in deep learning. These methods enhance existing models without modifying their architectures, and do not require any assumptions about specific architectures. As a result, they can be employed for various models and provide uniform benefits.

For CNN architecture, the widely-used general training scheme is structural re-parameterization [6; 9; 10; 8] (Fig. 1(b)). This approach decouples the training and inference stages of the CNN model by using a multi-branch structure during training, which is then equivalently converted back to the original single-branch structure during inference. Structural re-parameterization achieves improved performance without increasing inference cost. However, the multi-branch structure used during training may slow down the training speed (as shown in the second column of Tab. 1, its training

*Corresponding Author (eetchen@fudan.edu.cn). †Equal Contribution.

Preprint. Under review.



(a) Training-time & Inference-time MoE model **(b) Training-time & Inference-time ACNet**

(c) Training-time & Inference-time EWA model

Figure 1: Training-time & inference-time comparison of (a) recent MoE block, (b) typical structural re-parameterization block, and (c) our EWA training block. Each FFN denotes an expert of MoE.

Table 1: The comparison of achieved effects between vanilla training, structural reparam., MoE training, and our EWA training. The testbed is a GTX3090. For structural reparam., the baseline is ResNet-18, and the comparison model is its ACNet counterpart [6]. For MoE training, the baseline is standard ViT-S, and the comparison model is our V-MoE-S which replaces every other FFN of ViT-S with an 8-expert MoE using top-1 routing [12]. For EWA training, we replace top-1 routing with random uniform partition for V-MoE-S. To test latency, we use $128\times3\times224\times224$ as the input.

-	Training Latency	Training Params	Inference Latency	Inference Params	Architecture	Applications
Vanilla Training	1.0×	1.0×	1.0×	1.0×	-	-
Structural Reparam.	2.18×	1.63×	1.0×	1.0×	CNN	2D
MoE Training	1.12×	3.25×	1.12×	3.25×	ViT	2D
EWA Training	**1.07×**	**3.25×**	**1.0×**	**1.0×**	**ViT**	**2D & 3D**

latency is up to 2.18× of vanilla training). Additionally, structural re-parameterization only applies to CNNs, not to other recent stronger architectures like ViTs (see the sixth column of Tab. 1).

For transformer architecture, the recent general training method involves replacing the feed-forward network (FFN) layer with a sparse mixture of experts (MoE) [23; 12; 11; 34; 16] (Fig. 1(a)). The MoE is a novel network component that includes multiple experts (FFNs) with unique weights and a trainable routing network. During the training and inference stages, the routing network selects a sparse combination of experts for each input, enabling efficient scaling of the capacity of transformer models through sparse computation. In the field of natural language processing, many sparse MoE models [23; 12; 11] have demonstrated that they outperform their dense counterparts.

However, MoE still has some limitations that restrict its widespread use in Vision Transformers (ViTs). Firstly, V-MoE [34] and SwinV2-MoE [16] were pre-trained on large-scale datasets (JFT-300M [38], ImageNet-22K [33]), and when pre-trained on ImageNet-1K [5], V-MoE's performance is lower than ViT [44]. Furthermore, there is no research studying MoE Vision Transformer on smaller datasets such as CIFAR-100 [20] and Tiny-ImageNet [21]. Secondly, current MoE ViTs [34; 45; 44; 16; 19] are focused on 2D visual tasks, and there are no MoE ViTs for 3D visual tasks (see the fourth row of Tab. 1). Thirdly, the top-k routing mechanism of MoE itself is not efficient and effective. As shown in the fourth row of Tab. 1, the training and inference latency both increase to 1.12× since top-k routing requires additional computation to make routing decisions for each input, and the training and

inference parameters both largely increase to 3.25× that burdens model employing. Besides, top-k routing easily leads to load imbalance across experts, thus many studies are devoted to handling this problem through various designs, such as adding auxiliary load balancing losses [23; 12], making routing decisions globally [24], or redesigning the routing algorithm [35; 50].

Therefore, the following questions arise: *Is there a new training method that can efficiently train the sparsely activated experts during the training phase, and can a sparse model be converted to a dense model without additional computation cost during the inference phase? Can we further extend the effectiveness of MoE to both 2D visual datasets and 3D visual tasks?*

In this paper, we design a new general training strategy for ViTs called Experts Weights Averaging (EWA) training. Our approach achieves performance improvement without increasing inference latency and parameters, as shown in the second and last rows of Tab. 1. In detail, we decouple the training and inference stages of ViT, as shown in Fig. 1(c). During training, we replace some FFNs of vanilla ViT with specially designed, more efficient MoEs that assign tokens to experts by Random Uniform Partition (RUP), requiring no additional parameters, special designs, or auxiliary losses. Further, at the end of each iteration, we perform Experts Weights Averaging (EWA) on each MoE. During inference, by simply averaging the experts of each MoE into a single FFN, we can convert the sparse MoE model into a vanilla ViT model without any performance loss.

We conduct comprehensive experiments on various 2D and 3D visual tasks, ViT architectures, and datasets to validate the effectiveness and generalization of the proposed training scheme. For example, our proposed training method achieves 1.72% improvement on the image classification task and 1.74% mIoU improvement on point cloud semantic segmentation task. Furthermore, EWA training can also be used for finetuning pretrained ViT models. Our training method improves the ViT-B finetuning on CIFAR100 from 90.71% to 91.42% ($\uparrow 0.71\%$). Finally, we find that the key technology of Experts Weights Averaging in EWA training can greatly improve the effectiveness of naive MoE in various 2D visual small datasets and 3D visual tasks.

The main contributions are summarized as:

- We propose a new general training scheme for ViTs, achieving performance improvement without increasing inference latency and parameters. During training, we replace some of the FFNs in the ViT architecture with MoE using Random Uniform Partition and perform Experts Weights Averaging on these MoE layers after each model weight update. After training, we convert each MoE layer in the model into an FFN layer by averaging the experts, thus transforming the model back to the original ViT architecture for inference.
- We validate that EWA training can bring unified performance improvement to various Vision Transformer architectures on different 2D/3D visual tasks and datasets. Furthermore, we find that EWA training can also be used for fine-tuning pretrained ViT models, achieving even further performance improvement. We provide theoretical analysis to show why and how it works.
- We explore the effectiveness of naive MoE on small 2D visual datasets and 3D visual tasks. We find that using naive MoE results in lower performance than dense counterparts, but the proposed Experts Weights Averaging technique can greatly improve the effectiveness of naive MoE.

2 Related works

2.1 MoE

The Sparsely-Gated Mixture-of-Experts (MoE) is first introduced in [36] and has shown significant improvements in training time, model capacity, and performance. Furthermore, when MoE is introduced into transformer-based language models, it receives increasing attention [23; 12; 11]. To date, many works have improved MoE's key component, that is the routing network [12; 24; 35; 50; 51]. On the one hand, for traditional learnable routing networks, Switch Transformer [12] simplifies routing by selecting only the top-1 expert for each token, Base Layer [24] handles a linear assignment problem to perform routing, and Expert Choice [50] routes top-k tokens to each expert. On the other hand, Hash Layer [35] designs deterministic hash for token routing, and THOR [51] randomly activates experts for each input. In addition to routing network improvements, many works have explored MoE's implementation on modern hardwares [14; 31; 28; 16; 27], scaling properties [12; 1; 11; 4], and applications [23; 34; 46; 26; 45; 44; 19; 49; 3]. However, most of MoE works are based on transformer-based language models, and only a few MoE Vision Transformer

works exist in the computer vision field [34; 45; 44; 16; 19]. Riquelme et al. [34] create V-MoE by replacing some FFN layers of ViT with MoE layers, pre-train V-MoE on JFT-300M or JFT-3B and achieve great success in image classification. Hwang et al. [16] add MoE layers to the Swin-transformer V2 architecture for pre-training on ImageNet-22K for image classification and object detection tasks. Xue et al. [45] propose a more parameter-efficient and effective framework, WideNet, by replacing FFN with MoE and sharing the MoE layer across transformer blocks, achieving performance improvement on ImageNet-1K pre-training. Komatsuzaki et al. [19] use a pre-trained dense transformer checkpoint to warm-start the training of a MoE model, and achieve promising results on JFT-300M pre-training and ImageNet-1K fine-tuning. Xue et al. [44] achieve knowledge distillation from a pre-trained MoE model to learn a dense model, and verify it on ImageNet-1K.

In this paper, we propose a new general training for improving the performance of ViTs without increasing inference cost, which has different targets and implementation methods from MoE. Besides, previous MoE-ViT studies have primarily focused on larger 2D visual datasets, with little exploration conducted on smaller 2D visual datasets and 3D vision. This paper shows that adding naive MoE to ViT degrades the performance of ViTs on smaller 2D visual datasets and 3D vision, and introduces a novel Early EMA training approach that can greatly improve their performance.

2.2 Structural Re-parameterization

The essence of structural re-parameterization is to use a multi-branch structure instead of a convolution or fully-connected layer to enhance the model during training. After training, multiple branches are fused for inference using equivalent parameter transformations or specific algorithms. The typical examples are ACNet [6], DBB [9] and RepVGG [10]. Specifically, ACNet [6] replaces common $K \times K$ convolutions in CNN with ACBlocks that have parallel structures of $K \times 1$, $1 \times K$, and $K \times K$ convolutions during training. After training, ACBlocks are equivalently converted to a single $K \times K$ convolution layer. DBB [9] uses a multi-branch structure of 1×1 convolution, $K \times K$ convolution, 1×1-$K \times K$ convolution, and 1×1 convolution-average pooling to replace the original $K \times K$ convolution during training, which is equivalently converted to a single $K \times K$ convolution during inference. RepVGG [10] adds parallel 1×1 convolution branch and identity branch to each 3×3 convolution layer during training and is equivalently converted to a single 3×3 convolution layer after training. Due to the performance improvement without increasing inference cost it brings, structural re-parameterization has received increasing attention and application in various computer vision tasks. For example, ExpandNets [13] uses it to design compact models, RepNAS [48] uses it in neural architecture search, and ResRep [7] combines it with pruning.

However, previous studies on structural re-parameterization have mainly focused on CNN architectures, making them incompatible with ViT architectures. This paper introduces a novel approach that decouples the training and inference stages of ViT architectures. During training, we use efficient MoE sparse computation with Random Uniform Partition and Experts Weights Averaging mechanisms. During inference, we convert the model to the original ViT architecture, resulting in performance improvements without increasing inference parameters and latency. This approach is the first of its kind to apply MoE sparse computation to improve the performance of ViT architectures.

2.3 Weight Averaging

Weight averaging is a commonly used technique in deep learning and has been widely used in various tasks. For instance, in self-supervised and semi-supervised learning, [39; 2] use the exponential moving average (EMA) weights of the student model as the teacher to provide a smoother target for the student. Similarly, in online knowledge distillation, [42] employs the EMA weights of each branch as the peer mean teacher to collaboratively transfer knowledge among branches. Additionally, many studies use weight averaging to enhance the model's generalization ability [17; 41; 40; 32]. For example, stochastic weight averaging (SWA) [17] averages the weights of multiple points on the SGD optimization trajectory and achieves better generalization ability than conventional training. Wortsman et al. [41] average the original zero-shot model and the fine-tuned model, resulting in improvements in both in-distribution and out-of-distribution generalization. Model Soup [40] averages models with the same pre-trained initialization but different hyperparameter configurations during fine-tuning to obtain a better model. DiWA [32] provides theoretical analysis for the success of weight averaging in OOD and proposes to reduce the covariance between predictions and improve OOD generalization by averaging the weights of diverse independently trained models.

Unlike the studies mentioned above that conduct weight averaging at the model level to obtain a single model, our approach performs weight averaging at the expert level in MoE during the training process. By averaging the weights of other experts onto each expert, we obtain multiple experts.

2.4 Dropout and Other variants for ViT

Dropout [37] is a common technique used to improve the network generalization ability by randomly dropping neurons and their corresponding connections during training to prevent overfitting. For ViTs, most works use dropout with fixed drop rate for the self-attention or FFN layers. Meanwhile, Stochastic Depth [15] is often used in ViTs to randomly drop some blocks and applies higher drop rates to deeper blocks to improve generalization. Recently, a new dropout variant for ViT called dropkey is introduced. Dropkey [25] assigns an adaptive operator to each attention block by randomly dropping some keys during attention computation. This strategy constrains the attention distribution, making it smoother and more effective in alleviating overfitting. Moreover, dropkey gradually reduces the drop rate as the number of blocks increases, avoiding the instability that can result from a constant drop rate in traditional dropout methods during the training process.

In our method, for each MoE layer, we randomly and evenly divide all tokens among all experts during training. This is equivalent to randomly dropping a large number of tokens at each expert. Such an operation may help prevent overfitting while ensuring the training latency remains almost equivalent to vanilla training.

3 Method

3.1 PRELIMINARY

MoE. As shown in Fig. 1(a), a standard MoE layer consists of a set of N experts (i.e., N FFNs) $\{E_1, E_2, \ldots, E_N\}$, and a routing network G with weight W_g. Given an input example x, the output of both the training-time and inference-time MoE layer can be written as:

$$y = \sum_{i=1}^{N} G(x)_i \cdot E_i(x) \tag{1}$$

$$G(x) = TopK(softmax(x \cdot W_g), k) \tag{2}$$

where $G(x)_i$ denotes the routing score to the i-th expert, $E_i(x)$ stands for the output of i-th expert, $TopK(,k)$ means to select top k experts and set $G(x)$ of other experts as 0. Usually, $k \ll N$, which means $G(x)$ a sparse N-dimensional vector. When $G(x)_i = 0$, $E_i(x)$ does not need to be computed.

Structural re-parameterization. As shown in Fig. 1(b), structural re-parameterization employs a multi-branch CNN structure to replace the common convolutional layer during training. Let us denote $\{f_1, f_2, \ldots, f_M\}$ as the total M branches (usually M different operators with compatible sizes). For a given input x, the output of training-time structure can be expressed as:

$$y = \sum_{i=1}^{M} f_i(x) \tag{3}$$

After training, these M branches will be equivalently converted to a single convolutional layer F for inference. For an inference-time input x, the output becomes $y = F(x)$.

3.2 A New General Training Scheme

3.2.1 Motivation

This paper aims to design a new general training scheme for ViTs, achieving performance improvement without increasing inference cost. As shown in Eq. 3, classical structural re-parameterization only applies to CNNs composed of convolutions and significantly increases the training cost due to the multi-branch structure. As shown in Eq. 2, recent MoE consists of N experts and a routing network. First, the routing network brings additional parameters needed to learn and easily leads to the load imbalance problem. Second, N experts largely increase the burden of deploying the model. Third, the routing score $G(x)$ is a bottleneck in converting MoE into FFN. Considering these, we first

propose a specially designed, more efficient MoE to replace some FFNs of ViT for training, which assigns tokens to experts by Random Uniform Partition. Then, we propose a simple-but-effective technique called Experts Weights Averaging, which averages the weights of other experts to each expert at the end of each training iteration. As such, after training, we can convert each MoE to a single FFN for inference. We show how to apply it during training and fine-tuning below.

3.2.2 EWA Training

Overall. When applying EWA training to train a ViT model from scratch, we first create a randomly initialized MoE ViT by replacing some FFNs with Random Uniform Partition based MoE layers. During training, we perform Experts Weights Averaging with share rate β for each MoE layer after each weight update, and the β increases linearly with the training epoch from 0 to the hyperparameter $share\ rate$. After training, we convert each MoE into an FFN by averaging the experts.

MoE layer with Random Uniform Partition. Given a MoE layer that consists of N experts (namely N identical FFNs), $\{E_1, E_2, \ldots, E_N\}$, assuming these are T inputs $\{x_1, x_2, \ldots, x_T\}$ and T can be divisible by N. The outputs of MoE layer with Random Uniform Partition (RUP) can be written as:

$$\{x_1, x_2, \ldots, x_T\} \rightarrow \{X_1, X_2, \ldots, X_N\} \tag{4}$$

$$y = E_i(x) \qquad if\ x \in X_i \tag{5}$$

As shown in Eq. 4, the T inputs are randomly and uniformly partitioned into N parts, where X_i represents the $\frac{T}{N}$ inputs that assigned to the i-th expert. Further, as shown in Eq. 5, for each input in X_i, its output is $E_i(x)$. **We show the pytorch-like code of RUP in Appendix**.

Experts Weights Averaging. Given the weights of N experts $\{W_1, W_2, \ldots, W_N\}$ and share rate β, Experts Weights Averaging (EWA) performed on each MoE layer can be formulated as:

$$\overline{W_i} = (1-\beta)W_i + \sum_{j \neq i}^{N} \frac{\beta}{N-1} W_j \tag{6}$$

$\overline{W_i}$ denotes the new weight of the i-th expert. In short, we average the weights of other experts to each expert and obtain multiple experts. **The pytorch-like code of EWA is provided in Appendix**.

Convert MoE into FFN. After training, each MoE layer will be converted into an FFN layer by simply averaging the experts weights. Given a MoE layer consisting of N experts $\{E_1, E_2, \ldots, E_N\}$, the corresponding inference-time FFN can be expressed as:

$$FFN = \frac{1}{N} \sum_{i=1}^{N} E_i \tag{7}$$

By this way, the training-time MoE model will be converted into a vanilla ViT model for inference. **We conduct experiments to show the rationality in Appendix**.

3.2.3 EWA Fine-tuning

Overall. When applying EWA training to fine-tune a pre-trained ViT model, the only difference from training ViT from scratch is the weight initialization of the created MoE ViT model. Rather than random initialization, we initialize the MoE ViT with the given ViT model checkpoint. When replacing some FFNs with MoE layers, each expert of a specific MoE layer is initialized as a copy of the corresponding FFN layer in the pre-trained ViT model. For other layers that remain unchanged, their weights are directly inherited from the original ViT checkpoint.

4 Theoretical Analysis

In this section, we theoretically analyze *why and how EWA training works*. For the convenience of analysis, we focus on one MoE layer and m training steps. Assuming that the current training step is t, following Eq. 6, the new weight of i-th expert after experts weights averaging can be written as:

$$\overline{W_i^t} = (1-\beta)W_i^t + \sum_{j \neq i}^{N} \frac{\beta}{N-1} W_j^t \tag{8}$$

Table 2: Performance comparison between vanilla and EWA training on various architectures and datasets. For ViT architectures (-XT, -XS, -l2, -T) and training methods (+SL) specifically designed for small datasets [22], the addition of our EWA method can still improve their performance.

(a) on CIFAR-100 dataset

Model	Vanilla Top-1	EWA Top-1	Δ Top-1
ViT-S	72.61	74.33	+1.72
ViT-XT	73.93	75.20	+1.27
ViT-XT + SL	77.04	77.94	+0.90
PiT-XS	74.99	75.65	+0.66
PiT-XS + SL	79.73	80.39	+0.66
T2T-12	77.19	77.80	+0.61
T2T-12 + SL	78.35	78.81	+0.46
Swin-T	77.38	77.65	+0.27
Swin-T + SL	80.31	80.52	+0.21

(b) on Tiny-ImageNet dataset

Model	Vanilla Top-1	EWA Top-1	Δ Top-1
ViT-S	57.41	59.50	+2.09
ViT-XT	55.92	57.86	+1.94
ViT-XT + SL	59.72	60.75	+1.03
PiT-XS	59.03	61.61	+2.58
PiT-XS + SL	63.11	64.53	+1.42
T2T-12	60.39	61.65	+1.26
T2T-12 + SL	62.57	63.36	+0.79
Swin-T	59.70	60.66	+0.96
Swin-T + SL	64.34	65.13	+0.79

Then, the $(t+1)$-th training step begins, and the weights of each expert of MoE are updated as $\{W_1^{t+1}, W_2^{t+1}, \ldots, W_N^{t+1}\}$, where $W_i^{t+1} = \overline{W_i^t} + \eta\nabla\overline{W_i^t}$, η denotes the learning rate. Further, the experts weights averaging is performed, as shown in Eq. 9. Based on Eq. 8 and W_i^{t+1}, Eq. 9 can be further reformulated as Eq. 10.

$$\overline{W_i^{t+1}} = (1-\beta)W_i^{t+1} + \sum_{j\neq i}^{N}\frac{\beta}{N-1}W_j^{t+1} \tag{9}$$

$$= (1-\beta)^2 W_i^t + \eta(1-\beta)\nabla\overline{W_i^t} + \sum_{j\neq i}^{N}\left[\frac{\beta}{N-1}W_j^{t+1} + \frac{\beta(1-\beta)}{N-1}W_j^t\right] \tag{10}$$

Similarly, we can get $\overline{W_i^{t+2}}$, $\overline{W_i^{t+3}}$ and $\overline{W_i^{t+m}}$. Through mathematical induction, we get Eq. 12.

$$\overline{W_i^{t+m}} = (1-\beta)W_i^{t+m} + \sum_{j\neq i}^{N}\frac{\beta}{N-1}W_j^{t+m} \tag{11}$$

$$= (1-\beta)^{m+1}W_i^t + \eta\sum_{k=1}^{m}(1-\beta)^k\nabla\overline{W_i^{t+m-k}} + \frac{\beta}{N-1}\sum_{j\neq i}^{N}\sum_{k=0}^{m}(1-\beta)^{m-k}\cdot W_j^{t+k} \tag{12}$$

According to the above derivation, there are two findings: 1) Observing the first term of Eq. 9 and Eq. 11, *there is a layer-wise weight decay when performing experts weights averaging*; 2) Observing the last term of Eq. 10 and Eq. 12, *there continuously aggregates multi-experts historical exponential average weights along the training iteration of EWA training*. **More Details are shown in Appendix.**

5 Experiments

When applying EWA training to train or fine-tune ViTs, we always use random uniform partition based MoE to replace some FFNs. Unless otherwise specified, the number of each MoE's experts is set to 4 by default. The hyperparameter $share\ rate$ is simply adjusted within {0.1, 0.2, 0.3, 0.4, 0.5} for different ViT architectures to get the best performance. Following V-MoE [34], where to insert these MoE layers is simply adjusted within {every-2, last-4}. **All details are shown in Appendix.**

5.1 EWA Training

5.1.1 2D Classification on various architectures and datasets

Settings. We conduct comprehensive experiments on various architectures and datasets to evaluate the EWA training scheme. For datasets, we employ CIFAR-100 and Tiny-ImageNet. For architectures, besides standard ViT-S, we also adopt various ViT architectures specifically designed for small datasets such as smaller ViT-Tiny (we call it ViT-XT), PiT-XS, T2T-12, Swin-T, and their SL

Table 3: Performance comparison between vanilla and EWA training on point cloud classification and point cloud semantic segmentation.

(a) Classification on ModelNet40.

Scheme	Acc.(w/o vote)	Acc.(w vote)
Vanilla	92.42	92.67
EWA	**92.54**	**92.79**

(b) Semantic segmentation on S3DIS Area5.

Scheme	OA	mAcc.	mIoU
Vanilla	89.53	72.43	66.62
EWA	**90.05**	**73.83**	**68.36**

Table 4: Performance comparison of vanilla and EWA fine-tuning on various models and datasets.

(a) on Food-101 dataset

Model	Vanilla Top-1	EWA Top-1	Δ Top-1
ViT-S	88.04	88.42	+0.38
ViT-B	86.78	87.32	+0.54

(b) on CIFAR-100 dataset

Model	Vanilla Top-1	EWA Top-1	Δ Top-1
ViT-S	90.22	90.67	+0.45
ViT-B	90.71	91.42	+0.71

enhanced counterparts [22]. SL means to apply shifted patch tokenization and locality self-attention, which enable ViTs to learn better on small-size datasets [22]. **Details of all used ViTs are shown in Appendix**. For vanilla and EWA training, we always apply consistent CutMix, Mixup, Auto Augment, random erasing, and label smoothing for all models. On CIFAR-100, the input size is 32×32, the patch size of ViT is set to 4, while 2 for PiT and Swin. On Tiny-ImageNet, the input size is 64×64, the patch size is set to 8 for ViT, while 4 for PiT and Swin. All models are trained with the AdamW optimizer, the cosine schedule and a batch size of 128. For ViT-S, we train it for 300 epochs with 30 warm-up epochs, the learning rate is set to 0.0006, the weight decay is set to 0.06. For the remaining models, we train them for 100 epochs with 10 warm-up epochs, the weight decay is set to 0.05, the learning rate is set to 0.003 (0.001 for Swin-T).

Results. As shown in Tab. 2, compared with vanilla training, EWA training can bring consistent performance improvement for various ViT architectures on different datasets. For standard ViT-S, EWA training can bring great improvements {1.72%, 2.09%} on {CIFAR-100, Tiny-ImageNet} respectively. Even for the specifically designed ViT architectures and their SL-enhanced counterparts, the addition of EWA training scheme can still improve their performance, which further confirms its effectiveness and generalization. For instance, SL-enhanced Swin-T with vanilla training has already achieved a relatively high top-1 accuracy, 80.31% on CIFAR-100 and 64.34% on Tiny-ImageNet. It can still obtain {0.21%, 0.79%} performance boosts from EWA training for each dataset.

5.1.2 3D Vision tasks

Settings. To demonstrate the general training ability of our method, we also conduct experiments on point cloud classification (sparse) and semantic segmentation (dense) tasks. Following the experimental setting of PointBERT [47], we conduct the point cloud classification on the ModelNet40 [30]. Following the experimental setting of Pix4point [29], we conduct the point cloud semantic segmentation task on the S3DIS [43] and leave area 5 as testing, other areas as training. Following PointBERT and Pix4point, we use the standard encoder in ViT-S as transformer backbone.

Results. As shown in Tab. 3, the proposed EWA training scheme obtains consistent performance gain on both point cloud classification and semantic segmentation tasks. Notably, our method achieves an improvement of 0.52% in overall accuracy (OA), 1.40% in mean accuracy (mAcc) and 1.74% in mean IoU (mIoU) on the segmentation task.

5.2 EWA Fine-tuning

Settings. To evaluate EWA fine-tuning, we take pre-trained ViT-B and ViT-S from timm library and fine-tune them on CIFAR-100 and Food-101. For both CIFAR-100 and Food-101 [18], the batch size is set to 128 and all images are scaled to 224×224 for fine-tuning. For CIFAR-100, the total fine-tuning steps is set to 5000 with 500 warm-up steps. For Food-101, we first warm up 100 steps and the number of total steps is 1000. We use SGD optimizer with 0.9 momentum. The learning rate is 0.01 for CIFAR-100 and 0.03 for Food-101. We just simply place the MoE on every other layer and set the number of MoE layer's experts to 4. The β of Experts Weights Averaging (Eq. 6) is linearly increased with current step from 0 to the given hyperparameter $share\ rate$.

Results. Tab. 4 shows the fine-tuned performance comparison between vanilla and EWA fine-tuning. Although the performance of vanilla fine-tuning is pretty high, EWA fine-tuning can still bring further improvements. Specifically, equipped with EWA fine-tuning, ViT-S and ViT-B obtain {0.38%, 0.45%} and {0.54%, 0.71%} performance boosts on {Food-101, CIFAR-100} respectively.

5.3 Ablation Studies

Share schedule and Share rate. We first study how to set the hyperparameter $share\ rate$ and share schedule for Experts Weights Averaging. In the above, the share schedule is set to linear increasing by default, we here compare it with the constant share schedule. For $share\ rate$, we adjust it within {0.1, 0.2, 0.3, 0.4, 0.5}. We train ViT-S on CIFAR100 using EWA training with various share schedules and $share\ rate$. As shown in Fig. 2, EWA training with different schedules and $share\ rate$ can always outperform vanilla training, and EWA training with the linear increasing schedule performs better than the constant schedule under most cases. **Ablation studies about the number of experts are shown in the Appendix**.

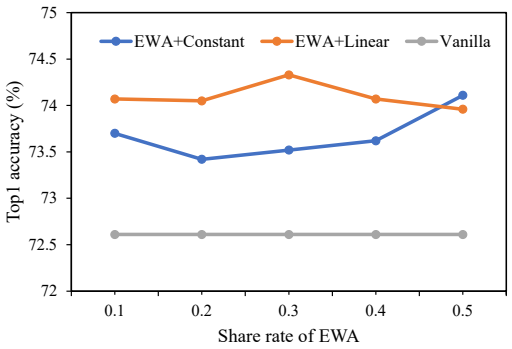


Figure 2: Ablation study on share schedules and share rates of the proposed EWA scheme.

6 More Discussions And Findings

More interestingly, we find that, on small 2D visual datasets and 3D visual tasks, naive MoE results in lower performance than its original dense counterparts, and our EWA technology can be seamlessly applied to naive MoE and help it learn better.

naive MoE. Following the settings of training ViT-S on CIFAR-100, Tiny-ImageNet, ModelNet40 and S3DIS, we further train our V-MoE-S, which replaces some FFN of ViT-S with naive top-1 routing MoE [12] every other block. We set the number of experts as 4, the weight of load balance loss λ as 0.01 and the capacity ratio C as 1.05 for 2D datasets and 1.2 for 3D datasets. As shown in Tab. 5, with vanilla training, naive V-MoE-S degrades performance compared to ViT-S on different datasets.

Table 5: Comparison between different models with different training schemes. Compared to ViT-S with vanilla training, V-MoE-S with vanilla training degrades the performance, while V-MoE-S with Early EWA improves the performance by a large margin, across 2D/3D different datasets.

Model +scheme	ViT-S +Vanilla	V-MoE-S +Vanilla	V-MoE-S +Early EWA
C100 Acc	72.61	72.17	**74.31**
Tiny-Img Acc	57.41	56.28	**59.91**
MN40 Acc	92.42	92.27	**92.63**
S3DIS mIoU	66.62	66.36	**67.67**

naive MoE + Early EWA. Things change when we introduce Experts Weights Averaging (EWA) into naive MoE. Considering that using EWA throughout the entire training process will finally lead to nearly identical weights across experts, we only perform EWA on naive MoE in the early stage of training. By default, we use constant share schedule and only perform Early EWA in the first half of total training epochs. With Early EWA training, the performance of V-MoE-S is greatly improved, resulting in a top-1 accuracy gain of {2.14%, 3.63%} on {CIFAR-100, Tiny-ImageNet} respectively. In addition, our Early EWA training can improve the performance of V-MoE-S on various 3D vision tasks, with 0.36% accuracy gain on ModelNet40 classification, and 0.30% OA gain, 1.40% mAcc gain, 1.31% mIoU gain on S3DIS Area5 segmentation. **Further studies are shown in Appendix.**

7 Conclusion

In conclusion, our proposed training scheme for ViTs achieves performance improvement without increasing inference latency and parameters. By designing efficient MoEs and EWA during training, and converting each MoE back into a FFN by averaging the experts during inference, we decouple

the training and inference phases of ViTs. Comprehensive experiments across various 2D and 3D visual tasks, ViT architectures, and datasets demonstrate the effectiveness and generalizability. The theoretical analysis further supports our approach, and our training scheme can also be applied to fine-tuning ViTs and improve the effectiveness of naive MoE in various visual tasks. Overall, our proposed training scheme provides a promising direction for enhancing the performance of ViTs in various visual tasks, and we believe it could lead to further research and development in this field.

References

[1] Mikel Artetxe, Shruti Bhosale, Naman Goyal, Todor Mihaylov, Myle Ott, Sam Shleifer, Xi Victoria Lin, Jingfei Du, Srinivasan Iyer, Ramakanth Pasunuru, et al. Efficient large scale language modeling with mixtures of experts. *arXiv preprint arXiv:2112.10684*, 2021.

[2] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[3] Zitian Chen, Yikang Shen, Mingyu Ding, Zhenfang Chen, Hengshuang Zhao, Erik Learned-Miller, and Chuang Gan. Mod-squad: Designing mixture of experts as modular multi-task learners. *arXiv preprint arXiv:2212.08066*, 2022.

[4] Aidan Clark, Diego De Las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, et al. Unified scaling laws for routed language models. In *International Conference on Machine Learning*, pages 4057–4086. PMLR, 2022.

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

[6] Xiaohan Ding, Yuchen Guo, Guiguang Ding, and Jungong Han. Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1911–1920, 2019.

[7] Xiaohan Ding, Tianxiang Hao, Jianchao Tan, Ji Liu, Jungong Han, Yuchen Guo, and Guiguang Ding. Resrep: Lossless cnn pruning via decoupling remembering and forgetting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4510–4520, 2021.

[8] Xiaohan Ding, Chunlong Xia, Xiangyu Zhang, Xiaojie Chu, Jungong Han, and Guiguang Ding. Repmlp: Re-parameterizing convolutions into fully-connected layers for image recognition. *arXiv preprint arXiv:2105.01883*, 2021.

[9] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Diverse branch block: Building a convolution as an inception-like unit. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10886–10895, 2021.

[10] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13733–13742, 2021.

[11] Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR, 2022.

[12] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1):5232–5270, 2022.

[13] Shuxuan Guo, Jose M Alvarez, and Mathieu Salzmann. Expandnets: Linear over-parameterization to train compact convolutional networks. *Advances in Neural Information Processing Systems*, 33:1298–1310, 2020.

[14] Jiaao He, Jiezhong Qiu, Aohan Zeng, Zhilin Yang, Jidong Zhai, and Jie Tang. Fastmoe: A fast mixture-of-expert training system. *arXiv preprint arXiv:2103.13262*, 2021.

[15] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 646–661. Springer, 2016.

[16] Changho Hwang, Wei Cui, Yifan Xiong, Ziyue Yang, Ze Liu, Han Hu, Zilong Wang, Rafael Salas, Jithin Jose, Prabhat Ram, et al. Tutel: Adaptive mixture-of-experts at scale. *arXiv preprint arXiv:2206.03382*, 2022.

[17] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

[18] Parneet Kaur, Karan Sikka, and Ajay Divakaran. Combining weakly and webly supervised learning for classifying food images. *arXiv preprint arXiv:1712.08730*, 2017.

[19] Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. Sparse upcycling: Training mixture-of-experts from dense checkpoints. *arXiv preprint arXiv:2212.05055*, 2022.

[20] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[21] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[22] Seung Hoon Lee, Seunghyun Lee, and Byung Cheol Song. Vision transformer for small-size datasets. *arXiv preprint arXiv:2112.13492*, 2021.

[23] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

[24] Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, pages 6265–6274. PMLR, 2021.

[25] Bonan Li, Yinhan Hu, Xuecheng Nie, Congying Han, Xiangjian Jiang, Tiande Guo, and Luoqi Liu. Dropkey. *arXiv preprint arXiv:2208.02646*, 2022.

[26] Basil Mustafa, Carlos Riquelme, Joan Puigcerver, Rodolphe Jenatton, and Neil Houlsby. Multimodal contrastive learning with limoe: the language-image mixture of experts. *arXiv preprint arXiv:2206.02770*, 2022.

[27] Xiaonan Nie, Xupeng Miao, Zilong Wang, Zichao Yang, Jilong Xue, Lingxiao Ma, Gang Cao, and Bin Cui. Flexmoe: Scaling large-scale sparse pre-trained model training via dynamic device placement. *arXiv preprint arXiv:2304.03946*, 2023.

[28] Xiaonan Nie, Pinxue Zhao, Xupeng Miao, and Bin Cui. Hetumoe: An efficient trillion-scale mixture-of-expert distributed training system. *arXiv preprint arXiv:2203.14685*, 2022.

[29] Guocheng Qian, Xingdi Zhang, Abdullah Hamdi, and Bernard Ghanem. Pix4point: Image pretrained transformers for 3d point cloud understanding. *arXiv preprint arXiv:2208.12259*, 2022.

[30] Shi Qiu, Saeed Anwar, and Nick Barnes. Geometric back-projection network for point cloud classification. *IEEE Transactions on Multimedia*, 24:1943–1955, 2021.

[31] Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. In *International Conference on Machine Learning*, pages 18332–18346. PMLR, 2022.

[32] Alexandre Rame, Matthieu Kirchmeyer, Thibaud Rahier, Alain Rakotomamonjy, Patrick Gallinari, and Matthieu Cord. Diverse weight averaging for out-of-distribution generalization. *arXiv preprint arXiv:2205.09739*, 2022.

[33] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. *arXiv preprint arXiv:2104.10972*, 2021.

[34] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595, 2021.

[35] Stephen Roller, Sainbayar Sukhbaatar, Jason Weston, et al. Hash layers for large sparse models. *Advances in Neural Information Processing Systems*, 34:17555–17566, 2021.

[36] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[37] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[38] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[39] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017.

[40] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pages 23965–23998. PMLR, 2022.

[41] Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7959–7971, 2022.

[42] Guile Wu and Shaogang Gong. Peer collaborative learning for online knowledge distillation. In *Proceedings of the AAAI Conference on artificial intelligence*, volume 35, pages 10302–10310, 2021.

[43] Qiangeng Xu, Xudong Sun, Cho-Ying Wu, Panqu Wang, and Ulrich Neumann. Grid-gcn for fast and scalable point cloud learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5661–5670, 2020.

[44] Fuzhao Xue, Xiaoxin He, Xiaozhe Ren, Yuxuan Lou, and Yang You. One student knows all experts know: From sparse to dense. *arXiv preprint arXiv:2201.10890*, 2022.

[45] Fuzhao Xue, Ziji Shi, Futao Wei, Yuxuan Lou, Yong Liu, and Yang You. Go wider instead of deeper. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8779–8787, 2022.

[46] Zhao You, Shulin Feng, Dan Su, and Dong Yu. Speechmoe: Scaling to large acoustic models with dynamic routing mixture of experts. *arXiv preprint arXiv:2105.03036*, 2021.

[47] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19313–19322, 2022.

[48] Mingyang Zhang, Xinyi Yu, Jingtao Rong, and Linlin Ou. Repnas: Searching for efficient re-parameterizing blocks. *arXiv preprint arXiv:2109.03508*, 2021.

[49] Xiaofeng Zhang, Yikang Shen, Zeyu Huang, Jie Zhou, Wenge Rong, and Zhang Xiong. Mixture of attention heads: Selecting attention heads per token. *arXiv preprint arXiv:2210.05144*, 2022.

[50] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

[51] Simiao Zuo, Xiaodong Liu, Jian Jiao, Young Jin Kim, Hany Hassan, Ruofei Zhang, Tuo Zhao, and Jianfeng Gao. Taming sparsely activated transformer with stochastic experts. *arXiv preprint arXiv:2110.04260*, 2021.