Mixture of Experts for Data-Efficient Image Classification

Pengfei Sun Xuan Jin Xin He Huiming Zhang Yuan He
Hui Xue
Alibaba Group
Hangzhou, China
{yeqing.spf,jinxuan.jx,siwang.hx,zhm220845,heyuan.hy,hui.xueh}@alibaba-inc.com

Abstract

Data-efficient image classification means model learning from a few of image samples without any pretrained models and any additional data. And 2nd VIPrior Image Classification Challenge has be successfully hosted in the past few months. In this paper, a method based on Mixture of Experts(MoE) in our competition solutions will be introduced. Different from ensemble several single models, MoE is an approach to improving the capacity of networks without making models deeper, which means that several branch are built as experts in parallel. The depth of each expert dynamically changes during training for a better regularization. Additionally, a dynamic weight module is proposed for a better information fusion among several experts. Experimental results shows that our method could improve classification performance significantly. Finally, we won the 1st Place in VIPriors image classification competition. In this paper, classification experiments are based on Pytorch Image Models(timm)[19] which is easy to reproduce for others.

1. Introduction

Convolutional Neural Networks (CNNs) have achieved state-of-the-art performance in image classification, object detection, semantic segmentation, etc. From AlexNet[9], VGG[14], ResNet[5], EfficientNet[16, 17], RegNet[13], ResNeSt[23], etc., the classification accuracy on ImageNet has been increased significantly. Recently, with the help of large scale dataset pretraining and extra training data, vision transformer models show impressive performance than CNNs, such as ViT[21], BEiT[1],Swin Transformer[11], etc. But without pretrained model and large scale dataset, vision transformer is more difficult for training and easily overfit.

MoE is an effective way to improve the capacity of networks. In MoE architecture, each expert learning different features from input data. In dynamic neural networks, the expert in MoE could be executed selectively and the output results of experts could be fused by data-dependent weight. In [8, 4], a real-valued weight is used to dynamically weight the output from different experts. In the whole process, all experts will be executed. To reduce the computing consumption during test, a router is trained in [18, 18] that dynamically assign ambiguous sample to additional experts. During test time, the logits of assigned experts will be collected for final decision. In HydraNet[12], convolutional blocks in the last stage is replaced by multiple branches.

In this paper, a MoE architecture is proposed to improve the accuracy for data-efficient image classification. Although in VIPrior classification trainset, every class has 50 images. But the classification results show that the accuracy of different classes are different: some samples are easy for learning, but some samples with occlusion or ahaotic background are difficult for learning. Therefore, in this paper, a MoE architecture is designed to learning different features from samples of different difficulty. To avoid experts to learn the same feature from samples, a loss function for calculating the diversity of output features from experts is used to force experts to learn differently from training samples. In each expert, a dynamic neural architecture based method Stochastic depth[7] is used for regularization. During training, a teacher model is used for distillation to generate soft target for label smoothing. After training each expert separately, a fusion module is designed to fused the feature of all experts.

2. Proposed Method

To learning from classification dataset more efficiently, first, a dynamic neural backbone[7] based on ResNest[23] is designed. For a more knowledgeable label smoothing, a teacher model is used to generate soft label during training. Then, a MoE architecture based on backbone is designed which shares early layers in the backbone. Compared with mean of logits from experts, a weighted module can fused the imformation of different experts which can improve the performance of MoE.

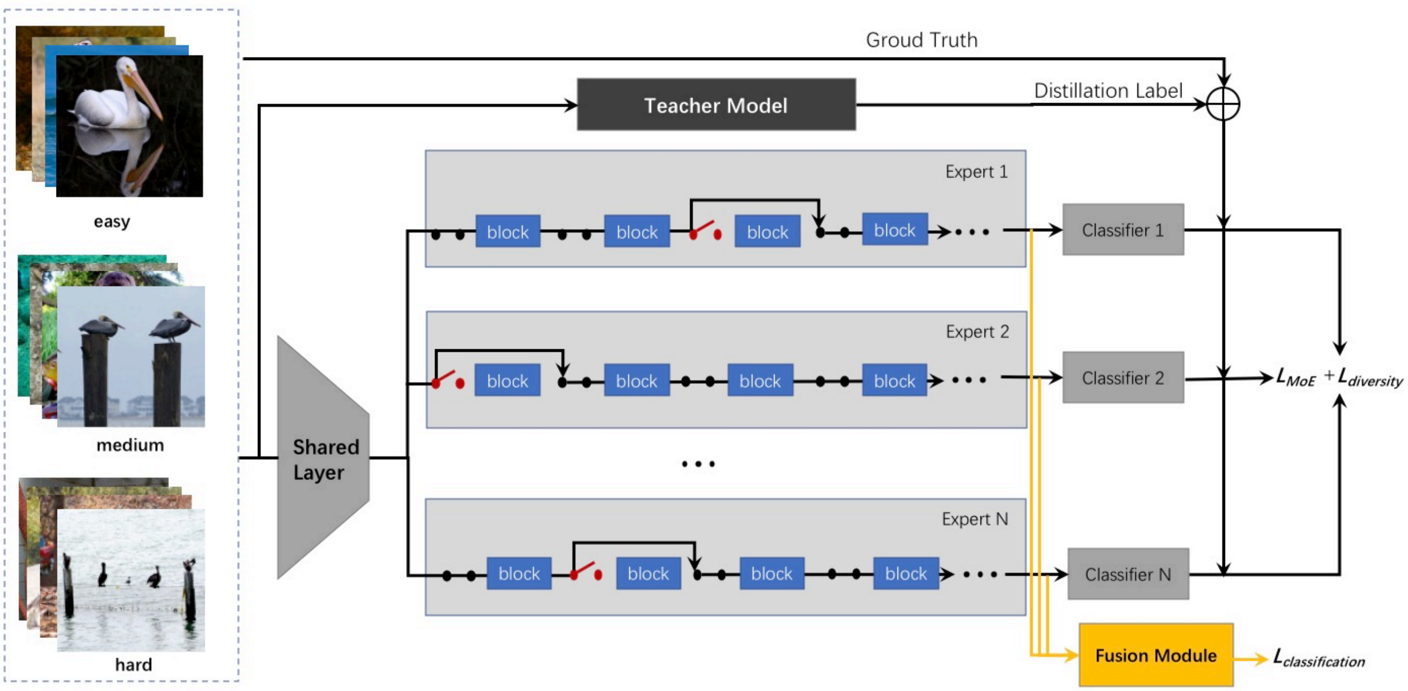

Figure 1. The proposed MoE architecture in this paper.

2.1. Training Single Expert with Distillation

For each expert in MoE, ResNeSt[23] is used as a backbone $\mathcal{M}$. Compared with ResNet[5], EfficientNet[16, 17] and RegNet[13], ResNeSt has the best performance and faster convergence rate in our experiments.

The spatial resolution operations in backbone, such as pooling and strided convolution are harmful for shift equivariance. Therefore, Blur pool[24] is used to improve the ability of translation invariant. For a better regularization, stochastic depth[7] is used in backbone. During training, blocks in backbone will randomly skipped, the depth of backbone will dynamically change, which make the learning more robust. As a regularization method, label smooth is an efficient method to make the model constrain more easily. For label l_i of class $i (i \in C)$, label smooth can be represent as:

$$l_{ls} = \begin{cases} 1-\varepsilon & if\, x = y, \\ \varepsilon/C - 1 & otherwise, \end{cases} \tag{1}$$

Considering the noise of the label and multi-label in trainset, a teacher model $\mathcal{M}_t$ pretrained on trainset is used for generating soft label to direct the student model $\mathcal{M}$ learning. For an image I_{input}, the final label l_{train} for training can be represented as :

$$l_{train} = \lambda * l_{ls} + (1-\lambda) * \mathcal{M}_t(I_{input}) \tag{2}$$

For data augmentation during training, cutmix[20] and mixup[22] is used, and a soft label cross-entropy loss $L_{soft-CE}$ is used which can be presented as :

$$L_{soft-CE} = -l_{train} * log(\mathcal{M}(I_{input})) \tag{3}$$

Additionally, cosine loss [2] is add into loss function:

$$L = L_{soft-CE} + (1 - cosine(l_{train}, \mathcal{M}(I_{input}))) \tag{4}$$

2.2. Mixture of Experts

To improve the accuracy of backbone and make maximum use of the features of training data in a single model, a MoE architecture is designed based on backbone in 2.1. For an expert $\mathcal{M}_i$ in MoE has same architecture and independent parameters. All experts shared the early layers whose features are basic features. Taking ResNest for example, during training, a MoE with N experts, the loss function can be presented as :

$$L_{MoE} = \sum_{i=1}^{N} L(l_{train}, \mathcal{M}_i(I_{input})) \tag{5}$$

Besides, because of same architecture in MoE, although parameters initialize randomly, experts would learning similar features and the outputs of experts will be similar which are little more than traditional model ensemble. To direct each expert to learn in an interactive way, a loss function $L_{diversity}$ for calculating the diversity of output features from experts is designed. For the output p_i for i_{th} expert, the average outputs p of all experts can be represented as:

$$p = \frac{1}{N}\sum_{i=1}^{N} p_i \tag{6}$$

Then the diversity loss of MoE is :

$$L_{diversity} = -\sum_{i=1}^{N} KL(p_i, p) \tag{7}$$

Finally, the training loss function L of MoE is :

$$L = L_{MoE} + \lambda L_{diversity} \tag{8}$$

In this paper, λ is set to 0.2.

2.3. Fusion of Experts

In 2.2, during training stage, each expert is calculated classification loss individually and a diversity loss is calculated among all experts. During testing, the final output is an average result of all output which ignore the relationship of different experts. So a fusion module $\mathcal{M}_{fusion}$ is designed to concatenate the embeddings E_i of each expert which can be expressed as :

$$\mathcal{M}_{fusion} = Linear(concat(E_1, E_2, ..., E_N), C) \tag{9}$$

During fusion module $\mathcal{M}_{fusion}$ training, all of parameters in $\mathcal{M}_{MoE}$ are fixed.

3. Experiments

3.1. Implementation Details

In order to reproduce the experimental results more easily, an open source image classification framework timm[19](`https://github.com/rwightman/pytorch-image-models`) is adopted in our competition scheme which includes newest image classification models such as CNNs models and vision transformers. In training stage, 32 GPUs with 12GB memory are used. To save the usage of GPU memory and speed up training process, mixed-precision training technology in Pytorch is used.

Additionally, self-supervised learning is also useful for data-efficient learning in our experiments. Each model would be self-supervised pretrained by MOVO-v2[3] on training data, and then used for parameters initialization in training.

All of models are training with 360/720 epochs. Learning rate warms up in the first 25 epochd from 0.0001 to 0.4. Then cosine scheduler is applied in the rest epochs. Batchsize on a single GPU is set from 8 to 32 which depends on GPU memory usage. After 300/640 epochs training, mixup and cutmix will be disabled in the last 60/80 epochs for complete image learning. Original AutoAugment, horizontal flip randomly, erasing randomly with prob 0.2, dropout with prob 0.2, label smoothing with value 0.1, stochastic depth with prob 0.2 are used from first to last. And after 360/720 epochs training, there are 10 epochs to cool down the model with a learning rate of 1e-5. And weight decay is set to 0.0001.

During training, the input image size of model is set to 352*352 and would be cropped randomly to 320*320. During test, test time augmentation is applied for a better performance including scale up input size and ten crops.

To get a better ensemble result, multiple series of networks are trained in our expertiments including ResNeSt[23], RegNet[13], EfficientNet[16], EfficientNet-V2[17], SeResNet[6] and SKNet[10]. The best performance of single model in our experiments is 74.03% which is based on ResNeSt series. For resize operation in SENet and ResNeSt, 'bilinear' is appilied, and 'bicubic' is used in SKNet, EfficientNet, EfficientNet-V2 and RegNet.

3.2. Experimental Results

The MoE architecture in this paper is based on ResNeSt which is represented as 'RS' in 3.2. In RS, N represents the number of experts in MoE. For distillation in MoE, a teacher model would be pretrained firstly on training data which has same architecture with student model. Then the student would be trained with the direction of teacher model. Fi-

Method	top-1 acc. (%)
MoE-RS-50 (N=1)	67.14
MoE-RS-SD-50 (N=1)	68.66
MoE-RS-SD-50 (N=1) + Distill	69.54
MoE-RS-SD-50 (N=1) + Distill + Fusion	-
MoE-RS-SD-50 (N=3)	70.14
MoE-RS-SD-50 (N=3) + Distill	71.21
MoE-RS-SD-50 (N=3) + Distill + Fusion	71.46
MoE-RS-SD-50 (N=5)	70.55
MoE-RS-SD-50 (N=5) + Distill	71.17/71.67*
MoE-RS-SD-50 (N=5) + Distill + Fusion	71.50/71.87*/73.11†
MoE-RS-101 (N=1)	69.47
MoE-RS-SD-101 (N=1)	70.71
MoE-RS-SD-101 (N=1) + Distill	71.84/72.73†
MoE-RS-SD-101 (N=1) + Distill + Fusion	-
MoE-RS-SD-101 (N=3) w/o diversity loss	70.88
MoE-RS-SD-101 (N=3)	71.44
MoE-RS-SD-101 (N=3) + Distill	72.05/72.52*/73.47†
MoE-RS-SD-101 (N=3) + Distill + Fusion	72.50/72.67*/74.03†
MoE-RS-SD-200 (N=1)	71.12
MoE-RS-SD-200 (N=1) + Distill	72.02/73.17†

Table 1. Results of MoE architecture proposed in this paper.

Method	top-1 acc. (%)
RegNetY-080	69.51
RegNetY-080 + Distill	70.17
RegNetY-120	69.76
RegNetY-120 + Distill	70.44/71.92†
RegNetY-160	70.04
RegNetY-160 + Distill	70.87/72.12†
RegNetY-320	70.02
RegNetY-320 + Distill	70.90/72.21†

Table 2. Results of RegNetY.

Method	top-1 acc. (%)
EfficientNet-b4	67.27
EfficientNet-b4 + Distill	68.48/69.67†
EfficientNet-b5	68.29
EfficientNet-b5 + Distill	69.90/71.00†
EfficientNet-b6	69.01
EfficientNet-b6 + Distill	70.26/71.38†
EfficientNet-V2-s	68.54
EfficientNet-V2-s + Distill	69.29/70.47†
EfficientNet-V2-m	69.66
EfficientNet-V2-m + Distill	70.06/71.20†

Table 3. Results of EfficientNet and EfficientNet-V2.

nally, a fusion module is trained with fixing all parameters in MoE-RS-SD. In Table2.2, values with * means models are trained with 720 epochs and † means models are tested with ten crops and a larger size than training.

Among the models in above tables, a weighted score average method[15] is used for ensemble in which the weight of higher performance models is set to 3 and rest is 1. Finally, the ensemble result got 75.46% top1 accuracy on testset.

4. Conclusion

In this paper, we discuss and explore MoE architecture for data-efficient learning on VIPriors Image Classification dataset. In each expert in MoE, a dynamic depth method is used for regularization. To make experts learn differenetly from each other, a diversity loss function is applied. Additionally, a fusion module is designed for information fusion among all experts. In all of our experiments, there are no weights pretrained on other/testing data or additional data used. Finally, we won the 1st Place in VIPriors Image Classification Competition.

References

[1] H. Bao, L. Dong, and F. Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

[2] B. Barz and J. Denzler. Deep learning on small datasets without pre-training using cosine loss. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 1371–1380, 2020.

[3] X. Chen, H. Fan, R. Girshick, and K. He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[4] D. Eigen, M. Ranzato, and I. Sutskever. Learning factored representations in a deep mixture of experts. *arXiv preprint arXiv:1312.4314*, 2013.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[7] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016.

[8] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[10] X. Li, W. Wang, X. Hu, and J. Yang. Selective kernel networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 510–519, 2019.

[11] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.

[12] R. T. Mullapudi, W. R. Mark, N. Shazeer, and K. Fatahalian. Hydranets: Specialized dynamic architectures for efficient inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8080–8089, 2018.

[13] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020.

[14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[15] P. Sun, X. Jin, W. Su, Y. He, H. Xue, and Q. Lu. A visual inductive priors framework for data-efficient image classification. In *European Conference on Computer Vision*, pages 511–520. Springer, 2020.

[16] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[17] M. Tan and Q. V. Le. Efficientnetv2: Smaller models and faster training. *arXiv preprint arXiv:2104.00298*, 2021.

[18] X. Wang, L. Lian, Z. Miao, Z. Liu, and S. X. Yu. Long-tailed recognition by routing diverse distribution-aware experts. *arXiv preprint arXiv:2010.01809*, 2020.

[19] R. Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

[20] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6023–6032, 2019.

[21] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer. Scaling vision transformers. *arXiv preprint arXiv:2106.04560*, 2021.

[22] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[23] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, Z. Zhang, H. Lin, Y. Sun, T. He, J. Mueller, R. Manmatha, et al. Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955*, 2020.

[24] R. Zhang. Making convolutional networks shift-invariant again. *arXiv preprint arXiv:1904.11486*, 2019.