Collaboration of Experts: Achieving 80% Top-1 Accuracy on ImageNet with 100M FLOPs

Yikang Zhang
Huawei
zhangyikang7@huawei.com

Zhuo Chen
Huawei
chenzhuo43@huawei.com

Zhao Zhong
Huawei
zorro.zhongzhao@huawei.com

Abstract

In this paper, we propose a Collaboration of Experts (CoE) framework to pool together the expertise of multiple networks towards a common aim. Each expert is an individual network with expertise on a unique portion of the dataset, which enhances the collective capacity. Given a sample, an expert is selected by the delegator, which simultaneously outputs a rough prediction to support early termination. To fulfill this framework, we propose three modules to impel each model to play its role, namely weight generation module (WGM), label generation module (LGM) and variance calculation module (VCM). Our method achieves the state-of-the-art performance on ImageNet, 80.7% top-1 accuracy with 194M FLOPs. Combined with PWLU activation function and CondConv, CoE further achieves the accuracy of 80.0% with only 100M FLOPs for the first time. More importantly, our method is hardware friendly and achieves a 3~6x speedup compared with some existing conditional computation approaches.

1 Introduction

From small systems like organic cells to complicated human society, the accomplishment of various tasks relies on the collaboration of multiple individuals. It can pool together the skills and creativity of each individual towards a common aim, increasing the chance of success. This is a consensus since long ago: as early as 1624, John Donne has popularised the phrase ‘No man is an island’. Similarly, in machine learning society, researchers have also studied the collaboration of different models for decades. Just like human beings, different models naturally have diverse properties. For example, neural networks with more layers in early stages can extract low-level feature more delicately, otherwise, high-level semantic feature is improved [1]. It is still the truth even if models share the same architecture because the random seed yields a great deal of randomness for the optimization [2]. Therefore, the outcome on a given task should be improved if the expertise of multiple models is combined.

There are many approaches for model collaboration, among which ensemble learning [2, 3, 4] is a popular one. It uses a consensus scheme to decide the collective result by vote. However, it requires n forward passes, leading to a significant runtime cost. MIMO [5] draws inspiration from model sparsity [6] and tries to ensemble several subnetworks within one regular network. It only needs one single forward pass of the regular network but is incompatible with compact models. Conditional computation methods [7, 8, 9, 10] alleviates this problem via delegation scheme, i.e. assigning one or several, rather than all, models conditionally to make the prediction. Some recently proposed conditional computation methods [11, 12, 7] achieve remarkable performance with low computation cost based on dynamic convolution. Nonetheless, they are not so hardware-friendly. Overall, the
above methods usually have high memory access cost and low degree of parallelism, which obviously increases the real latency \[13\].

Motivated by this, we propose the Collaboration of Experts (CoE) framework to both eliminate the need for multiple forward passes and keep hardware friendly. Each expert is an individual model with expertise on a unique portion of the dataset. The delegator will give a rough prediction and expert selection simultaneously. Only when the rough prediction is unreliable \[14, 15\], the selected expert makes the refined prediction. Moreover, we only need to load the selected expert into memory, thus keep the ratio of memory access cost to FLOPs a constant. By contrast, dynamic convolution methods \[11, 7\] need to load a large number of parameters, namely basis models or experts, to synthesize the input-dependent ones. It enlarges the memory access cost and reduces the degree of parallelism, resulting in a significant deceleration.

As shown in Fig. 1, we propose a novel training algorithm and optimization strategy for CoE to impel each model to play its role. To make each expert focus on the unique portion of the dataset assigned to itself, we propose the weight generation module (WGM) to reweight losses of each expert. Taking expert selection as input, WGM will provide loss weight for each expert. The label generation module (LGM) is used to generate the selection label which indicates the suitable expert. After that, the delegator is optimized based on the selection label with weighted cross-entropy loss \[\text{Loss}_S\], where the weight \(v\) is obtained from the variance calculation module (VCM). Specifically, if there exist multiple suitable experts for the given input, the value of \(v\) will get smaller.

Experimental results on ImageNet classification task \[16\] demonstrate the superiority of our method. The proposed method achieves 78.2/80.7\% top-1 accuracy with only 100/194M FLOPs, while the ensembled models can only achieve 79.6\% accuracy with 920M FLOPs. Compared with dynamic network approaches \[12, 7, 8\], our method is more hardware friendly. We not only outperform the state-of-the-art dynamic method BasisNet (80.0\% accuracy with 198M FLOPs) \[7\], but also achieve a 3.1x speedup on hardware compared with it. Moreover, our method can be equipped with CondConv \[11, 12\] and further improve the accuracy to 79.2/81.5\% with 102/214M FLOPs. More surprisingly, we achieve the accuracy of 80.0\% with only 100M FLOPs for the first time if Piecewise Linear Unit (PWLU) activation function \[17\] is further used.

The contributions of this paper can be summarized as follows:

- We propose a collaboration framework named Collaboration of Experts (CoE) and demonstrate its effectiveness. It can lead to outstanding performance with little computation cost. More importantly, this framework is hardware-friendly and can achieve real speedup.

- We propose a novel training algorithm and optimization strategy for CoE. The core insight is to promote diversity within experts by distributing the expertise of each one over different portions of the dataset.

- We update the state-of-the-art on ImageNet for mobile setting, achieving 80.0\% top-1 accuracy with only 100M FLOPs for the first time.
2 Related Work

2.1 Ensemble Learning

Ensemble learning \cite{2} aims at combining the predictions from several models to get a more robust one. Some recently proposed literatures \cite{3,4} demonstrate that significant gains can be achieved with negligible additional parameters compared to the original model. However, these methods still require multiple (typically, 4-10) forward passes for prediction, leading to a significant runtime cost. Differently, our method utilizes a delegator to select only one expert for the refined prediction, thus at most two forward passes are needed. MIMO \cite{5} draws inspiration from model sparsity \cite{6} and holds the view that multiple independent subnetworks can be concurrently trained within one regular network because of the heavy parameter redundancy. Therefore, those subnetworks can be ensembled with single forward pass of the regular model. However, MIMO cannot be applied to compact models which have already been pruned or the ones constructed by AutoML methods \cite{18,19,20,21,22}. It is because these models are compact enough and have few redundant parameters. By contrast, our method is free from the compactness of experts and compatible with various models.

2.2 Dynamic Networks

Dynamic network methods can achieve high performance with low computation cost by conditionally varying the network parameters \cite{11,12,7,23} or network architectures \cite{8,24}. HD-CNN \cite{9} and HydraNet \cite{25} select branches based on the category, they cluster all categories into n groups, where n is the number of branches. While our method learns the model selection pattern automatically, it can be based on any property, rather than limited to the category. MoE \cite{10} and Switch Transformer \cite{26} enable the direct training of Router by scaling the output feature of experts with the predicted gate-values of Router. These methods aim at conditionally selecting a specific layer or block. Differently, our method can take more advantage of conditional computation as the selection of whole network making every parameter input-dependent. The recently proposed Dynamic Convolution methods \cite{11,12,23} share the same idea and achieve remarkable performance with low FLOPs but high latency. It is because these methods need to load many basic models or experts to synthesize the dynamic parameters, causing high memory access cost and low degree of parallelism \cite{13}. By contrast, our method only needs to load the selected expert into memory, avoiding these problems. Finally yet importantly, batch processing is an important method to enhance the degree of parallelism. Because of the input-dependent parameters \cite{11,12,7} or architectures \cite{8,24}, these methods cannot process samples in batch. Differently, our method supports batch processing because the number of experts is limited and each one of them corresponds to many test samples.

3 Method

In this paper, we propose the CoE framework which pools together the expertise of each network towards a common aim. As shown in Fig. 1, the framework includes a delegator and n experts, a total of n + 1 individual neural networks. To make the best of limited capacity, we encourage each expert to have expertise on a unique portion of the dataset. Meanwhile, the delegator, shown in Fig. 2, is adopted for expert selection. To take advantage of early termination \cite{14,15}, the delegator also outputs a rough prediction. By varying the threshold, we can achieve anytime prediction \cite{27}. Since the inference of delegator is conducted all the time, we prefer to make it more lightweight than expert. It consists of three modules: feature extractor, task predictor and expert selector. Based on the feature derived from the extractor, task predictor and expert selector output the probabilities for classification and expert selection respectively.

There are three core modules in the framework, including weight generation module (WGM), label generation module (LGM) and variance calculation module (VCM). WGM is used to reweight the losses for n experts, thus nurtures the diversity among experts. LGM generates the selection label, based on which the delegator is optimized. VCM adjusts the weight for loss of expert selection: if there exist multiple suitable experts for the given input, the weight will get smaller. We describe these modules and training strategies comprehensively in the following subsections. The number of samples and experts are denoted as m and n respectively throughout this section.

Figure 2: The architecture of the delegator.
3.1 Weight Generation Module (WGM)

Our proposed method achieves remarkable performance by the collaboration of multiple networks. To maximize the collective capacity, we encourage each expert to focus on a unique portion of the dataset. To accomplish this goal, we propose WGM to reweight losses of different experts.

The expert selection can be denoted as a probability matrix $P_{m \times n} \in \mathbb{R}^{m \times n}$, whose element $P_{j,k} \in [0, 1]$ represents the probability of assigning the j-th sample to the k-th expert. WGM takes $P_{m \times n}$ as input and outputs a weight matrix $W_{m \times n}$. WGM firstly obtain an assignment matrix $A_{m \times n}$, which is a binary matrix with one-hot row vectors, then get $W_{m \times n}$ via smoothing and normalizing. For example, $A_{j,k} = 1$ means the delegator assigns the j-th sample to the k-th expert, while $W_{j,k}$ should be larger.

Given that the main concern of this paper is not the optimization of network architectures, we can suppose they have similar accuracy/FLOPs trade-off, thus make an assumption about the experts:

**No Superiority Assumption**  The overall accuracy/FLOPs trade-off for each expert is same, only the one specific to a certain sample is various.

Based on this assumption, the number of samples assigned to each expert should be same, i.e. the assignment matrix $A_{m \times n}$ needs to satisfy $\sum_j A_{j,k} = m/n$. Thus, $A_{m \times n}$ can be determined by solving the following problem:

$$
\min \sum_{j,k} -P_{j,k} * A_{j,k}
$$

s.t. $A_{j,k} \in \{0, 1\}$

$$
\sum_k A_{j,k} = 1
$$

$$
\sum_j A_{j,k} = \frac{m}{n}
$$

This problem can be modeled as the balanced transportation problem (BTP, [28]), where each sample $x_j$ is a supply source with a supply of one, each expert $E_{xpert_k}$ is a demand source with a demand of $m/n$. $-P_{j,k}$ is the per-unit transportation cost from $x_j$ to $Spcl_k$. We solve this problem via Vogel approximation method (VAM, [28]), which is a short-cut approach to solve BTP and invariably obtains a very good solution. It is introduced in appendix A.1.

$A_{m \times n}$ is a sparse matrix with many zero elements. It will cause an overfitting problem if the weight matrix $W_{m \times n}$ is acquired by directly normalizing $A_{m \times n}$. Thus, we firstly smooth $A_{m \times n}$ to $\overline{A}_{m \times n}$:

$$
\overline{A}_{j,k} = \begin{cases} 
\alpha + \frac{1-\alpha}{n}, & \text{if } A_{j,k} = 1 \\
\frac{1-\alpha}{n}, & \text{if } A_{j,k} = 0 
\end{cases}
$$

where, $\alpha \in [0.2, 0.8]$ grows linearly with the training going on. Then, $W_{m \times n}$ is obtained by normalizing $\overline{A}_{m \times n}$ with the coefficient $Z = \sum_j \overline{A}_{j,k} = \frac{m}{n}$:

$$
W_{j,k} = \frac{\overline{A}_{j,k}}{Z}.
$$

3.2 Label Generation Module (LGM)

LGM is used to generate the selection label which indicates the suitable expert. To achieve this goal, we need to measure the confidence of each expert firstly. Thus, we leverage the task labels to obtain the True Class Probability (TCP, [29]), which is an effective criterion for model confidence. Given the input $x_j$, task label $y_j$ and the expert $E_{xpert_k}$, TCP can be obtained by:

$$
TCP_{j,k} = P(Y = y_j|x_j, E_{xpert_k}),
$$

here, $Y$ denotes the predicted category.

However, TCP can only reflect precision and have nothing to do with FLOPs. Thus, we can directly use TCP value as the suitability. We denote the suitability matrix as $S_{m \times n}$. Taking FLOPs into account, the element values for each column vector of $S_{m \times n}$ should satisfy the same distribution according to **No Superiority Assumption**. To achieve this, we can individually standardize the
TCP values for each expert. Denoting the mean value and standard deviation of TCP for $Spcl_k$ as $\text{Mean}(TCP_{:,k})$ and $\text{Std}(TCP_{:,k})$ individually, then the elements of $S_{m \times n}$ satisfy:

$$S_{j,k} = \frac{TCP_{j,k} - \text{Mean}(TCP_{:,k})}{\text{Std}(TCP_{:,k})}.$$  \hspace{1cm} (5)

We can denote the output of LGM as a binary matrix $L_{m \times n}$, thus each row of $L_{m \times n}$ is a selection label. Because we aim at maximizing the total suitability, $L_{m \times n}$ can be determined by solving the following problem:

$$\min \sum_{j,k} S_{j,k} * L_{j,k}$$

s.t. $L_{j,k} \in \{0, 1\}$

$$\sum_{k} L_{j,k} = 1$$

$$\sum_{j} L_{j,k} = \frac{m}{n}$$ \hspace{1cm} (6)

This problem can also be modeled as BTP, and solved via VAM as described in section 3.1.

### 3.3 Variance Calculation Module (VCM)

The expert selection output by the delegator involves probabilities for the n experts. Based on LGM, we can obtain a one-hot label $L_{j,:}$, which indicates the suitable expert for the j-th sample. Then we can optimize the delegator by the cross-entropy loss $\text{Loss}_{CE}$ calculated based on $L_{j,:}$. This enables the delegator to assign experts properly. It should be noted that for some samples there may exist multiple suitable experts. In this case, it only causes little to no drop in accuracy if assign samples to other experts rather the most suitable one. Considering this, the weight of $\text{Loss}_{CE}$ can be set smaller. Therefore, we expect to dynamically adjust the weight of $\text{Loss}_{CE}$ and propose a module named Variance Calculation Module (VCM).

The output of VCM can be denoted as $\{V_j|j = 1, \ldots, m\}$, which are the weights for $\text{Loss}_{CE}$ over different samples. To obtain these values, VCM firstly calculates the suitability matrix $S_{m \times n}$ via Equation. 5 Then variance for elements of each row vector $S_{j,:}$ can reflect the diversity for the experts. A small diversity means $x_j$ can be assigned to an arbitrary expert with little to no drop of accuracy, thus $V_j$ can be smaller. Therefore, VCM determines the value of $V_j$ based on the normalized standard deviation of $S_{j,:}:

$$V_j = \frac{\text{Std}(S_{j,:})}{\sum_i \text{Std}(S_{i,:})}.$$ \hspace{1cm} (7)

### 3.4 Training method

The training procedure of our proposed method is composed of two stages. During the first stage, only the feature extractor and task predictor of delegator are trained to minimize the regular cross-entropy loss $\text{Loss}_{p}$. After that, we fix these two modules and jointly train the expert selector and the experts with the following loss:

$$\text{Loss}_{\text{Total}} = \eta * \text{Loss}_{S} + \text{Loss}_{T},$$ \hspace{1cm} (8)

here, the hyperparameter $\eta$ is set as 0.8.

$\text{Loss}_{S}$ is used to optimize the expert selector. Based on the selection label $L_{j,:}$ obtained from LGM, we can get the cross-entropy loss $\text{Loss}_{CE}^{S}$ for the j-th sample. Then $\text{Loss}_{S}$ is determined by the weighted sum of $\{\text{Loss}_{CE}^{S}|j = 1, \ldots, m\}$ with weights $\{V_j|j = 1, \ldots, m\}$:

$$\text{Loss}_{S} = \sum_j V_j * \text{Loss}_{CE}^{S}.$$ \hspace{1cm} (9)

$\text{Loss}_{T}$ is used to optimize the experts. Based on the task labels, we can get $m \times n$ cross entropy losses $\{\text{Loss}_{j,k}|j = 1, \ldots, m; k = 1, \ldots, n\}$ for the experts. Then $\text{Loss}_{T}$ is obtained by the weighted sum of them with weights $W_{j,k}$ output by WGM.
Each of our experiments includes either four or sixteen experts in this paper. When using four experts, the training method is the same as described. However, we will meet the slow-convergence problem if the number of experts is sixteen. It is because most of the samples, specifically \( \frac{15m}{16} \), have tiny weights for each expert according to WGM. To alleviate this problem, we propose a strategy which is described in Appendix A.2.

4 Experiments

In this section, we conduct experiments on ImageNet classification task [16]. After comparing with some popular methods, we analyze the effect of expert number and early termination respectively. Then, we compare with some existing model collaboration methods. Finally, we try to analyze the reasonability of the learned expert selection pattern. Results are implemented with the following setting unless otherwise stated.

4.1 Implementation Details

We conduct experiments with two settings: **CoE-Small** and **CoE-Large**. For CoE-Small, we take TinyNet-E [30] with 24M FLOPs as the feature extractor of delegator by removing the last fully connected layer. We use OFA-110 [22] with 110M FLOPs as the expert. For the CoE-Large, MobileNetV3-Small [31] with 56M FLOPs is adopted to construct the delegator by analogy. We use OFA-230 as the experts. To combine with CondConv [11], we simply replace the convolutions within each inverted residual block of the experts with CondConv (\( \text{experts} = 4 \)). To take advantage of PWLU activation function [17], we replace all activation layers except those that have tiny input feature maps as illustrated in [17]. Models are trained using SGD optimizer with 0.9 momentum. We use a mini-batch size of 4096, and a weight decay of 0.00002. Cosine learning rate decay is adopted and the number of training iterations is 313000. We use fixed auto-augment [32] as well. Inspired by BasisNet [7], we use knowledge distillation [33] with EfficientNet-B2 [34, 35] as the teacher. The learning rate is 0.8/1.6 for CoE-Small/Large and dropout rate is 0.2. The stochastic depth [36] is used for models except for TinyNet-E with a survival probability of 0.8.

4.2 Results and Analysis

4.2.1 Accuracy and Computation Cost

We compare with some efficient networks in Table 1 and Fig. 3, where statistics on referenced baselines are directly cited from original papers. Our method achieves 78.2% and 80.7% top-1 accuracy with the averaged FLOPs as 100M and 194M respectively. Compared with OFA [22], our method reduces the FLOPs from 230M to 100M and from 595M to 194M respectively, with better top-1 accuracy. Compared with EfficientNet-B1 with noisy student training [34], our method also reduces the FLOPs by 3.6x while improving the accuracy by 0.5%. Though dynamic networks like GFNet [37], CondConv [12] and BasisNet [7] are more efficient than traditional networks, our method still has significantly higher accuracy with smaller FLOPs. Compared with these approaches, our method improves the accuracy by 2.2/2.4/0.7% respectively. When combined with CondConv, we can achieve 79.2% and 81.5% top-1 accuracy with only 102M and 214M FLOPs respectively, which indicates our method is complementary to dynamic networks like CondConv. On the contrary, as CondConv and BasisNet share similar essence, namely using a group of basis to dynamically synthesize the input-dependent convolution kernel, the combination of them only arouses little collaborative benefit with the top-1 accuracy of only 80.5%. More surprisingly, we achieve the accuracy of 80.0% with only 100M FLOPs for the first time by further making use of PWLU activation function [17].

4.2.2 Speed on The Hardware

Compared with conditional computation methods [12, 7], our method is more hardware friendly. To verify the advantage, we also analyze the real latency on the hardware. The experiments are conducted on CPU platform (Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz) with PyTorch version as 1.8.0. When the mini-batch size is larger than one, our method firstly uses the delegator to partition the ImageNet validation set into \( n + 1 \) groups, where one group is classified based on rough prediction and others rely on the refined prediction of the corresponding expert.

We report the averaged latency on the ImageNet validation set in Table 2. From the table, we notice the discrepancy between the FLOPs and real speed. For example, OFA-230 has 1.6x FLOPs compared...
Figure 3: Visualization for Table 1, which compares our method (CoE) with other efficient networks on ImageNet. By varying the threshold of early-termination, CoE can get a series of results with different accuracies and FLOPs. Specifically, we accept the rough prediction of delegator if its Maximum Class Probability (probability of the predicted class) is larger than the threshold, otherwise, adopt the selected expert for a refined prediction. Statistics on referenced baselines are cited from original papers: BasisNet† [7], Dy-MobileNetV3-Large† [11], CC-MnasNet-A1† [12], GFNet† [37], WeightNet† [38], GhostNet† [39], TinyNet† [30], OFA† [22], EfficientNet (Noisy Student)† [35]. † indicates conditional computation approach, ‘CC’ and PWLU means CondConv and PWLU activation function [17].

4.2.3 Effect of Expert Number

We analyze the number of experts in this section, including 1, 4, and 16 experts. The results are shown in Table 3. Using one expert brings little improvement compared with the original model. When increasing the number of experts, the accuracy becomes 2.0% better with four experts and 3.0% better with sixteen experts. It demonstrates our method can make full use of multiple experts, leading to a large collaborative benefit. What’s more, the accuracy also reaches 79.9% by combining CondConv with OFA-230. In this manner, our method can further enhance the accuracy to 80.8/81.5% with 4/16 experts.

4.2.4 Effect of Early Termination

The original OFA-230 has 77.9% top-1 accuracy with 230M FLOPs. We can introduce a MobileNetV3-Small to conduct the early termination, i.e. accepting the prediction of MobileNetV3-Small if its Maximum Class Probability (probability of the predicted class) is larger than the given threshold, otherwise adopt OFA-230 for a reliable prediction. By varying the threshold of early termination, we can get a series of accuracies and FLOPs and we plot them in Fig. 4. We can see that the accuracy becomes 78.0% with 220M FLOPs. This indicates the computation cost brought by MobileNetV3-Small can be eliminated via early termination strategy. Inspired by this, we expect to eliminate the computation cost brought by the delegator via early termination as well. From Fig. 4,

7
Table 1: Comparison with some popular networks on ImageNet. Statistics on referenced baselines are cited from original papers. † indicates the conditional computation approach.

| Method                  | FLOPs | TOP-1 Acc |
|-------------------------|-------|-----------|
| WeightNet† [38]         | 141M  | 72.4%     |
| GhostNet 1.0x [39]      | 141M  | 73.9%     |
| MobileNetV3-Large [31]  | 219M  | 75.2%     |
| GhostNet 1.3x [39]      | 226M  | 75.7%     |
| OFA-230 [31]            | 230M  | 76.9%     |
| TinyNet-A [30]          | 339M  | 77.7%     |
| GFNet† [37]             | 400M  | 78.5%     |
| CondConv-EfficientNet-B0† [12] | 413M | 78.3% |

Table 2: The CPU latency for different methods. Accuracy of referenced baselines are cited from original papers. † indicates the conditional computation approach.

| Models                          | CPU Latency/Instance (ms) | FLOPs | Accuracy |
|---------------------------------|---------------------------|-------|----------|
| MobileNetV3-Small [31]          | 14.77                      | 56M   | 67.4%    |
| GhostNet 1.0x [39]              | 39.91                      | 141M  | 73.9%    |
| TinyNet-B [30]                  | 34.58                      | 202M  | 75.0%    |
| GhostNet 1.3x [39]              | 43.94                      | 226M  | 75.7%    |
| OFA-230 [22]                    | 33.52                      | 230M  | 76.9%    |
| EfficientNet-B0 [35]            | 700M                       | 80.2% |
| BasisNet† [7]                   | 290M                       | 80.3% |
| FBNetV3-E [40]                  | 752M                       | 80.4% |
| BasisNet + CondConv† [12]       | 308M                       | 80.5% |
| CoE-Large†                      | 194M                       | 80.7% |
| CoE-Large + CondConv†           | 214M                       | 81.5% |

we can see, it does reduce the computation cost by 60/66M FLOPs, demonstrating the effectiveness of early termination.

4.2.5 Analysis of the Generalization

As described in paper [41], the validation set labels have a set of deficiencies that makes recent progress on the ImageNet classification benchmark suffer from overfitting to the artifacts. To verify the generalization, we use their Reassessed Labels (ReaL) to re-evaluate our method. The results are

Table 3: Comparison among different number of experts.

| Method                  | Experts | FLOPs | TOP-1 Acc |
|-------------------------|---------|-------|-----------|
| OFA-230                 | -       | 230M  | 77.9%     |
| CoE-Large               | 1       | 220M  | 78.0 (0.1↑)% |
|                         | 4       | 220M  | 79.9 (2.0↑)% |
|                         | 16      | 220M  | 80.9 (3.0↑)% |
| CondConv-OFA-230        | -       | 242M  | 79.9%     |
| CoE-Large + CondConv    | 1       | 214M  | 79.9 (0.0↑)% |
|                         | 4       | 214M  | 80.8 (0.9↑)% |
|                         | 16      | 214M  | 81.5 (1.6↑)% |
Figure 4: The accuracy and FLOPs for different methods. ’ET’ means Early Termination and ’CC’ indicates CondConv.

Table 4: ReaL and original top-1 accuracies. CC means CondConv.

| Method          | FLOPs  | ReaL Acc. | Ori. Acc. |
|-----------------|--------|-----------|-----------|
| OFA-595         | 595M   | 86.0%     | 80.0%     |
| S4L MOAM        | 4B     | 86.6%     | 80.3%     |
| ResNeXt-101     | 16B    | 85.2%     | 79.2%     |
| ResNet-152      | 11B    | 84.8%     | 78.2%     |
| CoE-Large       | 194M   | 86.5%     | 80.7%     |
| CoE-Large + CC  | 214M   | 86.9%     | 81.5%     |

Table 5: Comparison with model ensemble.

| Method          | FLOPs   | Acc.   |
|-----------------|---------|--------|
| Seed1           | 230M    | 78.1%  |
| Seed2           | 230M    | 78.0%  |
| Seed3           | 230M    | 78.1%  |
| Seed4           | 230M    | 78.0%  |
| Ensemble        | 920M    | 79.6%  |
| CoE-Large       | 4 Experts | 220M   | 79.9% |
| CoE-Large       | 16 Experts | 220M   | 80.9% |

shown in Table. [41]. It can be seen that our method still has a remarkable performance, achieving higher accuracy than the compared methods with significantly smaller FLOPs.

4.3 Comparison with Other Model Collaboration Approaches

4.3.1 Comparison with Regular Model Ensemble

We train four OFA-230 models with different initialization seeds. The results are shown in Table. 5. The random seed only causes minor variety in accuracy. However, an improvement of 1.6% is still achieved via ensemble. It is because these models fall into different local minima, yielding the diversity of output. Compared with the ensemble, our method achieves 0.3/1.3% higher accuracies. It indicates our method can realize more potential of individual models. Besides, our method keeps the computation cost constant, while model ensemble increases the computation cost by four times.

4.3.2 Comparison with Model Selection Method

HD-CNN [9] and HydraNets [25] select branches based on the category. MoE [10] and Switch Transformer [26] enable the direct training of Router by scaling the output feature of experts with gate-values predicted from Router. Despite these methods are originally designed to conditionally select a specific layer or block, we apply them to the expert selection.

To select the expert based on category, the categories should be partitioned into n groups, where n is the number of experts. We try two schemes: random or clustering-based partition [25]. Then, an expert can be selected according to the rough prediction of delegator. During the training procedure, we also reweight losses of each expert based on the assignment matrix $A_{m \times n}$ with Equation 2 and $A_{m \times n}$ is obtained directly based on the rough prediction. We can enable the direct training of expert selector [10, 26] as well. It is achieved by scaling the loss of each expert, rather than the output feature of each branch as the original paper. While our method optimizes the expert selector via cross-entropy loss.

The results with 4 experts are shown in Table 6. It can be seen that our method outperforms the compared methods, demonstrating a better collaboration pattern is learned.
Table 6: Comparison with other conditional computation methods.

| Method                        | FLOPs | Top-1 Acc. |
|-------------------------------|-------|------------|
| Category-Based                |       |            |
| Random Partition              | 220M  | 78.3%      |
| Clustering-Based Partition    | 220M  | 77.5%      |
| Selector-Based                |       |            |
| Direct Training               | 220M  | 78.7%      |
| CE-Based Training (Ours)      | 220M  | 79.9%      |

4.4 Reasonability of the Learned Expert Selection Pattern

Considering TCP can measure the complexity of a given sample, we conduct an experiment to analyze the relationship between sample complexity and expert selection. If the experts have different architectures, it is reasonable to assign easy samples to smaller experts and complex samples to heavier experts. To verify this, we take four architectures searched via OFA [22] as the experts, i.e. OFA-110, OFA-163, OFA-230 and OFA-595. The delegator is also MobileNetV3-small as described in section 4.1. We obtain the TCP value of each sample on the validation set based on the delegator as well. As shown in Fig. 5, we count the selection probability for each expert at different TCP values. It meets our expectation that the selection probability for smaller models increases with the increase of TCP.

![Figure 5: The selection probabilities for each expert at different TCP value.](image)

We also analyze the selection pattern when all experts share the same architecture, and some interesting laws are found. Taking images predicted as ‘meat market’ by the delegator as an example, we find the ones containing humans are most likely to be assigned to the fourth expert. Specifically, 27 images are assigned to the fourth expert, among which 22 images contain humans with a ratio of 81.5%. By contrast, among the 32 images that are assigned to the other experts, only 7 images contain humans with a ratio of 21.9%. This indicates our method learns the expert selection pattern automatically, it can be based on properties other than the category. More analysis is given in Appendix B.1.

5 Conclusion

We propose a Coe framework to pool together the expertise of multiple networks towards a common aim. Experiments in this paper demonstrate the superiority of our method on both accuracy and real speed. We also analyze the collaboration pattern and find it has interpretability. In the future, the COE will be extended to the trillion parameters level. Meanwhile, we will try to implement CoE to more tasks and verify its compatibility with quantification and other technologies. Besides, the CoE can be conducted to solve the problems of lifelong learning by updating experts.

6 Acknowledgment

We thank our colleagues from 2012CMI-Media I&A Lab, who provided expertise that greatly assisted the research. We are grateful to Yucong Zhou and Zezhou Zhu for providing the implementation of PWLU activation function.
A Extra Details for Method

A.1 Introduction of Vogel Approximation Method (VAM)

In weight generation module (WGM) and label generation module (LGM), we need to solve the balanced transportation problem (BTP,\ref{28}) via Vogel approximation method (VAM,\ref{28}). We will introduce it in this section.

The BTP involved in WGM and LGM has \(m\) supply sources, each of which is denoted as Silo\(_j\) with a supply of one, as well as \(n\) demand sources, each of which is denoted as Mill\(_k\) with a demand of \(\frac{m}{n}\). \(C_{j,k}\) is the per-unit transportation cost from Silo\(_j\) to Mill\(_k\). Specifically, \(C_{j,k} = -P_{j,k}\) in WGM and \(C_{j,k} = -S_{j,k}\) in LGM. To make it easier, we illustrate this algorithm with a toy example, where the problem is simplified as Fig.\ref{6}(a) with \(m = 4, n = 2\). In the first step, we calculate the penalty cost \(p_{\text{row}_j}\) for each row and \(p_{\text{col}_k}\) for each column of the tableau in Fig.\ref{6}(a). Penalty cost is determined by subtracting the lowest unit cost in the row (column) from the next lowest unit cost. The penalty costs of the respective rows and columns have been marked in red color for clarity in Fig.\ref{6}(b). Since the third row has the largest penalty cost (\(p_{\text{row}_3} = 11\)) and \(C_{3,1}\) is the lowest unit cost of that row, Silo\(_3\) is allocated to Mill\(_1\), i.e. \(A_{3,1} = [1,0]\) in WGM or \(L_{3,1} = [1,0]\) in LGM. Then the corresponding row should be crossed out and the demand of Mill\(_1\) should minus one, if this results in a zero demand, the first column will be crossed out as well. After adjusting penalty cost for each row and column, the tableau becomes Fig.\ref{6}(c), where the changed values are marked in orange. The described procedure will be looped until no rows remained.

Considering the calculation of \(p_{\text{col}_k}\) is much more time-consuming compared with \(p_{\text{row}_j}\) because \(m \gg n\) in WGM and LGM, we modify VAM by only seeking lowest penalty cost among \(\{p_{\text{row}_1}, \ldots, p_{\text{row}_m}\}\). We find this modification makes VAM more efficient while keeps the superiority of the solution. It is because the mechanics of VAM makes it meaningful to take \(p_{\text{col}_k}\) into account only when the demand of Mill\(_k\) is one, which rarely happens. Thus, we adopt this modification to promote efficiency in this paper.

![Figure 6: The Vogel approximation method.](image)

\[\text{Tableau of the transportation problem}\]

\[\text{Penalty costs of respective rows and columns}\]

\[\text{Tableau after allocating Silo}_3\text{ to Mill}_1\]

\[\text{Tableau after allocating Silo}_3\text{ to Mill}_1\]

\[\text{Tableau after allocating Silo}_3\text{ to Mill}_1\]

\[\text{Tableau after allocating Silo}_3\text{ to Mill}_1\]

A.2 A Strategy to Face the Large Number of Experts

Each of our experiments involves either four or sixteen experts. For the four-experts setting, the training method is the same as described. However, we will meet the slow-convergence problem when the number of experts is sixteen. It motivates us to decompose the task into four subtasks, each of which only involves four experts and can be trained with the proposed training method. With this strategy, the number of samples assigned to each expert increases from \(\frac{m}{n}\) to \(\frac{m}{4}\). Because these samples contribute most to the optimization of the expert, the rate of convergence becomes nearly four times faster.

To fulfill task decomposition, we introduce a new module to delegator, named subtask selector as shown in Fig.\ref{7}. The subtask selector is used to allocate the input samples into different subtasks, each of which involves four experts. The expert selector outputs sixteen probabilities, which are partitioned into four groups. For each subtask, only one group of probabilities is visible. The experts
within each subtask and the corresponding weights of the expert selector are jointly optimized. As for the feature extractor, task predictor, and subtask selector, their weights directly derive from the delegator trained with the setting of four experts and then fixed. During this procedure, the weights of subtask selector derive from the expert selector.

Figure 7: The modified architecture of the delegator.

B Extra Analysis for Experiments

B.1 Extra Analysis for the Reasonability of Learned Expert Selection Pattern

We have analyzed the selection pattern when the experts have different FLOPs, here we focus on the case that all experts share the same architecture. We adopt the CoE-Large setting with four experts. Considering many works [9, 25] select branches based on the category, we firstly experiment to observe the relationship between selection pattern and rough prediction of the delegator. We count the probabilities for each expert to be selected within each predicted class on the ImageNet validation set. For better visualization, we cluster the 1000 probability vectors and then plot them on Fig. 8. It can be seen that samples with the same predicted class are assigned to different experts. Therefore, we can conclude the expert is not always selected based on category.

Besides, we further make qualitative analysis on the ImageNet validation set and find some interesting patterns. For example, we find that images predicted as ‘meat market’ are most likely to be assigned to the fourth expert if humans are contained. We show those images in Fig. 9. It can be seen, 27 images are assigned to the fourth expert, among which 22 images contain humans with a ratio of 81.5%. By contrast, among the 32 images that are assigned to the other experts, only 7 images contain humans with a ratio of 21.9%. This indicates our method learns the expert selection pattern automatically, it can be based on properties other than the category.
Figure 8: The selection probabilities for each expert within each predicted class. The 1000 probability vectors are clustered for better visualization.

Samples assigned to the first expert:

Samples assigned to the second expert:

Samples assigned to the third expert:

Samples assigned to the fourth expert:

Figure 9: The images that are predicted as ‘meat market’ by the delegator. They are partitioned into four groups based on which expert is selected. The red border indicates humans are contained, green border indicates humans are not contained.
References

[1] Yikang Zhang, Jian Zhang, and Zhao Zhong. Autobss: An efficient algorithm for block stacking style search. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 10259–10270. Curran Associates, Inc., 2020.

[2] Lars Kai Hansen and Peter Salamon. Neural network ensembles. IEEE transactions on pattern analysis and machine intelligence, 12(10):993–1001, 1990.

[3] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. arXiv preprint arXiv:2002.06715, 2020.

[4] Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. arXiv preprint arXiv:2006.13570, 2020.

[5] Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew M Dai, and Dustin Tran. Training independent subnetworks for robust prediction. arXiv preprint arXiv:2010.06610, 2020.

[6] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv:1803.03635, 2018.

[7] Mingda Zhang, Chun-Te Chu, Andrey Zhmoginov, Andrew Howard, Brendan Jou, Yukun Zhu, Li Zhang, Rebeca Hwa, and Adriana Kovashka. Basenets: Two-stage model synthesis for efficient inference. arXiv preprint arXiv:2105.03014, 2021.

[8] An-Chieh Cheng, Chieh Hubert Lin, Da-Cheng Juan, Wei Wei, and Min Sun. Instanets: Instance-aware neural architecture search. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 3577–3584, 2020.

[9] Zhicheng Yan, Hao Zhang, Robinson Piramuthu, Vignesh Jagadeesh, Dennis DeCoste, Wei Di, and Yizhou Yu. Hdcnn: hierarchical deep convolutional neural networks for large scale visual recognition. In Proceedings of the IEEE international conference on computer vision, pages 2740–2748, 2015.

[10] 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.

[11] Yikang Zhang, Jian Zhang, Qiang Wang, and Zhao Zhong. Dynet: Dynamic convolution for accelerating convolutional neural networks. arXiv preprint arXiv:2004.10694, 2020.

[12] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. arXiv preprint arXiv:1904.04971, 2019.

[13] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In Proceedings of the European conference on computer vision (ECCV), pages 116–131, 2018.

[14] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. arXiv preprint arXiv:1703.09844, 2017.

[15] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In 2016 23rd International Conference on Pattern Recognition (ICPR), pages 2464–2469. IEEE, 2016.

[16] 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.

[17] Yucong Zhou, Zezhou Zhu, and Zhao Zhong. Learning specialized activation functions with the piecewise linear unit. CoRR, abs/2104.03693, 2021.

[18] Z. Zhong, J. Yan, W. Wu, J. Shao, and C. Liu. Practical block-wise neural network architecture generation. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 2423–2432, Los Alamitos, CA, USA, jun 2018. IEEE Computer Society.

[19] Zhao Zhong, Zichen Yang, Boyang Deng, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Blockqnn: Efficient block-wise neural network architecture generation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(7):2314–2328, 2021.
[20] Muyuan Fang, Qiang Wang, and Zhao Zhong. BETANAS: balanced training and selective drop for neural architecture search. *CoRR*, abs/1912.11191, 2019.

[21] Xinyu Zhang, Qiang Wang, Jian Zhang, and Zhao Zhong. Adversarial autoaugment. In *International Conference on Learning Representations*, 2020.

[22] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint [arXiv:1908.09791]*, 2019.

[23] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11030–11039, 2020.

[24] Kun Yuan, Quanquan Li, Dapeng Chen, Aojun Zhou, and Junjie Yan. Dynamic graph: Learning instance-aware connectivity for neural networks. *arXiv preprint [arXiv:2010.01097]*, 2020.

[25] Ravi Teja Mullapudi, William R Mark, Noam Shazeer, and Kayvon Fatahalian. Hydranets: Specialized dynamic architectures for efficient inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8080–8089, 2018.

[26] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint [arXiv:2101.03961]*, 2021.

[27] Alex Grubb and Drew Bagnell. Speedboost: Anytime prediction with uniform near-optimality. In Neil D. Lawrence and Mark Girolami, editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 458–466, La Palma, Canary Islands, 21–23 Apr 2012. PMLR.

[28] Harvey H Shore. The transportation problem and the vogel approximation method. *Decision Sciences*, 1(3-4):441–457, 1970.

[29] Charles Corbière, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez. Addressing failure prediction by learning model confidence. *arXiv preprint [arXiv:1910.04851]*, 2019.

[30] Kai Han, Yunhe Wang, Qiulin Zhang, Wei Zhang, Chunjing Xu, and Tong Zhang. Model rubik’s cube: Twisting resolution, depth and width for tinynets. *CoRR*, abs/2010.14819, 2020.

[31] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019.

[32] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–123, 2019.

[33] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint [arXiv:1503.02531]*, 2015.

[34] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[35] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020.

[36] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016.

[37] Yulin Wang, Kangchen Lv, Rui Huang, Shiji Song, Le Yang, and Gao Huang. Glance and focus: a dynamic approach to reducing spatial redundancy in image classification. *arXiv preprint [arXiv:2010.05300]*, 2020.

[38] Ningning Ma, Xiangyu Zhang, Jiawei Huang, and Jian Sun. Weightnet: Revisiting the design space of weight networks. *CoRR*, abs/2007.11823, 2020.

[39] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. *CoRR*, abs/1911.11907, 2019.

[40] Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Bichen Wu, Zijian He, Zhen Wei, Kan Chen, Yuandong Tian, Matthew Yu, Peter Vajda, et al. Fbnetv3: Joint architecture-recipe search using neural acquisition function. *arXiv preprint [arXiv:2006.02049]*, 2020.
[41] Lucas Beyer, Olivier J. Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. Are we done with imagenet? *CoRR*, abs/2006.07159, 2020.

[42] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[43] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[44] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.