Deep Mixture of Experts via Shallow Embedding

Xin Wang[1] **Fisher Yu**[1] **Lisa Dunlap**[1] **Yi-An Ma**[1] **Ruth Wang**[1] **Azalia Mirhoseini**[2]
Trevor Darrell[1] **Joseph E. Gonzalez**[1]
[1]EECS Department, UC Berkeley [2] Google Brain

Abstract

Larger networks generally have greater representational power at the cost of increased computational complexity. Sparsifying such networks has been an active area of research but has been generally limited to static regularization or dynamic approaches using reinforcement learning. We explore a mixture of experts (MoE) approach to deep dynamic routing, which activates certain experts in the network on a per-example basis. Our novel DeepMoE architecture increases the representational power of standard convolutional networks by adaptively sparsifying and recalibrating channel-wise features in each convolutional layer. We employ a multi-headed sparse gating network to determine the selection and scaling of channels for each input, leveraging exponential combinations of experts within a single convolutional network. Our proposed architecture is evaluated on four benchmark datasets and tasks, and we show that DeepMoEs are able to achieve higher accuracy with lower computation than standard convolutional networks.

1 INTRODUCTION

Increasing network depth has been a dominant trend [11] in the design of deep neural networks for computer vision. However, increased network depth comes at the expense of computational overhead and increased training time. To reduce the computational cost of machine translation models, Shazeer et al. [22] recently explored the design of "outrageously" wide sparsely-gated mixture of experts models, which employs a combination of simple networks, called experts, to determine the output of the overall network. They demonstrated that these relatively shallow models can reduce computational costs and improve prediction accuracy. However, their resulting models needed to be many times larger than existing translation models to recover state-of-the-art translation accuracy. Expanding on this, preliminary work by Eigen et al. [9] demonstrated the advantages of stacking *two* layers of mixture of experts models for MNIST digits classification. With these results, a natural question arises: *can we stack and train many layers of mixture of experts models to improve accuracy and reduce prediction cost without radically increasing the network width?*

In this paper, we explore the design of *deep mixture of experts models (DeepMoEs)* that compose hundreds of mixture of experts layers. DeepMoEs combines the improved accuracy of deep models with the computational efficiency of sparsely-gated mixture of expert models. However, constructing and training DeepMoEs has several key challenges. First, mixture decisions interact across layers in the network requiring joint reasoning and optimization. Second, the discrete expert selection process is non-differentiable, complicating gradient-based training. Finally, the composition of multiple mixture of experts models increases the chance of degenerate (i.e., singular) combinations of experts at each layer.

To address these challenges we propose a general DeepMoE architecture that combines a deep convolutional network with a shallow embedding network and a multi-headed sparse gating network (see Fig. 1). The shallow embedding network terminates in a soft-max output layer that computes *latent mixture weights* over a fixed set of latent experts. These latent mixture weights are then fed into the multi-headed sparse gating networks (with ReLU outputs) to *select* and *re-weight* the channels in each layer

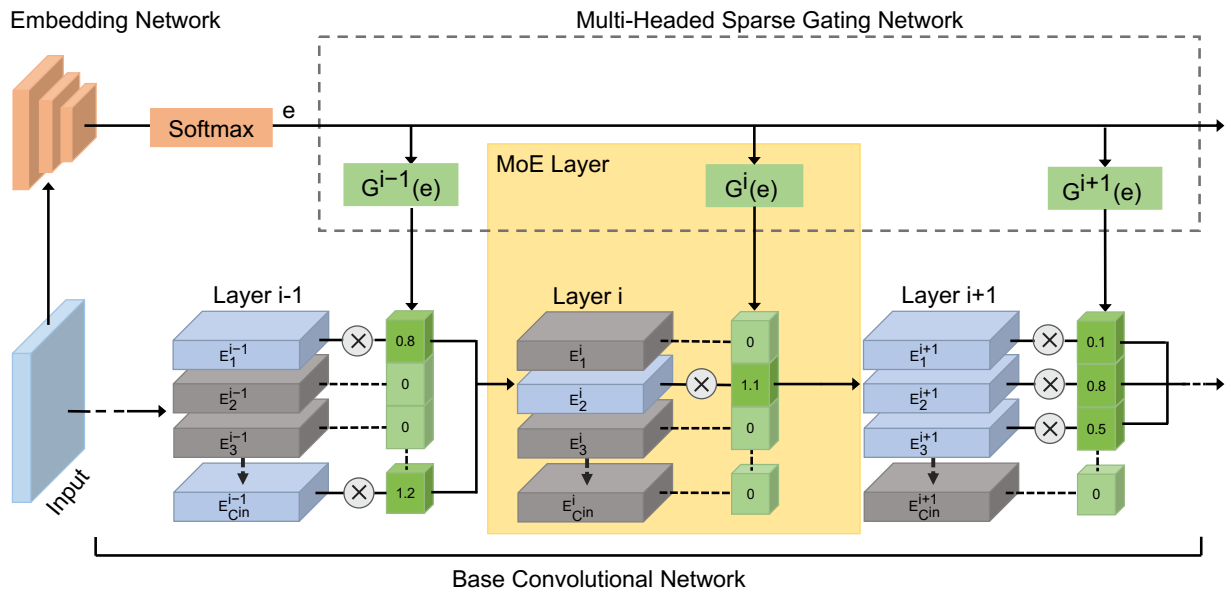


Figure 1: DeepMoE architecture. The input is fed into both the base convolutional network and the shallow embedding network. The embedding network outputs the latent mixture of weights, which is then inputted into the multi-headed sparse gating network, to select the experts to activate for that specific layer. The architecture within a single layer of DeepMoE strongly resembles the traditional Mixture of Experts structure.

of the base convolutional network. We then jointly train the base model, shallow embedding network, and multi-headed gating network with an auxiliary classification loss function over the shallow embedding network and sparse regularization on the gating network outputs to encourage diversity in the latent mixture weights and sparsity in the layer selection. This helps balance expert utilization and keep computation costs low.

Recent work [5] proves that the expressive power of a deep neural network increases super-exponentially with its depth, based on the width. By stacking multiple mixture of expert layers and dynamically generating the sparse channel weights, we analyze in Sec. 4 that DeepMoEs reserve the expressive power of the unsparsified deep networks.

Based on this theoretical analysis, we further propose two variants *wide-DeepMoE* and *narrow-DeepMoE* to improve prediction accuracy while reducing the computational cost compared to standard convolutional networks. For wide-DeepMoEs, we first double the number of channels in the standard convolutional networks and then replace the widened convolutional layers with MoE layers. We examine in experiments that if only half of channels in the widened layers are selected at inference, wide-DeepMoE is able to achieve higher prediction accuracy due to the increase of model capacity while maintaining the same computational cost as the unwidened network. For narrow-DeepMoEs, we directly replace the convolutional layers with MoE layers in the standard convolution networks which generalizes the existing work [18] on dynamic channel pruning and produces models that are more accurate and more efficient than the existing channel pruning literature.

We empirically evaluate the DeepMoE architecture on both image classification and semantic segmentation tasks using four benchmark datasets (i.e.,CIFAR-10, CIFAR-100, ImageNet2012, CityScapes) and conduct extensive ablation study on the gating behavior and the network design in Sec. 5. We find that DeepMoEs achieves the goal of improving the prediction accuracy with reduced computational cost on various benchmarks.

Our contributions can be summarized as: (1) We first propose a novel DeepMoE design which allows the network to dynamically select and execute part of the network at inference. (2) We theoretically analyze that the proposed DeepMoE design preserves the expressive power of a standard convolutional network with reduced computational cost. (3) We further introduce two DeepMoE variants that are more accurate and efficient than the prior methods on different benchmarks.

2 RELATED WORK

Mixture of experts. Jacobs et al. [14] introduced the original formulation of mixture of experts (MoE) models. In this early work, they describe a learning procedure for systems composed of many separate neural networks each devoted to subsets of the training data. Later work [6, 7, 15] applied the MoE idea to classic machine learning algorithms such as support vector machines. More recently, several [22, 10, 1] have proposed MoE variants for deep learning in language modeling and image recognition domains. These more recent efforts to combine deep learning and mixtures of experts have focused on mixtures of deep sub-networks rather than stacking many mixture of expert models. While preliminary work by Eigen et al. [9] explored stacked MoE models, they only successfully demonstrated networks up to depth two and only evaluated their design on MNIST Digits. In contrast, we construct deep models with hundreds of MoE layers based on a shared shallow embedding rather than the layer outputs [9] which makes DeepMoE more suitable to parallel hardware with batch parallelism as the gate decisions are predetermined. We also address several of the key challenges around the design and training of multi-layer MoE models. More recently, the mixture of experts design has been applied in different applications, e.g., video captioning [26], multi-task learning [20], etc.

Conditional computation. Related to mixture of experts, recent works by Bengio et al. [2, 3, 4] explored conditional computation in the context of neural networks which selectively executes part of the network based on the input. They use reinforcement learning (RL) for the discrete selection decisions which are delicate to train while our sparsely-gated DeepMoE design can be embedded into standard convolutional networks and optimized with stochastic gradient descent.

Dynamic channel pruning. To reduce storage and computation overhead, many [17, 12, 19] have explored channel level pruning which removes entire channels at each layer in the network and thus leads to structured sparsity. However, permanently dropping channels limits the network capacity. Bridging conditional computation and channel pruning, recent works [18, 27, 28] have explored dynamic pruning, which use per-layer gating networks to dynamically drop individual channels or entire layers based on the output of previous layers. Therefore the channels to be dropped are dependent on the input, resulting in a more expressive network than one that applies static pruning techniques. Like the work on conditional computation [25], dynamic pruning relies on sample inefficient reinforcement learning techniques to train many convolutional gates. In this work, we generalize the earlier work on dynamic channel pruning by introducing a more efficient shared convolutional embedding and simple ReLU based gates to enable sparsification and feature re-calibration and allowing end-to-end training using stochastic gradient descent.

3 DEEP MIXTURE OF EXPERTS

In this section, we first describe the DeepMoE formulation and then introduce the detailed architecture design and loss function formulation.

3.1 MIXTURE OF EXPERTS

The original mixture of experts [14] formulation combines a set of experts (classifiers), $E_1, ..., E_C$, using a mixture (gating) function G that returns a distribution over the experts given the input $\mathbf{x}$:

$$y = \sum_{i=1}^{C} G(\mathbf{x})_i E_i(\mathbf{x}). \tag{1}$$

Here $G(\mathbf{x})_i$ is the weight assigned to the i^{th} expert E_i. Later work [7] generalized this mixture of experts formulation to a non-probabilistic setting where the gating function G outputs arbitrary weights for the experts instead of probabilities. We adopt this non-probabilistic view since it provides increased flexibility in re-scaling and composing the expert outputs.

3.2 DEEPMOE FORMULATION

In this work, we propose the DeepMoE architecture which extends the standard single-layer MoE model to multiple layers within a single convolutional network. While traditional MoE frameworks focus on the model level combinations of experts, DeepMoE operates within a single model and treats each channel as an expert. The experts in each MoE layer consist of the output channels of the previous convolution operation. In this section, we derive the equivalence between gated channels in a convolution layer and the classic mixture of experts formulation.

A convolution layer with tensor input $\mathbf{x}$ having spatial resolution $W \times H$ and C^{in} input channels, and $C^{\text{in}} \times k \times k \times C^{\text{out}}$ convolutional kernel $\mathbf{K}$ of dimension $k \times k$ can be written as:

$$\mathbf{z}_{o,s,t} = \sum_{i=1}^{C^{\text{in}}} \sum_{u=0}^{k-1} \sum_{v=0}^{k-1} \mathbf{K}_{i,u,v,o} \mathbf{x}_{i,s+u,t+v}, \tag{2}$$

where $\mathbf{z}$ is the $C^{\text{out}} \times W \times H$ output tensor. To construct an MoE convolutional layer we scale the input channels by the gate values $\mathbf{g} \in \mathbb{R}^{C^{\text{in}}}$ for that layer and rearrange terms:

$$\mathbf{z}_{o,s,t} = \sum_{i=1}^{C^{\text{in}}} \sum_{u=0}^{k-1} \sum_{v=0}^{k-1} \mathbf{g}_i \mathbf{K}_{i,u,v,o} \mathbf{x}_{i,s+u,t+v} \tag{3}$$

$$= \sum_{i=1}^{C^{\text{in}}} \mathbf{g}_i \left(\sum_{u=0}^{k-1} \sum_{v=0}^{k-1} \mathbf{K}_{i,u,v,o} \mathbf{x}_{i,s+u,t+v} \right), \tag{4}$$

Defining convolution operator $*$, we can eliminate the summations and subscripts in (4) to obtain:

$$\mathbf{z} = \sum_{i=1}^{C^{\text{in}}} \mathbf{g}_i \mathbf{K}_i * \mathbf{x}_i = \sum_{i=1}^{C^{\text{in}}} \mathbf{g}_i E_i(\mathbf{x}). \tag{5}$$

Thus, we have shown that gating the input channels to a convolutional network is equivalent to constructing a mixture of experts for each output channel. In the following sections, we describe how the gate values $\mathbf{g}$ are obtained for each layer and then present how individual mixture of experts layers can be efficiently composed and trained in the DeepMoE architecture.

3.3 DEEPMOE ARCHITECTURE

DeepMoE is composed of three components: a base convolutional network, a shallow embedding network, and a multi-headed sparse gating network.

The **base convolutional network** is a deep network where each convolution layer is replaced with an MoE convolution layer as described in the previous section. In our experiments we use ResNet [11] and VGG [23] as the base convolutional networks.

The **shallow embedding network** maps the raw input image into a latent mixture weights to be fed into the multi-headed sparse gating network. To reduce the computational overhead of the embedding network, we use a 4-layer (for CIFAR) or 5-layer (for ImageNet) convolutional network with 3-by-3 filters with stride 2 (roughly 2% of the computation of the base models).

The **multi-headed sparse gating network** transforms the latent mixture weights produced by the shallow embedding network into sparse mixture weights for each layer in the convolutional network. The gate for layer l is defined as:

$$G^l(\mathbf{e}) = \texttt{ReLU}(W_g^l \cdot \mathbf{e}), \tag{6}$$

where $\mathbf{e}$ is the output of the shared embedding network $\mathbf{M}$ and W_g^l are the learned parameters which, using the ReLU operation, project the latent mixture weights into *sparse* layer specific gates.

We refer to this gating design as *on demand gating*. The number of experts chosen at each level is data-dependent and the expert selection across different layers can be optimized jointly. Unlike the "noisy Top-K" design in [22], it is not necessary to determine the number of experts at each layer and indeed each layer can learn to use a different number of experts.

3.4 DEEPMOE TRAINING

As with standard convolutional neural networks, DeepMoE models can be trained end-to-end using gradient based methods. The overall goals of the DeepMoE are threefold: (1) achieve high prediction accuracy, (2) lower computation costs, and (3) keep the network highly expressive. Thus, DeepMoE must learn a gating policy that selects a diverse, low-cost mixture of experts for each input. To this end, given the input $\mathbf{x}$ and the target y, we define the learning objective as

$$\mathcal{J}(\mathbf{x}; y) = \mathcal{L}_b(\mathbf{x}; y) + \lambda \mathcal{L}_g(\mathbf{x}) + \mu \mathcal{L}_e(\mathbf{x}; y), \tag{7}$$

$\mathcal{L}_b$ is the cross entropy loss for the base convolutional model, which encourages a high prediction accuracy.

The $\mathcal{L}_g$ term defined:

$$\mathcal{L}_g(\mathbf{x}) = \sum_{l=1}^{L} ||G^l(M(\mathbf{x}))||_1, \tag{8}$$

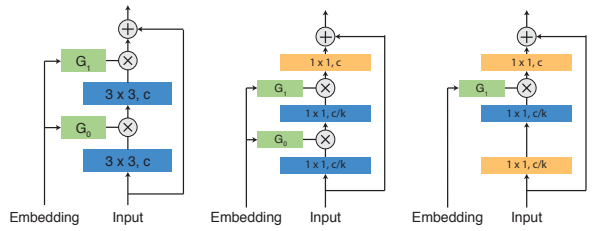
is used to control the computational cost (via the λ parameter) by encouraging sparsity in the gating network.

Finally, we introduce an additional embedding classification loss $\mathcal{L}_e$, which is the cross-entropy classification loss. This encourages the embedding or some transformation of the embedding to be predictive of the class label, preventing the phenomenon of gating networks converging to an imbalanced utilization of experts [22]. The intutition behind this loss construction is that examples from the same class should have similar embeddings and thus similar subsequent gate decisions, while examples from different classes should have divergent embeddings, which would in turn discourage the network from over-using a certain subset of channels.

Because the DeepMoE loss is differentiable we train all three sub-networks jointly using stochastic gradient descent. Once trained, we then set λ and μ to 0 and continue to train a few more epochs to refine the base convolutional network. The full training algorithm is described in Procedure 1.

Procedure 1 Training Algorithm for DeepMoE

repeat
  $e \leftarrow \texttt{EmbeddingNetwork}(x)$
  for i from 1 to L **do**
    $g^l \leftarrow G^l(e)$
  end for
  $output \leftarrow \texttt{BaseNetwork}(x, g^1, \ldots, g^L)$
  $\mathcal{L}_b \leftarrow \texttt{CrossEntropy}(output, y) + \lambda \sum_{l=1}^{L} ||g^l||_1 + \mu\texttt{CrossEntropy}(e, y)$
  Optimize $\mathcal{L}_b$ with SGD
until The model has been trained for n_0 epochs
Freeze `EmbeddingNetwork` and G^l for $l = 1, \ldots, L$, $\lambda \leftarrow 0$, $\mu \leftarrow 0$
Repeat the training loop for another n_1 epochs



(a) Basic Block (b) Bottleneck-A (c) Bottleneck-B

Figure 2: Gated Residual Block Designs. The bottleneck-A and B are used in ResNet-50 on ImageNet and the basic block for in all the other models. In wide-DeepMoE, we increase the number of channels of layers in blue.

4 EXPRESSIVE POWER

The expressive power of deep neural networks is associated with both the width and the depth of the network. Intuitively, the wider the network is, the more expressive power the network has. Cohen *et al.* [5] proves that the expressive power of a deep neural network increases super-exponentially with respect to the network depth, based on the network width. In this section, we demonstrate that due to the dynamic execution nature and the multi-layer stacking design, DeepMoE preserves the expressive power of a standard unsparsified neural network with reduced runtime computational cost.

We define the expressive power of a convolutional neural network as the ability to construct labeling to differentiate input values. Following Cohen *et al.* [5], we view a neural network as a mapping from a particular example to a cost function (e.g., negative log probability) over labels. The mapping can be represented by a tensor $\mathcal{A}^y$ operated on the combination of the representation functions.

More concretely, the rank of $\mathcal{A}^y$, which scales as n^{2^L} with measure 1 over the space of all possible network parameters (n is the number of channels of a convolutional layer, a.k.a, network width; and L is the network depth), is a measure of the expressive power of a neural network as established in [5]. In static channel pruning, if m channels are kept, then the expressive power of the pruned network becomes m^{2^L} which is a strict subspace of n^{2^L} as $m < n$.

What makes our DeepMoE prevail is that (the sparsity pattern of) our mapping $\mathcal{A}^y$ depends on the data. We prove in the Appendix A.1 that DeepMoE has an expressive power of n^{2^L} with probability $1 - \binom{m}{n}^{-L}$ when stacking multiple MoE layers, indicating DeepMoE preserves the expressive power of the unsparsified network.

Motivated by the theoretical analysis, we propose two variants of DeepMoE: *wide-DeepMoE* and *narrow-DeepMoE*. In the former one, we first increase the number of channels in the convolutional networks to increase the expressive power of the network and then replace the widened layers with MoE layers. By controlling the number of channels selected at runtime, we can improve the prediction accuracy with the same amount of the computation as the unwidened network. This design has the potential to be applied to the real-world deployment on the new hardware architecture supporting dynamic routing, e.g., TPU, where we can place a wide network on it and only execute part of the network at runtime instead of placing a static thin network with the same amount of computation. Narrow-DeepMoE is closer to the dynamic channel pruning setting in [17] and comparable to the traditional static channel pruning.

5 EXPERIMENTS

In this section, we first evaluate the performance of both wide-DeepMoE and narrow-DeepMoE on the image classification (Sec. 5.1 and 5.2) and semantic segmentation tasks (Sec. 5.4). We observe that DeepMoEs can achieve lower prediction error rate with reduced computational cost. We also analyze the behavior of our gating network, DeepMoEs regularization effect, and other strategies for widening the network in Sec. 5.3.

Datasets. For the image classification task, we use the CIFAR-10 [16], CIFAR-100 [16] and ImageNet 2012 datasets [21]. For the semantic segmentation task, we use the CityScapes [8] dataset, which provides pixel-level annotations in the images with a resolution of 2048×1024. We apply standard data augmentation using basic mirroring and shifting [24] for CIFAR datasets and scale and aspect ratio augmentation with color perturbation [29] for ImageNet. We

Table 1: Wide-DeepMoE with ResNet-18, ResNet-34, and ResNet-50 on ImageNet. Wide-DeepMoE improves the accuracy on ImageNet by ∼1%.

Model	Top-1 Error Rate (%)
ResNet-18	30.24
Hard MoE [10]	30.43
Wide-DeepMoE-18	**29.05**
ResNet-34	26.70
Wide-DeepMoE-34	**25.87**
ResNet-50	23.85
Wide-DeepMoE-50	**22.88**

Table 2: Wide-DeepMoE with ResNet-56 and ResNet-100 on CIFAR datasets. Wide-DeepMoE improves the prediction accuracy of the baseline ResNet by 3∼4% on CIFAR-100 and 0.5% on CIFAR-10.

Dataset	Model	Top-1 Error Rate (%)
CIFAR-10	ResNet-56	6.55
	Wide-DeepMoE-56	**6.03**
CIFAR-100	ResNet-56	31.46
	Wide-DeepMoE-56	**29.77**
	ResNet-110	29.45
	Wide-DeepMoE-110	**26.14**

follow [30] to enable random cropping and basic mirroring and shifting augmentation for the CityScapes dataset.

Models. We examine DeepMoE with VGG [23] and ResNet [11] network designs as the base convolutional network (a.k.a, backbone network). VGG is a typical feed-forward network without skip connections and feature aggregation while ResNet, which is composed of many residual blocks, has more complicated connections. To construct DeepMoE, we add a gating header after each convolutional layer in VGG and modify the residual blocks in ResNet (Fig. 2).

In wide-DeepMoE, we increase the number of channels in each convolutional layer by a factor of two unless stated otherwise. In narrow-DeepMoE, we retain the same channel configuration as the original base convolutional model.

Training. To train DeepMoE we follow common training practices [11, 31]. For the CIFAR datasets, we start training with learning rate 0.1 for ResNet and 0.01 for VGG16, which is reduced by 10× at 150 and 250 epochs with total 350 epochs for the baselines and 270 epochs for DeepMoE joint optimization stage and another 80 epochs for fine-tuning with fixed gating networks.

For ImageNet, we train the network with initial learning rate 0.1 for 100 epochs and reduce it by 10× every 30 epochs. We do not further fine-tune the base convolutional network on ImageNet as we find the improvement from fine-tuning is marginal compared to that on CIFAR datasets.

We set the computational cost parameter λ in the DeepMoE loss function (Eq. 7) between [0.001, 8] (larger values reduce computation) and $\mu = 1$ for the CIFAR datasets to match the scale of the cross entropy loss on the base model. For ImageNet we set $\mu = 0$ to improve base model feature extraction. The training schedule for semantic segmentation is detailed in Sec. 5.4.

5.1 WIDE-DEEPMOE

In this section, we evaluate the performance of wide-DeepMoE as well as its memory usage.

5.1.1 Improved Accuracy with Reduced Computation

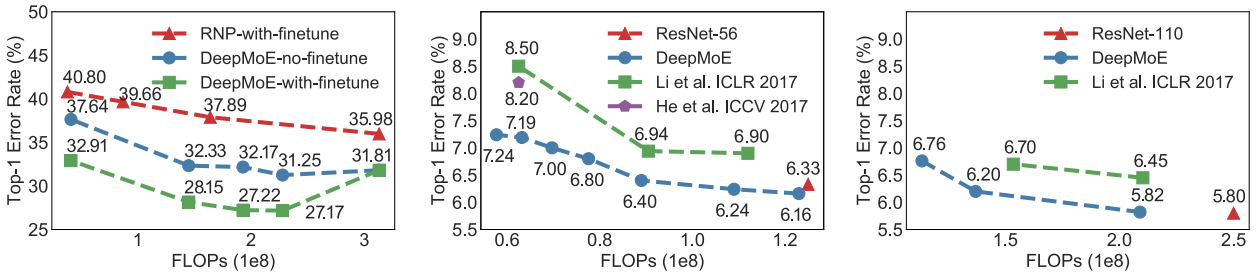
To conduct the evaluation, we first increase the number of channels in the residual networks by a factor of 2 and then control the sparsification so that on average half of the convolutional channels are selected at the inference time. Through our evaluations we find that wide-DeepMoE has lower prediction error rate than the standard ResNets on ImageNet (Tab. 1), CIFAR-10 and CIFAR-100 (Tab. 2).

We evaluate ResNet-56 and ResNet-110 on the CIFAR-10 and CIFAR-100 datasets and ResNet-18, ResNet-34, ResNet-50 on ImageNet, using the basic block (Fig. 2a) for 18 and 34 and bottleneck-A(Fig. 2b) for 50. The more memory efficient bottleneck-B (Fig. 2c) is also adopted on ImageNet.

As expressed in Tab. 1, wide-DeepMoE is able to reduce the error rate of the ImageNet benchmark without increasing the computational cost (measured by FLOPs) with networks of different depths. In particular, wide-DeepMoE with ResNet-18 and 34 reduce $\sim$ 1% Top-1 error on ImageNet on which the previous work [10] fails to show any improvement. Similar results can be observed on CIFAR datasets as shown in Tab. 2, where wide-DeepMoE improves the prediction accuracy of the baseline ResNet by 3∼4% on CIFAR-100 and 0.5% on CIFAR-10.

5.1.2 Memory Usage

Another aspect to consider about DeepMoE is its memory footprint (proportional to the number of parameters). We examine the memory usage of wide-DeepMoE with widened ResNet-50 as the backbone network and compare it to the standard ResNet-101



(a) DeepMoE *vs* RNP (b) DeepMoE *vs* Static Pruning (c) DeepMoE *vs* Static Pruning

Figure 3: (a) DeepMoE vs the dynamic pruning approach RNP on CIFAR-100 with VGG16. DeepMoE not only outperforms RNP on the accuracy-computation trade-off but improves the accuracy over the baseline VGG model. (b) and (c) DeepMoE vs static pruning approaches on CIFAR-10.

which has a similar prediction accuracy. We find that wide-DeepMoE using Bottleneck-A (Fig. 2b) achieves a 22.88% Top-1 error rate, which compared to ResNet-110 with an error rate of 22.63%, is only 0.2% lower in error but requires 20% less computation. Moreover, wide-DeepMoE using Bottleneck-B (Fig. 2c), which is more memory efficient than Bottleneck-A (Fig. 2b), achieves 22.84% top-1 error with 6% less parameters and 18% less FLOPs than the standard ResNet-101 indicating that wide-DeepMoE is competitive on the memory usage.

5.2 NARROW-DEEPMOE

In this section we compare DeepMoE to current static and dynamic channel pruning techniques. We show that DeepMoE is able to out-preform both dynamic and static channel pruning techniques in prediction accuracy while maintaining or reducing computational costs.

5.2.1 Narrow-DeepMoE *vs* Dynamic Channel Pruning

DeepMoE generalizes existing channel pruning work since it both dynamically prunes and *re-scales* channels to reduce the computational cost and improve accuracy. In previous dynamic channel pruning work [18], channels are pruned based on the outputs of previous layers. In contrast, the gate decisions in DeepMoEs are determined in advance based on the shared embedding (latent mixture weights) which enables improved batch parallelism at inference.

We compare DeepMoE to the latest dynamic channel pruning work RNP [18] with VGG-16[1] as the base model on CIFAR-100. As we can see from Fig. 3a, without fine-tuning, the prediction error and computation trade-off curve (dotted blue line) of DeepMoE is much flatter than RNP (dotted red line) which indicates DeepMoE has a greater reduction in computation without loss of accuracy. Moreover, when fine-tuning DeepMoE for only 10 epochs (dotted green line in Fig. 3a), DeepMoE improves the prediction accuracy by a large margin by 4% which is a $\sim$ 13% improvement over the baseline VGG model due to the regularization effect of DeepMoE (Sec. 5.3.2).

5.2.2 Narrow-DeepMoE *vs* Static Channel Pruning

Similarly, DeepMoE outperforms the state-of-the-art static channel pruning results [19, 11, 17, 13] on both ImageNet shown in Tab. 3 and the CIFAR-10 dataset in Fig. 3b and 3c. DeepMoE with ResNet-50 reduces 56.8% of the computation of the standard ResNet-50 with a top-1 error rate of 26.21%, approximately 2% better than He *et al.* [12], which currently has the best accuracy for an equivalent amount of computation among previous work on ImageNet. Fig. 3b and 3c show that DeepMoE achieves a higher accuracy less computation times than current techniques.

Table 3: Pruned ResNet-50 on ImageNet. Top-1/5 error rate and computation FLOPs are reported. DeepMoE is able to achieve a 26.2% Top-1 error rate, which is 0.6-2.8% lower than the other models, while using the least amount of computational cost.

Model	Top-1	Top-5	FLOPs(x10^9)	Reduct.(%)
SSS [13]	26.8	-	3.0	20.3
Li et al. [17]	27.0	8.9	3.0	19.0
He et al. [12]	-	9.2	1.9	50.0
ThiNet [19]	29.0	10.0	1.7	55.8
DeepMoE	**26.2**	**8.4**	**1.6**	**56.8**

[1]Our baseline accuracy is higher than RNP since we use a version with batch normalization in contrast to the published method.

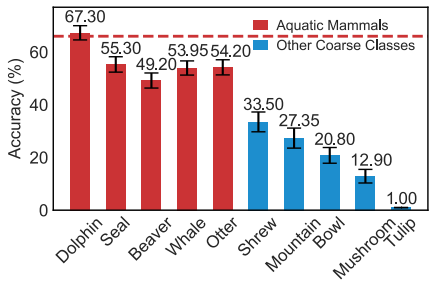


Figure 4: Gate embedding shuffling. The in-group shuffling has an accuracy 20-65% higher than out-of-group shuffling.

Table 4: Different widening strategies for VGG16 on CIFAR-100. When controlling the computation FLOPs or both the computation FLOPs and the parameters, the prediction accuracy of widening all convolutional layers is higher than all other widening techniques.

Control	Model	Params	FLOPs (x10^8)	Acc. (%)
Params	W1-High	24.15M	3.51	71.96
	W1-Mid	24.16M	9.18	72.02
	W4-Low	24.18M	43.16	72.51
	W13-All	24.18M	8.15	**73.91**
Params & FLOPs	W1-High	24.15M	2.98	73.28
	W1-Mid	24.16M	2.74	72.68
	W4-Low	24.18M	2.45	73.33
	W13-All	24.18M	2.29	**73.39**

5.3 ANALYSIS

In this section, we first analyze the effectiveness of the gating behavior in generating embeddings that are predictive of the class label and thus its ability balance expert utilization. We then study the regularization effect of DeepMoEs sparsification of the channel outputs. Lastly, we explore the effects of widening certain combinations of layers in a network as opposed to widening all convolutional layers as we do in DeepMoE.

5.3.1 Gating Behavior Analysis

To analyze the gating behavior of DeepMoE, we evaluate the trained DeepMoE with VGG-16 as follows: for a given fine-grained class A (e.g., *dolphin*), we reassign the gate embedding for each input in class A with a randomly chosen gate embedding from other classes either within the same coarse category (referred to as *in-group shuffling*) or different categories (referred to as *out-of-group shuffling*).

In Fig. 4, we plot the test accuracy of class *dolphin*, belonging to the coarse category *aquatic mammals* with randomly selected gate embeddings (repeated 20 times for each input) from 5 classes in the same coarse category (in red) and 5 classes for other coarse categories (in blue). Fig. 4 shows that the test accuracy with in-group embeddings is 20-60% higher than with out-of-group shuffling. Especially when applying the gate embeddings from the tulip category, the test accuracy drops to 1% while the accuracy with in-group shuffling is mostly above 50%. This indicates that the latent mixture of weights are similar for semantically related image categories, and since DeepMoE is never given this coarse class structure, our results are significant.

5.3.2 Regularization Effect of DeepMoE

Since DeepMoE sparsifies the channel outputs during training and testing, we study the regularization effect of such sparsification. We increase the number of channels of a modified ResNet-18 with bottleck-B (in Fig. 2c) by 2-8× on CIFAR-100. In Fig. 5, we plot the accuracy and computation FLOPs of the baseline widened ResNet-18 models (in blue) and wide-DeepMoE (in orange) with $\lambda = 2$.

Fig. 5 suggests that DeepMoE has a lower computation cost and higher accuracy than the baseline widened ResNet, and the advantages of DeepMoE increase with the width of the base convolutional network. This indicates a potential regularization effect to the DeepMoE design.

5.3.3 DeepMoE *vs* Single-Layer MoE

So far in our experiments, we have widened the network by increasing the number of experts/channels for all convolutional layers. Here, we study other strategies for widening the network. We try to widen the VGG-16 model in four different kinds of layers: the top layer (W1-High), the middle layer (W1-Mid), the lower 4 layers (W4-Low), and finally all the 13 convolutional layers (W13-All) as used in all the other experiments (details in Sec. A.2).

As shown in Tab. 4, the prediction accuracy of W13-All is strictly better than that of a single-layer MoE, even though they have the same number of parameters. Adding MoE to the bottom or top layers is more effective than adding it to the middle layer. Alternatively, if we control both the number of parameters and the computation FLOPs, the accuracy differences between different strategies are reduced but W13-All is still favorable to other widening strategies.

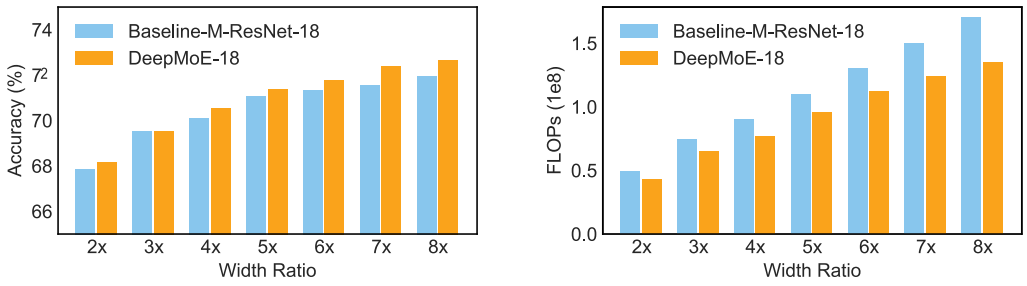


Figure 5: Regularization Effect of DeepMoE on Widened ResNet. Wide-DeepMoE is both more accurate and efficient than the widened baseline models.

Table 5: Segmentation Results on CityScapes. The more efficient version wide-DeepMoE-50-a beats the baseline by 1.5% of mIoU with a slight increase in FLOPs, while the more accurate version wide-DeepMoE-50-b outperforms the wide baseline by almost 2% of mIoU with lower FLOPs.

	Road	Sidewalk	Building	Wall	Fence	Pole	Light	Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorcycle	Bicycle	mIoU	FLOPs($\times 10^9$)
DRN-A-50	96.9	77.4	90.3	35.8	42.8	59.0	66.8	74.5	91.6	57.0	93.4	78.7	55.3	92.1	43.2	59.5	36.2	52.0	75.2	67.3	**703**
wide-DeepMoE-50-A	97.2	78.9	90.3	45.6	48.4	56.2	61.6	72.9	91.6	60.7	94.2	77.4	50.6	92.5	48.7	68.7	44.1	52.7	74.2	**68.8**	804
wide-DRN-A-50	97.4	80.6	90.6	38.5	49.0	58.7	65.1	73.4	91.8	59.5	93.9	78.2	51.1	92.9	49.1	68.7	51.3	52.2	74.5	69.3	2173
wide-DeepMoE-50-B	97.5	80.4	91.0	48.9	50.6	58.5	65.7	75.3	92.0	60.1	94.7	79.2	54.7	93.2	53.8	73.2	53.2	54.8	75.6	**71.2**	**1738**

5.4 SEMANTIC SEGMENTATION

Semantic image segmentation requires predictions for each pixel, instead of one label for the whole image in classification. We evaluate DeepMoE on the segmentation task to understand its generalizability. In specific, we apply DeepMoE to DRN-A [30], which adopts ResNet architecture as the backbone, and evaluate the results on the popular segmentation dataset CityScapes [8]. We follow the same training procedure as Yu *et al.* [30] for fair comparison. The optimizer is SGD with momentum 0.9 and crop size 832. The starting learning rate is set to 5e-4 and divided by 10 after 200 epochs. The intersection-over-union (IoU) scores and computation costs in FLOPs of DeepMoE are presented in Tab. 5.

The hyper-parameter λ can adjust the trade-offs between computer efficiency and prediction accuracy. Our efficient model `wide-DeepMoE-50-A` beats the baseline by 1.5% of mIoU with a slight increase in FLOPs, while our accurate model `wide-DeepMoE-50-B` outperforms the wide baseline by almost 2% mIoU with lower FLOPs. These results indicate DeepMoE is effective on pixel-level prediction such as semantic segmentation as well as image classification.

6 CONCLUSION

In this work we introduced our design of deep mixture of experts models, which produces a more accurate and computationally inexpensive model for computer vision applications. Our DeepMoE architecture leverages a shallow embedding network to construct latent mixture weights, which is then used by sparse multi-headed gating networks to select and re-weight individual channels at each layer in the deep convolutional network. This design in conjunction with a novel sparsifying and diversifying loss enabled joint differentiable training, addressing the key limitations of existing mixture of experts approaches in deep learning. We provided theoretical analysis on the expressive power of DeepMoE and proposed two design variants. The extensive experimental evaluation indicated that DeepMoE can reduce computation and surpass accuracy over baseline convolutional networks, as well as improving upon the residual network result on the challenging ImageNet benchmark by a full 1%. Through our analysis we were also able to prove that our embedding and gating network is able to resolve coarse grain class structure in the underlying problem. This work shows promising results when applied to semantic segmentation tasks, and could be incredibly useful for various other problems.

References

[1] K. Ahmed, M. H. Baig, and L. Torresani. Network of experts for large-scale image categorization. In *European Conference on Computer Vision*, pages 516–532. Springer, 2016.

[2] E. Bengio, P.-L. Bacon, J. Pineau, and D. Precup. Conditional computation in neural networks for faster models. *arXiv preprint arXiv:1511.06297*, 2015.

[3] Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[4] K. Cho and Y. Bengio. Exponentially increasing the capacity-to-computation ratio for conditional computation in deep learning. *arXiv preprint arXiv:1406.7362*, 2014.

[5] N. Cohen, O. Sharir, and A. Shashua. On the expressive power of deep learning: A tensor analysis. In *Conference on Learning Theory*, pages 698–728, 2016.

[6] R. Collobert, S. Bengio, and Y. Bengio. A parallel mixture of svms for very large scale problems. In *Advances in Neural Information Processing Systems*, pages 633–640, 2002.

[7] R. Collobert, Y. Bengio, and S. Bengio. Scaling large learning problems with hard parallel mixtures. *International Journal of pattern recognition and artificial intelligence*, 17(03):349–365, 2003.

[8] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[9] D. Eigen, M. Ranzato, and I. Sutskever. Learning factored representations in a deep mixture of experts. *International Conference on Learning Representations Workshop*, 2014.

[10] S. Gross, M. Ranzato, and A. Szlam. Hard mixtures of experts for large scale weakly supervised vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6865–6873, 2017.

[11] 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.

[12] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *International Conference on Computer Vision (ICCV)*, volume 2, page 6, 2017.

[13] Z. Huang and N. Wang. Data-driven sparse structure selection for deep neural networks. *arXiv preprint arXiv:1707.01213*, 2017.

[14] 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.

[15] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural computation*, 6(2):181–214, 1994.

[16] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[17] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. *International Conference on Learning Representations*, 2017.

[18] J. Lin, Y. Rao, J. Lu, and J. Zhou. Runtime neural pruning. In *Advances in Neural Information Processing Systems*, pages 2178–2188, 2017.

[19] J.-H. Luo, J. Wu, and W. Lin. ThiNet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5058–5066, 2017.

[20] J. Ma, Z. Zhao, X. Yi, J. Chen, L. Hong, and E. H. Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1930–1939. ACM, 2018.

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[22] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *International Conference on Learning Representations*, 2017.

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

[24] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066, 2013.

[25] X. Wang, Y. Luo, D. Crankshaw, A. Tumanov, F. Yu, and J. E. Gonzalez. Idk cascades: Fast deep learning by learning not to overthink. *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018.

[26] X. Wang, J. Wu, D. Zhang, Y. Su, and W. Y. Wang. Learning to compose topic-aware mixture of experts for zero-shot video captioning. *arXiv preprint arXiv:1811.02765*, 2018.

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

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

[29] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5987–5995. IEEE, 2017.

[30] F. Yu, V. Koltun, and T. Funkhouser. Dilated residual networks. In *Computer Vision and Pattern Recognition*, volume 1, 2017.

[31] F. Yu, D. Wang, and T. Darrell. Deep layer aggregation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.