Deep Mixture of Experts via Shallow Embedding
Supplementary Material

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

APPENDIX

A.1 Expressive Power of DeepMoE

To characterize the expressive power of DeepMoE, we follow the tensor analysis approach of Cohen et al. [1]. We first represent an instance of data as a collection of vectors $(\mathbf{x}_1, \cdots, \mathbf{x}_N)$, where $\mathbf{x}_i \in \mathbb{R}^s$. For the image data, the collection $(\mathbf{x}_1, \cdots, \mathbf{x}_N)$ corresponds to vector arrangements of possibly overlapping patches around pixels. We represent different features in data using (positive) representation functions:

$$f_d(\mathbf{x}_i), \tag{1}$$

so that the convolution operations over data become multiplications over representation functions. For the representation functions, index $d \in \{1, \cdots, M\}$, where M is the number of different features in data that we wish to distinguish and can be combinatorially large with respect to the number of pixels.

For classification tasks, we view a neural network as a mapping from a particular instance to a cost function (e.g., the log probability) over labels y for that instance. With the new representation of data instances following Eq. (1), the mapping can be represented by a tensor $\mathcal{A}^y$ operated on the combination of the representation functions:

$$h_y(\mathbf{x}_1, \cdots, \mathbf{x}_N) = \sum_{d_1, \cdots, d_N = 1}^{M} \mathcal{A}^y_{d_1, \cdots, d_N} \prod_{i=1}^{N} f_{d_i}(\mathbf{x}_i). \tag{2}$$

To be able to distinguish data instances $\mathbf{x}$ from $\tilde{\mathbf{x}}$, we need $h_y(\mathbf{x}_1, \cdots, \mathbf{x}_N) - h_y(\tilde{\mathbf{x}}_1, \cdots, \tilde{\mathbf{x}}_N)$ to be nonzero. For a fixed mapping $\mathcal{A}^y$, this requirement is equivalent to:

$$\sum_{d_1, \cdots, d_N = 1}^{M} \mathcal{A}^y_{d_1, \cdots, d_N} \left(\prod_{i=1}^{N} f_{d_i}(\mathbf{x}_i) - \prod_{i=1}^{N} f_{d_i}(\tilde{\mathbf{x}}_i) \right) \neq 0,$$

for $\mathbf{x} \neq \tilde{\mathbf{x}}$. It can directly be seen that the inequality is satisfied when the difference $\prod_{i=1}^{N} f_{d_i}(\mathbf{x}_i) - \prod_{i=1}^{N} f_{d_i}(\tilde{\mathbf{x}}_i)$ is not in the null space of $\mathcal{A}^y_{d_1, \cdots, d_N}$. Therefore, the expressive power is equivalent to the *rank* of the tensor $\mathcal{A}^y$. This approach, taken by [1], establishes that for a certain type of networks, the rank of $\mathcal{A}^y$ scales as n^{2^L} with measure 1 over the space of all possible network parameters, where n is the number of channels between network layers (width) and L is the network depth.

If we directly apply the theorem to a wider network (width m satisfying $m > n$), then the rank of $\mathcal{A}^y$ will scale as m^{2^L}, which is $\left(\frac{m}{n}\right)^{2^L}$ times better. However, when the channels are gated with *static* sparse weights, the set of $\mathcal{A}^y$ with this restriction has measure 0 in the overall space of network parameters. In fact, if the number of nonzero weights over the channels is n, then the rank of $\mathcal{A}^y$ still scales as n^{2^L}.

What makes our DeepMoE prevail is that (the sparsity pattern of) our mapping $\mathcal{A}^y$ depends on the data. We hereby compare an L-layer DeepMoE with width equal to m and number of nonzero weights over the channels equal to $n < m$ against an L-layer fixed, non-sparse neural network with width equal to m. For the latter, we know that it will be able to distinguish between features in a subspace of dimension m^{2^L}. For the former, if $h_y(\mathbf{x}_1, \cdots, \mathbf{x}_N) - h_y(\tilde{\mathbf{x}}_1, \cdots, \tilde{\mathbf{x}}_N) \neq 0$ for the same choices of features (in the m^{2^L} dimensional subspace), then we know that it will have

expressive power of at least m^{2^L}:

$$
\begin{aligned}
& h_y(\mathbf{x}_1, \cdots, \mathbf{x}_N) - h_y(\tilde{\mathbf{x}}_1, \cdots, \tilde{\mathbf{x}}_N) \\
= & \sum_{d_1,\cdots,d_N=1}^{M} \mathcal{A}^y_{d_1,\cdots,d_N} \prod_{i=1}^{N} f_{d_i}(\mathbf{x}_i) - \qquad (3) \\
& \sum_{d_1,\cdots,d_N=1}^{M} \tilde{\mathcal{A}}^y_{d_1,\cdots,d_N} \prod_{i=1}^{N} f_{d_i}(\tilde{\mathbf{x}}_i) \\
= & \sum_{d_1,\cdots,d_N=1}^{M} \mathcal{A}^y_{d_1,\cdots,d_N} \left(\prod_{i=1}^{N} f_{d_i}(\mathbf{x}_i) - \prod_{i=1}^{N} f_{d_i}(\tilde{\mathbf{x}}_i) \right) \qquad (4) \\
& + \sum_{d_1,\cdots,d_N=1}^{M} \left(\mathcal{A}^y_{d_1,\cdots,d_N} - \tilde{\mathcal{A}}^y_{d_1,\cdots,d_N} \right) \prod_{i=1}^{N} f_{d_i}(\tilde{\mathbf{x}}_i). \qquad (5)
\end{aligned}
$$

Since the gating network is independent from the convolution neural network, to have Line (4) exactly equal to the negative of Line (5)—when they are both nonzero—has zero measure over the space of network parameters (even with the sparsity constraint). We simply need to focus on the cases where Line (4) is zero for the pair of $\mathbf{x}$ and $\tilde{\mathbf{x}}$, and discuss whether Line (5) is also zero. In those cases, we assume that the sparsity pattern of the weights over the gated channels is i.i.d. with respect to each channel. With this assumption, probability of choosing exactly the same channels for different data: $\mathcal{A}^y_{d_1,\cdots,d_N} = \tilde{\mathcal{A}}^y_{d_1,\cdots,d_N}$ is $\binom{m}{n}^{-L}$. When they are not equal, the difference $\mathcal{A}^y_{d_1,\cdots,d_N} - \tilde{\mathcal{A}}^y_{d_1,\cdots,d_N}$ can be represented as combinations of linearly independent basis in $\mathbb{R}^{M^N}$ and positivity of the representation functions ensures that Line (5) is not zero with probability 1. Therefore, $h_y(\mathbf{x}_1, \cdots, \mathbf{x}_N) - h_y(\tilde{\mathbf{x}}_1, \cdots, \tilde{\mathbf{x}}_N) \neq 0$ holds with probability $1 - \binom{m}{n}^{-L}$. In other words, there is a $1 - \binom{m}{n}^{-L}$ probability that the expressive power of our DeepMoE equals to or is bigger than m^{2^L}.

A.2 Network Configurations of Wide VGG

In Sec. 5.3.3 of the main paper, we conduct experiments to investigate different widening strategies. We used four different strategies to widen the VGG-16 network which contains 13 convolutional layers in total: W1-High widens the top layer only, W1-Mid widens the middle layer only, W4-Low widens the lower 4 layers, and finally W13-All that widens all 13 convolutional layers in Tab. 1.

Table 1: Channel configurations of different widening strategies

Layers	W1-High	W1-Mid	W4-Low	W13-All
Conv1	64	64	512	128
Conv2	64	64	512	128
Max Pooling	-	-	-	-
Conv3	128	128	615	256
Conv4	128	128	615	256
Max Pooling	-	-	-	-
Conv5	256	2990	256	405
Conv6	256	256	256	405
Conv7	256	256	256	405
Max Pooling	-	-	-	-
Conv8	512	512	512	615
Conv9	512	512	512	615
Conv10	512	512	512	615
Max Pooling	-	-	-	-
Conv11	1536	512	512	615
Conv12	512	512	512	615
Conv13	512	512	512	615
Max Pooling	-	-	-	-
Soft-max	-	-	-	-

References

[1] 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.