OUTRAGEOUSLY LARGE NEURAL NETWORKS: THE SPARSELY-GATED MIXTURE-OF-EXPERTS LAYER

Noam Shazeer[1], Azalia Mirhoseini*†[1], Krzysztof Maziarz*[2], Andy Davis[1], Quoc Le[1] and Jeff Dean[1]

[1]Google Brain, {noam,azalia,andydavis,qvl,jeff}@google.com
[2]Jagiellonian University, Cracow, krzysztof.maziarz@student.uj.edu.pl

ABSTRACT

The capacity of a neural network to absorb information is limited by its number of parameters. In this work, we present a new kind of layer, the Sparsely-Gated Mixture-of-Experts (MoE), which can be used to effectively increase model capacity with only a modest increase in computation. This layer consists of up to tens of thousands of feed-forward sub-networks (experts) containing a total of up to tens of billions of parameters. A trainable gating network determines a sparse combination of these experts to use for each example. We apply the MoE to the task of language modeling, where model capacity is critical for absorbing the vast quantities of world knowledge available in the training corpora. We present new language model architectures where an MoE layer is inserted between stacked LSTMs, resulting in models with orders of magnitude more parameters than would otherwise be feasible. On language modeling and machine translation benchmarks, we achieve comparable or better results than state-of-the-art at lower computational cost, including test perplexity of 29.9 on the 1 Billion Word Language Modeling Benchmark and BLEU scores of 40.56 and 26.03 on the WMT'14 En to Fr and En to De datasets respectively.

1 INTRODUCTION

The exponential growth in training data as well as the increasing size and complexity of neural networks have been persistent trends in machine learning. It is evident that realizing the true potential of larger datasets demands training models with larger numbers of parameters. For typical deep learning models, where the entire model is activated for every example, This leads to a quadratic blow-up in training costs, as both the model size and the number of training examples increase. Unfortunately, the advances in computing power and distributed computation fall short of meeting such demand. To this end, it is imperative to design new scalable techniques that enable training ever larger models on ever larger datasets.

In this work we present a new technique that enables training extremely large models without increasing computation. Our approach is to increase the number of parameters by introducing a new type of neural network layer: a Sparsely-Gated Mixture-of-Experts (MoE). The MoE layer consists of a number of experts, each being a neural network model, and a trainable gating network which selects a sparse combination of the experts to process each input (see Figure 1). The sparse gating allows us to make the layer extremely large (billions of parameters), while keeping computation per example manageable (millions of FLOPs) and maintaining large enough batch sizes to allow for efficient GPU matrix multiplication.

While the concept of a mixture of experts has existed for some time, e.g., (Jacobs et al., 1991), our work advances prior art in several key aspects: We introduce the sparsity in the gating network which allows us increase the number of parameters while keeping the computation constant. We also pose MoEs as a general purpose neural network component which can be applied in a feed-forward, recurrent or convolutional manner. In addition, we develop novel system design practices that enable scaleable and parallel training of MoE layers with thousands of experts.

*Equally major contributors
†Work done as a member of the Google Brain Residency program (g.co/brainresidency)

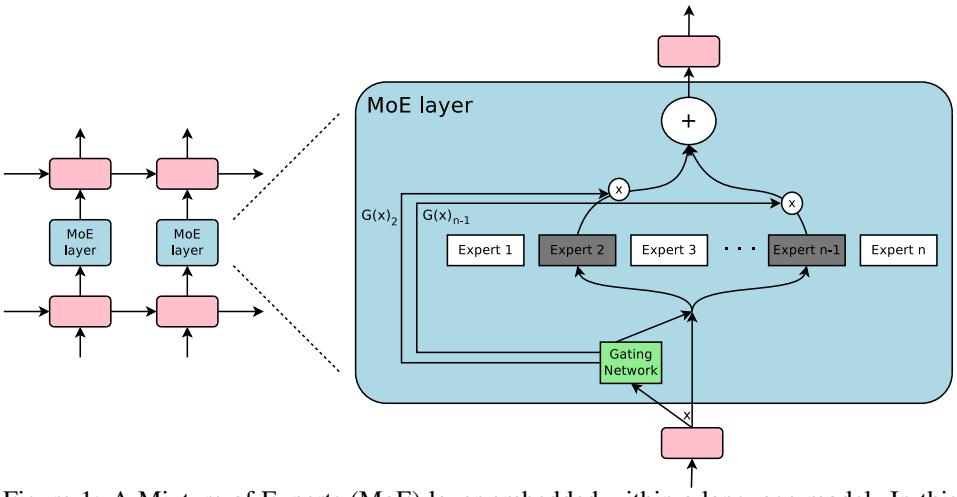

Figure 1: A Mixture of Experts (MoE) layer embedded within a language model. In this case, the sparse gating function selects two experts to perform computations. Their outputs are modulated by the outputs of the gating network.

While the introduced technique is generic, in this paper we focus on language modeling tasks, which are known to benefit from very large models. In particular, we apply a MoE convolutionally between stacked LSTM layers in a language model as in Figure 1. The MoE is called once for each position in the text, selecting a different combination of experts at each position. The different experts tend to become highly specialized based on syntax and semantics.

We applied our approach to Language Modeling (LM) and Neural Machine Translation (NMT) tasks. We trained models with up to 30 billion parameters, one or two orders of magnitude larger than models found in the literature. This allowed us to match and exceed best published results at a fraction of the computational cost.

Language Modeling: We experimented on the 1 Billion Word Language Model Benchmark (Chelba et al., 2013). On this dataset, we achieved a test perplexity of 29.9, improving on the best published result for a single model (Jozefowicz et al., 2016) while speeding up training by a factor of 8.

Machine Translation: On the WMT'14 En to Fr dataset, we achieved a BLEU score of 40.56 which is the highest reported value on this dataset. On the WMT'14 En to De dataset, we achieved the BLEU score of 26.03, also the highest value on this dataset. Both results were achieved while training the model for the same amount of time as (Wu et al., 2016). We were also able to improve the state-of-the-art by 1.0 BLEU score on an internal translation dataset that is 2 orders of magnitude larger than the WMT'14. This was done by training a model with 35 times more parameters than what is reported in (Wu et al., 2016) while speeding up training time by a factor of 6. These results are particularly significant as our baseline is highly optimized.

2 Related Work

Exploiting scale in both the number of available training examples and the model sizes has been central to the success of deep learning. When datasets are sufficiently large, increasing the size of neural networks can give much better prediction accuracy. This has been shown in domains such as text (Sutskever et al., 2014; Bahdanau et al., 2014; Jozefowicz et al., 2016; Wu et al., 2016), images (Krizhevsky et al., 2012; Le et al., 2012), and audio (Hinton et al., 2012; Amodei et al., 2015). Thus, new mechanisms should be designed to enable scalable training of very large models on large datasets.

Ever since its introduction more than two decades ago (Jacobs et al., 1991; Jordan & Jacobs, 1994), the mixture-of-experts approach has been the subject of much research. Prior work has focused on different aspects including using different types of expert models such as SVMs (Collobert et al.,

2002), Gaussian Processes (Tresp, 2001; Theis & Bethge, 2015; Deisenroth & Ng, 2015), Dirichlet Processes (Shahbaba & Neal, 2009), or different expert configurations such as a hierarchical structure (Yao et al., 2009), and infinite number of experts (Rasmussen & Ghahramani, 2002). Eigen et al. (2013) extends MoE to a deep model by stacking two layers of mixture of experts (with each expert being a feed forward network) followed by a Softmax layer. The experts output are weighted by a trainable gating network. Garmash & Monz (2016) suggests an ensemble model in the format of mixture of experts for machine translation. The gating network is trained on a pre-trained ensemble NMT model.

The key difference between our work and prior research on MoE is that our work enables training extremely large number of experts (e.g., thousands) and parameters (e.g., multiple billions). This is possible by a trainable gating network that assigns the inputs to only a sparse number of experts. We regard our MoE as a new type of neural network layer that can be used to increase the capacity of any conventional deep learning model in a computationally tractable manner. Our experts and the gating function are learnable feed forward networks. In our experiments, we create models with tens of billions of parameters whose training time is comparable or even better than that of previous state-of-the-art models that have only millions of parameters. We show that such dramatic increase in parameter size effectively results in new state-of-the-art inference accuracy in language modeling and translation tasks.

3 THE STRUCTURE OF THE MIXTURE-OF-EXPERTS LAYER

The Mixture-of-Experts (MoE) layer consists of a set of n "expert networks" $E_1, \cdots, E_n$, and a "gating network" G whose output is a sparse n-dimensional vector. Figure 1 shows an overview of the MoE module. The experts are themselves neural networks, each with their own parameters. Although in principle we only require that the experts accept the same sized inputs and produce the same-sized outputs, in our initial investigations in this paper, we restrict ourselves to the case where the models are feed-forward networks with identical architectures, but with separate parameters.

Let us denote by $G(x)$ and $E_i(x)$ the output of the gating network and the output of the i-th expert network for a given input x. The output y of the MoE unit can be written as follows:

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

We save computation based on the sparsity of the output of $G(x)$. Wherever $G(x)_i = 0$, we need not compute $E_i(x)$. In our experiments, we have up to thousands of experts, but only need to evaluate a handful of them for every example.

3.1 GATING NETWORK

Softmax Gating: A simple choice of non-sparse gating function (Jordan & Jacobs, 1994) is to multiply the input by a trainable weight matrix W_g and then apply the $Softmax$ function.

$$G_\sigma(x) = Softmax(x \cdot W_g) \tag{2}$$

Sparse Gating: To obtain a sparse gating vector, we multiply $G_\sigma(x)$ component-wise with a sparse mask $M(G_\sigma(x))$ and normalize the output. The mask itself is a function of $G_\sigma(x)$ and specifies which experts are assigned to each input example:

$$G(x)_i = \frac{G_\sigma(x)_i M(G_\sigma(x))_i}{\sum_{j=1}^{n} G_\sigma(x)_j M(G_\sigma(x))_j} \tag{3}$$

Top-K Mask: One choice of mask is to keep only the top k experts per example, for some small constant k such as 2. In the equation above, we let $M(v) = TopK(v, k)$, where:

$$TopK(v, k)_i = \begin{cases} 1 & \text{if } v_i \text{ is in the top } k \text{ elements of } v. \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Other Mask Functions: One can imagine experimenting with other choices of randomized and deterministic mask functions. In our larger distributed training setups, using a strictly balanced load helps ensure that we keep our computational devices across different machines equally busy. In Appendix A, we discuss an alternative mask function that enforces this property.

3.2 Ensuring Expert Utilization

We have observed that the gating network tends to converge to a state where it always produces large weights for same few experts. This imbalance is self-reinforcing, as the favored experts are trained more rapidly and thus are selected even more by the gating network. We address this problem by imposing an additional training loss to encourage the experts to be used equally across each training batch. This loss is added to the overall loss function for the model. For each batch X of inputs, we compute the ℓ_2 loss on the difference between the average output of the gating network across the batch and the uniform distribution:

$$L_{balance}(X) = \frac{1}{2} \sum_{i=1}^{n} \Big(\frac{\sum_{x \in X} G(x)_i}{|X|} - \frac{1}{n} \Big)^2 \tag{5}$$

3.3 Hierachical Mixture of Experts

If the number of experts is very large, we can reduce the branching factor by using a two-level hierarchical MoE. In a hierarchical MoE a primary gating network chooses a sparse weighted combination of "experts", each of which is itself a secondary mixture-of-experts with its own gating network. We have not found the need for deeper hierarchies.

4 Addressing MoE System Performance Issues

A major challenge that arises from a naive MoE implementation is the shrinking batch problem. If the gating network chooses k out of n experts for each example, then for a batch of b examples, each expert receives a much smaller batch of approximately $\frac{kb}{n} \ll b$ examples. The shrinking batch problem causes computational inefficiency, as computational devices such as CPUs or GPUs tend to benefit from the increased ratio between the number of arithmetic operations and memory accesses which happens when large batch sizes are used. Our solutions to the shrinking batch problem involve finding ways to increase the original batch size, which we discuss in the following.

Combination Across Time Steps: In our language models, we apply the MoE to each time step of the previous layer. If we wait for the previous layer to finish, we can apply the MoE to all the time steps together as one big batch, as shown in Figure 2. Doing so increases the size of the input batch to the MoE layer by a factor of the number of time steps, which we denote by n_s.[1]

Combination Across Batches: In addition to combination across time steps, we employ a second technique which combines examples from different training batches, as shown in Figure 2-Right. In a conventional distributed training setting, multiple copies of the model on different devices process distinct batches of data, and parameter updates are performed asynchronously by a parameter server. In our technique, these different batches run synchronously. We distribute the standard layers of the model and the gating network according to the conventional data-parallel schemes but only keep one shared copy of each expert. Each expert in the MoE layer receives a combined batch consisting of

[1]We could not use this trick if we were applying the MoE recurrently between time steps, which might be a direction of future work.

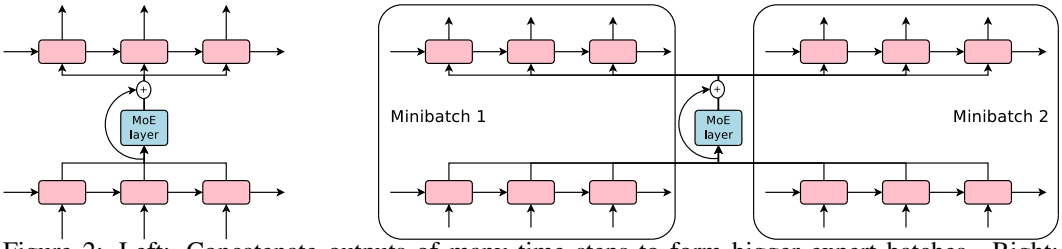

Figure 2: Left: Concatenate outputs of many time steps to form bigger expert batches. Right: Concatenate many minibatches from distributed data parallel models to form expert batches.

the relevant examples from all of the data-parallel input batches. If the model is distributed over n_p devices, and each device receives a batch of size b, each expert receives a batch of approximately $\frac{k \cdot b \cdot n_s \cdot n_p}{n}$ examples. Thus, we achieve a factor of $n_p\times$ improvement in expert batch size.

In the case of a hierarchical MoE (Section 3.3), the primary gating network employs data parallelism, secondary MoEs employ model parallelism, each one residing on its own device.

5 Experiments

In the following, we present our experimental results on scaling up neural language model parameter sizes. We will show how these large models improve baselines in language modeling and neural machine translation.

For language modeling, we tested our method on the 1 Billion Word Language Model Benchmark (Chelba et al., 2013). We improved on the best published result for a single, word-level model (Jozefowicz et al., 2016), using a fraction of the computation.

Table 1: Summary of our results against state-of-the-art. For more details see Section 5.1

	Test Perplexity	#Parameters excluding embedding and output layers	Computation per Word excluding output layer	Training Time
Ours	29.9	34431 million	58 million	15 hours on 128 k40s
Best Published Results	30.6	151 million	151 million	3 weeks on 32 k40s

For machine translation, we experimented with the WMT'14 En→Fr and En→De corpora, which have 36M sentence pairs and 5M sentence pairs, respectively. On these translation datasets, we compared our method against recently published Neural Machine Translation baselines in (Luong et al., 2015a; Zhou et al., 2016; Wu et al., 2016). We also experimented the performance of MoEs on Google's internal translation data for English to French language pair, which is two orders of magnitudes larger than the WMT corpora (Wu et al., 2016). A summary of our BLEU score results, computation, and model parameter sizes against state-of-the-art is shown in Table 2. All our results are based on a single model. We achieved our results by significantly increasing the model parameter sizes while incurring far less computation per word.

Table 2: Summary of our results against state-of-the-art. For more details, see Tables 4, 5, and 7.

	WMT En→Fr	WMT En→De	Production En→Fr	Total #Parameters	Computation per Word
Ours	40.56	26.03	36.57	8690 million	100 million
Best Published Results	39.92	24.91	35.56	250 million	215 million

5.1 1 Billion Word Language Model Benchmark

Dataset: We experimented with our method on the 1 Billion Word Language Model Benchmark, introduced by (Chelba et al., 2013). The dataset contains approximately 829 million words with a vocabulary of 793,471 words, including sentence boundary markers. All the sentences are shuffled and the duplicates are removed. The words that are out of vocabulary are replaced by a special UNK token, which constitutes 0.3% of the dataset. The test data consists of 159,658 examples.

5.1.1 SMALL MODELS

We investigated the effect of varying the number of experts while maintaining a very low computational budget: about 6 million multiply-and-adds per word excluding the output layer, or about 4% of the per-word computation used in the state-of-the art models described in (Jozefowicz et al., 2016).

Model Architecture: Our models consist of 2 recurrent Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997; Gers et al., 2000) layers, with a MoE layer between them. We added residual connections between layers to encourage gradient flow (He et al., 2015). The LSTM layers each contain 512 cells, and the experts in the MoE are each simple 512x512 matrices with no non-linearity. We varied the number of experts between models, using ordinary MoE layers with 8, 64 and 512 experts and hierarchical MoE layers with 512, 4096 and 32768 experts. For the hierarchical MoE layers, the first level branching factor was 32. We used the Batchwise Mask described in Appendix A. The average number of experts used per word was $k = 8$. For the hierarchical MoE models, we used $k = 2$ in the primary gating network and $k = 4$ in the secondary gating networks. As a computation-matched baseline, we also trained a model where the MoE layer was replaced with a third LSTM.

Training: The models were trained on a cluster of 32 K40 GPUs using the synchronous method described in Section 4. Each batch consisted of a set of sentences totaling roughly 220,000 words. We trained for 50,000 steps, or about 14 epochs. The step times on the order of 1 second, so training took less than a day. We used the Adam optimizer (Kingma & Ba, 2015). The learning rate was increased linearly for the first 200 training steps, held constant for the next 200 steps, and decreased after that so as to be proportional to the inverse square root of the step number. The Softmax output layer was trained efficiently using importance sampling similarly to the models in (Jozefowicz et al., 2016). We used a dropout rate of 0.2 between all layers during training.

Results: We evaluated our model using perplexity on the holdout dataset, used by (Chelba et al., 2013; Jozefowicz et al., 2016). We follow the standard procedure and sum over all the words including the end of sentence symbol. We report the best measured test perplexity in Table 3. For each model, we report the test perplexity, the number of parameters, and the amount of computation (measured by the number of multiply-and-adds per word in the forward pass excluding the output layer). Test perplexity continued to improve through training step 50,000 on all models except for the largest one (32768 experts), which began to degrade after step 32,000 due to over-fitting. For that model, we report the best test perpleixty. The 8-expert model achieved test perplexity of 50.2, similar to the computation-matched baseline model. Test perplexity improved as much as 26% as the number of experts increased, down to 36.8 with 32768 experts. There was no noticeable difference between the hierarchical and flat MoEs at 512 experts.

Table 3: Perplexity comparison of our method against previous state-of-art models on 1 Billion Word Language Model Benchmark.

Model	Test Perplexity	Computation per Word excluding output layer (millions)	#Parameters excluding embedding and output layers (millions)	Total #Parameters (billions)
LSTM-1024-512 (Jozefowicz et al., 2016)	48.2	4.7	4.7	0.8
LSTM-2048-512 (ibid.)	43.7	9.4	9.4	0.8
LSTM-8192-2048 (ibid.)	32.2	151	151	3.3
2 x LSTM-8192-1024 (ibid.)	30.6	151	151	1.8
2 x LSTM-8192-1024 + CNN inputs (ibid.)	30.0	151	151	1.0
3 x LSTM-512	50.5	6.3	6.3	0.8
2 x LSTM-512, 8 Experts	50.2	6.3	6.3	0.8
2 x LSTM-512, 64 Experts	48.4	6.3	21.0	0.8
2 x LSTM-512, 512 Experts (flat MoE)	41.9	6.6	138.7	1.0
2 x LSTM-512, 512 Experts (hierarchical MoE)	41.3	6.3	138.7	1.0
2 x LSTM-512, 4096 Experts	37.5	6.4	1080.0	1.9
2 x LSTM-512, 32768 Experts	36.8	7.4	8610.9	9.4
3 x LSTM-2048-1024	35.1	56.6	56.6	1.7
2 x LSTM-2048-1024, 32768 Experts	29.9	55.7	34431.1	36.1

5.1.2 BIG MODEL

We ran one additional model with larger LSTMs (2048 units, with a 1024-dimensional projection), and a larger hierarchical MoE, consisting of 32768 experts in a 128x256 hierarchy. Each expert consisted of a 1024x1024 weight matrix. We used $k = 3$ for the primary gating network and $k = 6$ for the secondary gating networks, so on average 18 experts were used per word. We trained the model synchronously on a cluster of 128 k40 GPUs. Each training batch consisted of sentences totaling about 900,000 words. Step times were about 4 seconds. A dropout rate of 0.3 was applied between layers during training. The model was trained for a total of 13300 steps, during which time test perplexity continued to improve. The final test perplexity was 29.9, better than the best single word-level model reported in the literature (Jozefowicz et al., 2016). Our model was also much faster to train (15 hours on 128 k40s vs. 3 weeks on 32 k40s). This is in large part due to the fact that the published model needs to employ much larger LSTMs to achieve the same quality as the MoE model, incurring roughly 2.5 times as much computation per training example. As with the small models, we also trained a computation-matched baseline model with 3 LSTMs according to a similar regimen. The results are reported at the bottom of Table 3.

5.2 MACHINE TRANSLATION ON WMT'14 EN→ FR AND EN→ DE

Dataset: We benchmarked our method on the WMT'14 En→Fr and En→De corpora, where the training sets have 36M sentence pairs and 5M sentence pairs, respectively. The experimental protocols were also similar to those in (Wu et al., 2016): newstest2014 was used as the test set to compare against previous work (Luong et al., 2015a; Zhou et al., 2016; Wu et al., 2016), while the combination of newstest2012 and newstest2013 was used as the development set.

Model Architecture: Our model was a modified version of the GNMT model described in (Wu et al., 2016). To reduce computation, we decreased the number of LSTM layers in the encoder and decoder from 9 and 8 to 3 and 2 respectively. We inserted MoE layers in both the encoder (between layers 2 and 3) and the decoder (between layers 1 and 2). We used an attention mechanism between the encoder and decoder, with the first decoder LSTM receiving output from and providing input for the attention. [2] All of the layers in our model have input and output dimensionality of 512. Our LSTM layers have 2048 hidden units, with a 512-dimensional output projection. We added residual connections around all LSTM and MoE layers to encourage gradient flow (He et al., 2015). Similar to GNMT, to effectively deal with rare words, we used sub-word units (also known as "wordpieces") (Schuster & Nakajima, 2012) for inputs and outputs in our system. We used a model with a shared source and target vocabulary of 32K wordpieces. We also used the same beam search technique as proposed in (Wu et al., 2016).

Each MoE layer in our model is composed of up to 2048 experts. Each expert in the MoE layer is a feed forward network with one hidden layer of size 2048 and ReLU activation. Thus, each expert contains $[512 * 2048] + [2048 * 512] = 2M$ parameters. The output of the MoE layer is passed through a sigmoid function.

Training: We trained our networks using the Adam optimizer and tuned the learning rate as described in previous section. Similarly to (Wu et al., 2016), we used dropout (Zaremba et al., 2014) between all layers during training. Our dropout rate was 0.4. We used longer periods of increasing and constant learning rate at the beginning of training (2000 steps and 8000 steps respectively). Training was done synchronously on a cluster of up to 64 GPUs as described in section 4.

Metrics: We evaluated our models using the perplexity and the standard BLEU score metric. We reported tokenized BLEU score as computed by the multi-bleu.pl script, downloaded from the public implementation of Moses (on Github), which was also used in (Luong et al., 2015a).

Results: We first report the performance of our method as we increase the number of experts in our model. This set of results is shown in Figure 3. Each step approximately processes 200k words. As can be seen from the Figure, as we increased the number of experts to approach 2048, the test

[2] For performance reasons, we used a slightly different attention function from the one described in (Wu et al., 2016) - See appendix B

perplexity of our model continued to improve. This supports the argument of using MoE for training large neural translation models.

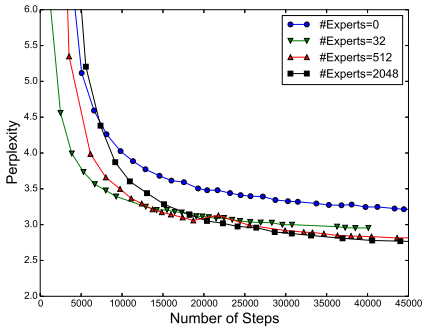

Figure 3: Perplexity on WMT'14n En→ Fr data. All models are identical except for the number of experts in MoE. In all the models except for the one with no experts, the gating network selects exactly 4 experts to process each example.

The results of our method on En→Fr with 2048 experts and other baselines are shown in Table 4. As can be seen from the table, our approach achieved a BLEU score of 40.56. Since our method did not use RL refinement, this result provides a significant gain of 1.34 BLEU score on top of a strong baseline (Wu et al., 2016). The perplexity is also much better given the same size of vocabulary as described in (Wu et al., 2016). Note that our best model (i.e., MoE with 2048 experts) reached to the reported BLEU and perplexities much faster than the baseline while using fewer GPUs.

Table 4: Perplexity and BLEU comparison of our method against previous state-of-art methods on the WMT'14 En→ Fr (newstest2014).

Model	Test Perplexity	Test BLEU	Computation per Word	Total #Parameters	Training Time
MoE with 2048 Experts	2.69	40.35	100.8M	8.690B	3 days/64 k40s
MoE with 2048 Experts (longer training)	2.63	40.56	100.8M	8.690B	6 days/64 k40s
GNMT (Wu et al., 2016)	2.79	39.22	214.2M	246.9M	6 days/96 k80s
GNMT+RL (Wu et al., 2016)	2.96	39.92	214.2M	246.9M	6 days/96 k80s
PBMT (Durrani et al., 2014)		37.0			
LSTM (6-layer) (Luong et al., 2015b)		31.5			
LSTM (6-layer+PosUnk) (Luong et al., 2015b)		33.1			
DeepAtt (Zhou et al., 2016)		37.7			
DeepAtt+PosUnk (Zhou et al., 2016)		39.2			

Table 5: Perplexity and BLEU comparison of our method against previous state-of-art methods on the WMT'14 En → De (newstest2014).

Model	Test Perplexity	Test BLEU	Computation per Word	Total #Parameters	Training Time
MoE with 2048 Experts	4.64	26.03	100.8M	8.690B	1 day/64 k40s
GNMT (Wu et al., 2016)	5.25	24.91	214.2M	246.9M	1 day/96 k80s
GNMT +RL (Wu et al., 2016)	8.08	24.66	214.2M	246.9M	1 day/96 k80s
PBMT (Durrani et al., 2014)		20.7			
DeepAtt (Zhou et al., 2016)		20.6			

The results of our method on En→De with 2048 experts and other baselines are shown in Table 5. As can be seen from the table, our method achieved a BLEU score of 26.03 which is 1.12 better than the best model in (Wu et al., 2016). The perplexity was also much better given the same size of vocabulary as described in (Wu et al., 2016).

We found that the experts indeed become highly specialized by syntax and/or semantics, as can be seen in Table 6. For example, one expert is used when the indefinite article "a" introduces the direct object in a verb phrase indicating importance or leadership.

Table 6: Contexts corresponding to a few of the 2048 experts in the MoE layer in the encoder portion of the WMT'14 En→ Fr translation model. For each expert i, we sort the inputs in a training batch in decreasing order of $G(x)_i$, and show the words surrounding the corresponding positions in the input sentences.

Expert 381	Expert 752	Expert 2004
... with **researchers** , ...	... plays **a** core ...	... with **rapidly** growing ...
... to **innovation** .	... plays **a** critical ...	... under **static** conditions ...
... tics **researchers** .	... provides **a** legislative ...	... to **swift** ly ...
... the **generation** of ...	... play **a** leading ...	... to **dras** tically ...
... technology **innovations** is ...	... assume **a** leadership ...	... the **rapid** and ...
... technological **innovations** , ...	... plays **a** central ...	... the **fast** est ...
... support **innovation** throughout ...	... taken **a** leading ...	... the **Quick** Method ...
... role **innovation** will ...	... established **a** reconciliation ...	... rec **urrent**) ...
... research **scienti** st ...	... played **a** vital ...	... provides **quick** access ...
... promoting **innovation** where ...	... have **a** central ...	... of **volatile** organic ...
...	...	...

5.3 Machine Translation on Production En→Fr Data

Dataset, Model Architecture, and Training: We also tested our method on our production English to French data, which is 2 orders of magnitude larger than the WMT'14 corpus. We used the exact same model architecture and training procedure as described in previous section.

Results: Figure 4 demonstrates the significance of increasing the number of parameters in the model in order to improve learning. It shows the improvement in perplexity on Google Production En→ Fr data on models with different number of experts, from a baseline of 0 expert to a MoE model of 2048 experts. If applicable, in all the models exactly 4 experts were used for processing each example. Each step approximately processes 200k words. As we increased the number of experts, from a baseline of 0 expert to the ones with 2048 experts the perplexity constantly improved. The improvement was achieved while the amount of computation per word was constant across all the models with non-zero experts.

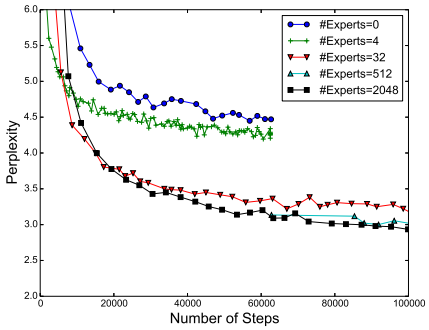

Figure 4: Perplexity on Google Production En→ Fr data. All models are identical except for the number of experts in MoE. In all the models except for the one with no experts, the gating network selects exactly 4 experts to process each example.

Table 7: Perplexity and BLEU comparison of our method against previous state-of-art methods on the Google Production En→ Fr dataset.

Model	Eval Perplexity	Eval BLEU	Test Perplexity	Test BLEU	Computation per Word	Total #Parameters	Training Time
MoE with 2048 Experts	2.60	37.27	2.69	36.57	100.8M	8.690B	1 day/64 k40s
GNMT (Wu et al., 2016)	2.78	35.80	2.87	35.56	214.2M	246.9M	6 days/96 k80s

In Table 7, we report the comparison between our 2048-expert model and the strong and highly optimized baseline models in (Wu et al., 2016). Our model achieved 1.01 higher BLEU score on test data while speeding up computation by a factor of 6× using fewer machines.

6 CONCLUSION

This work introduces Mixture-of-Experts (MoE) as a new type of layer in neural networks. The proposed approach trains an outrageously large network that contains thousands of experts and billions of parameters. We propose a sparse gating approach that enables training such a large network. We describe technical details on our data- and model- parallel system that is integral to MoE layer's efficiency and scalability. On the 1 Billion Word Language Modeling Benchmark and WMT'14 translation benchmarks, our approach's quality surpasses all currently published results. We also show that our approach can be applied to datasets with orders of magnitude more data, to deliver high quality language modeling and translation results. We find that increasing the model capacity well beyond the conventional approaches can lead to significant improvements on learning quality while incurring comparable or faster training time to the baseline models.

ACKNOWLEDGMENTS

We would like to thank all of the members of the Google Brain Team and the Google Translate Team who helped us with this project, in particular Geoffrey Hinton, Zhifeng Chen, Yonghui Wu, and Melvin Johnson.

REFERENCES

Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Y. Hannun, Billy Jun, Patrick LeGresley, Libby Lin, Sharan Narang, Andrew Y. Ng, Sherjil Ozair, Ryan Prenger, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Yi Wang, Zhiqian Wang, Chong Wang, Bo Xiao, Dani Yogatama, Jun Zhan, and Zhenyao Zhu. Deep speech 2: End-to-end speech recognition in english and mandarin. *arXiv preprint arXiv:1512.02595*, 2015.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

Ronan Collobert, Samy Bengio, and Yoshua Bengio. A parallel mixture of SVMs for very large scale problems. *Neural Computing*, 2002.

Marc Peter Deisenroth and Jun Wei Ng. Distributed Gaussian processes. In *ICML*, 2015.

Nadir Durrani, Barry Haddow, Philipp Koehn, and Kenneth Heafield. Edinburgh's phrase-based machine translation systems for wmt-14. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, 2014.

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts. *arXiv preprint arXiv:1312.4314*, 2013.

Ekaterina Garmash and Christof Monz. Ensemble learning for multi-source neural machine translation. In *staff.science.uva.nl/c.monz*, 2016.

Felix A. Gers, Jürgen A. Schmidhuber, and Fred A. Cummins. Learning to forget: Continual prediction with lstm. *Neural Computation*, 2000.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 2012.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computing*, 1991.

Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computing*, 1994.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

Quoc V. Le, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Jeffrey Dean, and Andrew Y. Ng. Building high-level features using large scale unsupervised learning. In *ICML*, 2012.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *EMNLP*, 2015a.

Minh-Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. *ACL*, 2015b.

Carl Edward Rasmussen and Zoubin Ghahramani. Infinite mixtures of Gaussian process experts. *NIPS*, 2002.

M. Schuster and K Nakajima. Japanese and Korean voice search. *ICASSP*, 2012.

Babak Shahbaba and Radford Neal. Nonlinear models using dirichlet process mixtures. *JMLR*, 2009.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

Lucas Theis and Matthias Bethge. Generative image modeling using spatial LSTMs. In *NIPS*, 2015.

Volker Tresp. Mixtures of Gaussian Processes. In *NIPS*, 2001.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Bangpeng Yao, Dirk Walther, Diane Beck, and Li Fei-fei. Hierarchical mixture of classification experts uncovers interactions between brain regions. In *NIPS*. 2009.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. Deep recurrent models with fast-forward connections for neural machine translation. *arXiv preprint arXiv:1606.04199*, 2016.

APPENDICES

A STRICTLY BALANCED EXPERT UTILIZATION

For performance purposes, we would like the different experts (which reside on different devices) to each receive the same number of examples from each training batch. To this end, we introduce an alternative mask function, $M_{batchwise}(X, m)$, which operates over batches of input vectors. Instead of keeping the top k values per example, we keep the top m values per expert across the training batch. We generally choose $m = \frac{k|X|}{n}$ for some small value k, so that each example is sent to an average of k experts.

$$M_{batchwise}(X, m)_{j,i} = \begin{cases} 1 & \text{if } X_{j,i} \text{ is in the top } m \text{ values for to expert } i \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

As our experiments suggest and also observed in (Ioffe & Szegedy, 2015), using a batchwise function during training (such as $M_{batchwise}$) requires modifications to the inference when we may not have a large batch of examples. Our solution to this is to train a vector T of per-expert threshold values to approximate the effects of the batchwise mask. We use the following mask at inference time:

$$M_{threshold}(x, T)_i = \begin{cases} 1 & \text{if } x_i > T_i \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

To learn the threshold values, we apply an additional loss at training time which is minimized when the batchwise mask and the threshold mask are identical.

$$L_{batchwise}(X, T, m) = \sum_{j=1}^{|X|} \sum_{i=1}^{n} (M_{threshold}(x, T)_i - M_{batchwise}(X, m)_{j,i})(X_{j,i} - T_i) \tag{8}$$

B ATTENTION FUNCTION

The attention mechanism described in GNMT (Wu et al., 2016) involves a learned "Attention Function" $A(x_i, y_j)$ which takes a "source vector" x_i and a "target vector" y_j, and must be computed for every source time step i and target time step j. In GNMT, the attention function is implemented as a feed forward neural network with a hidden layer of size n. It can be expressed as:

$$A_{GNMT}(x_i, y_j) = \sum_{d=1}^{n} V_d tanh((x_i U)_d + (y_j W)_d) \tag{9}$$

Where U and W are trainable weight matrices and V is a trainable weight vector.

For performance reasons, in our models, we used a slightly different attention function:

$$A(x_i, y_j) = \sum_{d=1}^{n} V_d tanh((x_i U)_d) tanh((y_j W)_d) \tag{10}$$

With our attention function, we can simultaneously compute the attention function on multiple source time steps and multiple target time steps using optimized matrix multiplications. We found little difference in quality between the two functions.