DeepSpeed-MoE: Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale

Samyam Rajbhandari [1] **Conglong Li** [1] **Zhewei Yao** [1] **Minjia Zhang** [1] **Reza Yazdani Aminabadi** [1] **Ammar Ahmad Awan** [1] **Jeff Rasley** [1] **Yuxiong He** [1]

Abstract

As the training of giant dense models hits the boundary on the availability and capability of the hardware resources today, Mixture-of-Experts (MoE) models have become one of the most promising model architectures due to their significant training cost reduction compared to quality-equivalent dense models. Their training cost saving is demonstrated from encoder-decoder models (prior works) to a 5x saving for auto-aggressive language models (this work). However, due to the much larger model size and unique architecture, how to provide fast MoE model inference remains challenging and unsolved, limiting their practical usage. To tackle this, we present DeepSpeed-MoE, an end-to-end MoE training and inference solution, including novel MoE architecture designs and model compression techniques that reduce MoE model size by up to 3.7x, and a highly optimized inference system that provides 7.3x better latency and cost compared to existing MoE inference solutions. DeepSpeed-MoE offers an unprecedented scale and efficiency to serve massive MoE models with up to 4.5x faster and 9x cheaper inference compared to quality-equivalent dense models. We hope our innovations and systems help open a promising path to new directions in the large model landscape, a shift from dense to sparse MoE models, where training and deploying higher-quality models with fewer resources becomes more widely possible.

1. Introduction

In the last three years, the largest trained model has increased in size by over 1000x, from a few hundred million parameters to half a trillion parameters (Megatron-Turing NLG 530B (Smith et al., 2022)). Improvements in model quality with size suggest that this trend will continue, with larger model sizes bringing better model quality.

However, sustaining the growth in model size is getting more and more difficult due to the increasing compute requirements. For example, the largest single dense model in existence as of Dec 2021, the Megatron-Turing NLG 530B model, took around 3 months to train on over 2000 A100 GPUs on the NVIDIA Selene Supercomputer, consuming over 3 million GPU hours (Microsoft & Nvidia, 2021). Another 3 to 5 times of increase in dense model size would be infeasible within a reasonable timeframe. Given the exorbitant compute resources required to train the state-of-art models, a natural question to ask is: "Is it possible to make non-trivial improvement to model quality without increasing the compute cost?" Or equivalently, "Is it possible to produce model with similar quality using 3 to 5 times less resources?"

There have been numerous efforts to reduce the compute requirements to train large models without sacrificing model quality. To this end, architectures based on Mixture-of-Experts (MoE) (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2021a) have paved a promising path, enabling sub-linear compute requirements with respect to the model parameters and allowing for improved model quality without increasing training cost. However, MoE based models have their own set of challenges that limit their use in a wide range of real world scenarios:

- **Limited Scope** The scope of MoE based models in the NLP area is primarily limited to encoder-decoder models and sequence-to-sequence tasks, with limited work done in exploring its application in other domains. Application of MoE to auto-regressive natural language generation (NLG) like GPT-3 and MT-NLG 530B, where the compute cost of training state-of-art language models can be orders of magnitude higher than for encoder-decoder models, is less explored.
- **Massive Memory Requirements** While MoE models require less compute to achieve the same model qual-

[1]Microsoft. Correspondence to: Samyam Rajbhandari <samyamr@microsoft.com>, Yuxiong He <yuxhe@microsoft.com>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

Author contributions listed at Appendix A.

ity as their dense counterparts, they need significantly more number of parameters. For example, the MoE based Switch-Base model has 10x more parameters than T5-large (7.4B vs 0.74B) and still it does not have the same model quality when compared across a wide range of downstream tasks (Fedus et al., 2021a). In other words, MoE based models have a much lower "parameter efficiency" compared to quality-equivalent dense models. For instance, if we scale the dense model size to MT-NLG equivalent with 530B parameters, achieving similar quality with MoE based model might need a model with over 5 trillion parameters (assuming the 10x scaling still holds), which would require over 5K GPUs to just fit the model states for training.

- **Limited Inference Performance** Due to the large model size and poor parameter efficiency mentioned above, fast inference of MoE based models is even more challenging. On one hand, the larger parameter size requires more GPUs to fit, and multi-gpu inference technology is not designed to work with MoE based models. On the other hand, as inference is often memory bandwidth bound, MoE based models, which can be 10x larger than their dense equivalent, could require 10x higher achievable memory bandwidth to achieve similar inference latency as the dense models.

Despite the promising and non-trivial reduction in training cost, these above mentioned challenges severely limits the practical applicability of MoE. In an effort to make MoE practical, accessible and applicable, in this paper, we address these challenges by offering three corresponding solutions:

- **We expand the scope of MoE based models** to auto-regressive NLG tasks, demonstrating training cost reduction of 5x to achieve same model quality for models like GPT-3 and MT-NLG (see Section 3). These results not only demonstrate clear opportunities to reduce the cost of training massive NLG models, but also opens up the possibilities to reach much higher next-generation model quality under the limitation of current generation hardware resource.
- **We improve parameter efficiency of MoE based models** by developing a novel MoE architecture that we call Pyramid-Residual MoE (PR-MoE). PR-MoE is a hybrid dense and MoE model created using residual connections, while applying experts only where they are most effective. PR-MoE can reduce parameters by up to 3x with no change to model quality. In addition, we leverage staged knowledge distillation to create a distilled version of PR-MoE, which we call Mixture-of-Students (MoS), that further reduce model size by 12.5% while retaining 99.3% performance of the teacher (see Section 4).
- **We develop DeepSpeed-MoE inference system**, a highly optimized MoE inference system which enables efficient scaling of inference workloads on hundreds of GPUs, providing up to 7.3x reduction in inference latency and cost when compared with existing MoE inference solutions (see Section 5). It offers ultra-fast inference latencies (under 25 ms) for trillion-parameter MoE models. DeepSpeed-MoE also offers up to 4.5x faster and 9x cheaper inference for MoE models compared to quality-equivalent dense models by combining both system and model optimizations.

Together, our innovations and systems enable MoE to be a more effective and economic alternative comparing to dense models, achieving significantly lower training and inference cost while obtaining the same model quality. We hope DeepSpeed-MoE helps open a promising path to new directions in the large model landscape, a shift from dense to sparse MoE models, where training and deploying higher-quality models with fewer resources becomes more widely possible.

Software The generic DeepSpeed-MoE end-to-end framework for training and inference of MoE-based models is open-sourced as part of the DeepSpeed software, and the experiments presented in this paper were conducted on the Microsoft Azure AI platform. Please find the code, tutorials, and documents at DeepSpeed GitHub (`https://github.com/microsoft/DeepSpeed`) and website (`https://www.deepspeed.ai/`).

2. Related Work

Large Scale Dense NLP Models To test and verify the upper bound of scaling law (Kaplan et al., 2020) for model capacity with respect to number of parameters, the pretrained natural language processing model size has been increasing 10x per year for the last several years, e.g., models with millions of parameters (Devlin et al., 2019; Yang et al., 2019; Liu et al., 2019; Lan et al., 2019; Radford et al., 2018), etc, to models with dozens of billions parameters (Radford et al., 2019; Rosset, 2020; Shoeybi et al., 2019; Raffel et al., 2019). The GPT-3 175B (Brown et al., 2020) and MT-NLG 530B model (Smith et al., 2022) further push this limit to hundreds of billions of parameters, and they are shown to have better generalization performance on various of natural language understanding and generation tasks (Paperno et al., 2016; Wang et al., 2018; 2019; Mostafazadeh et al., 2016; Berant et al., 2013; Joshi et al., 2017). As the training of MT-NLG takes 3 months on over 2000 A100 GPUs, it is no longer feasible to achieve better model quality by simply increasing the model size due to unsurmountable compute requirements.

Reducing Training Cost by MoE Architecture One promising way to reduce the training cost is using Mixture of Expert (MoE) (Masoudnia & Ebrahimpour, 2014). In (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2021a;a; Lin et al., 2021; Kim et al., 2021b; Zuo et al., 2021), different base model with various model sizes are trained for encoder-decoder models and sequence-to-sequence tasks, e.g., language modelings and machine translation, and significant training cost reduction is observed. A few recent and parallel works (Du et al., 2021; Artetxe et al., 2021) show that MoE model can also be applied to auto-regressive natural language generation tasks. However, our work has major differences compared to these parallel explorations: (1) our work investigates training, model design, and inference opportunities of MoE models while (Du et al., 2021; Artetxe et al., 2021) primarily focuses on MoE training; (2) we propose PR-MoE architecture and MoS knowledge distillation to achieve better MoE parameter efficiency and on-par/better zero-shot eval quality as described in Section 3; (3) we develop DeepSpeed-MoE inference system to efficiently serve large scale MoE models with high throughput and low latency. While recent studies like (Du et al., 2021) and (Artetxe et al., 2021) discuss reduction of FLOPs, it is pertinent to mention that unlike training, inference latency and cost do not depend on computation alone. Efficient inference depends on model size, memory bandwidth, and the capability of a system to read data from memory efficiently.

MoE Training Systems DeepSpeed MoE training system (Kim et al., 2021a) was primarily targeted for optimized training of MoE models at scale. It supports up to 8x bigger model sizes by leveraging flexible combinations of different types of parallelism including tensor-slicing, data parallelism, ZeRO (Rajbhandari et al., 2020)-powered data parallelism, and expert parallelism. FastMoE (He et al., 2021) is a research software developed to show how MoE models can be trained under data and expert (model) parallelism. The combination of various parallelism dimensions is not fully supported. Fairseq-MoE (Artetxe et al., 2021) offers an MoE API as well as a training pipeline for generic language models. The Fairseq system has been further optimized by Tutel (Microsoft, 2021), which offers up to 40 percent improvement over Fairseq. Unlike our work, we note that none of these systems are optimized for inference.

3. DeepSpeed-MoE for NLG: Reducing the Training Cost of Language Models by 5 Times

Transformer-based natural language generation (NLG) models offer convincing solutions to a broad range of language tasks. Given the tremendous compute and energy requirements for training NLG family of models, we explore the opportunities that MoE presents to reduce their training cost. We show that MoE-based NLG model can achieve 5x reduction in training cost to achieve the same model quality of a dense NLG model.

Model Architecture We studied MoE architecture for the GPT-3 like NLG model (Brown et al., 2020). The following models are selected: 350M/1.3B/6.7B (24/24/32 layers, 1024/2048/4096 hidden size, 16/16/32 attention heads). We use "350M+MoE-128" to denote a MoE model that uses 350M dense model as the base model and adds 128 experts on every other feedforward layer. We use a top-1 gating function to activate a single expert in the MoE layer for each token. Therefore, during both training and inference, our MoE model will have the same number of parameters to be activated for each token as their dense part (Figure 3 (left)).

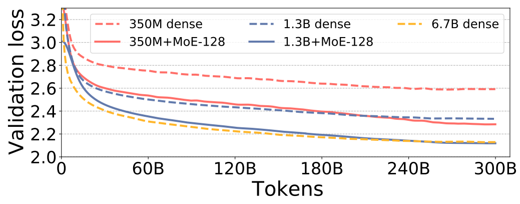

Figure 1: Token-wise validation loss curves for dense and MoE NLG models with different model sizes.

Training and Evaluation Settings We pre-trained both the dense and MoE models on 128 NVIDIA Ampere A100 GPUs (Azure ND A100 instances), using the same training data as described in (Microsoft & Nvidia, 2021). We use 300B tokens to train both dense and MoE models. In addition to the pre-training validation loss, we employ 6 zero-shot evaluation tasks to compare the final model quality: LAMBADA (Paperno et al., 2016), PIQA (Bisk et al., 2020), BoolQ (Wang et al., 2019), RACE-h (Lai et al., 2017), TriviaQA (Joshi et al., 2017), WebQs (Berant et al., 2013). Appendix B summarizes the hyperparameters for training the dense and MoE models. For dense models we followed the hyperparameters from the GPT-3 work. MoE models have two additional hyperparameters: the number of experts per MoE layer, and a coefficient when adding the MoE layer losses to the total training loss.

MoE Leads to Better Quality for NLG Models Figure 1 shows that the validation loss of the MoE models is significantly better than their dense counter parts (e.g., 1.3B+MoE-128 versus 1.3B dense). In addition, MoE models are on par with the validation loss of the dense models with 4-5x larger base (e.g., 1.3B+MoE-128 versus 6.7B dense). Furthermore, the model quality is also on par in terms of the zero-shot evaluation as shown in Table 1.

Same Quality with 5x Less Training Cost Adding MoE to the NLG model significantly improves the model quality. In addition, these experts do not change the compute requirements of the model as each token is only processed by a single expert. A 1.3B+MoE-128 model requires roughly the same amount of training compute as 1.3B dense model, while offering much better model quality. Furthermore, the 1.3B+MoE-128 model can achieve the model quality of the 6.7B dense model at the training cost of 1.3B parameter dense model, resulting in a 5x training compute reduction:

Table 1: Zero-shot evaluation results (accuracy metric) for different dense and MoE NLG models.

	Model (num. params)	LAMBADA	PIQA	BoolQ	RACE-h	TriviaQA	WebQs
Dense:	350M (350M)	52.03	69.31	53.64	31.77	3.21	1.57
	1.3B (1.3B)	63.65	73.39	63.39	35.60	10.05	3.25
	6.7B (6.7B)	71.94	76.71	67.03	37.42	23.47	5.12
Standard MoE:	350M+MoE-128 (13B)	62.70	74.59	60.46	35.60	16.58	5.17
	1.3B+MoE-128 (52B)	69.84	76.71	64.92	38.09	31.29	7.19

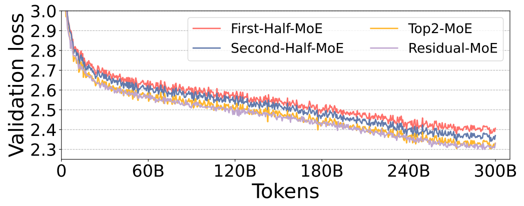

Figure 2: The validation loss of First-Half-MoE/Second-Half-MoE and Top2-MoE/Residual-MoE based on 350M+MoE models.

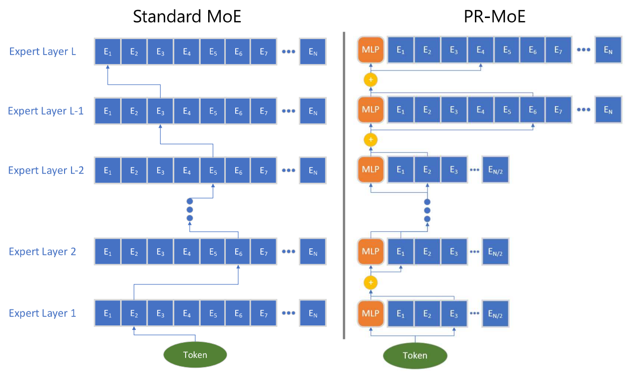

Figure 3: The illustration of standard MoE (left) and PR-MoE (right).

on 128 A100 GPUs the 6.7B dense model achieves a training throughput of 70 samples/sec, while the 1.3B+MoE-128 model achieves 372 samples/sec.

4. PR-MoE and MoS: Reducing the Model Size and Improving Parameter Efficiency

While MoE based models achieve the same quality with 5x training cost reduction in the NLG example, the resulting model has roughly 8x the parameters of the corresponding dense model. Such a massive MoE model requires significantly more memory during training, and it is challenging to meet latency requirements for such models during inference as memory bandwidth consumed to read the model weights is the primary performance bottleneck in inference. To reduce the number of parameters and improve the parameter efficiency of MoE based models, we present innovations in the MoE model architecture (called PR-MoE) and we design a novel MoE-to-MoE knowledge distillation technique to create a distilled version of PR-MoE, which we call Mixture-of-Students (MoS).

4.1. PR-MoE: Pyramid-Residual-MoE for Smaller Model Size and Fast Inference

4.1.1. Two Observations and Intuitions

Phenomenon-I First, the standard MoE architecture has the same number and structure of experts in all MoE layers. This reminds us a fundamental question in machine learning community: do all the layers in a Deep Neural Network learn the same representation? This question has been well-studied in Computer Vision (CV): shallow layers (close to inputs) learn general representations and deep layers (close to outputs) learn more objective specific representations (Zeiler & Fergus, 2014). This also inspired transfer learning in CV to freeze shallow layers for fine-tuning (Yosinski et al., 2014). This phenomenon, however, has not been well-explored in NLP, particularly for MoE architectures. To investigate this question, we compare the performance of two different Half-MoE architectures based on the 350M+MoE model. More specifically, (1) we put MoE layers in the first half layers of the model and leave the second half of layers identical to dense model (referred to as First-Half-MoE), and (2) we switch the MoE layers to the second half and use dense at the first half (referred to as Second-Half-MoE). The results are shown in Figure 2. As can be seen, Second-Half-MoE has significantly better performance than its counterpart. This confirms that not all MoE layers learn the same level of representations. Deeper layers benefit more from large number of experts.

Phenomenon-II Second, to improve the generalization performance of MoE models, there are two common methods: (1) increasing the number of experts while keeping the Top-1 expert selection (aka for each token, the number of experts it goes through) to be the same; (2) using Top-2 expert selection at the expense of slightly more computation (33%) while keeping the same number of experts. However, for (1), the memory requirement for training resources needs to be increased due to larger number of experts; for (2), higher capacity also doubles the communication volume which can significantly slow down training and inference. Is there a way to keep the training/inference efficiency while getting generalization performance gain? One intuition of why larger expert capacity helps accuracy is that those extra experts can help correct the "representation" of the first one. However, does this first expert need to be changed every time? Or can we fix the first expert and only assign different extra experts to different tokens? To investigate this unknown property, we perform a comparison in two ways (1) doubling the capacity (referred to as Top2-MoE), and (2) fixing one expert and varying the second expert across different experts (referred to as Residual-MoE). Particularly, for (2), a token will always pass a dense MLP module and an expert from MoE module, which can be viewed as a special case of residual network. Afterward, we add the output of these two branches together to get the final output. The main intuition is to treat the expert from MoE module as an error correction term of the dense MLP module. Such that, we can achieve the benefit of using 2 experts per layer with the same amount communication volume as Top-1 gating function. We perform the comparison for the 350M+MoE model with 32 experts and the validation curves are presented in Figure 2. We find out that the generalization performance of these two (aka Top2-MoE and Residual-MoE) is on-par with each other. However, the training speed of our new design, Residual-MoE, is more than 10% faster than Top2-MoE due to the communication volume reduction.

4.1.2. Pyramid Residual MoE Architecture and Its Training System Design

Based on the above, we propose our novel MoE architecture. As Phenomenon-I in Section 4.1.1 suggested that leveraging MoE at the later layers bring more benefits, our new architecture utilizes more experts in the last few layers as compared to previous layers. This gives the Pyramid-MoE design, where we show an example in Figure 3 (right)–the last two layers have 2x experts as the previous layers. Meanwhile, considering Phenomenon II, we propose the Residual-MoE architecture, where each token separately passes one fixed MLP module and one chosen expert as shown in Figure 3 (right), where orange blocks are the fixed MLP. By combining Pyramid-MoE and Residual-MoE together, we have our Pyramid-Residual-MoE model (PR-MoE in short), where all standard MoE layers are replaced by the new PR-MoE layer. Figure 3 shows the illustration of standard-MoE and PR-MoE architectures.

Designing a training infrastructure that can efficiently train PR-MoE model is non-trivial due to the presence of different number of experts at different stages of the model. To address the training balance (e.g., token per-expert and/or memory requirements per GPU) of different MoE layers, we develop and implement a flexible multi-expert and multi-data parallelism design on top of DeepSpeed-MoE, that allows for training different parts of the model with different expert and data parallelism degree. For instance, a PR-MoE model running on 128 GPUs, with 32, 64, and 128 experts at different MoE layers, can be trained with 128-way data parallelism for the non-expert parallelism, and {32, 64, 128} expert parallelism plus {4, 2, 1} data parallelism for MoE parameters. Please see Appendix C.1 for more details.

4.1.3. Evaluation of PR-MoE

We evaluate our new architecture, PR-MoE on two different sizes of models, i.e., base size of 350M and 1.3B, and compare the performance with larger Standard-MoE architectures. More specifically, we compare 350M+PR-MoE-32/64 with 350M+MoE-128, and we compare 1.3B+PR-MoE-64/128 with 1.3B+MoE-128. The results are shown in Table 2. For both 350M and 1.3B cases, our PR-MoE uses much fewer parameters but achieves comparable accuracy as Standard-MoE models. Particularly, (1) for 350M case, PR-MoE only uses less than 1/3 of the parameters as Standard-MoE; (2) for 1.3B case, PR-MoE only uses about 60% of the parameters as Standard-MoE, while achieving similar accuracy. We also conduct an experiment to compare the performance between different MoE architectures in Appendix C.2.

4.2. Mixture-of-Students: Distillation for Even Smaller Model Size and Faster Inference

Knowledge distillation (KD) has been proven to be a successful way to compress a large model into a small one (Hinton et al., 2015), which contains much fewer parameters and computations but still obtaining competitive results. There have been some works that apply KD to task-specific distillation of pre-trained large LMs into small models (Sanh et al., 2019; Sun et al., 2019; Wang et al., 2020; Sun et al., 2020). However, they only consider small transformers (a few hundreds of parameters) and dense encoder-based LM models (e.g., BERT). In contrast, we focus on studying KD for sparse MoE-based auto-generative LMs models on multi-billion parameter scale. The only other analysis of MoE distillation we are aware of are by (Fedus et al., 2021b; Artetxe et al., 2021), who study distillation of MoE into dense models. In contrast, our study show that it is possible to reach similar performance, such as zero-shot evaluation on many downstream tasks, for smaller MoE model pretrained with knowledge distillation, resulting in models that are lighter and faster during inference time.

4.2.1. Mixture-of-Students via Staged KD

To apply knowledge distillation for MoE, we reduce the depth of each expert branch in the teacher model to obtain a corresponding Mixture-of-Students (MoS) model. Since MoE structure brings significant benefits by enabling sparse training and inference, our task-agnostic distilled Mixture-of-Students inherits these benefits while preserving the inference advantage over its quality equivalent dense model. We then take a general formulation of the KD loss (Yu et al., 2013) to force the MoS to imitate the outputs from the teacher MoE as:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim D}[\mathcal{L}(x;\theta) + \alpha \mathcal{L}_{KD}(x';\theta)], \qquad (1)$$

Table 2: Zero-shot evaluation comparison between standard MoE, PR-MoE, MoS.

Model (num. params)	LAMBADA	PIQA	BoolQ	RACE-h	TriviaQA	WebQs
350M+MoE-128 (13B)	62.70	**74.59**	**60.46**	35.60	**16.58**	5.17
350M+PR-MoE-32/64 (4B)	**63.65**	73.99	59.88	**35.69**	16.30	4.73
350M+PR-MoE+L21+MoS (**3.5B**)	63.46	73.34	58.07	34.83	13.69	**5.22**
1.3B+MoE-128 (52B)	69.84	76.71	64.92	**38.09**	**31.29**	7.19
1.3B+PR-MoE-64/128 (31B)	**70.60**	**77.75**	**67.16**	**38.09**	28.86	7.73
1.3B+PR-MoE+L21+MoS (**27B**)	70.17	77.69	65.66	36.94	29.05	**8.22**

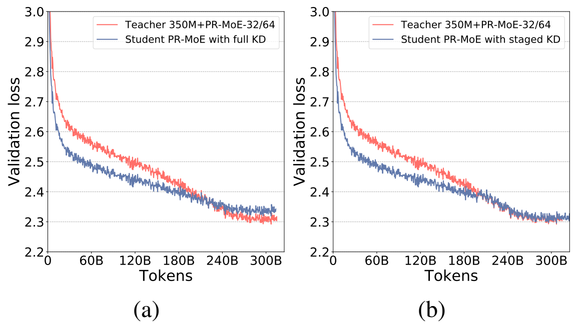

Figure 4: (a) The validation curves of training without distillation from scratch vs. performing knowledge distillation for the entire pre-training process. In the figure the student PR-MoE is trained with 21-layer. KD helps initially but starts to hurt accuracy towards the end of training. (b) The validation curves of training the student PR-MoE with staged knowledge distillation obtains almost the same validation loss as the teacher PR-MoE on 350M+PR-MoE.

which measures a weighted sum of the cross-entropy loss between predictions and the given hard label and the KL divergence loss between the predictions and the teacher's soft label (e.g., $\alpha = 1$). Furthermore, given the excellent performance of PR-MoE, we combine PR-MoE together with KD (PR-MoS) to further reduce the MoE model sizes.

Improving Student Accuracy with Staged Knowledge Distillation One interesting observation during distilling MoE model is that using the teacher PR-MoE leads to lower student accuracy than a PR-MoE student trained from scratch. As KD often improves the student generalization, this raises a question on why KD does not improve accuracy for pre-training MoE on generative language model. Since no prior experiments reported distillation experiment results on distilled MoE for NLG, we dig deeper into the results. Figure 4a shows a comparison of validation loss between a PR-MoE trained from scratch and using knowledge distillation with its teacher. We find that while KD loss improves validation accuracy initially, it begins to hurt accuracy towards the end of training (e.g., after 400K steps).

We hypothesize that because the PR-MoE already reduces the capacity compared with the standard MoE by exploiting the more parameter efficient architecture (e.g., reducing experts in lower layers), further reducing the depth of the model causes the student to have insufficient capacity, making it fall into the underfitting regime. Therefore, the student PR-MoE may not have enough capacity to minimize both the training loss and the knowledge distillation loss, and might end up minimizing one loss (KD loss) at the expense of the other (cross entropy loss), especially towards the end of training. The aforementioned hypothesis suggests that we might want to either gradually decay the impact from KD or stop KD early in the training process and perform optimization only against the standard language modeling loss for the rest of the training.

4.2.2. Evaluation of Mixture-of-Students

We evaluate our approach on two different PR-MoE model configs, 350M+PR-MoE-32/64 and 1.3B+PR-MoE-64/128. We build student models by reducing the depth of the teachers to 21 (12.5% depth reduction) in both cases and compare the resulting MoS model with its teacher using the method described in the previous section.

Figure 4b shows the validation curve comparison of stopping KD at 400K steps and its comparison with the teacher model. We find that this staged version of KD now gives us the promised benefit of knowledge distillation: the student model now has a similar validation curve as the teacher towards the end of training. The evaluation results on downstream tasks in Table 2 also show that the MoS via staged KD achieves an average accuracy of 42.87 and 47.96, retaining 99.5% and 99.1% performance of the 350M (43.08) and 1.3B teacher model (48.37) despite having 12.5% fewer layers. We also conduct ablation studies to compare (1) MoS with alternative baselines in Appendix C.3 and (2) DeepSpeed-MoE with other concurrent work (Du et al., 2021; Artetxe et al., 2021) in Appendix C.4.

5. DeepSpeed-MoE Inference: Serving MoE Models at Unprecedented Scale and Speed

Optimizing inference latency and cost is crucial for MoE models to be useful in practice. During inference, the batch size is generally small, so the inference latency of an MoE model depends primarily on the time it takes to load the model parameters from the main memory, contrasting with the conventional belief that lesser compute should lead to faster inference. Therefore, the MoE inference performance depends on two main factors: the overall model size and the overall achievable memory bandwidth.

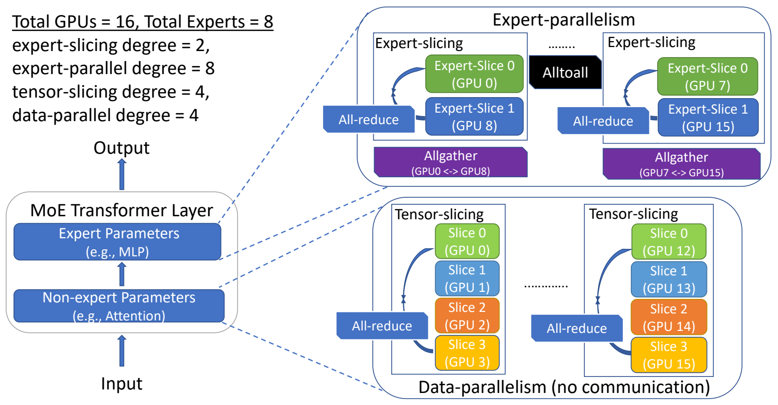

Figure 5: DeepSpeed-MoE design that embraces the complexity of multi-dimensional parallelism for different partitions (expert and non-expert) of the model.

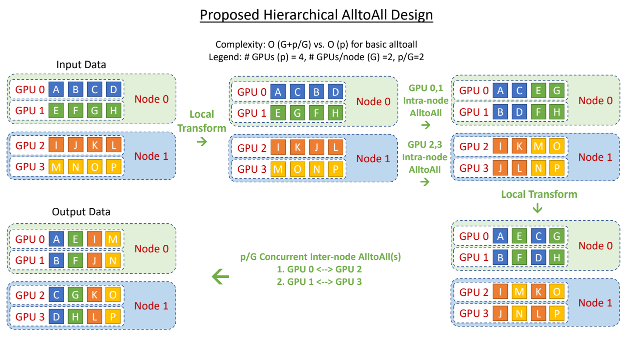

Figure 6: Proposed hierarchical all-to-all design.

While PR-MoE and MoS (previous section) help to reduce the MoE model size while preserving the model accuracy, we now present our multi-GPU MoE inference system that maximizes and leverages the aggregated memory bandwidth across hundreds of GPUs to speed up inference at an unprecedented scale.

5.1. Design of DeepSpeed-MoE Inference System

MoE inference performance is an interesting paradox. From the best-case view, each input token of an MoE model only activates a single expert at each MoE layer, resulting in a critical data path that is equivalent to the base dense model size (e.g. 1.3 billion), orders-of-magnitude smaller than the actual model size (52 billion). From the worst-case view, the aggregate parameters needed to process a batch of tokens (e.g., a sentence or a paragraph of text) can be as large as the full model size since different tokens could activate different experts making it challenging to achieve short latency and high throughput.

The design of DeepSpeed-MoE inference system (Figure 5) ensures that we steer the performance toward the best case. Here, we offer a brief summary of the main optimizations. For more details, please refer to (Rajbhandari et al., 2022).

Expert, Tensor and Data parallelism We use tensor parallelism, referred in Figure 5 as tensor-slicing (for non-expert parameters) and expert-slicing (for expert parameters), to split individual parameters across multiple GPUs to leverage the aggregate memory bandwidth across GPUs. However, tensor parallelism can only scale efficiently to a few GPUs due to communication overhead and fine-grained parallelism. To address this, we use expert parallelism in conjunction with tensor parallelism to scale experts parameters to hundreds of GPUs. Expert parallelism does not reduce computation granularity of individual operators, therefore allowing our system to leverage aggregate memory bandwidth across hundreds of GPUs. To scale the non-expert computation to the same number of GPUs, we use data parallelism at no communication overhead.

Hierarchical All-to-all Hierarchical tree-based algorithms are often used with communication collectives like allreduce, broadcast, etc to reduce the number of communication hops. We implement a hierarchical all-to-all as a two-step process with a data-layout transformation, followed by an intra-node all-to-all, followed by a second data-layout transformation, and a final inter-node all-to-all. This reduces the communication hops from $O(p)$ to $O(G + p/G)$, where G is the number of GPUs in a node and p is the total number of GPU devices. Figure 6 shows the design overview of this implementation. Despite the 2x increase in communication volume, this hierarchical implementation allows for better scaling for small batch sizes as communication at this message size is more latency-bound than bandwidth-bound.

Parallelism Coordinated Communication All-to-all communication latency increases linearly with the number of GPUs. To avoid this linearly increase, we leverage the data redundancy created by tensor-slicing (Rajbhandari et al., 2020) to limit the GPUs that participate in all-to-all . Since each GPU in tensor parallelism contains the same data, the all-to-all communication can be limited within GPUs with the same tensor-parallelism rank. This reduces the total number of GPUs participating in the communication by the tensor-slicing degree.

Kernel Optimizations We designed custom multi-GPU kernel optimizations to maximize aggregate memory bandwidth for transformer operators. For MoE gating functions, we use sparse data structures instead of commonly used dense representations that contains cubic number of zeros and quadratic number of non-zeros with respect to the number of input tokens. Thus our approach reduces the compute complexity from cubic to quadratic. The details of our strategy are as follows:

First, we fuse the gating function into a single kernel, and use a dense token-to-expert mapping table to represent the assignment from tokens to experts, greatly reducing the kernel launch overhead, as well as memory and compute overhead from the dense representation of sparse data. More specifically, gating kernel includes top-k, *cumsum*, and scatter operations in order to distribute the right tokens to each expert. The top-k operator selects the *k* experts with the k-highest logits for each input token, and since *k* is normally small (e.g., 1 or 2) for the MoE models, we store the best expert-indices in a mapping table rather than creating a mask for the rest of gating function operations. *Cumsum* calculates the ID for the tokens processed by each expert, that is defined by the capacity-factor in the MoE configuration. We use the so-called Blelloch scan algorithm to parallelize *cumsum* on GPU architecture. Finally, we use the mapping table and token IDs in order to route the correct tokens to the MoE experts.

Second, to optimize the remaining two sparse einsums, we implement them as data-layout transformations using the above-mentioned mapping table, to first sort them based on the expert id and then back to its original ordering without requiring any sparse einsum, reducing the complexity of these operations from $S \times E \times M \times c_e$ to $S \times M \times c_e$. Together with the data transformation, we use the corresponding gating logits (in the probability domain) to update the expert output. Combined, these optimizations result in over 6x reduction in MoE Kernel related latency.

5.2. Performance Evaluation of DeepSpeed-MoE Inference

In this section, we explore how two broad goals of *high throughput* and *low latency* can be realized for MoE model inference at scale. We also explore how MoE model inference is different compared to their dense counterparts. Model configurations are shown in Table 6.

Achieving Low Latency and Super-linear Throughput Increase Simultaneously We scale a 52B MoE model (1.3B base model and 128 experts) from 8 GPUs to 64 GPUs and observe the latency and throughput trends on DeepSpeed-MoE inference system comparing with a strong baseline, a full-featured distributed PyTorch implementation that is capable of both tensor-slicing and expert-parallelism (Kim et al., 2021a). Figure 8 shows that both DeepSpeed-MoE and PyTorch reduce the inference latency as we increase the number of GPUs, as expected, although PyTorch is much slower compared to DeepSpeed-MoE. Furthermore, DeepSpeed-MoE obtains increased throughput per GPU when we increase the number of GPUs from 8 to 64 and hence a super-linear increase in total throughput as shown in Figure 8. This is in stark contrast to dense models (best case throughput scaling is linear with respect to the number of GPUs) and shows the major benefit of scaling MoE models over dense models.

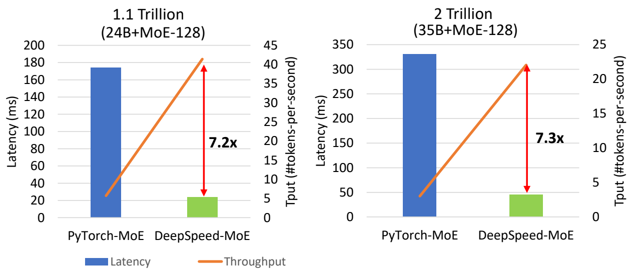

Figure 7: Latency/throughput improvement offered by DeepSpeed-MoE (Optimized) over PyTorch (Baseline).We ran the experiments on 256 GPUs (throughput is reported per GPU).

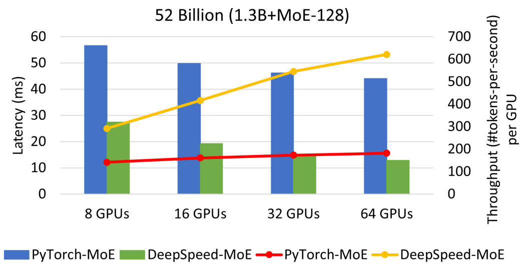

Figure 8: Latency (bars) and throughput (lines) improvement offered by DeepSpeed-MoE inference system (Optimized) over PyTorch (Baseline) for a 52-Billion standard MoE model with 128 experts, using between 8 to 64 GPUs.

Low Latency and High Throughput at Unprecedented Scale Figure 7 shows the performance and throughput of two MoE models with one and two trillion parameters. Compared to baseline, DeepSpeed-MoE achieves up to 7.3x reduction in latency while achieving up to 7.3x higher throughput. By effectively exploiting hundreds of GPUs in parallel, DeepSpeed-MoE achieves an unprecedented scale for inference at incredibly low latencies - a staggering trillion parameter MoE model can be served under 25ms.

Enhanced Benefits of PR-MoE and MoS By leveraging model innovations of PR-MoE and MoS, DeepSpeed-MoE delivers two more benefits as shown in Figure 9: (1) reduce

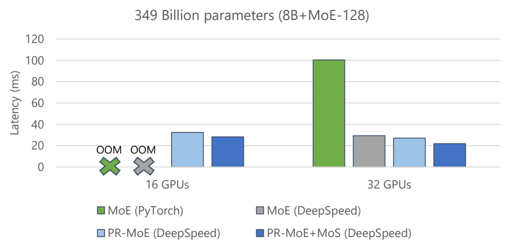

Figure 9: 2x fewer resources needed for MoE inference when using PR-MoE+MoS.

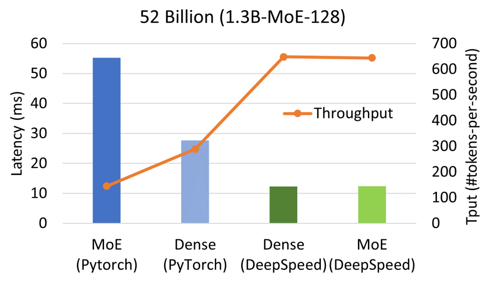

Figure 10: Inference latency comparison of MoE models and the quality-equivalent 6.7 billion-parameter dense model (throughput is reported per GPU). We used 128 GPUs for MoE models and one GPU for the dense model.

the minimum number of GPUs required to perform inference and (2) further improve both latency and throughput.

Better Latency and Throughput than Quality-equivalent Dense Models We now show the comparison between two MoE models with their quality-equivalent dense models: (1) a 52 billion-parameter MoE model (1.3B-MoE-128) compared to a 6.7 billion-parameter dense model in Figure 10, and (2) a 1.5 trillion-parameter MoE model compared to a 175 billion-parameter dense model in Figure 11. When using PyTorch, MoE model inference is more expensive and slower compared to its quality-equivalent dense models. However, the optimizations in DeepSpeed-MoE reverse this trend and make MoE model inference both faster and cheaper than quality-equivalent dense models. This is a critical result: showing the benefit of MoE models over dense models not only on training but also on inference latency and cost, where real-world deployments care the most.

The benefits increase for larger models because DeepSpeed-MoE leverages parallelism-coordinated optimization to reduce communication overhead when using tensor-slicing on the non-expert part of the model. Furthermore, we can take advantage of expert-slicing at this scale, which enables us to scale to a higher number of GPUs compared to the PyTorch baseline. We also observe 2x additional improvement in throughput over latency because MoE models can run with half the tensor-slicing degree of the dense model (8-way vs. 16-way) and thus two times higher batch size. With benefits that scale with model size and hardware resources, we believe that MoE models could be crucial to bring the next generation of advances in AI scale.

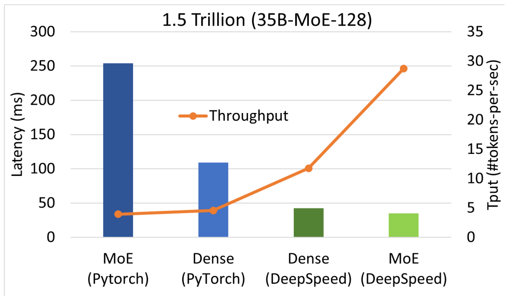

Figure 11: Measured inference latency comparison of 1.5T MoE models and assumed quality-equivalent 175 billion dense model (throughput is reported per GPU). We ran MoE experiments on 128 and 256 GPUs for Baseline and XYZ-MoE, while using 16 GPU for the dense model.

6. Towards the Next Generation of AI Scale

With the exponential growth of model size recently, we have arrived at the boundary of what modern supercomputing clusters can do to train and serve large dense models. We, along with recent literature (Du et al., 2021; Artetxe et al., 2021), have demonstrated how MoE-based models can reduce the training cost of the large NLG models by several times compared to their quality-equivalent dense counterparts. However, prior to this work, to our knowledge, there have been no existing works on how to serve the MoE models (with many more parameters) with latency and cost comparable to or better than the dense models. To enable practical and efficient inference for MoE models, we offer novel PR-MoE model architecture and MoS distillation technique to significantly reduce the memory requirements (up to 3.7x) of these models. We also offer an MoE inference framework to achieve incredibly low latency (up to 4.5x) and cost (up to 9x) at an unprecedented model scale. The new innovations and infrastructures offer a promising path towards training and inference of the next generation of AI scale, without requiring an increase in compute resources. A shift from dense to sparse MoE models can open a path to new directions in the large model landscape, where deploying higher-quality models with fewer resources becomes more widely possible.

Acknowledgment

We thank Olatunji Ruwase from the Microsoft DeepSpeed Team for his contributions on developing, debugging, testing, and releasing the DeepSpeed-MoE software. This work was done in collaboration with Brandon Norick, Zhun Liu, and Xia Song from the Microsoft Turing Team, Young Jin Kim, Alex Muzio, and Hany Hassan Awadalla from the Microsoft Z-Code Team, and both Saeed Maleki and Madan Musuvathi from the Microsoft SCCL team.

References

Artetxe, M., Bhosale, S., Goyal, N., Mihaylov, T., Ott, M., Shleifer, S., Lin, X. V., Du, J., Iyer, S., Pasunuru, R., Anantharaman, G., Li, X., Chen, S., Akin, H., Baines, M., Martin, L., Zhou, X., Koura, P. S., O'Horo, B., Wang, J., Zettlemoyer, L., Diab, M., Kozareva, Z., and Stoyanov, V. Efficient large scale language modeling with mixtures of experts. *arXiv preprint arXiv:2112.10684*, 2021.

Berant, J., Chou, A., Frostig, R., and Liang, P. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1533–1544, 2013.

Bisk, Y., Zellers, R., Gao, J., Choi, Y., et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, 2019.

Du, N., Huang, Y., Dai, A. M., Tong, S., Lepikhin, D., Xu, Y., Krikun, M., Zhou, Y., Yu, A. W., Firat, O., et al. Glam: Efficient scaling of language models with mixture-of-experts. *arXiv preprint arXiv:2112.06905*, 2021.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2021a.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *CoRR*, abs/2101.03961, 2021b.

He, J., Qiu, J., Zeng, A., Yang, Z., Zhai, J., and Tang, J. Fastmoe: A fast mixture-of-expert training system. *CoRR*, abs/2103.13262, 2021. URL https://arxiv.org/abs/2103.13262.

Hinton, G. E., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

Joshi, M., Choi, E., Weld, D. S., and Zettlemoyer, L. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kim, Y. J., Awan, A. A., Muzio, A., Cruz-Salinas, A. F., Lu, L., Hendy, A., Rajbhandari, S., He, Y., and Awadalla, H. H. Scalable and efficient moe training for multitask multilingual models. *CoRR*, abs/2109.10465, 2021a. URL https://arxiv.org/abs/2109.10465.

Kim, Y. J., Awan, A. A., Muzio, A., Salinas, A. F. C., Lu, L., Hendy, A., Rajbhandari, S., He, Y., and Awadalla, H. H. Scalable and efficient moe training for multitask multilingual models. *arXiv preprint arXiv:2109.10465*, 2021b.

Lai, G., Xie, Q., Liu, H., Yang, Y., and Hovy, E. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*, 2017.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. ALBERT: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2019.

Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., and Chen, Z. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Lin, J., Yang, A., Bai, J., Zhou, C., Jiang, L., Jia, X., Wang, A., Zhang, J., Li, Y., Lin, W., et al. M6-10t: A sharing-delinking paradigm for efficient multi-trillion parameter pretraining. *arXiv preprint arXiv:2110.03888*, 2021.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Masoudnia, S. and Ebrahimpour, R. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42(2): 275–293, 2014.

Microsoft. Tutel: An efficient mixture-of-experts implementation for large dnn model training. https://www.microsoft.com/en-us/research/blog/tutel-an-efficient-mixture-of-experts-implementation-for-large-dnn-model-training/, 2021.

Microsoft and Nvidia. Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, the World's Largest and Most Powerful Generative Language Model. https://developer.nvidia.com/blog/using-deepspeed-and-megatron-to-train-megatron-turing-nlg-530b-the-worlds-largest-and-most-powerful-generative-language-model/, 2021.

Mostafazadeh, N., Chambers, N., He, X., Parikh, D., Batra, D., Vanderwende, L., Kohli, P., and Allen, J. A corpus and evaluation framework for deeper understanding of commonsense stories. *arXiv preprint arXiv:1604.01696*, 2016.

Paperno, D., Kruszewski, G., Lazaridou, A., Pham, Q. N., Bernardi, R., Pezzelle, S., Baroni, M., Boleda, G., and Fernández, R. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*, 2016.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving language understanding by generative pre-training. *OpenAI Blog*, 2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

Rajbhandari, S., Rasley, J., Ruwase, O., and He, Y. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–16. IEEE, 2020.

Rajbhandari, S., Li, C., Yao, Z., Zhang, M., Yazdani Aminabadi, R., Awan, A. A., Rasley, J., and He, Y. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. ArXiv, January 2022. URL https://www.microsoft.com/en-us/research/publication/deepspeed-moe-advancing-mixture-of-experts-inference-and-training-to-power-next-generation-ai-scale/.

Rosset, C. Turing-nlg: A 17-billion-parameter language model by microsoft. *Microsoft Blog*, 1:2, 2020.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019. URL http://arxiv.org/abs/1910.01108.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., and Catanzaro, B. Megatron-LM: Training multi-billion parameter language models using gpu model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Smith, S., Patwary, M., Norick, B., LeGresley, P., Rajbhandari, S., Casper, J., Liu, Z., Prabhumoye, S., Zerveas, G., Korthikanti, V., et al. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*, 2022.

Sun, S., Cheng, Y., Gan, Z., and Liu, J. Patient knowledge distillation for BERT model compression. In Inui, K., Jiang, J., Ng, V., and Wan, X. (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 4322–4331. Association for Computational Linguistics, 2019.

Sun, Z., Yu, H., Song, X., Liu, R., Yang, Y., and Zhou, D. Mobilebert: a compact task-agnostic BERT for resource-limited devices. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. R. (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 2158–2170. Association for Computational Linguistics, 2020.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. Superglue: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint arXiv:1905.00537*, 2019.

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pp. 5753–5763, 2019.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792*, 2014.

Yu, D., Yao, K., Su, H., Li, G., and Seide, F. Kl-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, pp. 7893–7897. IEEE, 2013.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Zuo, S., Liu, X., Jiao, J., Kim, Y. J., Hassan, H., Zhang, R., Zhao, T., and Gao, J. Taming sparsely activated transformer with stochastic experts. *arXiv preprint arXiv:2110.04260*, 2021.

A. Author Contributions

Samyam Rajbhandari designed the NLG training experiments and architected the inference system.

Conglong Li led the NLG training experiments (Section 3) and contributed to Section 4.

Zhewei Yao led the design and experiments of PR-MoE, and its system support (Section 4.1).

Minjia Zhang led the design and experiments of MoS (Section 4.2) and memory-efficient checkpointing.

Reza Yazdani Aminabadi and **Ammar Ahmad Awan** led the development and experiments of the inference system (Section 5).

Jeff Rasley developed, debugged and integrated multiple software features into DeepSpeed.

Yuxiong He designed, managed and led the overall research project.

B. MoE-based NLG Model Training and Evaluation Settings

Table 3 summarizes the hyperparameters for training the dense and MoE models. For dense models we followed the hyperparameters from the GPT-3 work (Brown et al., 2020). For MoE models, we find that using a lower learning rate and longer learning rate decay duration compared to the dense counter parts (e.g., dense 1.3B versus 1.3B+MoE-128) could provide better convergence. We believe that this is because MoE models have much larger number of parameters.

C. Appendix for PR-MoE and MoS

C.1. System Design of PR-MoE

In this section, we begin by discussing how an MoE model can be trained efficiently using expert parallelism. Then we discuss the limitation of such an approach when applying it to PR-MoE model. Finally, we discuss how we can extend existing expert-parallelism based training systems to efficiently train PR-MoE models.

Efficiently Training an MoE model Training an MoE model efficiently requires having sufficiently large batch size for each expert in the MoE module to achieve good compute efficiency. This is challenging since the number of input tokens to an MoE is partitioned across all the experts which reduces the number of tokens per expert proportionally to the number of experts when compared to the rest of the model where no such partition is done. The simplest way to avoid this reduction in tokens per expert is to train the model with data parallelism in combination with expert parallelism (Kim et al., 2021a) equal to the number of experts. This increases the aggregate tokens in the batch per MoE replica that will be partitioned between the experts, resulting in no reduction of the tokens per expert compared to rest of the model.

Challenges of PR-MoE Designing a training infrastructure that can efficiently train PR-MoE model is non-trivial due to the presence of different number of experts at different stages of the model. As discussed above, the most efficient approach to training MoE based models is to make expert parallelism equal to the number of experts, to avoid reducing input tokens per experts. However, due to variation in the number of experts in PR-MoE, there is no single expert parallelism degree that is optimal for all MoE layers. Furthermore, if expert parallelism is set to the smallest number of experts in the model, then it would require multiple experts per GPU for MoE layers with larger number of experts, resulting in poor efficiency due to reduced batch size per expert, as well as an increase in memory required per GPU. On the other hand, if we set the expert parallelism to be the largest number of experts in the model, then this would result in a load balancing problem, where some GPUs have more experts to process than the others, ultimately limiting training throughput efficiency.

DeepSpeed-MoE with Multi-expert and Multi-data Parallelism Support To address these challenges, we develop and implement a flexible multi-expert and multi-data parallelism design on top of DeepSpeed-MoE, that allows for training different parts of the model with different expert and data parallelism degree. For instance, a PR-MoE model running on 128 GPUs, with 32, 64, and 128 experts at different MoE layers, can be trained with 128-way data parallelism for the non-expert parallelism, and $\{32, 64, 128\}$ expert parallelism plus $\{4, 2, 1\}$ data parallelism for MoE parameters. Note that each GPU can now train exactly 1 expert per MoE layer regardless of the number of experts in it, resulting in no reduction in input tokens per expert, no load-imbalance, or increase in memory requirements per GPU.

Through this flexible extension, DeepSpeed-MoE can train PR-MoE models, along with any other future MoE variations that may require different experts at different stages of the model, without compromising on training efficiency or the memory requirements.

C.2. Ablation Study of Different MoE Architectures

To fully study the performance of different MoE architectures, particularly the comparison between Standard-MoE and Residual-MoE/Pyramid-MoE/PR-MoE, we evaluate 5 different MoE-based models, including 350M+MoE-32, 350M+MoE-128, 350M+Pyramid-MoE-32/64 (which has 10 MoE layers using 32 experts and 2 MoE layers using

Table 3: Hyperparameters for different dense and MoE NLG models.

	Dense 350M	Dense 1.3B	Dense 6.7B	350M+ MoE-128	1.3B+ MoE-128	350M+PR-MoE-32/64	1.3B+PR-MoE-64/128
Num. layers	24	24	32	24	24	24	24
Hidden size	1024	2048	4096	1024	2048	1024	2048
Num. attention heads	16	16	32	16	16	16	16
Num. experts per layer	N/A	N/A	N/A	128	128	32/64	64/128
Num. parameters	350M	1.3B	6.7B	13B	52B	4B	31B
Context/sequence length	2K	2K	2K	2K	2K	2K	2K
Training tokens	300B	300B	300B	300B	300B	300B	300B
Batch size	256	512	1024	256	512	256	512
Batch size rampup tokens	0B	4B	10B	0B	0B	0B	0B
Learning rate	3.0e-4	2.0e-4	1.2e-4	2.0e-4	1.2e-4	3.0e-4	1.2e-4
Min. learning rate	3.0e-5	2.0e-5	1.2e-5	2.0e-6	1.0e-6	1.0e-6	1.0e-6
LR linear warmup tokens	375M	375M	375M	375M	375M	375M	375M
LR cosine decay tokens	260B	260B	260B	300B	300B	300B	300B
Model parallel degree	1	1	8	1	1	1	1
MoE loss coefficient	N/A	N/A	N/A	0.01	0.01	0.01	0.01

Table 4: Zero-shot evaluation comparison (last six columns) between PR-MoE and PR-MoE + MoS.

Model (num. params)	LAMBADA	PIQA	BoolQ	RACE-h	TriviaQA	WebQs
350M+PR-MoE+L24(4B)	63.65	73.99	59.88	35.69	16.30	4.73
350M+PR-MoE+L21 (3.5B)	62.33	73.88	52.35	32.54	8.81	4.48
350M+PR-MoE+L21+KD only (3.5B)	61.56	73.18	57.89	33.78	12.13	4.87
350M+PR-MoE+L21+MoS (3.5B)	63.46	73.34	58.07	34.83	13.69	5.22
1.3B+PR-MoE+L24 (31B)	70.60	77.75	67.16	38.09	28.86	7.73
1.3B+PR-MoE+L21 (27B)	69.14	76.99	60.8	37.42	28.9	5.61
1.3B+PR-MoE+L21+KD only (27B)	69.73	76.93	64.16	36.17	26.17	6.25
1.3B+PR-MoE+L21+MoS (27B)	70.17	77.69	65.66	36.94	29.05	8.22

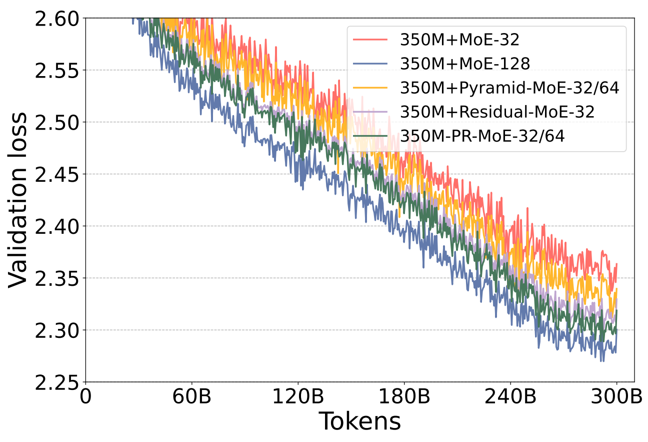

Figure 12: The validation curves of different MoE models based on 350M+MoE.

64 experts), 350M+Residual-MoE-32, and 350M+PR-MoE-32/64 (same expert setting as 350M+Pyramid-MoE-32/64).

The validation curves for the full training are shown in Figure 12. As can be seen, the validation loss gap between 350M+MoE-128 and 350M+MoE-32 can be significantly reduced by Pyramid-MoE and Residual-MoE. When we use PR-MoE, a combination of Pyramid-MoE and Residual-MoE, the loss gap can be further reduced to around 0.01, demonstrating PR-MoE's great parameter efficiency with minimum quality impact.

C.3. Ablation Study of MoS

In this part, we conduct ablation studies of zero-shot evaluation on MoS, including (1) MoS without KD: This config trains an MoE model with reduced depth, (2) MoS with full KD: This config trains an MoS model but with KD during the entire training process, and (3) MoS with staged KD: This is the config described in Section 4.2. The results on each of the tasks are shown in Table 4. We make a few observations. First, with the same amount of depth reduction but without MoS-based KD (row 2), the PR-MoE model encounters noticeable accuracy drop on several tasks such as LAMBADA (1.3 points) and BoolQ (7.5 points), indicating that directly reducing the expert depth can hurt the model accuracy. Second, with staged KD (row 4), we are able to improve the student PR-MoE's performance and observe accuracy improvements on 5 out of 6 tasks. Notably, 1.1 points improvement for LAMBADA, 6.5 points higher for BoolQ, 1.7 points higher for RACE-h, 4.5 points higher for TriviaQA. One exception is PIQA, in which case the student PR-MoE experiences some small accuracy drop. These results indicate the effectiveness of our proposed

Table 5: Zero-shot evaluation comparison between standard MoE, PR-MoE, MoS, and related works (last 2 rows).

Model (num. params)	LAMBADA	PIQA	BoolQ	RACE-h	TriviaQA	WebQs
350M+MoE-128 (13B)	62.70	**74.59**	**60.46**	35.60	**16.58**	5.17
350M+PR-MoE-32/64 (4B)	**63.65**	73.99	59.88	**35.69**	16.30	4.73
350M+PR-MoE+L21+MoS (**3.5B**)	63.46	73.34	58.07	34.83	13.69	**5.22**
1.3B+MoE-128 (52B)	69.84	76.71	64.92	**38.09**	**31.29**	7.19
1.3B+PR-MoE-64/128 (31B)	**70.60**	**77.75**	**67.16**	**38.09**	28.86	7.73
1.3B+PR-MoE+L21+MoS (**27B**)	70.17	77.69	65.66	36.94	29.05	**8.22**
8B+MoE-64 (143B) (Du et al., 2021)	67.3	78.6	72.2	43.4	55.1	10.7
355M+MoE-512 (52B) (Artetxe et al., 2021)	N/A	76.8	56.0	N/A	N/A	N/A

Table 6: The configuration of different MoE models used for the performance evaluation of DeepSpeed-MoE inference system. These configurations represent the standard MoE architecture cases, and we also test the case of PR-MoE and PR-MoE+MoS, which will have smaller sizes but same (projected) quality.

Model	Size (billions)	#Layers	Hidden size	MP degree	EP degree	Expert-slicing	#GPUs
1.3B+MoE-128	52	24	2048	1	128	1	128
2.4B+MoE-128	107.7	16	3584	1	128	1	128
8B+MoE-128	349.0	30	4096	4	128	1	128
24B+MoE-128	1064.9	40	8192	8	128	2	256
47B+MoE-128	2024.0	58	8192	8	128	2	256

Mixture-of-Students method as a novel KD technique for MoE models. Third, performing KD for the entire training process (full KD, row 3) hurts the downstream task accuracy on LAMBADA (0.8 points lower) and PIQA (0.7 points lower). As explained in the previous part, this is because the student model does not have sufficient capacity to optimize both the KD loss and the standard LM loss towards the end of training, due to under-fitting. In contrast, our proposed staged-KD is able to resolve this issue and brings the promised benefits from KD. Overall, the distilled MoE model through staged KD achieves an average accuracy of 42.87 and 47.96, retaining 99.5% and 99.1% performance of the 350M (43.08) and 1.3B teacher model (48.37) despite having 12.5% fewer layers. This enables an additional latency reduction and throughput reduction for inference, which we show in Section 5.

C.4. Comparison with Related Works

Please note that with the caveats that the training data, training time (e.g., the number of training tokens), and other hyperparametrs are not the same, a direct model quality comparison is challenging. As such, we here give a more detailed summary about the difference between our DeepSpeed-MoE work and (Du et al., 2021; Artetxe et al., 2021).

Comparisons with (Du et al., 2021) We list the results of (Du et al., 2021) in Table 5, and as can be seen, on certain tasks (LAMBADA/PIQA) our MoE models can achieve on-par quality with less number of parameters. The differences are as follows: (1) While we use Top-1 gating function to training the MoE model, (Du et al., 2021) uses Top-2 gating function, which can potentially improve model quality but has higher computational and communication cost based on our experiments; (2) We use the same number of training samples/tokens to train the MoE model as the dense model, but (Du et al., 2021) increases the training tokens for MoE models to up to 600B, 2 times of what we used for our dense and MoE models; (3) Based on (i) and (ii), the training cost reduction from dense to our MoE model (5x) is much higher than (Du et al., 2021) ($\sim$2.8x); (4) We propose a new MoE architecture, i.e., PR-MoE, to increase the parameter efficiency. To support PR-MoE, we also design a new system to support flexible expert and data parallelism at each MoE layer. This is not supported in (Du et al., 2021); (5) To further reduce the inference cost, we propose a new knowledge distillation technique, Mixture-of-Students, to reduce the depth of PR-MoE model; (6) To efficiently run inference for MoE models, we develop a whole new inference system which can transfer the training cost reduction to **real** inference as compared to dense model. In (Du et al., 2021), only FLOP reduction is provided, which is not the full picture of MoE model inference as we discussed in Section 5.

Comparisons with (Artetxe et al., 2021) Compared to (Artetxe et al., 2021), our MoE models are able to achieve better PIQA/BoolQ accuracy with 1.9x less number of parameters. Meanwhile, the authors only show the training reduction for MoE models as compared to dense with Top-2 experts as (Du et al., 2021) and this makes our MoE model's 5x training cost reduction higher than the 4x reduction in (Artetxe et al., 2021). In addition, similar to the comparisons with (Du et al., 2021) above, comparing with (Artetxe et al., 2021) our novel contributions include (1) new MoE architecture and its system support; (2) MoE-to-MoE knowledge distillation; and (3) end-to-end inference system design and evaluation.