Carnegie Mellon University

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

11-868 Presentation

Yichen(Eason) Lu, Yifan He, Kath Choi

Introduction

Background: Compute is the primary challenge of training massive models.

Model	Model Size	Hardware	Days to Train
Megatron-LM GPT-2	8.3B	512 V100 GPU	9.2 days
OPT	175B	992 A100 GPU	56 days
MT-NLG	530B	2200 A100 GPU	60 days
PaLM	540B	6144 TPU v4	57 days

Mixture of Experts (MoE) is a promising path for improved model quality without increasing training cost.

Challenges of Previous MoE

- Limited on encoder-decoder models and sequence-to-sequence tasks
- Require more parameters to achieve the same model quality as their dense counterparts
- MoE models' large size makes inference difficult and costly

Carnegie Mellon University

NLG Model Training with MoE

Dense vs. MoE on GPT-style NLG model family

- MoE significantly improve model quality with the same training cost
- MoE achieve 5x reduction in training cost to achieve the same model quality

Base Models

- 350M (24 layers, 1024 hidden size, 16 attention heads)
- 1.3B (24 layers, 2048 hidden size, 16 attention heads)
- 6.7B (32 layers, 4096 hidden size, 32 attention heads)

MoE Architecture

- add 128 experts on every other feedforward layer
- top-1 gating vs. top-2 gating (heavy computation/communication overhead)

Deepspeed MoE Code Example

```
class Net(nn.Module):
    def __init__(self, args):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.moe = args.moe
        if self.moe:
            fc3 = nn.Linear(84, 84)
            self.moe_layer_list = []
            for n_e in args.num_experts:
                # Create moe layers based on the number of experts.
                self.moe_layer_list.append(
                    deepspeed.moe.layer.MoE(
                        hidden_size=84,
                        expert=fc3,
                        num_experts=n_e,
                        ep_size=args.ep_world_size,
                        use_residual=args.mlp_type == "residual",
                        k=args.top_k,
                        min_capacity=args.min_capacity,
                        noisy_gate_policy=args.noisy_gate_policy,
                    )
                )
            self.moe_layer_list = nn.ModuleList(self.moe_layer_list)
            self.fc4 = nn.Linear(84, 10)
        else:
            self.fc3 = nn.Linear(84, 10)
```

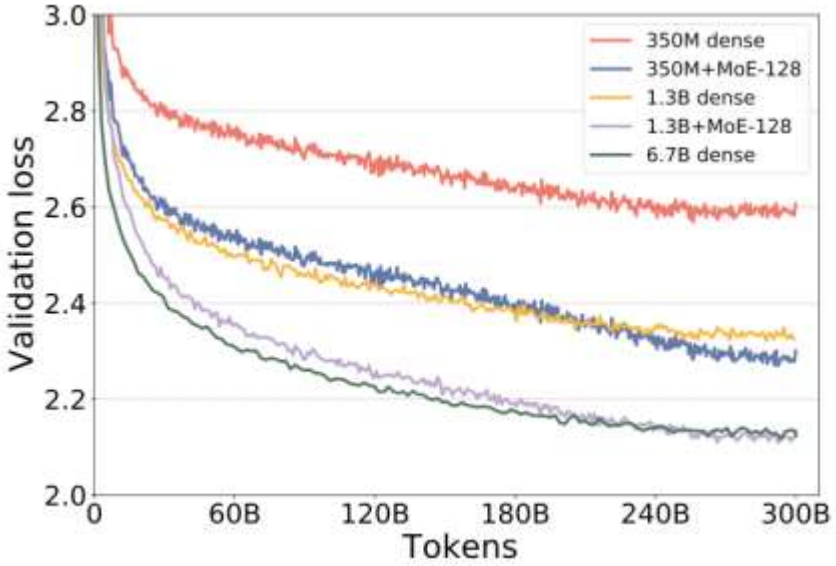
```
import torch
import deepspeed
import deepspeed.utils.groups as groups
from deepspeed.moe.layer import MoE

WORLD_SIZE = 4
EP_WORLD_SIZE = 2
EXPERTS = 8

fc3 = torch.nn.Linear(84, 84)
fc3 = MoE(hidden_size=84, expert=self.fc3, num_experts=EXPERTS, ep_size=EP_WORLD_SIZE, k=1)
fc4 = torch.nn.Linear(84, 10)
```

https://github.com/microsoft/DeepSpeed/blob/master/deepspeed/moe/layer.py

https://github.com/microsoft/DeepSpeedExamples/blob/master/training/cifar/cifar10_deepspeed.py

Carnegie Mellon University

Evaluation on Validation Loss and Throughput



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

	Training samples per sec	Throughput gain/ Cost Reduction
6.7B dense	70	1x
1.3B+MoE-128	372	5x

Training throughput (on 128 A100 GPUs) comparing MoE based model vs dense model that can both achieve the same model quality.

Carnegie Mellon University

Evaluation on Zero-shot Benchmarks

Case	Model size	LAMBADA: completion prediction	PIQA: commonsense reasoning	BoolQ: reading comprehension	RACE-h: reading comprehension	TriviaQA: question answering	WebQs: question answering
Dense NLG:							
(1) 350M	350M	52.03	69.31	53.64	31.77	3.21	1.57
(2) 1.3B	1.3B	63.65	73.39	63.39	35.60	10.05	3.25
(3) 6.7B	6.7B	**71.94**	**76.71**	**67.03**	37.42	23.47	5.12
Standard MoE NLG:							
(4) 350M+MoE-128	13B	62.70	74.59	60.46	35.60	16.58	5.17
(5) 1.3B+MoE-128	52B	69.84	**76.71**	64.92	**38.09**	**31.29**	**7.19**

Zero-shot evaluation results on different benchmarks. Metrics are accuracy.

- 5x lower training cost to same accuracy using MoE
- 8x more parameters to same accuracy using MoE

Carnegie Mellon University

PR-MoE: Intuitions

Intuition 1: In Computer Vision, deeper layers learn more objective specific representations []

Question 1: Are all the MOE layers equally important?

Intuition 2: More experts (more memory) and more expert capacity (higher latency) can help improve generalization.

Question 2: Can we achieve a balance by fixing the first expert and only assign different extra experts to different tokens?

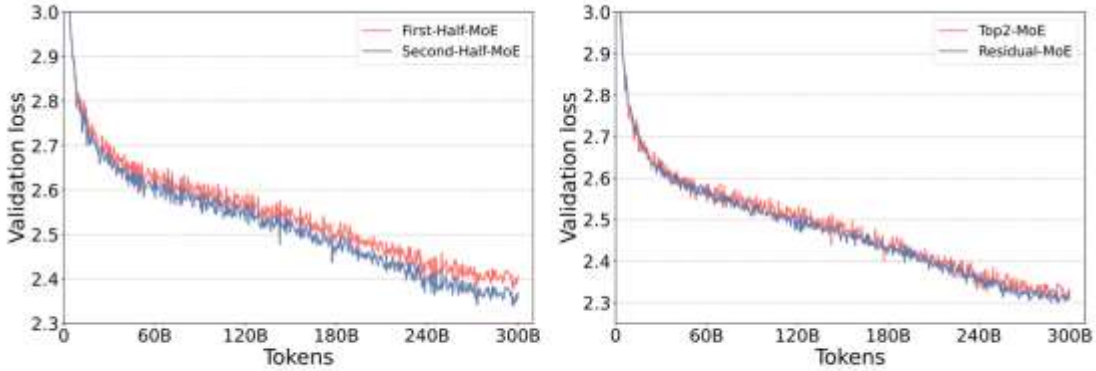


Figure 2: The validation loss of First-Half-MoE/Second-Half-MoE (left) and Top2-MoE/Residual-MoE (right) based on 350M+MoE models.

Carnegie Mellon University

Pyramid-Residual-MoE

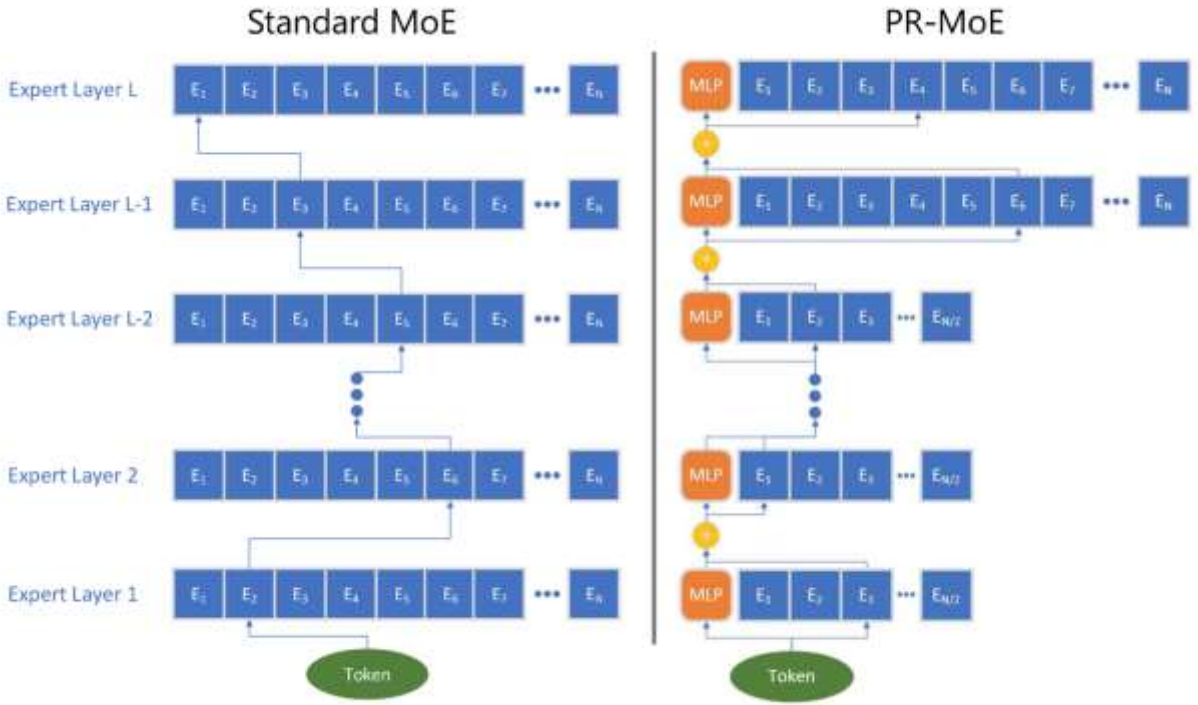


Figure 3: The illustration of standard MoE (left) and PR-MoE (right).

Carnegie Mellon University

PR-MoE: System Design

Training MoE: data parallelism in combination with expert parallelism

Challenge: PR-MoE is Pyramid shaped, no optimal expert parallelism

- parallelize smallest number of experts: multiple experts per GPU, reduced batch size and increased memory requirement
- parallelize largest number of experts: load balancing problem

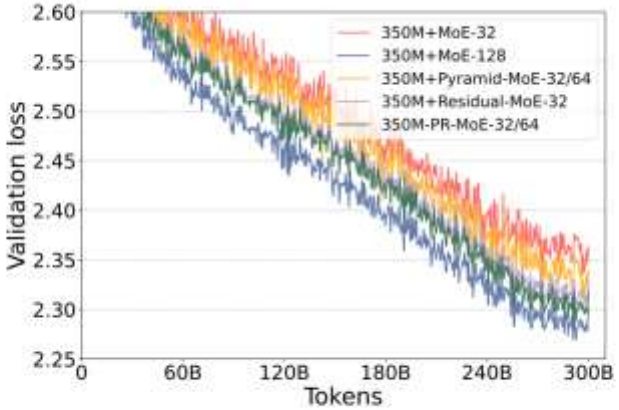
Multi-expert and Multi-data Parallelism Support

- provide flexible training for different parts of the model with different expert and data parallelism degree
- each GPU train exactly 1 expert per MoE layer

PR-MoE: Evaluation

350M+PRMoE-32/64 vs 350M+MoE-128, 1.3B+PR-MoE-64/128 vs 1.3B+MoE128

comparable accuracy, ⅓ parameters for 350M, 60% parameters for 1.3B

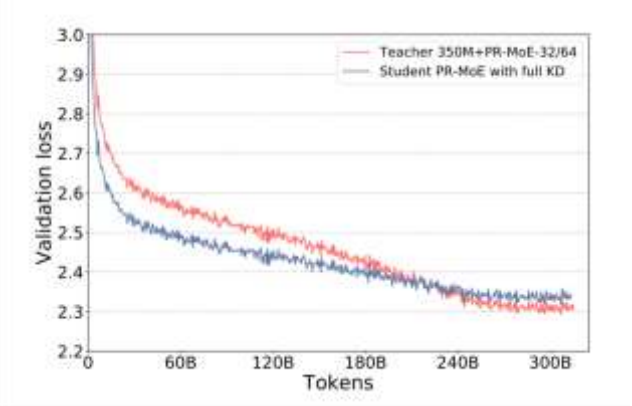


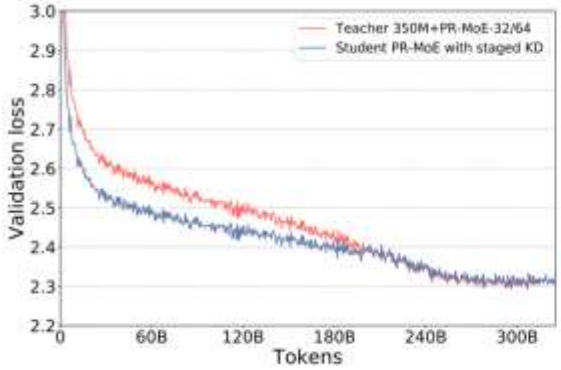
Model (num. params)	LAMBADA	PIQA	BoolQ	RACE-h	TriviaQA	WebQs
350M+MoE-128 (13B)	62.70	**74.59**	**60.46**	35.60	**16.58**	5.17
350M+PR-MoE-32/64 (**4B**)	**63.65**	73.99	59.88	**35.69**	16.30	**4.73**
1.3B+MoE-128 (52B)	69.84	76.71	64.92	**38.09**	**31.29**	7.19
1.3B+PR-MoE-64/128 (**31B**)	**70.60**	**77.75**	**67.16**	**38.09**	28.86	**7.73**

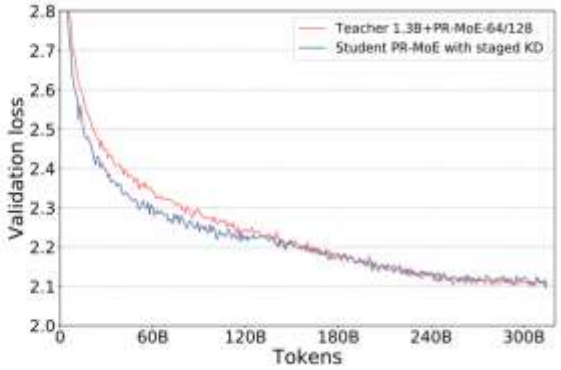
Mixture-of-Students

Main idea: layer reduction through knowledge distillation

- reduce the depth of each expert branch
- gradually decay the impact from KD or stop KD early (at 400K steps)







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

Carnegie Mellon University

MoS evaluation

Up to 3.7 times drop in model size together with PR-MoE compared to standard MoE

Table 5: 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+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

Carnegie Mellon University

Expert Parallelism and Expert-slicing (Expert Params)

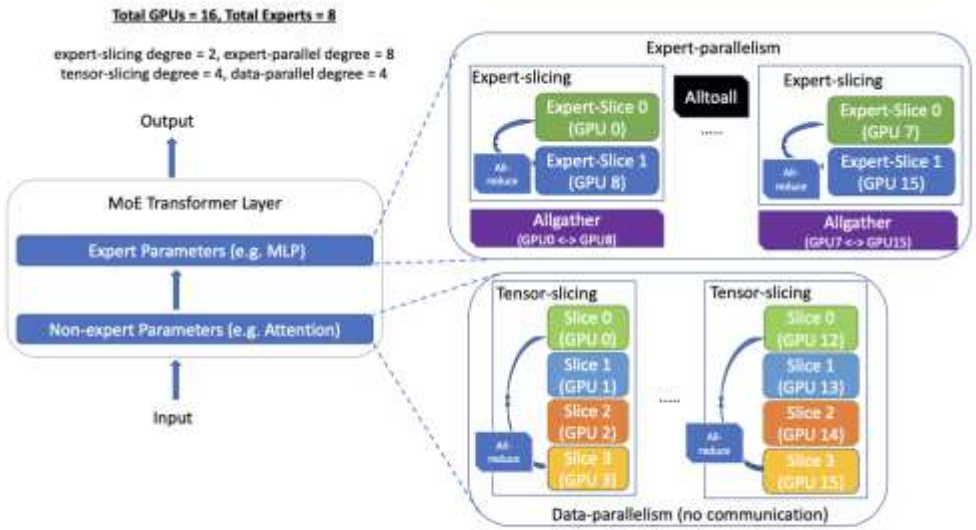


Figure 7: DS-MoE design that embraces the complexity of multi-dimensional parallelism for different partitions (expert and non-expert) of the model.

Group all input tokens assigned to the same experts under the same critical data path, and parallelize processing of the token groups with different critical paths among different devices using expert parallelism.

Partitions the expert parameters horizontally/vertically across multiple GPUs. (Design for GPU num > Experts num)

Carnegie Mellon University

Data Parallelism and Tensor-slicing (Non-Expert Params)

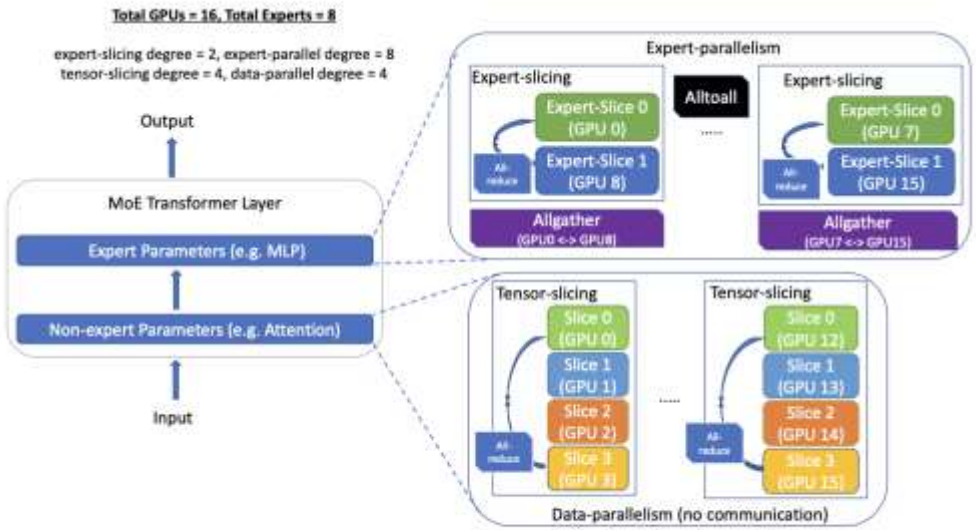


Figure 7: DS-MoE design that embraces the complexity of multi-dimensional parallelism for different partitions (expert and non-expert) of the model.

Data-parallelism by creating non-expert parameter replicas processing different batches across nodes

tensor-slicing within a node allowing for hundreds of billions of non-expert parameters by leveraging aggregate GPU memory, while also leveraging the aggregate GPU memory bandwidth across all GPUs within a node

Carnegie Mellon University

Hierarchical All-to-all for communication

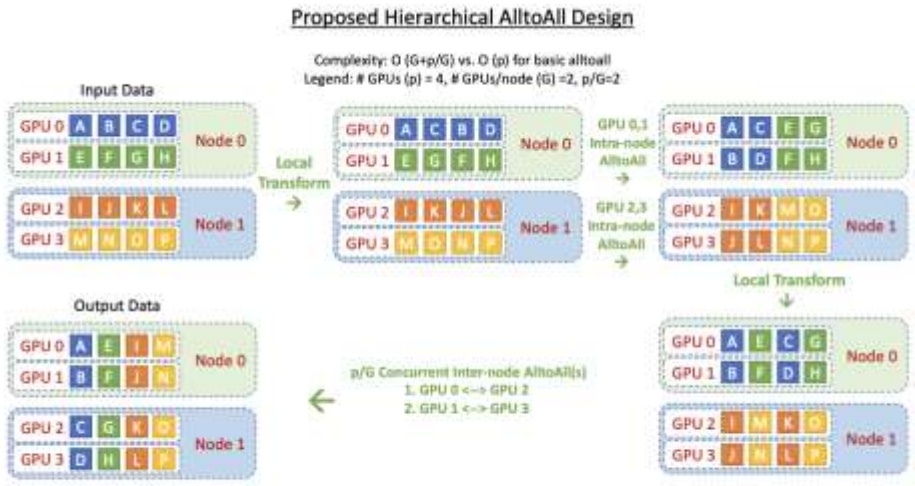


Figure 8: Illustration of the proposed hierarchical all-to-all design.

Implemented 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.

Reduces the communication hops from O(p) to O(G+ p/G),

Carnegie Mellon University

Experiments - inference

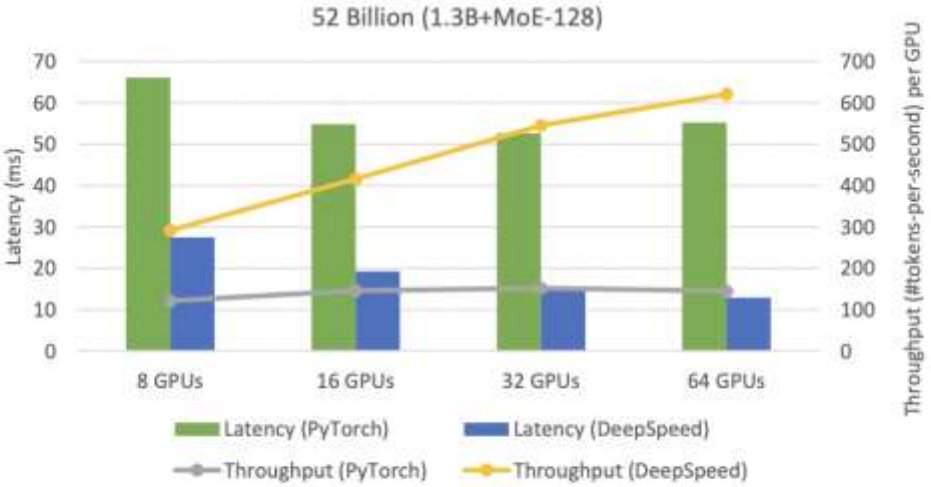


Figure 10: Latency and throughput improvement offered by DeepSpeed-MoE inference system (Optimized) over PyTorch (Baseline) for a 52-Billion MoE model with 128 experts, using between 8 to 64 GPUs.

Carnegie Mellon University

Experiments - inference

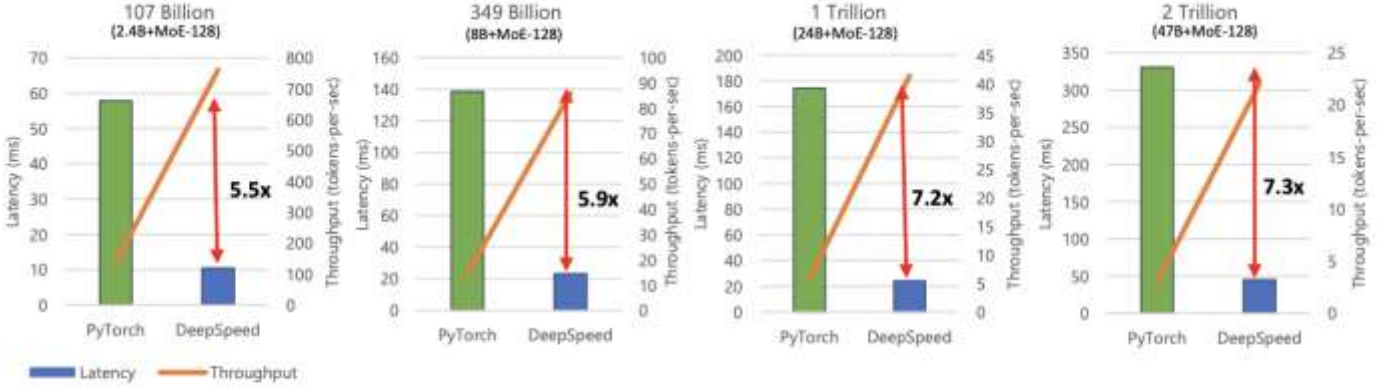
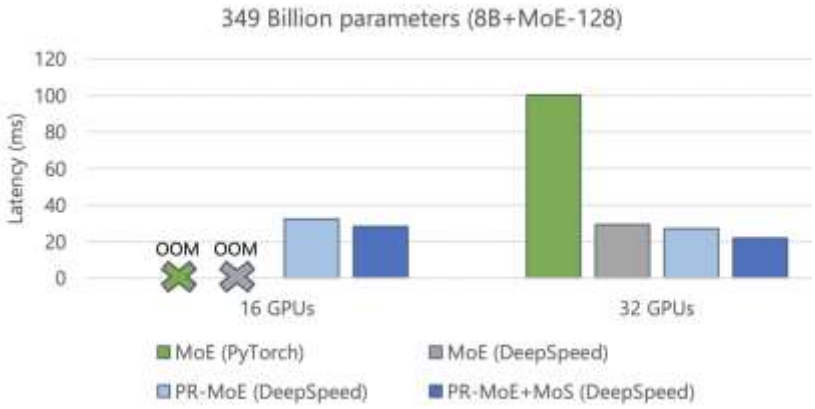
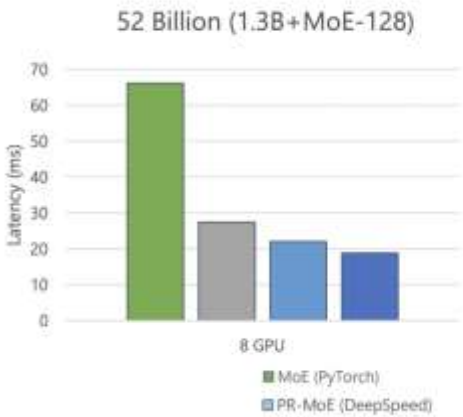


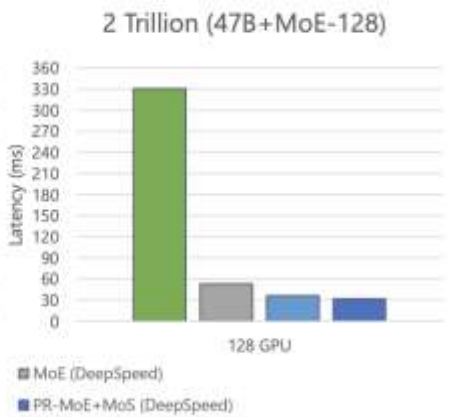
Figure 11: Latency and throughput improvement offered by DeepSpeed-MoE (Optimized) over PyTorch (Baseline) for different MoE model sizes (107 billion to 2 trillion parameters). We use 128 GPUs for all configurations for baseline, and 128/256 GPUs for DeepSpeed-MoE (256 GPUs for the trillion-scale models). The throughputs shown here are per GPU and should be multiplied by number of GPUs to get the aggregate throughput of the cluster.

Carnegie Mellon University

Performance of MoS and PR-MOE

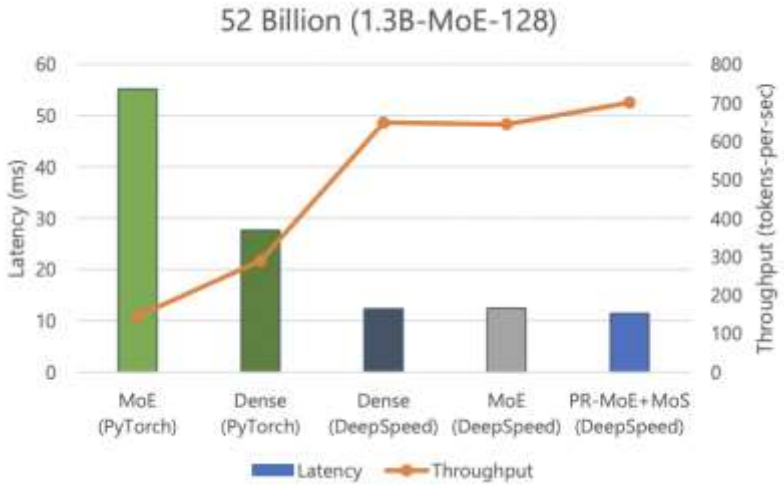


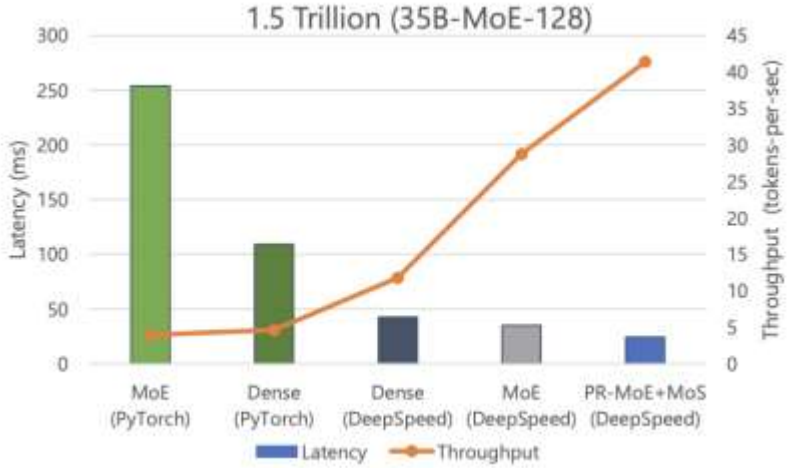




Carnegie Mellon University

Performance of MoS and PR-MOE





Carnegie Mellon University

Reference

Rajbhandari, S., Li, C., Yao, Z., Zhang, M., Aminabadi, R. Y., Awan, A. A., … He, Y. (2022). DeepSpeed-MoE: Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale. arXiv [Cs.LG]. Retrieved from http://arxiv.org/abs/2201.05596