Summary of Optimizing Mixture of Experts Models

MAYA RAO (RAOMAYA), AIDAN DELWICHE (DELWICHE), CHRISTOPHER KOK (CHRISKOK)

Problem and Motivation

It's generally accepted that one of the most important ways to increase a model's quality is to scale up a model. Given we only have a certain amount of computing resources (GPUs aren't cheap afterall), we're generally able to get better results while training a large model for fewer steps than a smaller model. While one could continue adding more and more parameters to a traditional dense model, the amount of computing resources used to pretrain said dense model would increase substantially. Not only is it more expensive to pretrain, but during compute time it's more resource intensive too since the model utilizes all parameters in the network. The goal then is to find a model which is not under full parameter utilization at all times, and the Mixture of Experts (MoE) model attempts to address this problem. MoE models have been around since the 90's, but no such effort had been made to adapt MoEs to natural language processing (NLP) or, in the case of our papers, natural language generation (NLG) like GPT.

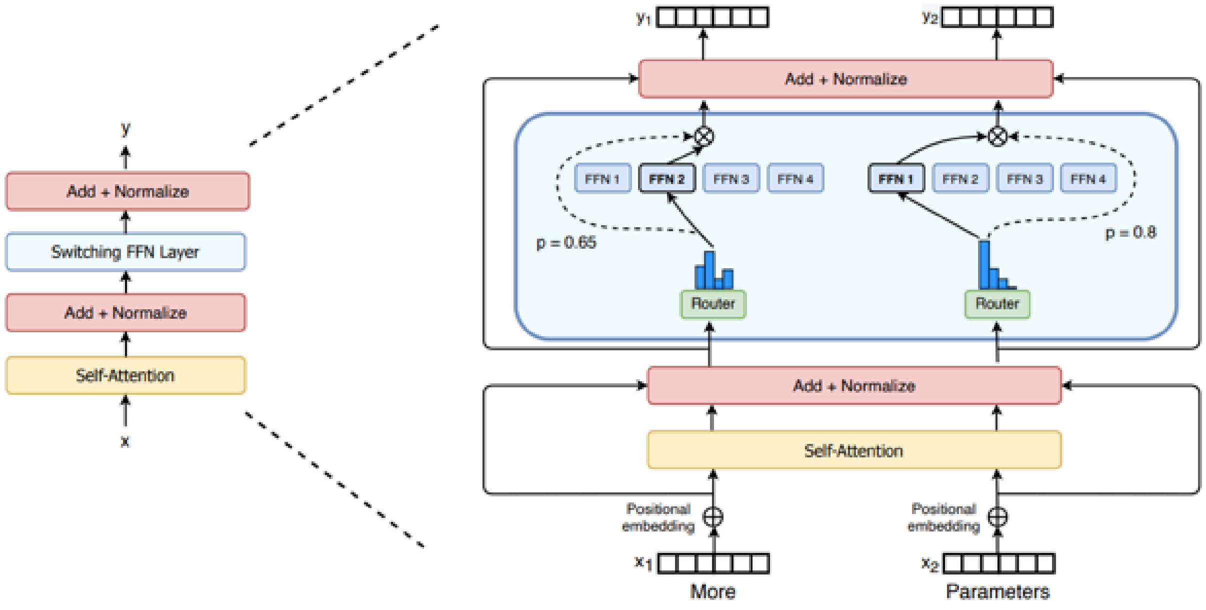
Related Works

The presenters mentioned a severe lack of related/current work for MoE's especially in relation to the critical topics within ML like NLG. This is reflected in both papers presented today. However, we wanted to include a short summary of the related works presented in the DeepSpeed-MoE paper for reference/context:

In essence, the study of Large Scale Dense NLP Models (that the papers compare against) has shown that the scale of pre-trained NLP model size has been rapidly increasing, leading to increasingly advanced models (e.g. BERT, XLNet, RoBERTa, ALBERT, GPT, etc.). However, as training these models can take up to three months and requires significant computational power, a new method was needed to achieve better model quality without merely increasing the model size. To address this issue, research has focused on Reducing Training Cost by using the Mixture of Expert (MoE) Architecture. This approach significantly improves the scaling of model size without increasing the computational cost. The study includes a focus on MoE training and inference systems, like the DeepSpeed MoE training system. Despite these advances, however, researchers have yet to develop a MoE system specifically designed and optimized for inference (which is what the authors tackle with the solutions below!).

Solution Overview

MoEs are a type of ensemble model with the unique attribute of only utilizing part of the model at a time. The model is designed to have different “experts”, or neural networks that specialize in distinct areas, so not all of the model will have to be fired up all the time. The MoE uses a gating network to decide which expert it wants to use at a given time. For example, if you’re processing images, an expert might be a “vehicular identification” expert, so when an image of a car is the input it might be sent to that specific expert based on the gating network’s identification.



MoEs come with some large challenges, but both papers address some of the common issues with MoEs and design solutions to mitigate their effects.

Shrinking Batch Size:

A challenge with MoEs has been the model’s batch sizes decreasing due to gating becoming sparse, however without large batch sizes performance decreases significantly. One very effective solution to increase batch size is expert parallelism whether that be to giving each expert its own device or just spreading out the experts across many devices. Another solution is to take advantage of the convolutionality of MoEs by applying the same MoE model to every time step of the previous layer. Then have the model wait for previous timesteps to finish to apply the MoE to all timesteps in one big batch increasing the batch size immensely.

Network Bandwidth:

Another challenge with MoEs implementation is its network bandwidth. Typically the communication and computation time have not been equal leaving the expert inactive while training examples are still being processed. Thus, one solution to the issue is to increase the amount of computation the expert does by increasing the size of the hidden layers the expert runs.

Expert balancing:

MoEs also are limited by their design as the model tends to favor certain experts causing them to be selected and trained more by the gating network. A solution the authors implemented to help place some constraint ensuring experts are selected evenly was to create an additional loss function for variation in gate values and loads. They then added the loss functions they created to the main loss function the model utilizes to choose experts.

Scope of MoE:

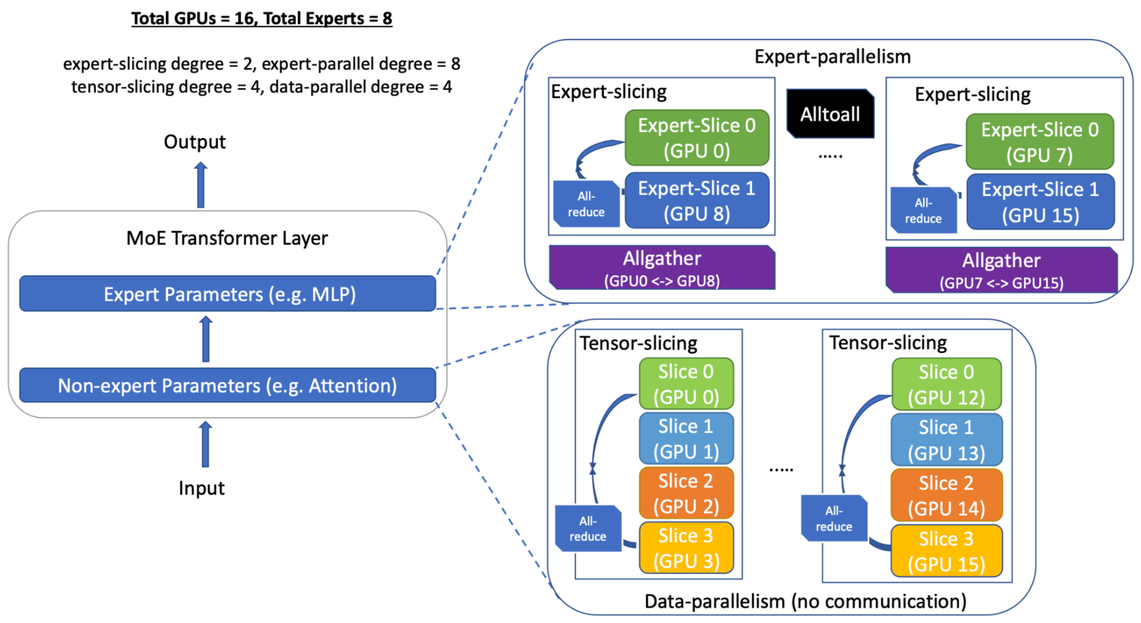
Another primary challenge lies in enhancing the usage of MoE models in NLP, particularly in auto-regressive NLG (which is currently less explored despite being incredibly popular). These tasks still require considerable computational power and energy for training and improvements are needed to reduce costs. To overcome these barriers, the MoE-based NLG model architecture proposed introduces an extra layer of expert models each assigned with distinct aspects of the task. Initial training tests have demonstrated this model's potential to boost the quality and cost-efficiency of NLG, showing promise for future research and application.

Massive memory requirements:

While MoEs reduce computational cost, they create a different fundamental issue due to the extreme increase in memory needed. MoEs require around 10x more parameters to be stored and loaded than dense models. Two primary solutions proposed to help reduce the memory are the Pyramid-Residual-MoE and Mixture-of-Students models. The Pyramid-Residual- MoE was designed after an experiment was run revealing it is more beneficial to have MoE running in the second half of the model. The pyramid model's two major changes are: increasing the amount of experts in the second half of the model, and having one fixed primary expert. The Mixture-of-Students model implements the knowledge distillation technique to help reduce memory by first creating a "teacher" model, and then reducing the branch depths to create "student" models. The model then enforces KD loss as a weighted sum of cross-entropy loss between student predictions and real labels, however after experiments were run it was revealed the model should stop using KD early in training due to a loss in performance.

Inference performance:

Lastly, to combat inference challenges in MoE models, several technical solutions have been observed. These include the use of small batch size in inference, and strategies such as Expert Parallelism and Expert-Slicing for expert parameters/processing, as well Data Parallelism and Tensor-Slicing for non-expert parameters/processing. By combining these methods, the approach has delivered an up to 4.5 times faster and up to 9 times cheaper MoE model inference compared to serving quality-equivalent dense models using PyTorch!



Limitations

Despite the best efforts to optimize the MoE model as presented in the DeepSpeed paper, there are still some limitations to the MoE approach that still need to be further explored. While the PR-MoE and MoS models do reduce memory usage, MoEs as a whole are *still* limited by their high memory usage. For example, in most benchmarks, a PR-MoE model needs 31B parameters to have performance comparable to a 6.7B parameter dense NLG model, and an MoS model needs around 27B parameters. Another limitation is the sole usage of FFNs for experts by the authors; because FFNs are a more “simple” type of NN, it’s worth considering if using FFNs is limiting the performance of an MoE, or if that’s the correct solution based on the memory constraints we’ve already discussed.

Future Research Directions

In speaking to the presenters after class, they described that future focus would be on exploring real-world applications of MoE Architecture and Multimodality. As MoE models continue to evolve, their real-world applications become increasingly diverse and imperative to investigate. This investigation can provide valuable insights into the practical utility of these models across a range of sectors and applications, from medicine to NLP. Multimodality (the ability of a system to interpret, understand, and process information from various forms and sources simultaneously) in MoE models would enable these systems to operate effectively across different data types and conditions, thereby enhancing their flexibility, robustness, and overall performance. This area holds a good amount of promise in their eyes.

Summary of Class Discussion

The in-class discussion revolved around the following questions/topics:

- Firstly, the discourse highlighted the importance of 'runtime systems' and the use of 'dynamic partitions.' This setup promotes better equality and utilization, according to the NSDI paper by utilizing resources more efficiently, potentially leading to improved performance in certain contexts. (link: https://www.usenix.org/conference/nsdi23/presentation/zheng)
- The conversation also touched on the issue of network bandwidth and overfitting. It was mentioned that an increase in layers could result in more overfitting. This issue can also be exacerbated when additional parameters are added to the model.
- The absence of control flow in the model was another focal point. Initially missing in systems like with TensorFlow, control flows have now been incorporated to enable conditional operations and loops, thereby enhancing their capacity to handle complex computations.
- Further, the class discussed the strategic use of experts, for instance, deploying the top 4/8 experts according to usage to optimize the overall system performance. This could be a potential area of further research!
- Finally, the benefits of knowledge distillations were also examined. The presenters mentioned that PR MoE and MoS (although marginal), both impact the efficiency and productivity of the model, making the system more robust and dynamic.