ST-MoE: Designing Stable and Transferable Sparse Expert Models

Main Idea(s): this paper provides a thorough study on MoEs. It tackles the biggest challenges presented to MoE models at the time of its release, with those being instabilities in training and poor fine-tuning performance. Its main goal is therefore to improve the practicality and reliability of sparse models.

- Trains a 269B sparse encoder-decoder model.
- Introduces router z-loss to resolve instability issues.

Stabilizing Training of Sparse Models

Transformer models today are normally trained by using float32 to compute gradients and float16 to compute the forward and backward pass. Sparse models contain several exponential functions (like softmax), which can lead to large values flowing through the network. Float16 does not handle large numbers well, as the larger the number, the larger its resulting rounding error. It is proposed that this abundance of exponential functions in MoE is what causes training instability. Router z-loss is a trick to penalize large values from flowing through the network, thus improving stability.

- Router z-loss is a function that stabilizes the training of MoE models without degradation in model quality by penalizing large values from flowing through the network.
    - Stability is referred to as constant/smooth decrease in the training loss.

- Router z-loss is a complement to the overall loss function, as cross-entropy and auxiliary load-balancing loss are also used.
- So total loss = cross-entropy + auxiliary load-balancing loss + router z-loss.

Fine-Tuning Sparse Models

Model characteristics:

- Dense and sparse models both pre-trained on 500B tokens.
- Both roughly match T5 (encoder-decoder), which has 770M parameters.
- Sparse model has 32 experts and a sparse layer every 4 layers.
- Train capacity factor = 1.25, 2.0 at eval time.
- Fine-tuned on 2 SuperGLUE tasks, one with 100,000 training examples and the other with 250 to analyze overfitting of sparse models during fine-tuning.

Sparse models are thought of to be more prone to overfitting during fine-tuning, especially when there is little data to work with (the more data, the lesser the risk of overfitting). This is observed on the smaller task (250 training examples), where the sparse model performs better against the training set but worse in the evaluation set (classic overfitting). This does not happen in the larger task (which has 100k training examples), where the sparse model performs better than the dense one on both the training and evaluation sets. This leads us to the conclusion that sparse models have fine-tuning advantages if enough data is available to prevent the model from overfitting. Increasing regularization did not seem to have much effect at the small-scale fine-tuning, showing that small amounts of data are hard to overcome in this scenario.

To explore fine-tuning MoEs further, the authors experiment with exclusively updating a few layers while keeping the remaining layers frozen. They test this for different combinations.

- Most combinations yield similar results, except one -> only updating sparse layers, which resulted in degraded performance.
    - This indicates that the overfitting comes from sparse layers (although updating all parameters leads to better performance than updating all non-MoE parameters only).
    - Another explanation for this can be the frequency of sparse layers being too sparse (only 1 sparse layer for every 3 dense layers), so the number of parameters being updated is not large enough.

Another fine-tuning aspect analyzed was the batch size and learning rate of dense vs sparse models. Experiments showed that they do not respond the same way to changes in these training decisions.

- Sparse models benefit from smaller batch sizes and larger learning rates, while the opposite is observed for dense models.
    - The range of batch sizes used was 65K to 1M, and the range of learning rates used was 1e-4 to 1e-3.
    - This is consistent with the overfitting hypothesis proposed for MoE models, as smaller batch sizes have less accurate gradient updates (the higher the number of inputs used for an update, the more accurate the update). This reduced accuracy

can be thought of as added noise, which serves as regularization, helping with overfitting.

Load balancing is seen as a key challenge for effectively training and fine-tuning sparse models to optimize for modern hardware and prevent token dropping (expert overflow). Experiments conducted on this topic, however, contradict this (for fine-tuning):

- The percentage of tokens dropped (up to 15%) did not seem to have a significant impact in fine-tuning. So, token dropping in fine-tuning does not seem like a problem.
- High capacity factors used during fine-tuning do not seem to have an impact on model quality.
- The addition of an auxiliary load balancing loss seems to have very little impact on fine-tuning.

In terms of the number of experts to choose, works like Switch Transformers show that a large number of experts (up to 512) can improve model quality if designed correctly (although at a diminishing rate). However, to ensure hardware efficiency, an important constraint is necessary: each GPU/TPU core available should have a maximum of 1 expert to minimize memory transfer costs. The main reason for this can be attributed to modern hardware not being optimized for loading parameters to memory. If more than 1 expert is present in a core, whenever the other expert gets called, all experts in that core need to be loaded, leading to inefficiencies.

Deciding on CF (capacity factor) and k in top-k routing should depend on memory and computational resources, as it leads to performance boosts but at the expense of increased costs.

Results

The ST-MoE 32B (32B active parameters, 369B total parameters) becomes the new SOTA in the SuperGLUE tasks (it was trained on all the tasks concurrently).

A 4.1B sparse model designed to match the FLOPs of T5-L (800M active parameters) shows improved performance on all fine-tuning tasks except the two with fewer training examples (around 250 each) -> more signs of MoEs being prone to overfitting.

Finally, an observation was made that upon analysis, the encoder layers generally show specialization in areas such as punctuation, verbs, numbers, names, etc. While decoder expert layers do not show specialization.

My takeaways:

- Router z-loss seems to be a helpful loss function in terms of stabilizing pre-training of MoE models in a mixed-precision environment.
- As shown by the experiments with fine-tuning MoEs, the performance issues seem to come from the scenario where there is a lack of data to use for fine-tuning. If there is enough data available, fine-tuning MoEs obtains better performance than fine-tuning dense models.
    - This can perhaps be explained to the distribution shift fine-tuning data brings in comparison to pre-training data. Naturally, a few experts will be more suited to

the fine-tuning data, and thus will be used more than others, leading to overfitting of the more commonly used experts and underfitting of others.

- Although the fact that only updating sparse parameters during fine-tuning leads to worse performance, it would be worth exploring further if this can be caused by the lack of MoE layers present (only one for every 3 dense layers).
- It is also interesting how MoEs benefit from smaller batch sizes, which add a regularizing effect (adding strength to the claim that MoEs are prone to overfitting).
- Experiments done for this paper show that load balancing is not an issue for fine-tuning, which makes sense since the router should already be fully trained. This seems to indicate that routers can be frozen during fine-tuning.
- It is explained how, although having many experts may lead to performance boosts, a balance needs to be achieved depending on the number of GPU/TPU cores available. Loading more than 1 expert per core leads to inefficiencies.
- Deciding on CF (capacity factor) and k in top-k routing should depend on memory and computational resources, as it leads to performance boosts but at the expense of increased costs.