Unified Scaling Laws For Routed Language Models

Main idea: this paper investigates the scaling behaviors of routing networks, more specifically in the axis of parameter count (in terms of total number of parameters) and computational requirements (total number of active parameters).

Routing:

It experiments with 3 different routing techniques:

- An approach based on BASE (linear programming).
    - This represents a more traditional learned algorithm for routing. BASE in specific approaches routing as a linear programming problem, which naturally distributes tokens evenly through experts (no load balancing issues). The algorithm experimented with has slight modifications to BASE to be more efficient in accelerated hardware (they call it Sinkhorn-BASE).
- A non-parametric approach (hash layer).
    - HASH layers approaches routing as a fixed function of the input, meaning it does not have learnable parameters.
- A Reinforcement Learning approach.

Results:

- Although routing (sparse) performs better than no routing (dense) on all sizes experimented with (up to 1.3B active parameters, up to 512 experts – biggest model has around 200B parameters), the sparse gains over dense are diminishing with scale (BASE is more robust than other routing techniques).

- Scaling the number of experts when the number of active? parameters is fixed improves the validation loss during pre-training.
- Effective Parameter Count (EPC) is created to compare the performance of dense against sparse models based on an equation that considers the total number of parameters and the active parameters of a model.

Main takeaways (as listed in the paper):

- Routing improves performance across all model sizes and routing strategies (compared to dense aka no routing).
- RL routing is more effective than expected, although BASE is the best performer.
- Performance can be described by scaling the number of experts and dense model size.
- Development of an effective parameter count mapping for performance vs scaling.

Recommendations:

- Use routing when training any model with N (parameter count of base model) <= 1.3B.
- Sinkhorn-BASE is a good default routing algorithm.
- Although more experts lead to improved performance, it is recommended to use between 64 and 128 experts due to diminishing returns above that range.
- It is recommended to use k=1 experts.

<u>My takeaways</u>:

- Shows that learned routing (represented through BASE) is the best routing strategy.
    - Non-parametric routing can be used in cases where there might not be enough data to train specific experts (for example, on task/domain-level MoE like DEMix where we are not certain if the training load for each expert will be similar, which will lead to load balancing issues that cannot be solved through traditional auxiliary loss or adding noise – this might not happen at token-level routing)
    - RL routing performs worse than BASE but looks to not be too far off
- To describe performance, the number of experts and dense model size (number of active parameters for each forward pass) are the most relevant features. This is logic as the number of experts represents the horizontal scale of the model while the dense model size represents the vertical scale of the model. (dense model size corresponds to vertical scaling, does number of experts as mentioned here correspond to an increase in the number of experts with the same total parameter count or is this accounting for an increase in the total parameter count coming from the added experts).
- Sparse models seem to be the most useful at small scales, with diminishing returns over dense with an increase in the scale of active parameters, but this can be prevented to a certain extent by choosing a robust routing strategy.
- The result arrived at that scaling the number of experts when the number of active parameters is fixed is logical as this scales the model horizontally. However, my

intuition in this is that scaling the number of experts might make things difficult for fine-tuning (more data will be needed to update all experts while not overfitting on others). Therefore, a balance is needed. (authors recommend between 64 and 128 experts due to diminishing returns in increasing the number of experts). -> how does fine-tuning performance change with differing number of experts and in respect to more training data to use for fine-tuning (explore how scaling the number of experts while keeping the active parameter count constant impacts fine-tuning performance)?

- The EPC equation seems to be useful for practitioners looking to train a MoE model from scratch. This would help with design choices in number of active parameters and number of total parameters.
- Interesting how the authors recommend MoE in scenarios of training smallish models (up to 1.3B). I believe that this is because that was the bigger dense model studied, so it is not saying that dense models perform better when scaled above 1.3B, but just that a bigger dense model was not used in the experiments. It is important to note that the experiments showed diminishing returns for routing models -> did any other papers dive into this question?
- It is also interesting to note that the authors concluded that k=1 experts is the ideal number for k.