From Sparse to Soft Mixtures of Experts

Main Idea: Soft MoE presents a strategy to combat the issue of a traditional MoE of not having the property of continuous differentiability. By making a discrete choice (hard routing) to obtain sparsity, traditional MoE introduces training instabilities, as small changes in the input may lead to large changes in the model's output, since this small change may end up changing the expert(s) chosen. The soft MoE architecture is compatible with certain tasks such as image classification in Vision or machine translation in Language, but it is not compatible with Natural Language Generation (NLG). An equivalent approach that could be compatible with language generation was proposed as "Mixture-of-Tokens" (MoT) (in a different paper), but this MoT architecture seemed to also bring significant challenges that remain unsolved. The continuous differentiability property of Soft MoE is, therefore, only able to be applied to a limited set of tasks.

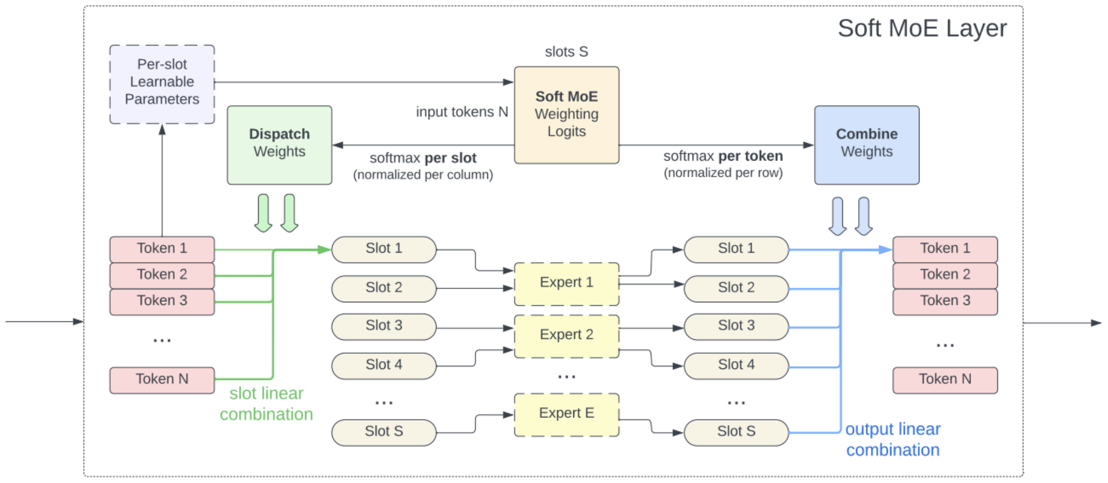
Traditional MoE Routing

In traditional MoE, each input is routed to its corresponding expert in a hard manner (SoftMax is converted to 1 or 0) and the available slots are then occupied by a single token at a time (each slot gets 1 token). This means the experts will be updated solely based on that token.

OBS: slot refers to each inference run supported by the expert until it reaches its maximum capacity.

Soft MoE Algorithm

- Each slot pi of each expert has learnable parameters.
- The input tokens X are passed through each slot and a SoftMax is applied at the column level.
    - This means that the input slots to be passed to each expert are simply a weighted linear combination of all the input tokens with the respective slot's learnable parameters.
- We then obtain the output slots by passing each input slot to a corresponding expert.
- The output slots are then merged through some combine weights, which are the inputs passed through the slot's learnable parameters but now softmaxed at the row level (per token).
- As explained in the paper's figure:



- Soft MoE first computes scores or logits for every pair of input token and slot, based on some learnable per-slot parameters.
- These logits are then normalized per slot (columns)
  - So now we have, for each slot, a weight to give to each input token, which sum up to one per slot.
- And every slot computes a linear combination of all the input tokens based on these weights.
  - The tokens' weights/embeddings are adjusted based on the weights/importance assigned to them per slot.
- Each expert (an MLP) then processes its slots.
  - Now we have the experts' outputs.
- Finally, the same original logits are then normalized per token (by row) and used to combine all the slot outputs, for every token.
  - To get the final output for each token, we then obtain the softmaxed weights now normalized per token (instead of the slot's weights sum up to 1, each token's weights sum up to 1) and combine the expert's outputs with those weights accordingly.

<u>Intuition for softmaxes</u>

- By slot (column)

    - Leads to scores being given for each token by the slot, used to measure the importance which should be given to each token for a specific slot (how much should the slot consider each token).
- By token (row)
    - Leads to scores being given for each slot (by the token), used to measure the importance which should be given to each slot for a specific token (to help determine the final output for each token) (how much should the token consider each slot).

<u>Properties of Soft MoEs</u>

- Usually to get past the token-expert assignment problem, MoE architectures resort to hard assignment methods such as top-k token-choice or expert-choice. These measures are discrete in nature, and thus non-differentiable. Soft MoE, on the other hand, is fully differentiable and continuous.
- Soft MoE does not suffer from token dropping or expert imbalance.
- Soft MoEs adjust better to hardware accelerators than “hard” MoE methods, mainly due to avoiding top-k/sorting routing operations (these are not well suited for hardware accelerators). Therefore, Soft MoEs are fast.
- Soft MoEs are neither sparse (since every token is a weighted average for all input tokens) nor dense (since every expert only processes a subset of the slots, and not all input tokens).

- Traditional MoE models are not so predictable at the sequence-level since inputting a single sequence may force the router to use every expert to balance the load and thus minimize the loss. This can lead to too generalist experts. Traditional MoEs are more predictable at the batch-level (more tokens) since a small number of tokens can fight for the same expert at the sequence level, but this risk is smaller at the batch level. Since in soft MoEs all tokens are grouped together and every expert handles tokens from every input, this risk is not present – leading to more deterministic/predictable and faster inference.

OBS: the number of slots in a soft MoE is a hyper-parameter (must be equal or greater than the number of experts).

<u>Limitation in NLG</u>:

- Soft MoE was only experimented with in an image classification scenario. Translating this method to an NLG setting is not so straightforward.
    - This is because soft MoE uses all input tokens to compute all output tokens at once. In NLG, each input token is generated at a time/separately (one-by-one) and is used as a part of the context to predict the next token. It is possible to use causal masking techniques to only take one token at a time, but this can lead to a bias in training (correlation between token position and a slot).
    - The sequential nature of token generation thus complicates the application of the Soft MoE architecture to language generation tasks.
- More research is needed to translate Soft MoE into an NLG setting.

Memory Consumption

Soft MoE works best when each expert is assigned to one slot only. Therefore, many experts need to be trained and stored, which comes with big costs in terms of memory.

Experiments (Image Classification only)

- Soft MoE is compared to other MoE methods – token-choice and expert-choice – and a dense setting and outperforms all of them in all hyperparameter scenarios.
- With cheaper training and inference costs, Soft MoE outperforms Vision Transformers at a large scale for a given compute budget in both pre-training and fine-tuning.
- Soft MoE scales the number of experts well (more experts = better). Additionally, scaling the number of experts in Soft MoE doesn't really change training time, while this can have a tremendous negative effect in training time with token-choice and expert-choice.

My takeaways:

- Soft MoE seems to instead of routing each token individually, to route all tokens to each expert. This means that the expert will choose how much importance to give to each input token. The weighted average of the experts is then summed up based on the weights given to each token (normalized per token) to get the final output for each token.