Fast Feedforward Networks + Exponentially Faster Language Modeling

Main Idea: the goal of this work is to introduce a new MoE architecture to improve inference time (up to 6x faster than other MoEs). They also claim that FFF (Fast FeedForward) has better training properties due to noiseless conditional execution (no randomness in the gating function).

- Traditional MoEs scale down inference time but remain linear in the width of the feedforward layer (increase in expert parameters). These models also rely on noisy gating for load balancing, which complicates training.
- FFF, on the other hand, uses a binary tree-like structure to improve on these challenges.

Method

- FFF uses nodes to aid the routing mechanism and leaves for the experts.
- The input representation goes through a first node (which is a common MLP layer). The node's output is then passed to a sigmoid to give a probability p. This probability p is used to route the input representation into the next node (left or right branch, as in a binary tree). This process is repeated until a leaf node (expert) is reached)
    - The number of nodes the input goes through (in case of hard routing) corresponds to the depth d of the network.
- MoE chooses an expert width e (size of expert) and trains n separate expert blocks by the partially randomized output of a gating network of width g = [w/e]. The target is then predicted based on the mixture of the k best scoring experts.

    - MoE cost of inference is k*e neurons plus the gating overhead g (g tends to be small).
    - FFFs of depth d learn a tree partition R1, … , R2^d of the input space determined by their nodes, and 2^d small leaf feedforward networks (experts) of width l.
    - FFFs uses a soft routing approach to training, meaning that backpropagation is done by considering the soft routing probabilities p, so training a FFF is more costly than even a feedforward network. However, a hard routing approach at inference (routing only happens through the most relevant nodes) ensures an inference gain over MoE.
        - This soft to hard routing transition is referred to as hardening.
- The FFF routing is more efficient than regular MoE routing.
    - In MoE, a gating network for each expert is needed to calculate the suitability of the specific expert to the input.
    - In FFF, the input is passed through d nodes. Since each node halves the number of experts (leaves) to be considered in future routings for the same input, and because the left/right decision is simpler and thus requires less parameters than a normal MoE gating function, FFF provides a logarithmic routing improvement over MoE in terms of computational overhead at inference. This is especially significant when scaling the number of experts.
- The strategy of soft routing during training comes with the idea that as the leaves specialize, the nodes will be more confident in the routing, leading to probabilities closer to 1 (to the correct path). This process is referred to as hardening. If hardening does not

occur at the expected rate, the hard routing required for inference might not work as well. In those cases, a hardening loss is used.

- Localized overfitting can occur with a high number of leaves, with each leaf being responsible for a very small part of the input space. To diminish this, one can add random child transpositions (flip the p scores given by a node to its child nodes randomly), which ensures the gradients are more diversely distributed, and exposing different nodes and leaves to areas of the input space they otherwise wouldn't see.
    - Hardening can also lead to a shrinking batch problem, mitigated by using larger batch sizes, gradient accumulation and smaller learning rates.

FFFs Applied to NLP

A variant of BERT, deemed UltraFastBERT, is developed, where the feedforward layers are replaced with FFFs.

FFFs provide a forward pass speedup over regular FFs of O(log^2n) compared to O(n), a logarithmic improvement (where n is neurons). This improvement comes from FFF's balanced binary tree structure, which only executes one branch of the tree conditionally on the input.

UltraFastBERT has 4095 neurons (leaves + nodes) and is compared to a 3072-neuron BERT.

UltraFastBERT only uses 1/341 of its neurons for inference while BERT uses all its 3072 neurons.

- This leads to a 78x speedup (not a 341x speedup, as would be expected) due to hardware optimization for matrix multiplication favoring FFs.

Results

- UltraFastBERT performs on-par with BERT on fine-tuning in downstream tasks, with a 78x inference speedup.
- UltraFastBERT shows that only a fraction of parameters of feedforward networks needs to be applied at inference.
- The concept of FFFs can technically be applied to decoder-only models as well.

My takeaways:

- The efficiency gains are a result of instead of passing the input to a routing mechanism which considers all experts, having the router only decide between two experts (sending the input to a specific side of the binary tree.
    - While traditional routing expects the router to choose the specific part of the input space of each expert, this binary tree approach has the router dividing the input space in half at every decision, eventually leading to the desired input space.
- FFF routing seems to be theoretically less expensive, but not allow parallelization (each node decision needs to be performed sequentially), so gains might not be as significant as expected.