AI 실전 11주차

여러가지 모델 합친 LLM 모델 MoE

MoE 모델 효율적으로 사용하기 위한 K Transformers

MoE?

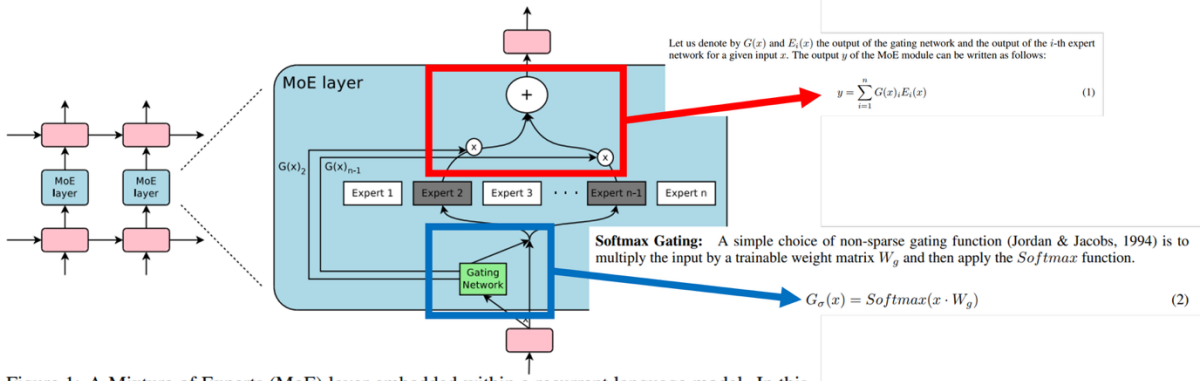

Figure 1: A Mixture of Experts (MoE) layer embedded within a recurrent language model. In this case, the sparse gating function selects two experts to perform computations. Their outputs are modulated by the outputs of the gating network.

- **Softmax Gating(파란색)과 MoE의 최종 output vector(빨간색)**
- 전문가를 활성화 여부를 사용하여 특정 모델 사용
- Output 영역으로 기존 모델에 계산하기 위한 레이어
- 효율적인 모델 추론과 성능 높히기 위한 방안

Outrageously Large Neural Networks: The Sparsely-gated Mixture-of-Experts Layer('17)

MoE 전문가 사용 여부

Noisy Top-K Gating: We add two components to the Softmax gating network: sparsity and noise. Before taking the softmax function, we add tunable Gaussian noise, then keep only the top k values, setting the rest to $-\infty$ (which causes the corresponding gate values to equal 0). The sparsity serves to save computation, as described above. While this form of sparsity creates some theoretically scary discontinuities in the output of gating function, we have not yet observed this to be a problem in practice. The noise term helps with load balancing, as will be discussed in Appendix A. The amount of noise per component is controlled by a second trainable weight matrix W_{noise}.

$$G(x) = Softmax(KeepTopK(H(x), k)) \tag{3}$$

$$H(x)_i = (x \cdot W_g)_i + StandardNormal() \cdot Softplus((x \cdot W_{noise})_i) \tag{4}$$

$$KeepTopK(v, k)_i = \begin{cases} v_i & \text{if } v_i \text{ is in the top } k \text{ elements of } v. \\ -\infty & \text{otherwise.} \end{cases} \tag{5}$$

- Noisy Top-K Gating 기법을 이용하여 전문가 활용 여부를 넣음.
- 계산에 필요한 특정 전문가 모델을 음수 무한으로 하여 반영하지 않게 함.
- **Normalization과 Softplus를 적용**

GShard, FFN 레이어를 MoE로 적용 사례

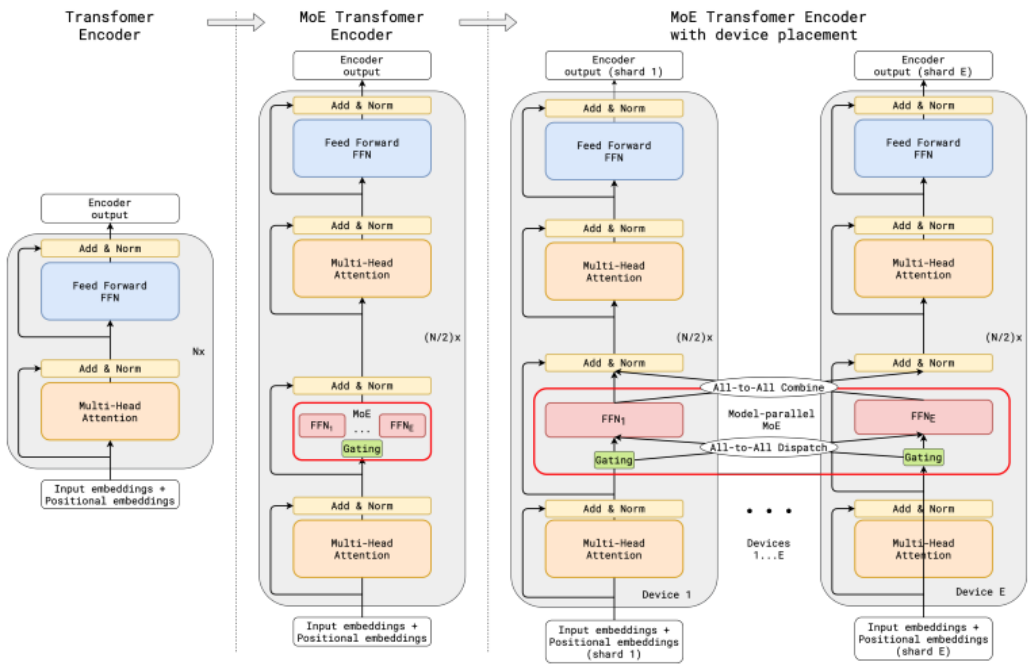

Figure 3: Illustration of scaling of Transformer Encoder with MoE Layers. The MoE layer replaces the every other Transformer feed-forward layer. Decoder modification is similar. (a) The encoder of a standard Transformer model is a stack of self-attention and feed forward layers interleaved with residual connections and layer normalization. (b) By replacing every other feed forward layer with a MoE layer, we get the model structure of the MoE Transformer Encoder. (c) When scaling to multiple devices, the MoE layer is sharded across devices, while all other layers are replicated.

- Transformer의 Encoder에서 FFN 레이어를 MoE로 대체함.
- 단점으로 특정 분야의 전문가가 대부분 활용됨.
- 다른 전문가로 계산하기 위한 딜레이 발생

GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding('20)

Switch Attention

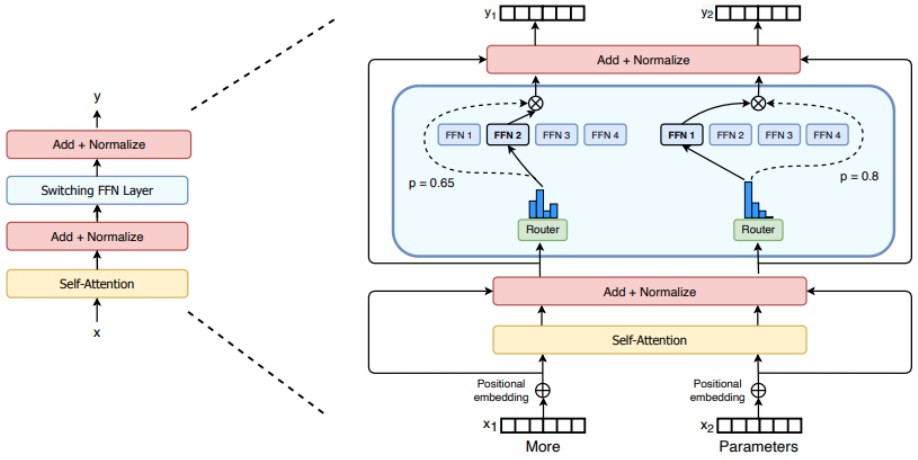

Figure 2: Illustration of a Switch Transformer encoder block. We replace the dense feed forward network (FFN) layer present in the Transformer with a sparse Switch FFN layer (light blue). The layer operates independently on the tokens in the sequence. We diagram two tokens (x_1 = "More" and x_2 = "Parameters" below) being routed (solid lines) across four FFN experts, where the router independently routes each token. The switch FFN layer returns the output of the selected FFN multiplied by the router gate value (dotted-line).

- 기존의 transformer에 FNN 레이어를 Switch FNN(MoE)으로 바뀐 모델
- FNN Layer에서 파라미터를 꽤나 많이 사용함!

각 토큰별 맞는 전문가 모델 추론

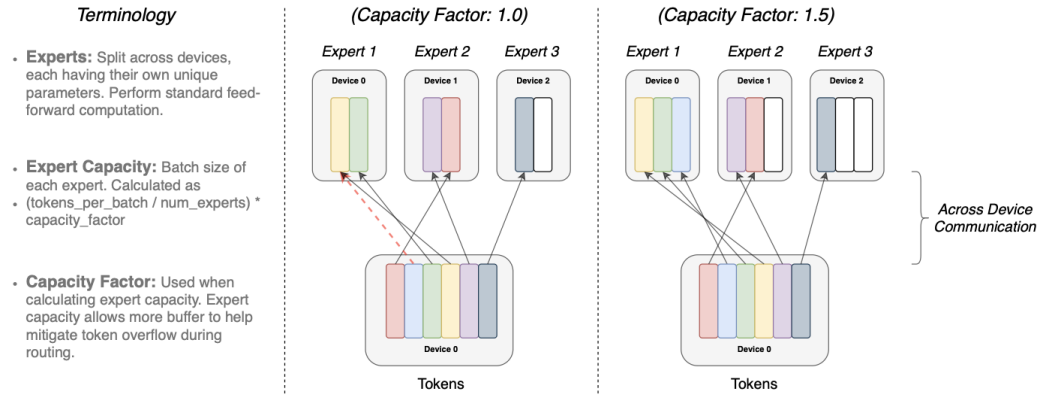

Figure 3: Illustration of token routing dynamics. Each expert processes a fixed batch-size of tokens modulated by the *capacity factor*. Each token is routed to the expert with the highest router probability, but each expert has a fixed batch size of (total_tokens / num_experts) × capacity_factor. If the tokens are unevenly dispatched then certain experts will overflow (denoted by dotted red lines), resulting in these tokens not being processed by this layer. A larger capacity factor alleviates this overflow issue, but also increases computation and communication costs (depicted by padded white/empty slots).

- 각 토큰에 따라 각 해당하는 전문가가 대답할 수 있게 모델 추론 함
- MoE의 router 적용하여 각 토큰에 관한 전문가 사용 여부 계산

MoE 프레임워크 K Transformers

- MoE의 모델을 필요한 레이어만 불러오기 및 효율적인 계산 가능 (배치 및 병렬계산)
- MoE 용량이 200GB 이지만 계산할 때 필요한 용량은 약 20GB 사용
- 사용한 용량 줄어드는 것 뿐만 아니라 추론속도 향상

http://github.com/kvcache-ai/ktransformers