7. Advanced Pretraining and Finetuning Techniques

Efficient transformers

Compute cost of transformers

Q, K, V projections:

$n \times d_e \xrightarrow{\text{linear}} n \times d$ $\qquad O(n \times d_e \times d)$

Scaled dot-product attentaion:

$(n \times d)(d \times n) \xrightarrow{\text{matmul}} n \times n$ $\qquad O(d \times n^2)$ — slow

Feed-forward layer (GPT-2):

$n \times d \xrightarrow{\text{linear+ReLU}} n \times d_h \xrightarrow{\text{linear+ReLU}} n \times d$ $\qquad O(n \times d \times d_h)$

Improve efficiency of transformers:

- Quantization (training and inference)
- Weight sharing (training)
- Sparsely-activated models (training and inference)
- Pruning (inference)
- Distillation (inference)

Specifically, improve efficiency of self-attention (reduce the $O(n^2)$ time and memory cost):

- Sparsify the attention matrix
  - Deterministic mask:
    - Blockwise self-attention [Qiu et al., 2020], Longformer [Beltagy et al., 2020]: attention within a local window
  - Data-dependent mask
    - Reformer [Kitaev et al., 2020]: attention within an adaptive local window
- Compress the key-value memory
  - Low-rank prediction (self-attention is low rank)
    - Linformer [Wang et al., 2020]: compute self-attention in a lower dimension
  - Attention-based projection
    - Perceiver [Jaegle et al., 2021]: use latent states to compress the KV memory

Efficient finetuning

- Finetune a subset of parameters

  - Freezing the first X layers [Lee et al., 2019]
  - BitFit [Ben-Zaken et al., 2022]: only finetune the bias term (0.1% of the parameters)
- Adapt the frozen pretrained model
  - Adapter [Houlsby et al., 2019]: insert small networks to the pretrained model
  - LoRA [Hu et al., 2021]: add low-rank matrices as additional parameters