Low rank attention denoising

Nikita Okhotnikov

MIPT

2025

Introduction

Problem

- Noisy attention weights distribution in modern LLMs hurts interpretability and potentially limits the performance
- Existing solution utilizes large number of additional parameters and requires training from scratch

Objective

Implement parameter and training time efficient attention score denoiser

Assumption

Attention noise has lower intrinsic dimensionality than the signal

Proposal

Low rank implementation of existing adaptive denoiser[1] that does not require training from scratch

[1] Yueyang Cang et al. DINT Transformer, 2025

Attention & DIFF attention

Attention mechanism

$X \in \mathbb{R}^{N \times d_{model}}$ – input, $W_Q, W_K, W_V \in \mathbb{R}^{d_{model} \times d}$ – projection matrices

$$Q = XW_Q, K = XW_K, V = XW_V$$

$Q, K, V \in \mathbb{R}^{N \times d}$ – projected queries, keys, values

$$\text{Attention} = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$

DIFF attention mechanism

$X \in \mathbb{R}^{N \times d_{model}}$ – input, $W_Q, W_K \in \mathbb{R}^{d_{model} \times 2d}$, $W_V \in \mathbb{R}^{d_{model} \times d}$ – projection matrices

$$[Q_1, Q_2] = XW_Q, [K_1, K_2] = XW_K, V = XW_V$$

$Q_1, K_1, Q_2, K_2, V \in \mathbb{R}^{N \times d}$ – projected queries, keys, values

$$\text{DIFFAttention} = \left(\text{softmax}\left(\frac{Q_1 K_1^T}{\sqrt{d}}\right) - \lambda \cdot \text{softmax}\left(\frac{Q_2 K_2^T}{\sqrt{d}}\right)\right)V, \ \lambda \in (0, 1)$$

DINT Attention & low rank modification

DINT Attention

$$A_1 = \text{softmax}\left(\frac{Q_1 K_1^T}{\sqrt{d}}\right), A_2 = \text{softmax}\left(\frac{Q_2 K_2^T}{\sqrt{d}}\right), A_3 = \text{repeat}\left(\frac{1}{N}\sum_{i=1}^{N} A_1[i,:], N\right)$$

$$\text{DINTAttention} = (\lambda \cdot A_3 + A_1 - \lambda \cdot A_2)\, V$$

Low rank DINT

$$Q_1 = XW_{Q_1}, K_1 = XW_{K_1}, Q_2 = XW_{Q_2}^{down} W_{Q_2}^{up}, K_2 = XW_{K_2}^{down} W_{K_2}^{up}$$

$$W_{Q_1}, W_{K_1} \in \mathbb{R}^{d_{model} \times d}; W_{Q_2}^{down}, W_{K_2}^{down} \in \mathbb{R}^{d_{model} \times r}; W_{Q_2}^{up}, W_{K_2}^{up} \in \mathbb{R}^{r \times d}$$

$$A_1 = \text{softmax}\left(\frac{Q_1 K_1^T}{\sqrt{d}}\right), A_2 = \text{softmax}\left(\frac{Q_2 K_2^T}{\sqrt{d_2}}\right), A_3 = \text{repeat}\left(\frac{1}{N}\sum_{i=1}^{N} A_1[i,:], N\right)$$

as linear layer of shape$[d_{in}, d_{out}]$ is initialized with $w[i,j] \sim U[-\sqrt{d_{in}}, \sqrt{d_{in}}]$

$$\mathbb{D}[W_{Q_2}^{down} W_{Q_2}^{up}[i,j]] = \mathbb{D}[W_{K_2}^{down} W_{K_2}^{up}[i,j]] = \frac{1}{3d_{model}} \cdot \frac{1}{3r} \cdot r = \frac{1}{9}\mathbb{D}[Q_1[i,j]]$$

$$\implies d_2 = \frac{d}{81}, \quad \text{LowRankDINTAttention} = (\lambda \cdot A_3 + A_1 - \lambda \cdot A_2)\, V$$

Experiments

SFT of Llama-3.2-1B model[2] on OpenMathInstruct-2 dataset[3]

HERE SHOULD BE TRAINING CURVES

[2] https://huggingface.co/meta-llama/Llama-3.2-1B
[3] https://huggingface.co/datasets/nvidia/OpenMathInstruct-2