CMSC499A Report, University of Maryland

Ananth Sankaralingam

December 12, 2024

1 Introduction

In the following report, I will cover my experience in CMSC499A in the Fall 2024 semester. I will include what I learned, what problems I solved and how I contributed to the Huang lab.

2 Problem Description

This semester, I worked under Dr. Tahseen Rabbani on the topic of language models. He introduced me to the subject through a few whiteboarding sessions, defining key terms like embeddings and next token prediction. Beyond the whiteboarding sessions, I initially struggled to grasp the nature of transformers in practice, but eventually built enough basic knowledge to understand our goal: compression.

Introduced in the renowned paper *Attention is All You Need*, the transformers architecture redefined the field of natural language processing through a variety of improvements in text processing and generation. Unlike previous RNNs, transformers introduced the self-attention mechanism, which enabled contextual understanding of tokens by processing entire sequences simultaneously. The attention module relies on the key ($\mathbf{K}$), value ($\mathbf{V}$), and query ($\mathbf{Q}$) embeddings of all available tokens, computing the softmax of the scaled dot product:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}.$$

This allows for contextual understanding, as embeddings of tokens are computed in relation to each other, creating abstract vector spaces for similar meaning tokens to coexist. To reduce redundant computation of key-value pairs for reappearing tokens in a sequence, the KV cache maintains all key-value pairs of existing tokens, significantly reducing time complexity. However, as context length grows, KV cache size grows linearly with it, introducing severe memory bottlenecks. The paper, *Model Tells You What to Discard: Adaptive KV Cache Compression for LLMs*, aimed to solve this by only storing high-attention tokens in the cache. However, this reduced output text quality significantly while still calculating attention, providing an inept answer to an unsolved problem.

3 Approach

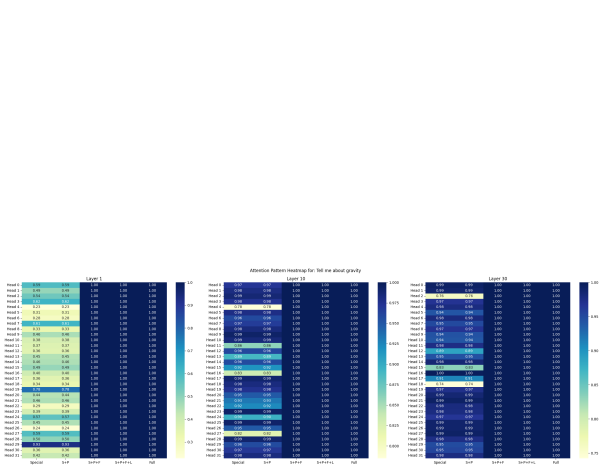
After discussing and reading papers with Tahseen on the memory bottleneck of the KV cache, we began to explore existing strategies. I learned of several methods such as H20, Scissorhands, etc., providing context behind the problem we were solving. Although the actual math behind our implementation was mainly derived by Tahseen, I offered various (probably incorrect) solutions and ideas along the way, boosting my own understanding of the problem and bringing a new perspective to the team. In our KV cache compression strategy, we maintain a fixed-size cache and dynamically add and drop tokens at each decoding step, estimating token relevance without performing full attention computation. As we process each query token in a sequence, we project it to a lower-dimensional space using Gaussian random projections and binarize the resulting values. These binary representations are then hashed using locality-sensitive hashing (LSH) to facilitate efficient similarity comparisons.

We approximate cosine similarity between the current query token and cached key tokens by computing the Hamming distance between their binary hashes. If the similarity is high enough (i.e., low dissimilarity), the token is estimated to be relevant and therefore retained in the cache. To maintain the fixed-size constraint, we replace the least relevant token in the KV cache, determined by its expected contribution to future attention scores. The hash table, which stores the binary representations, facilitates these updates but is distinct from the KV cache itself.

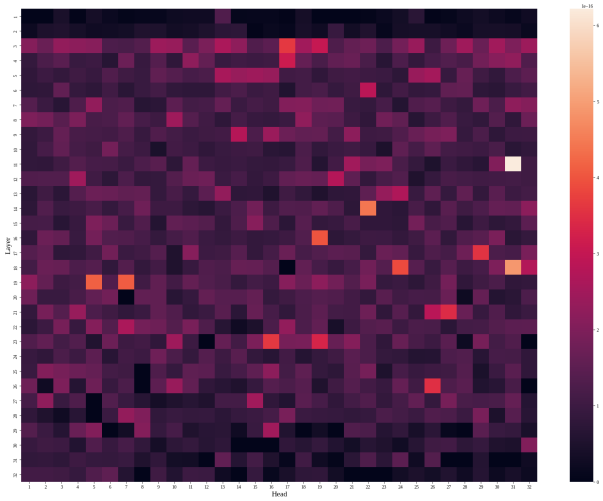
This strategy ensures that the KV cache remains fixed in size, independent of the context length, and reduces the computational cost by pre-filtering tokens before the full attention computation. By doing so, we achieve memory efficiency without significantly degrading model performance.

4 Contribution

During the course of this semester, I contributed to the Huang Lab in a variety of ways. In the initial stages of the paper, I started by hacking LLaMA 3 to extract attention scores and understand their nature. This involved creating additional parameters and hooks to the direct source code, modifying the forward pass in classes like the Transformer and Attention module. Through this, our team was able to gain insight into the various behaviors of each attention head, learn of attention sinks for the first token in a sequence, and visualize scores over different tokens. Although this was not the final approach for the compression algorithm, hacking LLaMA 3 allowed me to understand the architecture and drive early insights. Later on, I also helped build an attention profiler for different types of tokens (special characters, etc.) to replicate figures shown in the *Model Tells You What to Discard: Adaptive KV Cache Compression for LLMs* paper. As we continued past the initial stages of exploring other works, we began writing our own compression algorithm. This initially involved modifying LLaMA 3 source code again, but we quickly diverted after discovering "Cold Compress," an open source template for implementing KV Cache compression strategies. During this process, I mostly helped to debug various errors and completed other miscellaneous tasks. After our algorithm was written, subsequent tasks pertained to our end goal: the paper. I shifted my focus to contributing to experiments by writing custom tests (GPT-as-a-Judge), as well as implementing strategies directly offered from Cold Compress on Google Cloud. Furthermore, I visualized the effectiveness of our algorithm, measuring the Attention Loss Ratio (ALR) by comparing attention across various prompts between the modified and base model. Next, I helped with the written content of the paper, contributing a rough draft of the related works sections. By reading similar papers and their related works sections, I summarized adjacent algorithms before Tahseen cleaned it up for the final version.



(a) Attention patterns for layers 1, 10, and 30 for prompt "Tell me about gravity"



(b) ALR across 32 heads and 32 layers

Figure 1: Visualizations of attention patterns and ALR for LLaMA 3 modifications.

5 Results

Through the course of working in the Huang Lab and writing the paper *LSH Tells You What To Discard: An Adaptive Locality-Sensitive Strategy for KV Cache Compression*, I expanded my horizons in several ways. The research experience brought about a variety of challenges from vague tasks and technical understanding, but tackling each one allowed me to grow as both a student and researcher. From a technical perspective, I feel far more comfortable in the field of machine learning and natural language processing, and am eager to continue my journey. From an academic standpoint, consistently digesting papers where I had minimal background knowledge developed resillience, but more importantly confidence. After attending NeurIPS, I am even more inspired to continue research in AI/ML and have already started working with the team on the follow up paper. Moving forward, I aim to expand my knowledge and write more, but set myself tangible goals of writing a paper independently and teaching core ML concepts on my YouTube channel.