CORM: Coarse-to-fine-grained Offloading for SMoE LLM Inference on Consumer-grade GPU

Anonymous ACL submission

Abstract

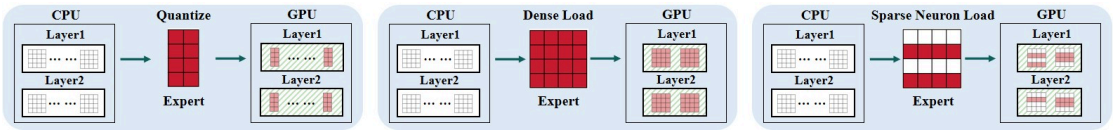
Large language models (LLMs) with sparse mixture-of-experts (SMoE) have shown empirical success in various tasks. The sparse routing strategy of SMoE increases the model capacity without proportionally increasing the computation cost by activating only a subset of the parameters ($i.e.$, the experts). Unfortunately, compared to previous LLMs without SMoE, the capacity benefit of SMoE LLMs also brings additional substantial memory resources. Thus, deploying SMoE LLM in resource-limited scenarios is a challenging issue. Previous approaches involved offloading the expert weights of MoE models to the CPU, which significantly increased inference latency due to the need to copy expert weights between the CPU and GPU. To address this issue, we propose CORM, a **C**oarse-to-fine-grained **O**ffloading framework fo**R** **SM**oE large language model inference. This framework leverages the sparsity present in both the expert and neuron levels of large models, offloading both levels accordingly. We have designed an efficient and memory-saving coarse-to-fine-grained sparsity prediction mechanism that allows inference and weight prefetching to occur in parallel. We also implement a coarse-to-fine-grained caching strategy which minimizes the need for repeated weight loading. Our method has been proven through experiments to significantly accelerate SMoE LLMs inference, reducing latency by up to 2.14× while the model's accuracy only decreases by 1%. These features enable our coarse-to-fine grained offloading framework to efficiently deploy large-scale SMoE LLMs on a consumer-grade GPU.

1 Introduction

The adoption of Sparse Mixture-of-Experts (SMoE) large language models (LLMs) in the AI field is becoming increasingly widespread, such as in chatbots and code generation (Reid et al., 2024; OpenAI, 2022; xAI, 2024). The core idea of SMoE is to activate parameters ($i.e.$, the experts) conditional on given inputs (Jacobs et al., 1991; Shazeer et al., 2017). Therefore, LLMs with SMoE can expand their model parameter scale with little or a minor increase in computation cost. The SMoE LLM has demonstrated superior performance compared to dense LLMs with equivalent computational demands, such as Mixtral (Jiang et al., 2024), Switch-Transformer (Fedus et al., 2022), and DeepSeek-MoE (Dai et al., 2024).

Deploying LLMs on consumer-grade GPUs has become a popular trend (Sheng et al., 2023; Song et al., 2023; Zhao et al., 2024) because it has the following advantages: safeguarding data privacy (Yan et al., 2024), reducing energy consumption (Xu et al., 2023), easier model customization (Lyu et al., 2023), etc. However, deploying SMoE LLMs on consumer-grade GPUs presents a significant challenge compared to dense LLMs, because SMoE LLMs are typically equipped with more model parameters than their dense counterparts, far exceeding the memory capacity of consumer-grade devices (Yi et al., 2023). For instance, Mixtral 8×7B (Jiang et al., 2024), a cutting-edge SMoE LLM, features each MoE layer with 8 experts and demands 94GB of memory in half-precision representation. At the same time, an NVIDIA RTX 3090Ti GPU has only 24GB of memory.

There are approaches (Yi et al., 2023; Eliseev and Mazur, 2023; Kong et al., 2023) aimed at enabling SMoE LLMs inference on devices with limited resources. However, these approaches either result in reduced accuracy or unacceptable inference latency. Again, we take Mixtral 8×7B as an example: the offloading technique offloads experts to CPU DRAM and loads them as needed. While this makes inference feasible with 24GB of GPU memory, it results in an inference latency of 3.9 seconds per word, significantly slower than the human reading speed of 0.2 to 0.3 seconds per word (Jin et al., 2023). Meanwhile, quantization



(a) Quantization (b) Offloading (c) CORM(Ours)

Figure 1: Illustration of CORM vs. existing methods. (a) Quantization uses low precision to represent model weights and stores them in GPU memory, avoiding weight loading but potentially causing accuracy degradation. (b) Offloading stores MoE layer weights in CPU DRAM and loads them into GPU memory on-demand during inference. This approach does not affect accuracy but introduces unacceptable loading latency. (c) Our CORM stores MoE layer weights in CPU DRAM and, during inference, loads the active neurons of experts into GPU memory. This maintains model accuracy while increasing throughput.

	Latency (s/token)	Accuracy (%)
Quantization	0.4	65.6
Offloading	3.9	71.2
CORM	1.28	70.3

Table 1: Comparison of Mixtral 8×7B inference latency and accuracy on MMLU dataset between CORM and existing methods. CORM achieves efficient inference while maintaining model accuracy.

can reduce the model's accuracy to unacceptable levels. For instance, applying INT2 quantization to MoE layers and INT4 quantization to Attention layers of Mixtral 8×7B allows it to fit into a GPU with 24GB of memory. However, this results in an average accuracy drop of 5% (Eliseev and Mazur, 2023). To summarize, deploying SMoE LLMs in consumer-grade devices makes it hard to strike balance between inference latency and accuracy. Thus, the challenge lies in achieving efficient inference while maintaining model accuracy.

To address this challenge, we propose CORM, an efficient framework for SMoE LLM inference with a coarse-to-fine-grained offloading, enabling SMoE LLMs deployment on consumer-grade GPUs, where coarse grain refers to expert level and fine grain refers to neuron level. The core idea of CORM is to leverage dual-levels of sparsity inherited from SMoE LLMs to accelerate inference: the sparse routing policy of SMoE and the neuron sparsity nature of LLMs.

The key components of CORM involve coarse-to-fine-grained loading. By utilizing the neuron sparsity of LLMs, we only load active neurons from the selected experts to GPU, minimizing weight loading. Additionally, although training SMoE models tends to balance expert usage, actual inference often shows imbalanced expert usage (Du et al., 2024). For example, (Li et al., 2023b) shows the most popular expert receives 4.02× and 5.56× tokens of the least popular ones in 4-expert and 16-expert inference tasks of Transformer-XL (Dai et al., 2019). Based on this observation, we employ mixed quantization for experts in SMoE LLMs, using lower precision for less frequently used experts. This reduces memory usage and weight loading latency without affecting model accuracy.

Moreover, we design a coarse-to-fine-grained prediction and prefetching mechanism to improve parallelism. Exploiting the high similarity of activations between adjacent layers in LLMs, we utilize the hidden state of current layer along with the weights of next layer to predict the selected experts and active neurons in the subsequent layer. This achieves accurate predictions without additional memory requirements. We effectively hide computation latency by overlapping weight loading and computation through prediction and prefetching.

To efficiently utilize GPU memory and reduce expert and neuron loading, CORM further includes coarse-to-fine-grained caching, maintaining caches for both expert and neuron levels. We maintain an expert cache for each MoE layer, with an internal neuron cache for each expert in the cache, updating the neuron cache upon expert cache hits.

Experimental results demonstrate the effectiveness of our approach when deployed on a single NVIDIA RTX 3090Ti GPU. As shown in Table 1, CORM takes both inference efficiency and accuracy into account. Applying CORM to state-of-the-art SMoE LLMs achieved up to 2.14× inference speedup compared to the state-of-the-art library FasterTransformer (NVIDIA, 2019) from NVIDIA. Accuracy degradation is 1% compared to the original SMoE LLMs across a wide range of natural language processing tasks including commonsense reasoning and multitasking from various branches of knowledge. Furthermore, we show several ablations on different components of CORM and their

contributions to end-to-end inference speedup.

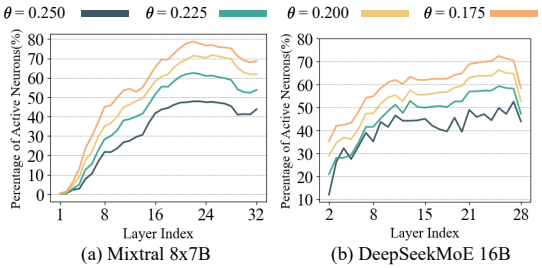
Our contributions are summarized as follows:

- We propose CORM, a dual-level offloading method efficiently deploys SMoE LLMs on consumer-grade GPUs. CORM utilizes two levels of sparsity from expert and neuron levels to optimize weight loading process.
- We design a coarse-to-fine-grained prediction and prefetching mechanism that allows for accurate predictions of the required weights without additional memory requirements, enabling the overlap of weight loading and computation to hide computation latency.
- We design a coarse-to-fine-grained caching strategy. This dual-level caching minimizes the need for repeated weight loading.
- We demonstrate the effectiveness of CORM through extensive experiments on state-of-the-art SMoE LLMs. Our results show up to 2.14× inference speedup with 1% accuracy degradation across a wide range of natural language processing tasks.

2 Related work

Recent years numerous LLMs have been proposed (Zhang et al., 2022; Touvron et al., 2023a,b; Almazrouei et al., 2023; Chowdhery et al., 2023), shows outstanding performance in natural language processing tasks. As a variant of LLMs, SMoE LLMs have become increasingly popular due to their superior performance compared to common LLMs under the same computational budget. Recently, many MoE LLMs have been introduced, such as GLaM (Du et al., 2022), Switch Transformer (Fedus et al., 2022), Mixtral (Jiang et al., 2024), DeepSeekMoE (Dai et al., 2024).

The latest advancements in LLMs and SMoE LLMs have highlighted the importance of their inference workloads, prompting various efforts to accelerate inference. These include quantization (Frantar et al., 2022; Xiao et al., 2023; Frantar and Alistarh, 2024), operator fusion (Dao et al., 2022; Dao, 2023; Hong et al., 2023; Li et al., 2023a), activation sparsity (Liu et al., 2023; Song et al., 2023), and sparsification (Alizadeh et al., 2023). Additionally, some works employ offloading methods to deploy models on edge devices and accelerate inference (Sheng et al., 2023; Song et al., 2023). Specifically for SMoE LLMs, several studies have analyzed expert usage characteristics during inference to enhance acceleration (Du et al., 2024; Jiang et al., 2024; Kamahori et al., 2024; Xue et al., 2024; Shen et al., 2022).



(a) Mixtral 8x7B (b) DeepSeekMoE 16B

Figure 2: Percentage of active neurons by layer on varying thresholds θ. We can ignore about 35% of neurons at θ of 0.175 and 55% at θ of 0.25. This observation demonstrates that neuron sparsity exists not only in dense LLMs but also in SMoE LLMs.

3 Observation

The feed-forward network (FFN) layer of an LLM contains three fully connected layers: W1, W2, W3 and an activation function (Touvron et al., 2023b). The computation process proceeds as follows: first, the input vector is supplied to W1 and W3. The output vector from W1 is then passed through the activation function. This activated result is multiplied element-wise with the output vector from W3, and the product is subsequently supplied to W2. We obtain the output vector of W2 as the final output of the FFN layer.

During inference, the activation function maps some of the activated values to a very small range (Agarap, 2018). When these small values are involved in further computations, the neurons of W2 that interact with them become negligible because their impact on the final result is minimal. Here, a neuron refers to a row or column of weights in a fully connected layer. Moreover, we can infer that the corresponding neurons in W1 and W3, which relate to values ultimately mapped to a small range by the activation function, are also insignificant. These insignificant neurons are termed inactive neurons, while those affecting the result are called active neurons. Previous studies (Liu et al., 2023; Song et al., 2023) have referred to this phenomenon as neuron sparsity, which is highly dynamic and dependent on the input data.

Existing research has primarily focused on neuron sparsity in dense LLMs. However, in SMoE LLMs, each expert constitutes an FFN layer, and

we have observed similar characteristics in SMoE LLMs. In our approach, a neuron is considered active if the absolute value of its activation exceeds a threshold, denoted as θ. Conversely, if the absolute activation value is below this threshold, the neuron is considered inactive. Our experiments on Mixtral 8×7B and DeepSeekMoE 16B revealed neuron sparsity, as shown in Figure 2. Neuron sparsity is approximately 35% at θ of 0.175 and 55% at θ of 0.25. Lower layers exhibit higher sparsity, which gradually decreases in deeper layers. This finding indicates leveraging neuron sparsity to accelerate the inference of SMoE LLMs is feasible.

4 Coarse-to-fine-grained Offloading for SMoE LLM Inference

In this paper, we introduce CORM, an efficient SMoE LLM offloading inference framework deployed on consumer-grade GPUs. CORM focus on sparsity at both coarse-grained and fine-grained levels: coarse-grained at the expert level and fine-grained at the neuron level. The core idea of CORM is to leverage two inherent sparsity conditions from SMoE LLMs to accelerate inference: the sparse routing policy of SMoE and the neuron sparsity nature of LLMs. Figure 3 presents an overview of our design. During inference, CORM only loads necessary neurons within the experts, reducing the weight loading volume and improving throughput. Additionally, CORM predict the required experts and neurons and use prefetching to improve parallelism. By utilizing the model's own features and weights for prediction, we effectively control memory overhead while maintaining model accuracy stability. Furthermore, CORM implement a cache strategy that transitions from coarse-grained to fine-grained levels, minimizing redundant weight loading and further enhancing inference efficiency.

4.1 Coarse-to-fine-grained Offloading and Expert Granularity Mixed Quantization

Based on the observations, we discovered that neuron sparsity is also present in SMoE LLMs. In CORM, we only need to use active neurons during inference, allowing us to implement "sparse neuron loading" — only loading active neurons. The process of sparse neuron loading is as follows: first, the active neurons are compressed into a matrix on CPU, then load the compressed matrix to GPU, and finally, decompressed on GPU. This process significantly decreases loading latency.

We further reduce the weight loading through expert granularity mixed quantization. Although each expert is selected with equal frequency during training, expert load imbalance still occurs in practical inference (Zhou et al., 2022; Fedus et al., 2022). In our inference framework, experts are quantized with different precisions; less frequently used experts are quantized at lower precision to accelerate inference with minimal impact on accuracy. To determine which experts should be quantized at lower precision, we conduct offline statistics by testing on general datasets like C4 (Raffel et al., 2020) and Wikipedia (Foundation, 2023).

4.2 Coarse-to-fine-grained Predicting and Prefetching

In LLM inference, prefetching means moving the needed weights to GPU in advance, in parallel with the computation processes on GPU (Hwang et al., 2023; Liu et al., 2023). This method hides part of the computation latency, thus reducing inference latency. However, in SMoE LLMs, both expert selection and neuron sparsity are dynamic. We only know the exact expert selection and active neurons upon reaching the respective MoE layer. This sequential execution prevents us from prefetching weights. Previous work added and trained a predictor for each FFN layer to predict the sparsity of the neurons ahead of time (Liu et al., 2023). However, implementing this approach in SMoE LLMs would incur unacceptable additional memory overhead in memory-constrained edge GPU scenarios, or would lead to unacceptable prediction accuracy. Specifically, small predictors achieve only 50-60% prediction accuracy, while larger predictors require an additional 50-100 MB of memory per expert per layer for state-of-the-art SMoE LLMs. For instance, the Mixtral 8×7B model, which employs eight experts per layer, would require an additional 7-14 GB of memory to achieve satisfactory prediction accuracy. This additional memory requirement is unacceptable for the NVIDIA RTX 3090Ti GPU, which has only 24 GB of memory.

To save memory, our framework utilizes the model's own weights to predict expert selection and neuron sparsity, representing a training-free approach that requires **no additional memory**. As shown in Figure 4, we add a predictor after the i^{th} Attention layer to predict the $(i+1)^{th}$ MoE layer. In the i^{th} predictor, the input is calculated with the gate function of the $(i+1)^{th}$ layer to predict

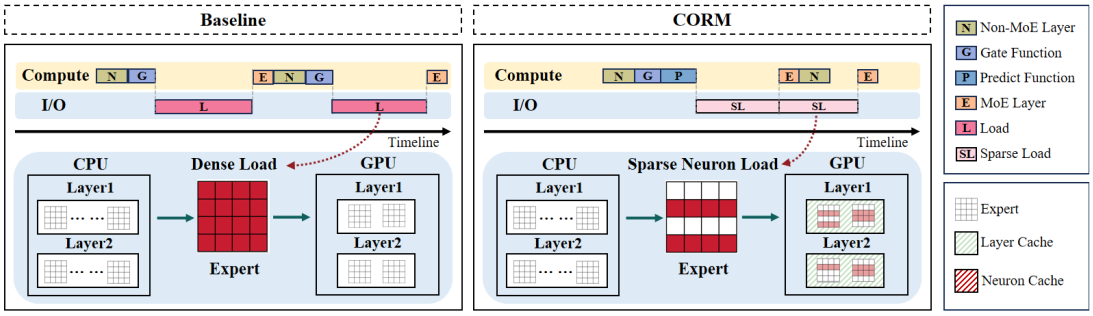


Figure 3: Overview of CORM. The core of CORM is loading the active neurons of experts to reduce weight loading during SMoE LLM inference. We add a predictor before each MoE layer. The predictor predicts the selected experts and active neurons of next MoE layer. After completing predictions, we perform prefetching and forward inference in parallel to mask computational latency. During weight loading, we maintain caches for expert and neuron levels.

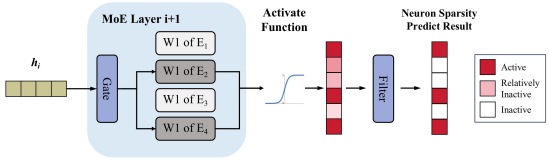
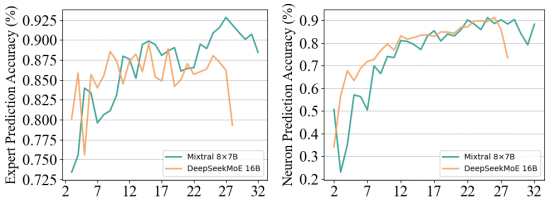


Figure 4: Structure of expert selection and neuron sparsity predictor. Given the input to MoE layer at Layer i, we utilize the gate function of MoE layer at Layer $i+1$ to predict the expert selection for Layer $i+1$. Then calculating input with predicted expert's W1 layer, and pass the result through an activation function. Using the output from activation function, we predict neuron sparsity by filtering out relatively inactive neurons.



(a) Expert Prediction Accuracy (b) Neuron Prediction Accuracy

Figure 5: CORM's prediction accuracy by layer. (a) illustrates Expert selection prediction accuracy, (b) illustrates neuron sparsity prediction accuracy. CORM shows expert selection prediction accuracy of around 85% and neuron sparsity prediction accuracy of around 80% on state-of-the-art SMoE LLMs.

expert selection for the $(i+1)^{th}$ layer. Next, we prefetch the predicted expert's W1 layer to GPU, calculate input with W1 layer, and then pass the result through an activation function. Based on obtained activation values, we employ a filter function to generate predictions for neuron sparsity. The filter selects neurons whose absolute activation values exceed a threshold θ to predict active neurons.

Our method for predicting expert selection and neuron sparsity achieves satisfactory accuracy. We tested Mixtral 8×7B and DeepSeekMoE 16B, with expert prediction accuracy shown in Figure 5(a). The accuracy of lower layers is around 75%, and up to 90% for middle and upper layers. Neuron prediction accuracy, shown in Figure 5(b), is 80% to 90% in most layers. This result demonstrates the feasibility of using the model's own weights for prediction. The similarity between adjacent layers' activations in LLMs, due to the residual network structure, is approximately 0.99 (Liu et al., 2023), making prediction accuracy satisfactory.

If the predictor misses, the model proceeds with the incorrect prediction rather than reloading the correct weights. This decision is based on a trade-off between accuracy and inference speed; reloading the correct weights in response to a miss would significantly increase latency. Our high accuracy rate ensuring that predictor misses have minimal impact on overall model accuracy.

By predicting sparsity in advance, we can prefetch the needed weights, hiding part of the computation latency and reducing inference latency. Throughout the prefetching process, model inference continues as normal, using CUDA streams to parallelize weight loading and computation.

4.3 Coarse-to-fine-grained Caching

After determining the experts and neurons required for inference, efficiently utilizing GPU memory to reduce expert and neuron loading needs further consideration. Expert caching can effectively reduce the latency caused by offloading in SMoE LLM inference (Eliseev and Mazur, 2023; Xue et al., 2024). Frequently reused experts in SMoE LLMs can be cached in GPU HBM, avoiding costly retrievals from CPU DRAM. Previous work showed satisfactory hit rates using a simple LRU strategy (Eliseev

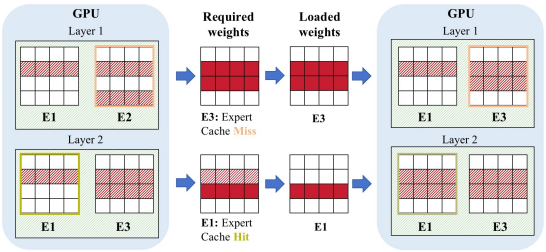


Figure 6: An illustrative example shows how CORM maintains coarse-to-fine-grained cache. We maintain a neuron cache in GPU memory for each cached expert. For cached experts, we only load active neurons that are not in the cache.

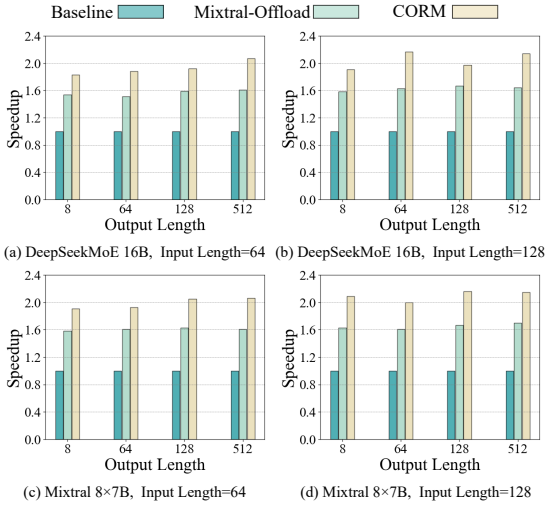


Figure 7: End-to-end speedup of Mixtral 8×7B and DeepSeekMoE 16B. The X axis indicates the output length. The Y axis represents the speedup. (a,c) is configured with an input length of around 64, (b,d) is configured with an input length of around 128.

and Mazur, 2023). In CORM, we allocate a fixed storage space in GPU memory before inference to cache a fixed number of experts. During inference, the LRU strategy maintains the cached experts.

However, since we introduce "sparse neuron loading", this doesn't guarantee correct inference even if the expert is cached, because different activation inputs lead to different neuron sparsity. Therefore, we design a neuron cache mechanism. As shown in Figure 6, we maintain a neuron cache on GPU for each cached expert. For cached experts, we only load active neurons that aren't in the neuron cache, updating the neuron cache with the union of cached and currently active neurons. For the experts that have just been cached, we set the expert's neuron cache to currently loaded neurons.

5 Evaluation

Implementation. CORM is implemented using NVIDIA's FasterTransformer (NVIDIA, 2019). We use HQQ (Badri and Shaji, 2023) data-free quantization algorithm to quantize SMoE LLMs. In addition to implementing the coarse-to-fine grained predicting, prefetching, and caching, we also adjusted quantization algorithm to suit neuron operations. For LLMs, one bit usually stores more than a value in quantized weights. We modified compression dimensions of quantization algorithm to ensure each neuron is compressed into the same bit, maintaining independence of each neuron's storage to avoid additional overhead when extracting neurons.

Hardware. We test most of the experiments on a server with 64 Intel(R) Xeon(R) Gold 6226R CPUs @ 2.90GHz and 8 NVIDIA RTX 3090Ti GPU with 24GB of HBM. The CPU and GPU communicate over a PCIe (gen4) channel with 32 GB/sec of data transfer bandwidth. In all our experiments, testing was conducted using only one GPU.

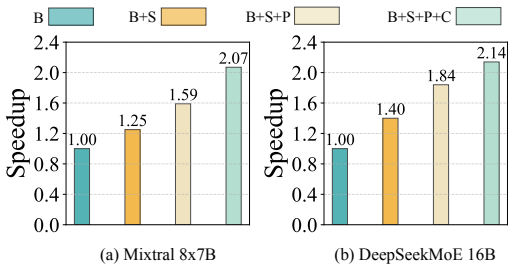
Model and datasets. We use Mixtral 8×7B model (Jiang et al., 2024) and DeepSeekMoE 16B (Dai et al., 2024) model. Mixtral 8×7B represent the current state of the art among open-access SMoE LLMs. We select three datasets from Winogrande (Sakaguchi et al., 2021), MMLU (Hendrycks et al., 2021) and PIQA (Bisk et al., 2020). Winogrande and PIQA test commonsense reasoning. MMLU is a massive multitask from various branches of knowledge.

5.1 End-to-End Result

Our end-to-end experiment focuses on edge device inference scenarios, where the batch size is set to 1. In our experiments, the input lengths are 64 and 128, while the number of output tokens varies from 8 to 512. We conducted prompt sampling on the C4 (Raffel et al., 2020) and Wikipedia (Foundation, 2023) datasets, which are high-quality text datasets covering a wide range of domains. Our baseline implementation uses expert offloading. Specifically, we quantize the model weights to INT4 and store intermediate activations in FP16. We store the non-MoE weights and activations of the SMoE LLM on the GPU, while the MoE layer weights are stored in CPU DRAM. The expert weights are loaded to GPU on-demand during inference. This is a common SMoE LLM offloading strategy used in previous work. Mixtral-Offload (Eliseev and Mazur, 2023) is a state-of-the-art method for accelerating SMoE LLM inference on edge devices, utilizing

	PIQA	Winogrande	MMLU
Mixtral 8×7B	82.9%	77.2%	71.2%
Mixtral 8×7B-CORM	82.7%	76.7%	70.3%
Difference	-0.2%	-0.5%	-0.9%
DeepSeekMoE 16B	80.2%	70.2%	45.0%
DeepSeekMoE 16B-CORM	79.3%	70.4%	43.7%
Difference	-0.9%	+0.2%	-1.3%

Table 2: Comparison of SMoE LLM accuracy between CORM optimized models and their original counterparts. Due to the potential to select incorrect experts and overlook actually active neurons, our approach inevitably introduces some degree of compromise to model accuracy. Experimental results demonstrate the extent of accuracy decline is within acceptable bounds.



(a) Mixtral 8x7B (b) DeepSeekMoE 16B

Figure 8: Speedup breakdown for each module of CORM. "B" denotes baseline implementation. "S" denotes coarse-to-fine-grained offloading and expert granularity mixed quantization module. "P" denotes coarse-to-fine-grained predicting and prefetching module. "C" denotes coarse-to-fine-grained caching module.

speculative expert loading and expert caching.

Figure 7 shows the comparison of end-to-end generation speedup of CORM under different input and output lengths. For Mixtral 8×7B, the average speedups are 1.92× and 1.99× compared to the baseline, and 1.23× and 1.24× compared to Mixtral-Offload for input lengths of 64 and 128, respectively. For DeepSeekMoE 16B, the average speedups are 2.04× and 2.09× compared to the baseline, and 1.26× and 1.27× compared to Mixtral-Offload for the same input lengths. As the generation length increases, the speedup of CORM improves. This enhancement occurs because a greater number of generated tokens reduces the weight loading latency through increased cache hits from coarse-to-fine-grained cache system.

5.2 Model Accuracy

Given that our method predicts the usage of experts and the sparsity of neurons, and some neurons are ignored during inference, it is crucial to explore the impact of our method on the accuracy of MoE LLMs. Table 2 compares the accuracy with and without the use of CORM. Experiments demonstrate that CORM maintains competitive accuracy across different models and various downstream tasks. The accuracy of MoE LLMs may fluctuate due to possible misselections of experts and the omission of some actually active neurons. However, sometimes this even enhance accuracy.

5.3 Ablation Studies

Speedup breakdown. In this section, we examine the contribution of each module of our method to overall acceleration. We incrementally tested the throughput of four schemes. Our baseline implementation mirrors that of the end-to-end speedup experiment. Next, we implemented and tested coarse-to-fine-grained offloading by sparse neuron loading and expert granularity mixed quantization. Then, we add coarse-to-fine-grained predicting and prefetching module. Finally, we add the coarse-to-fine-grained caching module as our full algorithm.

Figure 8 illustrates the generation speedup breakdown for each module of CORM. Adding the coarse-to-fine-grained offloading and expert granularity mixed quantization module results in speedup of 1.25× for Mixtral 8×7B and 1.4× for DeepSeek-MoE 16B, primarily by reducing weight loading through sparse neuron loading. Incorporating the coarse-to-fine-grained predicting and prefetching module achieves speedup of 1.59× and 1.84× by avoiding the serialized execution of computation and weight loading. Finally, adding coarse-to-fine-grained caching module results in speedup of 2.07× and 2.14×, minimizes the need for repeated weight loading by efficiently utilizing GPU memory.

Latency breakdown and predictor overhead. Weight loading time constitutes a significant portion of inference latency in offloading methods, making it an essential focus for optimization. To demonstrate the impact of CORM's optimizations, we measured and compared the weight loading costs and computation costs to the time between tokens (TBT). The latency breakdown results are shown in table 3. Weight loading times are indeed substantial relative to computation times, and CORM's optimizations help mitigate this bottleneck by implementing coarse-to-fine-grained control over weight loading process. We also present overlap latency in table 3, which can account for over 80% of the computation latency, highlighting effectiveness of our prediction and prefetching operations in masking computation latency.

The overhead introduced by our dual-level predictor is a critical factor to measure. Since the

	TBT	Weight Loading	Computation	Overlap
Mixtral 8×7B	2450ms	2100ms	350ms	N/A
Mixtral 8×7B-CORM	1280ms	1210ms	398ms	328ms
DeepSeekMoE 16B	836ms	706ms	130ms	N/A
DeepSeekMoE 16B-CORM	404ms	370ms	144ms	110ms

Table 3: Latency breakdown of the CORM optimized models and their original counterparts for generating one token with a prompt length of 128. "TBT" denotes the time between tokens, "Overlap" represents the overlapping time between weight loading and computation.

	TBT	Predictor	Predictor Latency Ratio
Mixtral 8×7B	1280ms	48ms	3.75%
DeepSeekMoE 16B	405ms	14ms	3.46%

Table 4: Predictor overhead of CORM for generating one token with a prompt length of 128. "TBT" represents time between tokens.

predictor requires no additional memory, we measured the latency overhead it introduces. In table 4, we provide a breakdown of the predictor latency compared to the time between tokens(TBT). As shown, the predictor latency represents a relatively small portion of the total time per token, ensuring minimal impact on overall inference performance.

Effect of neuron sparsity on throughput and accuracy. In our method, the sparse neuron loading module significantly reduces weight loading. We determine whether a neuron is active by comparing its absolute value of the activation value with a threshold θ, thus deciding whether it needs to be loaded. Therefore, the threshold is a critical factor affecting accuracy. We test impact of threshold settings on accuracy and throughput below.

Figure 9 shows the relationship between the latency of sparse neuron loading and the percentage of active neurons for an expert of Mixtral 8×7B. Sparse neuron loading consists of three parts: compressing active neurons, loading, and decompressing. In our experiments, the decompression operation takes an average of 0.1ms, accounting for about 1% of the total latency. Therefore we omitted this part from the figure. It can be observed that the latency of neuron compression, weight loading and the total latency are approximately proportional to the percentage of active neurons. If too many neurons are active, the latency of sparse neuron loading will exceed that of directly copying the full expert (as shown by the red dashed line in Figure 9). Therefore, we set an upper limit on the percentage of active neurons, beyond which the full expert is loaded. For Mixtral 8×7B, this upper limit is 75%.

As fewer active neurons result in lower inference latency, we control the number of active neurons by discarding neurons whose activation function outputs do not exceed the threshold. However, discarding too many neurons can lead to a decrease in model accuracy. Figure 10 illustrates the relationship between the threshold θ and both the throughput and accuracy of Mixtral 8×7B. As θ increases, the throughput gradually iecreases.The model's accuracy remains above 99% when θ is below 0.25. However, at θ of 0.275, we observe a sudden drop in accuracy. Therefore, we select θ of 0.25 as a balance between accuracy and throughput.

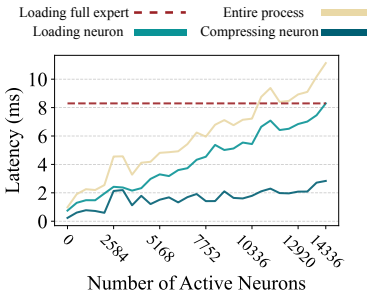


Figure 9: Sparse neuron loading latency breakdown of Mixtral 8×7B at varying number of active neurons. The red dashed line denote the latency of loading a full expert. It's more efficient to load the full expert weight if the sparse neuron loading latency exceeds the loading latency of the full expert weight.

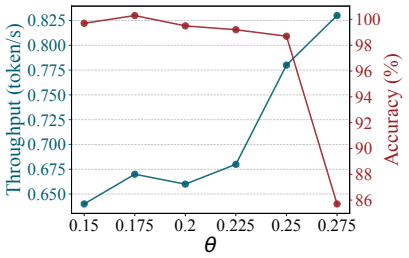


Figure 10: Effect of threshold θ setting on throughput and accuracy of Mixtral 8×7B. A larger value of θ leads to higher throughput but lower accuracy. We must select θ judiciously to achieve an optimal balance between throughput and accuracy. For Mixtral 8×7B, the optimal value of θ is 0.25.

6 Conclusion

This paper introduces CORM, an efficiently SMoE LLM inference framework on consumer-grade GPUs. By leveraging dual-levels of sparsity inherited from SMoE LLMs, it achieves efficient weight loading and cache management. CORM achieves up to 2.14× inference speedup while maintaining comparable model accuracy across various natural language processing tasks.

7 Ethical Statements

We used publicly available or synthetic datasets for our experiments to avoid any potential harm to individuals or groups. The data used in this study were carefully selected and processed to ensure privacy and confidentiality. No personally identifiable information was used, and all data were anonymized prior to analysis. All artifacts used in this study were consistent with their intended use. The artifact we created is designed to accelerate the inference process of SMoE LLMs on consumer-grade GPUs, and it is compatible with the original access conditions.

In considering the application of our research findings, we acknowledge the potential risks related to data privacy. In the event of vulnerabilities in LLMs deployed on edge devices, attackers could exploit these vulnerabilities to steal data or gain control of the devices.

For preprocessing, we employed the HQQ (Badri and Shaji, 2023) data-free quantization algorithm to quantize SMoE LLMs. We quantized the models to 4-bit with a group size of 64, scale group size 256, and 2-bit with a group size of 16, scale group size 128.

The models and datasets used in this study are open-source. These models include Mixtral 8×7B (Jiang et al., 2024) and deepseekMoE-16B (Dai et al., 2024), with licenses under Apache-2.0 and MIT, respectively. We tested the models on all problems in the model test datasets, which include Winogrande (Sakaguchi et al., 2021), PIQA (Bisk et al., 2020), and MMLU (Hendrycks et al., 2021). Winogrande and PIQA test commonsense reasoning, while MMLU is a large-scale multitask dataset encompassing various branches of knowledge. Winogrande contains a new collection of 44,000 problems, PIQA has 16,000 training examples, 2,000 for development, and 3,000 for testing. MMLU consists of 57 tasks with 23,000 problems. The usage license for Winogrande is CC-BY, and the usage license for MMLU is MIT. The usage license for PIQA was not explicitly found, but it has been widely used in numerous studies as a test dataset. The prompt datasets include C4 (Raffel et al., 2020) and Wikipedia (Foundation, 2023). C4 is a colossal, cleaned version of Common Crawl's web corpus based on the Common Crawl dataset, while the Wikipedia dataset contains cleaned articles in all languages, built from Wikipedia dumps. We randomly selected 5,000 prompts each from C4 and Wikipedia to profile the models, and selected 1,000 prompts each to test the inference speed and perform ablation experiments. The usage license for C4 is ODC-BY, and the usage license for Wikipedia is CC-BY-SA-3.0.

We did not use human annotators or research involving human subjects, nor did we employ AI assistants in our research, coding, or writing.

8 Limitations

The proposed approach in this work has several limitations:

- Our method applies offline mixed-precision quantization to the experts based on prior knowledge. Developing a mechanism for dynamic mixed-precision quantization during inference, tailored to the usage characteristics of the experts, could further minimize the performance impact on SMoE LLMs.

- In this study, the computational tasks of the SMoE LLM are executed exclusively on GPUs, without leveraging CPU resources. By employing efficient planning and scheduling, it may be possible to accelerate inference through parallel computation using both CPU and GPU resources.

- Our caching mechanism pre-allocates memory on GPUs at the expert granularity. A more efficient approach would be to dynamically maintain a cache at the neuron granularity based on the number of active neurons during inference, thereby reducing memory consumption.

References

Abien Fred Agarap. 2018. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

Keivan Alizadeh, Iman Mirzadeh, Dmitry Belenko, Karen Khatamifard, Minsik Cho, Carlo C Del Mundo, Mohammad Rastegari, and Mehrdad Farajtabar. 2023. Llm in a flash: Efficient large language model inference with limited memory. *arXiv preprint arXiv:2312.11514*.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. 2023. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*.

Hicham Badri and Appu Shaji. 2023. Half-quadratic quantization of large machine learning models.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, et al. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR.

Zhixu Du, Shiyu Li, Yuhao Wu, Xiangyu Jiang, Jingwei Sun, Qilin Zheng, Yongkai Wu, Ang Li, Hai Li, and Yiran Chen. 2024. Sida: Sparsity-inspired data-aware serving for efficient and scalable large mixture-of-experts models. *Proceedings of Machine Learning and Systems*, 6:224–238.

Artyom Eliseev and Denis Mazur. 2023. Fast inference of mixture-of-experts language models with offloading. *arXiv preprint arXiv:2312.17238*.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Wikimedia Foundation. 2023. Wikimedia downloads.

Elias Frantar and Dan Alistarh. 2024. Qmoe: Sub-1-bit compression of trillion parameter models. *Proceedings of Machine Learning and Systems*, 6:439–451.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Ke Hong, Guohao Dai, Jiaming Xu, Qiuli Mao, Xiuhong Li, Jun Liu, Kangdi Chen, Hanyu Dong, and Yu Wang. 2023. Flashdecoding++: Faster large language model inference on gpus. *arXiv preprint arXiv:2311.01282*.

Ranggi Hwang, Jianyu Wei, Shijie Cao, Changho Hwang, Xiaohu Tang, Ting Cao, Mao Yang, and Minsoo Rhu. 2023. Pre-gated moe: An algorithm-system co-design for fast and scalable mixture-of-expert inference. *arXiv preprint arXiv:2308.12066*.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Yunho Jin, Chun-Feng Wu, David Brooks, and Gu-Yeon Wei. 2023. s^3: Increasing gpu utilization during generative inference for higher throughput. *Advances in Neural Information Processing Systems*, 36:18015–18027.

Keisuke Kamahori, Yile Gu, Kan Zhu, and Baris Kasikci. 2024. Fiddler: Cpu-gpu orchestration for fast inference of mixture-of-experts models. *arXiv preprint arXiv:2402.07033*.

Rui Kong, Yuanchun Li, Qingtian Feng, Weijun Wang, Linghe Kong, and Yunxin Liu. 2023. Serving moe models on resource-constrained edge devices via dynamic expert swapping. *arXiv preprint arXiv:2308.15030*.

Dacheng Li, Rulin Shao, Anze Xie, Eric P Xing, Joseph E Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. 2023a. Lightseq: Sequence level parallelism for distributed training of long context transformers. *arXiv preprint arXiv:2310.03294*.

Jiamin Li, Yimin Jiang, Yibo Zhu, Cong Wang, and Hong Xu. 2023b. Accelerating distributed {MoE} training and inference with lina. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*, pages 945–959.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. 2023. Deja

vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pages 22137–22176. PMLR.

Hanjia Lyu, Song Jiang, Hanqing Zeng, Yinglong Xia, and Jiebo Luo. 2023. Llm-rec: Personalized recommendation via prompting large language models. *arXiv preprint arXiv:2307.15780*.

NVIDIA. 2019. Fastertransformer.

OpenAI. 2022. Introducing chatgpt.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Liang Shen, Zhihua Wu, WeiBao Gong, Hongxiang Hao, Yangfan Bai, HuaChao Wu, Xinxuan Wu, Jiang Bian, Haoyi Xiong, Dianhai Yu, et al. 2022. Semoe: A scalable and efficient mixture-of-experts distributed training and inference system. *arXiv preprint arXiv:2205.10034*.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pages 31094–31116. PMLR.

Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. 2023. Powerinfer: Fast large language model serving with a consumer-grade gpu. *arXiv preprint arXiv:2312.12456*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

xAI. 2024. Open release of grok-1.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR.

Daliang Xu, Wangsong Yin, Xin Jin, Ying Zhang, Shiyun Wei, Mengwei Xu, and Xuanzhe Liu. 2023. Llmcad: Fast and scalable on-device large language model inference. *arXiv preprint arXiv:2309.04255*.

Leyang Xue, Yao Fu, Zhan Lu, Luo Mai, and Mahesh Marina. 2024. Moe-infinity: Activation-aware expert offloading for efficient moe serving. *arXiv preprint arXiv:2401.14361*.

Biwei Yan, Kun Li, Minghui Xu, Yueyan Dong, Yue Zhang, Zhaochun Ren, and Xiuzheng Cheng. 2024. On protecting the data privacy of large language models (llms): A survey. *arXiv preprint arXiv:2403.05156*.

Rongjie Yi, Liwei Guo, Shiyun Wei, Ao Zhou, Shangguang Wang, and Mengwei Xu. 2023. Edgemoe: Fast on-device inference of moe-based large language models. *arXiv preprint arXiv:2308.14352*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. 2024. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. 2022. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114.