# Kernel Fusion in Attention Layer of Quantized LLMs

**CSE412 Parallel Computing: Project Report**

**Jaemin Kim (20211061)** **Jiseung Jeong (20211301)**
Department of Computer Science & Engineering, UNIST
{jm611, wjdwltmd1151}@unist.ac.kr

## 1 Introduction

Transformers have become the dominant architecture for large language models (LLMs), but their self-attention layers are both compute- and memory-intensive. A standard attention implementation materializes an $N \times N$ score matrix for sequence length $N$, incurring $O(N^2)$ memory traffic that often makes inference memory-bandwidth bound rather than compute-bound. Recent work such as FlashAttention reformulates attention as an IO-aware tiled computation, fusing softmax and value aggregation into a single GPU kernel and avoiding the explicit $N \times N$ matrix in high-bandwidth memory (HBM).

In parallel, weight quantization has emerged as a practical approach to reduce LLM memory footprint and bandwidth requirements. By storing weights in 8-bit or 4-bit formats and dequantizing them on-the-fly, one can substantially reduce VRAM usage and improve effective throughput. However, the dequantization step itself incurs additional memory accesses and kernel launches. In naïve implementations, each attention layer executes separate kernels to dequantize $W_Q, W_K, W_V$, followed by separate GEMM kernels to form $Q, K, V$, and finally an attention kernel. These extra kernels reintroduce memory bottlenecks and underutilize modern GPUs.

This work targets the attention layer of a quantized LLM and asks: *how should we fuse dequantization and attention computation to maximize performance on GPUs, particularly for **block-wise** 4-bit formats such as NF4, MXFP4, and NVFP4?* We study kernel fusion strategies that integrate block-wise dequantization directly into the QKV projection, while treating a FlashAttention-style attention kernel as an off-the-shelf backend. Our implementation on GPT-2 attention, evaluated on an NVIDIA RTX 3090, shows that a carefully designed two-stage pipeline—with a fused dequantization + QKV projection kernel followed by an unmodified FlashAttention kernel—significantly reduces end-to-end latency, while a fully fused single-kernel design can be slower due to register pressure and control-flow divergence. Our implementation is available at https://github.com/ginjae/fused-qattn.

## 2 Related Work

**Efficient attention.** FlashAttention (Dao et al., 2022) introduced an IO-aware attention algorithm that tiles $Q, K, V$ into shared memory and computes softmax and value aggregation incrementally in a single kernel. By never materializing the full $N \times N$ attention matrix, it drastically reduces memory traffic and achieves substantial speedups for long sequences. Follow-up implementations and variants have adopted similar tiling and fusion principles in production systems.

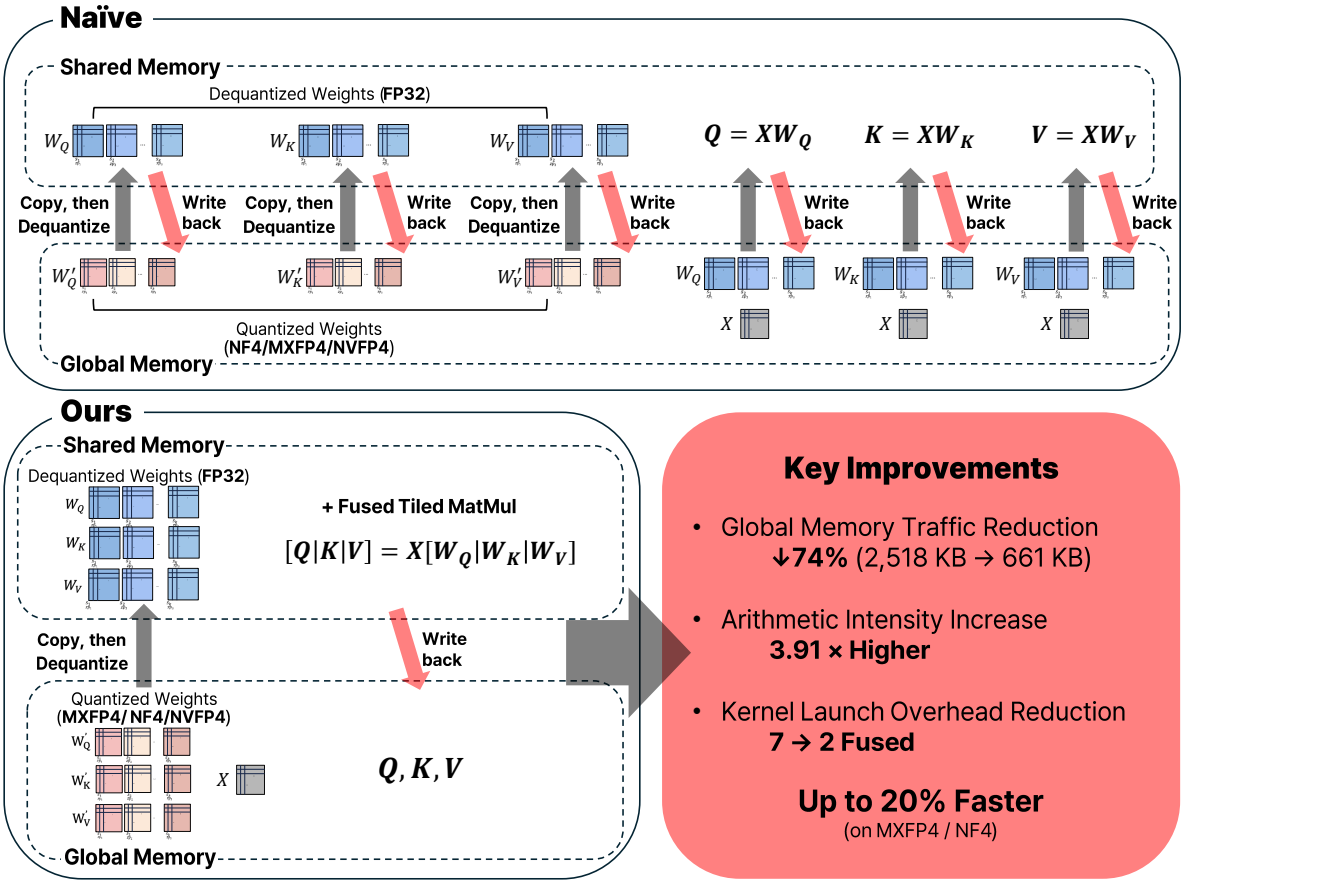
**Quantization of LLMs.** Post-training quantization of LLM weights to 8-bit and 4-bit formats has been widely explored to reduce memory and improve inference throughput. QLoRA (Dettmers et al., 2023) popularized 4-bit quantization in the form of NF4 (NormalFloat4), which uses a learned non-linear codebook and block-wise statistics to preserve model quality with small quantization error. Other 4-bit formats, such as MXFP4 (Darvish Rouhani et al., 2023), which uses a block-wise mini floating-point representation with a shared exponent, and NVIDIA's NVFP4 (NVIDIA et al., 2025), further exploit block-level statistics or hardware-friendly layouts.

**Fused quantized attention.** Recent work has begun to combine quantization with FlashAttention-style kernels. INT-FlashAttention (Chen et al., 2024) integrates INT8 quantized weights and scaling into a fused attention kernel. The PyTorch INT4 GQA kernel (PyTorch Team, 2024) fuses INT4 dequantization with grouped-query attention and KV-cache quantization using custom CUDA kernels. However, these works generally assume element-wise or row-wise quantization and do not systematically address block-wise 4-bit formats such as NF4, MXFP4, and NVFP4, which require more complex dequantization logic (look-up tables, shared exponents, or multi-stage conversions). To our knowledge, our study is among the first to analyze fusion strategies specifically for block-wise 4-bit quantized attention.

## 3 Methodology

We consider a single Transformer attention layer with input $X \in \mathbb{R}^{N \times d_{\text{model}}}$ and weight matrices $W_Q, W_K, W_V \in \mathbb{R}^{d_{\text{model}} \times d_h}$, where $N$ is the sequence length and $d_h$ the head dimension. The weights are stored in low-precision formats (INT8 or FP4-style) with associated block-wise metadata (scales, exponents, or codebook indices). The input activations remain in FP16/FP32.

Figure 1: Conceptual comparison of a naïve pipeline and the proposed fused approach which replaces the three dequanitization + projection kernels with a single fused dequantization + QKV projection kernel.




### 3.1 Baseline: unfused quantized attention

A baseline implementation performs:

1. Three dequantization kernels to convert quantized $W_Q, W_K, W_V$ to FP32 weights.
2. Three GEMM kernels to compute $Q = XW_Q$, $K = XW_K$, and $V = XW_V$.
3. A FlashAttention kernel that takes $Q, K, V$ and computes the attention output.

This yields seven kernels in total. Quantized weights are read from global memory, dequantized, written as FP32, and then read again for GEMM. Q, K, and V are also written to global memory before the attention kernel. The dominant cost is memory traffic and kernel launch overhead rather than arithmetic operations.

### 3.2 Proposed fusion strategy

Our goal is to reduce both memory traffic and kernel count by fusing dequantization with projection and by reusing activations across Q, K, and V:

- **Fused dequantization + GEMM.** For each weight block, we load quantized weights and block metadata into registers or shared memory, perform block-wise dequantization (e.g., NF4 table lookup or MXFP4 exponent scaling), and immediately use the resulting FP16/FP32 values in the dot products with tiles of $X$. No intermediate dequantized weight matrices are written back to global memory.
- **Joint Q/K/V projection.** Instead of separate kernels for Q, K, and V, we launch a single kernel that computes all three projections. A tile of $X$ is loaded once into shared memory, and three sets of quantized weights are streamed and dequantized on the fly. The kernel outputs tiles of Q, K, and V directly to global memory.

We implement this fused dequant + QKV kernel using a block-tiled design similar in spirit to GEMM and FlashAttention: each thread block processes a tile of $X$ and the corresponding tiles of quantized weights (for all three projections), accumulating partial results in registers and shared memory.

### 3.3 Explored Fusion Configurations

To systematically analyze the trade-offs between memory traffic reduction and hardware resource constraints (e.g., register pressure), we explored four distinct implementation strategies ranging from no fusion to complete fusion. These correspond directly to the configurations evaluated in our experiments:

- **Naïve (Baseline):** The standard unfused approach

using separate kernels for dequantization, GEMM, and attention (7 kernels total).

- **Projection-Fused:** Fuses dequantization into the GEMM kernel for each of $Q, K, V$ separately. This executes three fused projection kernels followed by attention (4 kernels total).
- **Semi-Fused:** Separates the pipeline into a consolidated dequantization kernel, a consolidated QKV projection kernel, and the attention kernel (3 kernels total). This isolates complex dequantization logic from the projection kernel.
- **Ours:** Our proposed design that fuses block-wise dequantization and QKV projection into a single kernel, followed by a standard FlashAttention kernel (2 kernels total). In this setup, the attention kernel itself is unchanged; all fusion happens in the QKV projection stage to maximize arithmetic intensity without over-consuming registers.

We initially hypothesized that the Single-Kernel Fully-Fused design would be optimal. However, as shown in our results, this design suffered from excessive register usage and large shared-memory requirements, leading to low occupancy and spilling. Consequently, the our design provided the best balance on the RTX 3090.

**Support for multiple quantization formats**

The fused kernel supports:

- **MXFP4** (block-wise shared-exponent FP4 format),
- **NF4** (block-wise normal-distribution-based 4-bit codebook),
- **NVFP4** (two-stage block-wise FP4 format).

For MXFP4 and NF4, the dequantization logic is implemented using small look-up tables or bit manipulations in constant/shared memory. For NVFP4, more complex logic is required, which becomes a key factor in the performance analysis.

As illustrated in Figure 3, the proposed two-kernel pipeline removes redundant global-memory traffic for dequantized weights while preserving a simple, high-performance attention kernel.

# 4 Experiments

## 4.1 Experimental Setup

We evaluate the proposed kernels on a single NVIDIA GeForce RTX 3090 GPU (Ampere architecture, 24 GB VRAM). As a workload, we use a single attention layer from GPT-2 with hidden dimension $d_{\text{model}} = 768$, head dimension $d_h = 64$, and 12 attention heads. The weight matrices $W_Q, W_K, W_V$ from a pretrained GPT-2 model are quantized offline to three 4-bit formats: MXFP4, NF4, and NVFP4.

Input activations $X$ are sampled as random FP16 values in $[-1, 1]$ with sequence length $N = 128$. We measure the end-to-end forward latency of the attention layer, including all kernels involved in QKV projection and attention, averaged over multiple runs. We also compute arithmetic intensity (FLOPs per byte) from operation counts and memory-access statistics.

Table 1: End-to-end attention latency (ms) and the number of CUDA kernels for different fusion strategies in the QKV projection stage and block-wise 4-bit quantization formats on RTX 3090. FlashAttention is used as a fixed attention kernel in all configurations. Lower latency and fewer kernels are better.

| Implementation | # Kernels | MXFP4 (ms) | NF4 (ms) | NVFP4 (ms) |
|---|---|---|---|---|
| Naïve | 7 | 0.1464 | 0.1444 | 0.1454 |
| Projection-Fused | 5 | 0.1382 | 0.1300 | 0.1434 |
| Semi-Fused | 3 | 0.1311 | 0.1280 | **0.1321** |
| Ours | 2 | **0.1167** | **0.1203** | 0.1464 |

We evaluate the four fusion configurations described in Section 3.3: Naïve, Projection-Fused, Semi-Fused, and Ours.

## 4.2 Results

Table 1 summarizes the measured end-to-end latency of each implementation for MXFP4, NF4, and NVFP4 quantization.

As shown in Table 1, the proposed two-kernel fused attention achieves the best latency for MXFP4 (0.1167 ms) and NF4 (0.1203 ms), improving over the naïve implementation by roughly 20–25%. For NVFP4, the Semi-Fused configuration performs best (0.1321 ms), while our two-kernel design is slightly slower than the naïve baseline due to heavier dequantization overhead.

As shown in Table 2, arithmetic-intensity analysis reveals a substantial shift in the compute-to-memory ratio as fusion is applied. While the naïve baseline achieves only 16.48 FLOPs per kilobyte, the proposed two-kernel design reaches an upper bound of 64.36 FLOPs per kilobyte—approximately a 3.9× increase. This improvement arises because fusion eliminates repeated global-memory reads of dequantized weight blocks and enables a single shared activation tile to be reused across all three projections (Q, K, and V). Structurally, this reduces memory traffic in the inner GEMM loop without changing its arithmetic workload, thereby raising the FLOP/kilobyte ratio by construction. As a result, a much larger fraction of execution time is spent on arithmetic rather than memory stalls.

## 4.3 Analysis

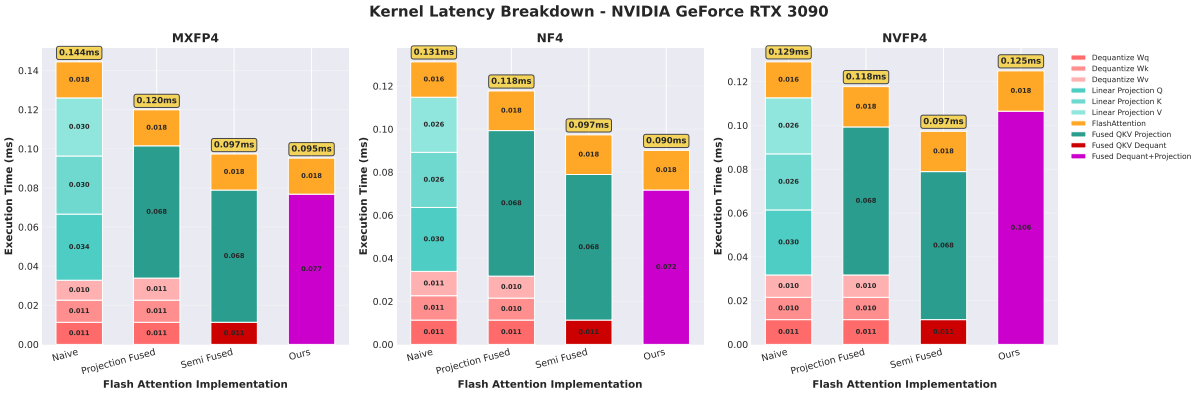
The results in Table 1 highlight two main observations:

**Fusion helps most when dequantization is simple and memory-bound.** For NF4 and MXFP4, dequantization consists of lightweight operations (e.g., look-up tables or shared exponents) that integrate naturally into the GEMM pipeline. This behavior is illustrated in Figure 2, where fusing dequantization with QKV projection eliminates redundant memory traffic and kernel launches, effectively hiding dequantization cost within

Table 2: Theoretical arithmetic intensity (AI) upper bound for NVFP4 quantization under different fusion strategies. AI is computed as FLOPs per kilobyte of memory access. Lower memory traffic in the fused design produces substantially higher AI.

| Baseline | Kernels | FLOPs (M) | Mem (KB) | AI (FLOPs/KB) | vs Naïve |
|---|---|---|---|---|---|
| Naïve | 7 | 43.52 | 2581 | 16.48 | 1.00x |
| Projection-Fused | 5 | 43.52 | 1813 | 23.46 | 1.42x |
| Semi-Fused | 3 | 43.52 | 1813 | 23.46 | 1.42x |
| Ours | 2 | 43.52 | 661 | 64.36 | 3.91x |

Figure 2: Kernel-level latency breakdown for different quantization formats and fusion strategies. The proposed method (Ours) removes independent dequantization and projection steps, significantly reducing total runtime for MXFP4 and NF4.




the main GEMM computation and shifting execution toward a more compute-bound regime.

**Impact of Dequantization Complexity (NVFP4 Case).** For NVFP4, more fusion is not always better. Unlike NF4 or MXFP4, NVFP4 requires a two-step dequantization process (FP4 $\rightarrow$ FP8 $\rightarrow$ FP16/FP32), which significantly increases register pressure when fused into the QKV projection kernel. On RTX 3090 (Ampere), which lacks native FP4 support, this limits occupancy and offsets the benefit of reduced memory traffic. As a result, the Semi-Fused configuration performs best by isolating the complex dequantization logic into a separate kernel.

**Limits of Aggressive Fusion.** To explore the theoretical limits of fusion, we implemented a single-kernel design fusing dequantization, projection, and attention into a monolithic kernel. However, this approach resulted in severe slowdowns (e.g., 7.76 ms for MXFP4 and 16.85 ms for NVFP4), rendering it over 60$\times$ slower than our proposed method. This degradation primarily stems from excessive **register pressure**: combining complex dequantization logic with the FlashAttention loop forces register spilling to local memory, drastically reducing occupancy on the RTX 3090. Furthermore, the conditional branching required for decoding introduces **warp divergence**. These findings confirm that our two-stage pipeline strikes a superior balance between arithmetic intensity and hardware constraints.

## 5 Conclusion

In this work, we investigated kernel fusion strategies for the attention layer in quantized LLMs, focusing on block-wise 4-bit formats (NF4, MXFP4, NVFP4). We proposed a two-kernel design that fuses block-wise dequantization with the QKV projection in a tiled GEMM-style kernel and then applies a FlashAttention-style kernel for the attention computation. On GPT-2 attention running on an RTX 3090, this design significantly reduces memory traffic, increases arithmetic intensity, and yields up to about 20–25% lower latency for NF4 and MXFP4 compared to a naïve unfused baseline, as summarized in Table 1.

**Future work.** There are several directions to extend this work. First, evaluating the fused kernels on additional GPUs (e.g., A100, H100) and in full end-to-end LLM inference pipelines would provide a more comprehensive understanding of their practical impact. Second, extending fusion beyond attention to include output projections, MLP blocks, and normalization layers could further improve utilization. Third, integrating fused block-quantized attention into mainstream frameworks would make these optimizations accessible to a wider audience. Finally, co-designing quantization schemes and kernels with upcoming hardware features (such as native FP4/FP8 tensor-core support and larger on-chip memory) is a promising path toward even more efficient low-precision LLM inference.

## References


Shimao Chen, Zirui Liu, Zhiying Wu, Ce Zheng, Peizhuang Cong, Zihan Jiang, Yuhan Wu, Lei Su, and Tong Yang. 2024. Int-flashattention: Enabling flash attention for int8 quantization. *Preprint*, arXiv:2409.16997.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Preprint*, arXiv:2205.14135.

Bita Darvish Rouhani, Ritchie Zhao, Venmugil Elango, Rasoul Shafipour, Mathew Hall, Maral Mesmakhosroshahi, Ankit More, Levi Melnick, Maximilian Golub, Girish Varatkar, Lai Shao, Gaurav Kolhe, Dimitry Melts, Jasmine Klar, Renee L'Heureux, Matt Perry, Doug Burger, Eric Chung, Zhaoxia (Summer) Deng, and 3 others. 2023. With shared microexponents, a little shifting goes a long way. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ISCA '23, New York, NY, USA. Association for Computing Machinery.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: efficient finetuning of quantized llms. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

NVIDIA, Felix Abecassis, Anjulie Agrusa, Dong Ahn, Jonah Alben, Stefania Alborghetti, Michael Andersch, Sivakumar Arayandi, Alexis Bjorlin, Aaron Blakeman, Evan Briones, Ian Buck, Bryan Catanzaro, Jinhang Choi, Mike Chrzanowski, Eric Chung, Victor Cui, Steve Dai, Bita Darvish Rouhani, and 70 others. 2025. Pretraining large language models with nvfp4. *Preprint*, arXiv:2509.25149.

PyTorch Team. 2024. Accelerating llm inference with int4 quantization and grouped query attention. `https://pytorch.org/blog/int4-decoding/`.