FlashSVD: Memory-Efficient Inference with Streaming for Low-Rank Models

Zishan Shao[1,2*], Yixiao Wang[2], Qinsi Wang[1], Ting Jiang[1], Zhixu Du[1], Hancheng Ye[1], Danyang Zhuo[3], Yiran Chen[1], Hai "Helen" Li[1]

[1]Department of Electrical & Computer Engineering, Duke University
[2]Department of Statistical Science, Duke University
[3]Department of Computer Science, Duke University
zishan.shao@duke.edu, yixiao.wang@duke.edu, qinsi.wang@duke.edu, justin.jiang@duke.edu, zhixu.du@duke.edu, hancheng.ye@duke.edu, danyang@cs.duke.edu, yiran.chen@duke.edu, hai.li@duke.edu

Abstract

Singular Value Decomposition (SVD) has recently gained traction as an effective compression technique for large language models (LLMs), with many studies reporting 20-80% parameter reduction at minimal accuracy cost. However, despite reducing weight memory, existing SVD-based approaches still rely on standard dense CUDA kernels during inference, which incur substantial-and ultimately unnecessary-activation memory overhead. Our analysis reveals that this kernel-induced cost, which grows with sequence length and hidden size, in worst case prevents any real reduction in peak inference memory, limiting the practical impact of SVD compression for on-device deployment.

To address this bottleneck, we propose **FlashSVD**, an end-to-end, rank-aware streaming inference framework for SVD-compressed LLMs. FlashSVD integrates seamlessly with any SVD-based model and directly fuses low-rank projection kernels into self-attention and feed-forward pipelines. This design avoids materializing large activation buffers by streaming small tiles of truncated factors through on-chip SRAM, performing on-the-fly multiplication and reduction, and immediately evicting results–thus preserving high GPU occupancy without introducing latency. On standard benchmarks (e.g., BERT-Base), FlashSVD reduces peak activation memory by up to 70.2% and transient memory by 75%, with zero accuracy loss against low-rank baselines, enabling truly memory-efficient deployment of low-rank LLMs.

Code — https://github.com/Zishan-Shao/FlashSVD
Extended version — https://arxiv.org/abs/2508.01506

Introduction

Recent advances in Transformer architectures have yielded state-of-the-art performance across diverse tasks with the cost of ever-increasing model size and computational complexity. Rapid scaling of model size poses non-negligible challenges on deploying large pre-trained models on edge devices due to stringent *peak memory* constraints: activations and weights together often exceed available high-bandwidth memory (HBM), leading to out-of-memory (OOM) failures even for moderately sized context. While parameter compression techniques such as truncated singular value decomposition (SVD) can reduce model size by over 50% with minimal accuracy loss (Hsu et al. 2022; Yuan and Others 2023; Wang et al. 2025a, 2024b; Chang et al. 2025), they do not address the *activation* memory spikes during inference. In practice, mainstream inference backends (e.g., FlashAttention (Dao et al. 2022, 2023), xFormers (Zhang et al. 2024), Megatron (Shoeybi et al. 2019)) offer no native support for low-rank layers, so peak memory remains dominated by large intermediate buffers if not properly optimized systematically.

Concurrent with advances in model compression, extensive research has targeted activation-aware optimizations for dense transformers. Key-value caching in autoregressive decoders reuses past attention states to eliminate redundant $\mathcal{O}(BM^2)$, where B is the batch size and M is the sequence length, computations and storage overhead (Pope et al. 2023). FlashAttention further accelerates both training and inference by fusing the softmax and matrix-multiply operations into a single GPU kernel, thereby obviating the need to materialize full attention score matrices (Dao et al. 2022, 2023). Complementary methods such as mixed-precision arithmetic, quantization, and structured pruning have also proven effective at reducing compute and memory demands without modifying the core transformer architecture across diverse tasks(Dettmers et al. 2023; Han, Mao, and Dally 2016; Jacob et al. 2018; Ye et al. 2025; Wang et al. 2024a, 2025b; Jacob et al. 2018; Zhou et al. 2025; Li, Chen, and Zeng 2025).

Despite these advances, existing SVD-based compression methods primarily factorize model weights but continue to rely on standard dense matrix kernels during inference–overlooking the opportunity to exploit the underlying low-rank structure for computational efficiency.

In this work we introduce **FlashSVD**, an end-to-end inference framework that enables rank-aware activation for SVD-based low-rank models. Our key contributions are:

- We identify that activation memory-alongside fixed parameter cost-dominates inference overhead, whereas prior work has largely targeted weight compression. We propose rank-aware fine-tuning as a promising direction to drive ranks lower without sacrificing accuracy, yielding further activation-cost reductions.
- We proposed two series of rank-aware streaming low-

*Corresponding Author.

rank kernels that consumes only low-rank activations in a single pass with no memory-expensive dense intermediates while ensure computational efficiency in moderate-large context window.

- By exploiting the multi-head structure when compressing the attention projection matrices, the necessary rank-loss ratio $1 - \frac{r_{\text{retained}}}{r_{\max}}$ is substantially lower than that required by single-head compression methods in prior works, thereby permitting more moderate rank reductions while still achieving comparable overall compression performance.

Related Work

Low-Rank Weight Compression SVD-based factorization is a classical approach for transformer compression, where small singular values are truncated under the Eckart-Young Mirsky theorem to reduce parameter counts at some cost of accuracy (Yuan and Others 2023; Wang et al. 2024b). Activation-aware SVD (ASVD) introduces rank awareness into both weight and KV-cache compression for decoders, factorizing cache states into low-rank components to achieve up to 30 % weight reduction and 50 % KV-cache memory savings with no retraining (Yuan and Others 2023). Building on this idea, SVD-LLM (v1-v2) (Wang et al. 2024b, 2025c) also stores KV-cache in low-rank form but further employs a lightweight calibration set to determine optimal truncation thresholds, thereby preserving downstream performance despite more aggressive compression. MoDeGPT partitions each Transformer block into modular subcomponents and applies Nystrom, CR decomposition, and SVD at the module level - enabling structured, fine-tuning-free compression that saves up to 30% compute and increases inference throughput by 46% on 13B models (Lin and Colleagues 2024). Dobi-SVD makes the truncation process differentiable by learning per-layer cut positions via gradient-based optimization and deriving optimal low-rank updates, thus maintaining performance under 40% parameter compression (Wang et al. 2025a). Palu (Chang et al. 2025) introduced a rank-aware KV-Cache compression scheme that dynamically adapts the low-rank approximation of cached keys and values per attention head that significantly reducing memory footprint without degrading retrieval accuracy for the SVD decomposed models.

Streaming Inference FlashAttention reorders the standard attention computation into tiled matrix-multiply and softmax steps that fit in on-chip SRAM, thereby minimizing HBM-SRAM transfers and achieving up to 3× end-to-end speedups on GPT-2 with exact numerical equivalence to dense attention (Dao et al. 2022).

FlashAttention-2 and -3 optimize work partitioning and kernel fusion to sustain 50-73 % of peak FLOPs on NVIDIA A100s, and FlashAttention-3 on H100s delivers 1.5-2.0× speedups in FP16 (up to 740 TFLOPs/s, 75 % utilization) and near 1.2 PFLOPs/s in FP8–with 2.6× lower numerical error than baseline FP8 attention (Dao et al. 2023; Shah et al. 2024). BlockLLM introduces a block-parallel inference engine that shards and caches KV blocks across multiple concurrent workflows, reducing redundant memory loads and lowering per-token latency by as much as 33% in multi-tenant serving scenarios (Shi and Team 2024). Star Attention partitions long-sequence inputs across hosts in two phases–a block-local intra-node attention followed by a global inter-node aggregation–enabling exact, distributed attention on sequences exceeding 64K tokens with sublinear communication overhead (Liu and Team 2024).

Fine-Tuning & Hybrid Compression LoRA inserts trainable, low-rank adaptation matrices into the frozen weights of each Transformer layer, reducing the number of fine-tuned parameters by over four orders of magnitude while matching full-fine-tuning performance on tasks such as GLUE and SQuAD (Hu et al. 2021). QLoRA builds on this by first quantizing all model weights to 4-bit integers using NormalFloat and GPTQ techniques, then applying LoRA to the quantized weights; this combination enables fine-tuning of 65B-parameter LLMs on a single 48 GB GPU with negligible quality loss (Dettmers et al. 2023).

Although low-rank SVD compressors and I/O-aware streaming kernels have each shown dramatic gains, they have developed in isolation: SVD methods focus on offline factorization with no regard for GPU memory-I/O or block-parallel execution, while streaming kernels target dense weights. There is limited work in developing a rank-awared streaming kernels for SVD-based low-rank models inference. We bridge this gap with two families of rank-aware GPU kernels that eliminate memory-intensive dense activation materialization. Our method is agnostic to the upstream SVD compression – applicable to any SVD-compressed model without degrading accuracy.

Methodology

Optimal Low-Rank Factorization via SVD Let $W \in \mathbb{R}^{m\times n}$ be a dense matrix (e.g. a full self-attention projection matrix). Its full singular value decomposition

$$W = U\,\Sigma\,V^\top, \Sigma = \operatorname{diag}\big(\sigma_1, \ldots, \sigma_{\min\{m,n\}}\big), \quad \sigma_i \geq \sigma_j, \quad \forall\, i > j. \tag{1}$$

yields the best rank-r approximation in Frobenius norm by the Eckart-Young-Mirsky theorem (Eckart and Young 1936). Concretely, truncating to the top-r singular values gives

$$W^{[r]} = U_{[:,1:r]}\,\Sigma_{[1:r,1:r]}\,V^\top_{[:,1:r]},$$

which provably minimizes $\|W - W^{[r]}\|_F$ over all rank-r matrices. For simplicity, we evenly distribute the singular values, rewriting as $W^{[r]} = \tilde{U}\,\tilde{V}^\top$ where $\tilde{U} = U_{[:,1:r]}\Sigma^{1/2}$ and $\tilde{V} = V_{[:,1:r]}\Sigma^{1/2}$.

Model Decomposition For a general large-language model $\Theta = \big\{\Theta^{(\ell)}_{\mathcal{A}} \cup \Theta^{(\ell)}_{\mathcal{F}}\big\}_{\ell=1}^{L}$, where $\Theta^{(\ell)}_{\mathcal{A}}$ and $\Theta^{(\ell)}_{\mathcal{F}}$ denote the attention and FFN parameters in block ℓ. Equivalently, grouping across layers gives $\Theta = \{\Theta^{(\ell)}_{\mathcal{A}}\}_{\ell=1}^{L} \cup \{\Theta^{(\ell)}_{\mathcal{F}}\}_{\ell=1}^{L}$, so truncated SVD can be independently applied layer-wise on each $\Theta^{(\ell)}_{\mathcal{A}}$ and $\Theta^{(\ell)}_{\mathcal{F}}$ subfactors, enabling fine-grained, per-layer rank selection and compression. We apply rank-r_{Θ^*}

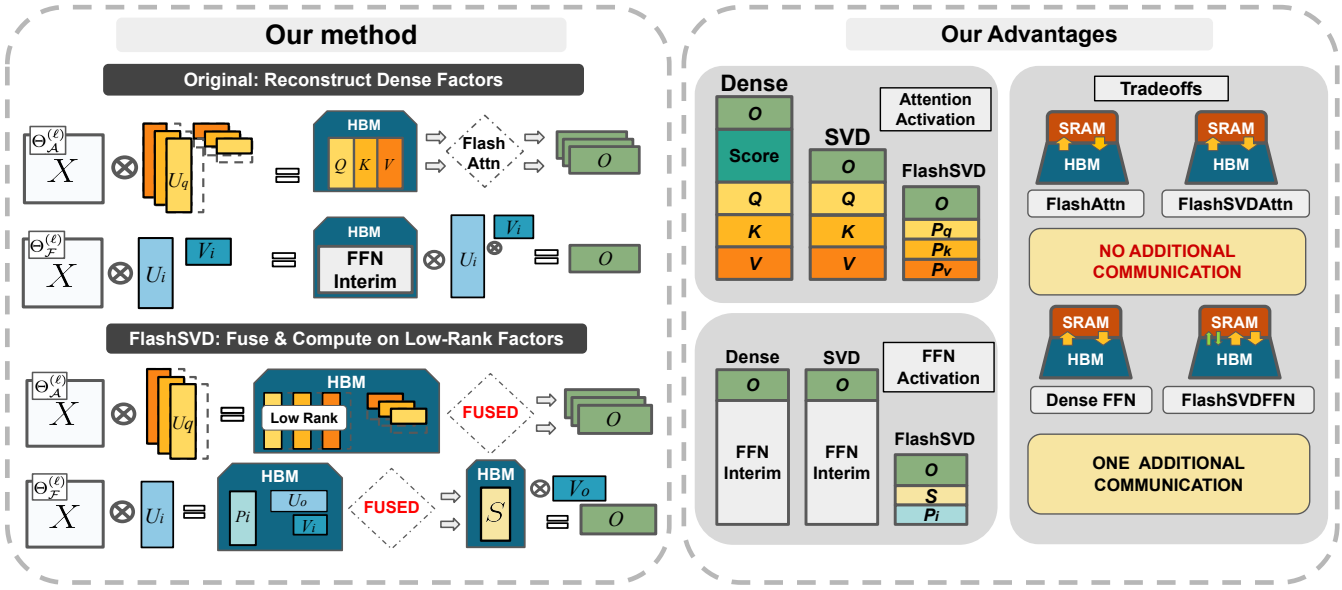

Figure 1: FlashSVD fuses low-rank projections into both the self-attention and FFN stages by streaming small tiles of the truncated U and V factors through on-chip SRAM instead of materializing dense activation matrices in HBM. **FlashSVDAttn** only consumes low-rank factors, reducing complexity from $O(BM\mathcal{D}_{\mathcal{A}}^{\Theta})$ to $O(BMr_{\Theta^*})$. In the FFN, inputs are first projected into the low-rank factor space via a cuBLAS GEMM (for standard matrix multiplication) and then passed through **FlashSVDFFN** in one pass, avoiding any $O(BM\mathcal{D}_{\mathcal{F}}^{\Theta})$ intermediate activations and ensuring the rank-awared HBM performance.

truncated SVD to each dense projection matrix in a Transformer block. The resulting compressed model is denoted Θ^*, so that $r_{\Theta}^* = r_{\Theta^*}$. The self-attention projections are square matrices of size $\mathcal{D}_{\mathcal{A}}^{\Theta}$, while the FFN projection matrices $W_i^\top$ and W_o are rectangular of shape $\mathcal{D}_{\mathcal{F}}^{\Theta} \times \mathcal{D}_{\mathcal{A}}^{\Theta}$.

Streaming Activations In standard multi-head self-attention calculation, one forms the full score matrix $\text{Softmax}(\frac{QK^\top}{\sqrt{d_{\Theta_{\mathcal{A}},h}}}) \in \mathbb{R}^{B\times M\times M}$, where $d_{\Theta_{\mathcal{A}},h} = \mathcal{D}_{\mathcal{A}}^{\Theta} \,/\, H$, $\forall\ h \in \{1,\ldots,H\}$, B is batch size, and M is sequence length, incurring $\mathcal{O}(BM^2)$ off-chip memory and bandwidth per head. FlashAttention (Dao et al. 2022, 2023) instead tiles and streams the queries, keys, and values, small blocks of Q, K and V, sized to fit the GPU's shared memory per streaming multiprocessor, are loaded on-chip, dot-products are then computed and softmaxed with streaming, and results are immediately accumulated in full FP32 precision without ever materializing the full $B \times H \times M^2$ matrix on HBM. Consequently, HBM memory complexity for attention computation drops from $\mathcal{O}\big(BHM^2\big)$ to $\mathcal{O}(BH\,M\,d_{\Theta_{\mathcal{A}},h})$ for H heads attention, enabling state-of-the-art attention computation performance in both training and inference.

FlashSVDAttention FlashSVDAttention enables rank-aware streaming by consuming only the low-rank factors of each head's query, key, and value projections:

$$P_q,\, P_k,\, P_v \in \mathbb{R}^{B\times M\times r_{\Theta^*}}, \quad V_q,\, V_k,\, V_v \in \mathbb{R}^{r_{\Theta^*}\times d_{\Theta_{\mathcal{A}},h}}.$$

where the P_a factors are formulated from XU_a, and V_a $\forall\ a \in \{q,k,v\}$. Internally the kernel tiles the $(M, d_{\Theta_{\mathcal{A}},h})$ plane into small $(B_M, d_{\Theta_{\mathcal{A}},h})$ blocks that stream through on-chip SRAM. For each tile it (1) forms the local low-rank query block $Q_{\text{tile}} = P_{q,\text{tile}}\, V_{q,\text{tile}} + b_q$ with iterative accumulation over r_{Θ^*}, (2) multiplies and accumulates score with corresponding reconstructed key blocks $K_{\text{tile}} = P_{k,\text{tile}}\, V_{k,\text{tile}} + b_k$, (3) applies a row-wise softmax, and (4) weights and reduces the value factors $P_{v,\text{tile}}\, V_{v,\text{tile}}$ to produce the output slice. By fusing the two small GEMMs and the softmax into one pass, and never materializing the full $B \times H \times M \times d_{\Theta_{\mathcal{A}},h}$ score or output buffers off-chip, FlashSVDAttn attains the rank-aware $\mathcal{O}\big(Hr_{\Theta^*}(B\,M + d_{\Theta_{\mathcal{A}},h})\big)$ HBM bound.

FlashSVDFFN Once the FlashAttention alleviated the peak-memory burden of self-attention (Dao et al. 2022, 2023), the feed-forward network (FFN) becomes the new dominant factor. Even when W_i and W_o admit low-rank decompositions $W_i^{[r_{\Theta^*}]} = U_i\, V_i$ and $W_o^{[r_{\Theta^*}]} = U_o\, V_o$ with ranks $r_{\Theta^*} \ll D_{\mathcal{F}}^{\Theta}$, the intermediate $\phi(XU_iV_i + b_i) \in \mathbb{R}^{(B\times M)\times D_{\mathcal{F}}^{\Theta}}$ must still be materialized in full, dominating HBM despite in low-rank case. We propose two rank-aware streaming FFN variants to eliminate this large interim activation that trade off compute versus I/O: a mild version that maximizes reuse of vendor-tuned GEMM kernels at the cost of extra off-chip traffic, and an extreme version that enable zero I/O to HBM.

FlashSVDFFN V1 (Rank-Aware Fusion) First, the input is projected into the factor space with $P = X\,U_i$ via the cuBlas GEMM, storing $(B \times M) \times r_i$ sized intermediate in HBM. Second, a lightweight streamed kernel applies the nonlinearity and the next projection in one fused pass by

Algorithm 1 FlashSVDAttention

PART I. SUBROUTINE LOAD_TILES:
Input: Factor $P_a \in \mathbb{R}^{B\times B_M\times r_{\Theta^*}}$, projection $V_a \in \mathbb{R}^{r_{\Theta^*}\times d_{\Theta_{\mathcal{A},h}}}$, bias $b_a \in \mathbb{R}^{d_{\Theta_{\mathcal{A},h}}}$, tile size B_R, rank r_{Θ^*}, padded rank $R = \lceil \frac{r_{\Theta^*}}{B_R} \rceil \cdot B_R$
Output: Reconstructed tiles A_{tile} for each r with padding mask
for $r = 0$ **to** R **step** B_R **do**
  $A_{\text{tile}} \leftarrow P_a[:, :, r : r + B_R]\, V_a[r : r + B_R, :] + b_a$
end for
PART II. FLASHSVDATTENTION FORWARD PASS:
Input: Low-rank factors $P_q, P_k, P_v \in \mathbb{R}^{B\times M\times r_{\Theta^*}}$, projections $V_q, V_k, V_v \in \mathbb{R}^{r_{\Theta^*}\times d_{\Theta_{\mathcal{A},h}}}$, biases $b_q, b_k, b_v \in \mathbb{R}^{d_{\Theta_{\mathcal{A},h}}}$, tile size B_M and B_R, sequence length M
Output: Output $O \in \mathbb{R}^{B\times M\times d_{\Theta_{\mathcal{A},h}}}$
Initialize $O \leftarrow 0$
for $\ell = 0$ **to** $M - 1$ **step** B_M **do**
  `Reconstruct Q block on-chip`
  $Q_{\text{tile}} \leftarrow \text{LOAD\_TILES}(\mathrm{P_q}[:, \ell : \ell + \mathrm{B_M}, :], \mathrm{V_q}[:, :, \ell : \ell + \mathrm{B_M}], \mathrm{b_q}, \mathrm{B_R})$
  for $m = 0$ **to** $M - 1$ **step** B_M **do**
    `Reconstruct K and V blocks on-chip`
    $K_{\text{tile}} \leftarrow \text{LOAD\_TILES}(\mathrm{P_k}[:, \ell : \ell + \mathrm{B_M}, :], \mathrm{V_k}[:, :, \ell : \ell + \mathrm{B_M}], \mathrm{b_k}, \mathrm{B_R})$
    $V_{\text{tile}} \leftarrow \text{LOAD\_TILES}(\mathrm{P_v}[:, \ell : \ell + \mathrm{B_M}, :], \mathrm{V_v}[:, :, \ell : \ell + \mathrm{B_M}], \mathrm{b_v}, \mathrm{B_R})$
    `Compute attention scores and softmax`
    scores $\leftarrow Q_{\text{tile}} (K_{\text{tile}})^\top / \sqrt{d_{\Theta_{\mathcal{A},h}}}$
    $\alpha \leftarrow \text{STREAM-SOFTMAX}(\text{scores}, \text{axis} = -1)$ (Dao et al. 2022)
    $O[:, \ell : \ell + B_M, :] \mathrel{+}= \alpha\, V_{\text{tile}}$
  end for
end for

computing $S = \phi(P\,V_i + b_i)\,U_o$ according to algorithm 2 without ever materializing the full $(B \times M) \times D_{\mathcal{F}}^{\Theta}$ activation. Finally, the output is reconstructed with $O = S\,V_o + b_o$ via a second cuBlas GEMM. Because both P and S remain bounded by the truncated rank r_{Θ^*}, the overall computation and intermediate memory scale with r_{Θ^*} with the trade-off for two extra I/O with HBM, directly yielding the tight bounds on activation memory and FLOP reduction.

FlashSVDFFN V2 (Extreme-Case) employs a single fused GEMM-Activation-GEMM GPU kernel to stream the low-rank FFN forward pass by directly loading X, projection factors U_i, V_i, U_o, V_o, and biases b_i, b_o on-the-fly, incuring no additional arithmetic cost or asymptotic complexity compared to a dense FFN (identical $\mathcal{O}\big(B\,M\,r_{\Theta^*}\,D_{\mathcal{F}}^{\Theta} + B\,M\,r_{\Theta^*}\,D_{\mathcal{A}}^{\Theta}\big)$ FLOPs). This unified pipeline supports arbitrary activations $\phi(\cdot)$, and theoretically pushes the lower bound for variable part of rank-aware activation compression to zero. By launching one kernel block per tile and accumulating directly into the output buffer, we avoid any materialized $(B \times M) \times \mathcal{D}_{\mathcal{F}}^{\Theta}$ mid-buffers, achieving the $O(r_{\Theta^*}(B\,M + \mathcal{D}_{\mathcal{F}}^{\Theta}))$ HBM bound.

Multi-Head Attention Amplifies SVD Savings Many recent compression schemes (e.g., ASVD (Yuan and Others 2023), Dobi-SVD (Wang et al. 2025a)) apply SVD to the entire $\mathcal{D}_{\mathcal{A}}^{\Theta} \times \mathcal{D}_{\mathcal{A}}^{\Theta}$ attention projection as a single matrix. This single head approach only yields parameter reduction when $r_{\Theta^*} < \frac{1}{2}\,\mathcal{D}_{\mathcal{A}}^{\Theta}$, and in practice forces drastic rank cuts that frequently degrade accuracy and limits the overall compression capacity of SVD-based method. In contrast, the native multi-head structure splits $\mathcal{D}_{\mathcal{A}}^{\Theta}$ into H heads of size $d_{\Theta_{\mathcal{A},h}} = \mathcal{D}_{\mathcal{A}}^{\Theta}/H$. By applying truncated SVD per head, the threshold for parameter compression drops to $r_{\Theta^*} < \frac{\mathcal{D}_{\mathcal{A}}^{\Theta}}{H+1}$. As Figure 2 illustrates, compressing BERT's head projections to 1.5 M parameters requires a 56% rank reduction under single-head SVD, risking severe quality loss, whereas multi-head SVD needs only a 19% cut, achieving the same compression with far gentler rank pruning.

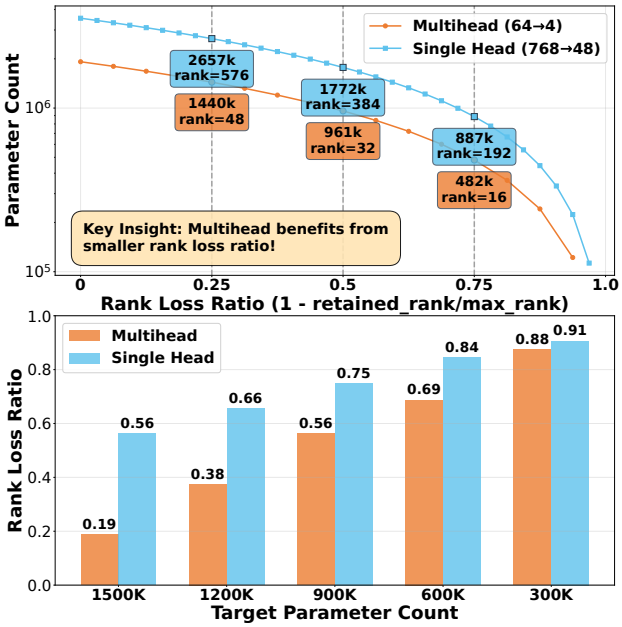

Figure 2: Multi-head SVD compresses model attention projection matrices more gently than single-head SVD.

Analysis

FlashSVD establishes provable memory efficient upper bounds on dominant activations during inference by exploiting low-rank structure. Below, we summarize the key theorems that characterize these activation bounds and clarify the fundamental time-memory trade-offs in FlashSVD. Detailed proofs are available in extended version.

Peak Memory Footprint Analysis Table 1 demonstrates that combining *FlashSVDAttention* with *FlashSVDFFN* reduces the activation memory footprint from $\mathcal{O}\big(BM(\mathcal{D}_{\mathcal{F}}^{\Theta} + \mathcal{D}_{\mathcal{A}}^{\Theta} + M)\big)$ to $\mathcal{O}\big(r_{\Theta^*}(BM + \mathcal{D}_{\mathcal{F}}^{\Theta} + \mathcal{D}_{\mathcal{A}}^{\Theta})\big)$. The FFN memory can be further reduced using *FlashSVDFFN-V2*, and ultimately brought variable part to zero. In the typical regime where $r_{\Theta^*} \ll \mathcal{D}_{\mathcal{A}}^{\Theta}, \mathcal{D}_{\mathcal{F}}^{\Theta}, BM$, these optimizations yield a ranked-bounded reduction in HBM activation.

Algorithm 2 FlashSVDFFN V1

Require: Input $X \in \mathbb{R}^{B\times M\times \mathcal{D}_{\mathcal{A}}^{\Theta}}$; SVD factors $U_i \in \mathbb{R}^{\mathcal{D}_{\mathcal{A}}^{\Theta}\times r_{\Theta^*}}$, $V_i \in \mathbb{R}^{r_{\Theta^*}\times \mathcal{D}_{\mathcal{F}}^{\Theta}}$, $b_i \in \mathbb{R}^{\mathcal{D}_{\mathcal{F}}^{\Theta}}$; $U_o \in \mathbb{R}^{\mathcal{D}_{\mathcal{F}}^{\Theta}\times r_{\Theta^*}}$, $V_o \in \mathbb{R}^{r_{\Theta^*}\times \mathcal{D}_{\mathcal{A}}^{\Theta}}$, $b_o \in \mathbb{R}^{\mathcal{D}_{\mathcal{A}}^{\Theta}}$; number of blocks G; nonlinearity $\phi(\cdot)$.

Ensure: $O \in \mathbb{R}^{B\times M\times \mathcal{D}_{\mathcal{A}}^{\Theta}}$

$B_M \leftarrow \lceil M/G \rceil, \quad B_{D_{\mathcal{F}}^{\Theta}} \leftarrow \lceil \mathcal{D}_{\mathcal{F}}^{\Theta}/G \rceil$

`GEMM 1st: project into factor-space (cuBlas)`

$P \leftarrow X\,U_i \{(BM)\times r_{\Theta^*}\}$

`On-Chip: rank-aware activation streaming`

for $\ell = 0$ **to** $M-1$ **step** B_M **do**

  initialize $Z_{\text{tile}} \leftarrow 0_{B\times B_M\times r_{\Theta^*}}$

  for $d = 0$ **to** $\mathcal{D}_{\mathcal{F}}^{\Theta} - 1$ **step** $B_{D_{\mathcal{F}}^{\Theta}}$ **do**

    $V_i^{(d)} \leftarrow V_i[:, d : d + B_{D_{\mathcal{F}}^{\Theta}}]\{(r_{\Theta^*}\times B_{D_{\mathcal{F}}^{\Theta}})\}$

    $U_o^{(d)} \leftarrow U_o[d : d + B_{D_{\mathcal{F}}^{\Theta}}, :]\{(B_{D_{\mathcal{F}}^{\Theta}}\times r_{\Theta^*})\}$

    $b_i^{(d)} \leftarrow b_i[d : d + B_{D_{\mathcal{F}}^{\Theta}}]\{(B_{D_{\mathcal{F}}^{\Theta}})\}$

    $Y \leftarrow \phi\big(P\,V_i^{(d)} + b_i^{(d)}\big)\ \{B\times B_M\times B_{D_{\mathcal{F}}^{\Theta}}\}$

    $Z_{\text{tile}} \mathrel{+}= Y\,U_o^{(d)}\ \{(B\times B_M\times B_{D_{\mathcal{F}}^{\Theta}})\times(B_{D_{\mathcal{F}}^{\Theta}}\times r_{\Theta^*})\}$

  end for

  $Z \leftarrow Z_{\text{tile}}\ \{(BM)\times r_{\Theta^*}\}$

end for

`GEMM 2nd: project to model-space (cuBlas)`

$O \leftarrow Z\,V_o + b_o \quad \{(BM)\times \mathcal{D}_{\mathcal{A}}^{\Theta}\}$

Rank Sensitivity The memory footprint of FlashSVD depends critically on the selected SVD rank r_{Θ^*}. For the attention module, lowering the rank by one in a single head saves $\Delta\mathcal{M}_{\text{flash-svd-attn-single}} = \mathcal{O}(BM + \mathcal{D}_{\mathcal{A}}^{\Theta})$ words. Applied across all H heads, the aggregate saving is $\Delta\mathcal{M}_{\text{flash-svd-attn-multi}} = \mathcal{O}(H(BM + d_{\Theta_{\mathcal{A}},h}))$. An analogous result holds for the feed-forward network. A unit decrease in rank reduces off-chip memory by

$$\Delta\mathcal{M}_{\text{flash-svd-ffn}} \le \mathcal{O}\big(BM + D_{\mathcal{F}}^{\Theta}\big)\,,$$

demonstrating that FFN layers benefit from rank reduction along both the batch-sequence and feature dimensions.

Module	$\mathcal{M}_{\text{dense}}$	$\mathcal{M}_{\text{FlashSVD}}$
Single-Head Attn.	$3BM\mathcal{D}_{\mathcal{A}} + BM^2$	$\mathcal{O}(r_{\Theta^*}(BM + \mathcal{D}_{\mathcal{A}}))$
Multi-Head Attn.	$BM\mathcal{D}_{\mathcal{A}}$	$\mathcal{O}(r_{\Theta^*}(BM + \mathcal{D}_{\mathcal{A}}))$
FFN-V1	$BM\mathcal{D}_{\mathcal{F}}$	$\mathcal{O}(r_{\Theta^*}(BM + \mathcal{D}_{\mathcal{F}}))$
FFN-V2	$BM\mathcal{D}_{\mathcal{F}}$	$\mathcal{O}(r_{\Theta^*}\mathcal{D}_{\mathcal{F}})$

Table 1. Peak HBM $\mathcal{M}$ Complexity for each sub-module.

Speedup and Latency Analysis We analyze the performance benefits of FlashSVD along three dimensions: FLOP count, memory I/O volume, and end-to-end latency.

Theoretical FLOP savings arise from replacing dense matrix multiplications with low-rank projections: the full-rank computation costs $\mathcal{O}\big((D_{\mathcal{A}}^{\Theta})^2 MB + D_{\mathcal{A}}^{\Theta} D_{\mathcal{F}}^{\Theta} MB\big)$, while the low-rank version requires only $\mathcal{O}\big(r_{\Theta^*}^2 MB + r_{\Theta^*} D_{\mathcal{A}}^{\Theta} MB + r_{\Theta^*} D_{\mathcal{F}}^{\Theta} MB\big)$. The resulting asymptotic speedup factor is:

$$\Theta\left(\frac{(D_{\mathcal{A}}^{\Theta})^2 + D_{\mathcal{A}}^{\Theta} D_{\mathcal{F}}^{\Theta}}{r_{\Theta^*}^2 + r_{\Theta^*} D_{\mathcal{A}}^{\Theta} + r_{\Theta^*} D_{\mathcal{F}}^{\Theta}}\right).$$

In terms of data movement, the full-rank model must load all query/key/value activations and FFN outputs, with total volume $\mathcal{O}(BM(D_{\mathcal{A}}^{\Theta} + D_{\mathcal{F}}^{\Theta}))$. In contrast, FlashSVD only needs to read and write low-rank components, yielding reduced volume

$$\mathcal{O}(MBr_{\Theta^*} + r_{\Theta^*} D_{\mathcal{A}}^{\Theta} + r_{\Theta^*} D_{\mathcal{F}}^{\Theta}).$$

Finally, following the roofline model, the end-to-end latency per layer is bounded below by the slower of the compute- and bandwidth-limited paths:

$$T \;\ge\; \max\Big\{\frac{\text{FLOPs}}{\text{PeakFLOP/s}}\,,\ \frac{4\,N_{\text{bytes}}}{\beta}\Big\},$$

where $N_{\text{bytes,in}}$ is the number of input bytes and β is the memory bandwidth in bytes/s.

Experiment

Experimental Setup

Model Setup Prior activation-aware methods (e.g., ASVD (Yuan and Others 2023), SVD-LLM (Wang et al. 2024b, 2025c)) have mainly targeted decoder-side KV-cache compression in causal LMs, leaving encoder-side activation overhead–dominated by feed-forward networks (FFN)–largely unexplored. In this work, we explicitly address the encoder scenario, focusing on popular compressed encoder models such as naive SVD-BERT, fine-tuned low-rank BERT (FWSVD-BERT), and our fully-fused FlashSVD-BERT variants. To isolate the impact of our kernel-level optimizations, we conduct thorough ablation studies, independently evaluating FlashSVDAttn and FlashSVDFFN under diverse input settings. While FlashSVD naturally extends to decoder models–and we indeed observe meaningful improvements there–our primary analyses emphasize the encoder-side scenario, where FFN activations dominate and prior methods fall short.

Evaluation Metrics For encoders, we measure accuracy on GLUE classification tasks (Wang et al. 2018) across sequence lengths (128, 256, 512 tokens), activation memory (peak MiB via PyTorch profiler), and inference latency per batch (milliseconds) on an NVIDIA L40S GPU. We assess how latency and memory usage scale across varying compression ratios ($\frac{\#\text{ Param Compressed}}{\#\text{ Param Dense}}$). For decoders, we evaluate perplexity (PPL) on WikiText-2, reporting peak memory (GiB), activation overhead, and wall-clock time (minutes:seconds). Although encoder scenarios (dominated by FFN activations) remain our primary focus, decoder-side experiments highlight FlashSVD's complementary benefits.

Main Results

Table 2 shows that FlashSVD delivers a strong tradeoff between compression, memory, and throughput on BERT model across GLUE benchmarks. FlashSVD matches

Methodology	Base	sst2					mnli				
		Peak	Tran	Lat	Acc	Ratio	Peak	Tran	Lat	Acc	Ratio
BERT Experiments											
Dense	418.7	695.0	277.3	79.7	92.32	1.00	1547.4	1129.7	310.2	84.06	1.00
Parameter Ratio: 25%											
Vanilla SVD	332.4	741.6	409.1	161.1	85.16	-	1990.0	1657.5	359.7	66.73	-
FlashSVD V1	332.4	576.7	244.2	188.8	85.16	**0.60**	1230.2	897.7	490.5	66.71	**0.54**
FlashSVD V2	332.4	576.7	244.2	253.5	85.16	**0.60**	1230.2	897.7	753.0	66.71	**0.54**
FWSVD	332.4	913.6	581.1	83.8	89.91	-	3853.0	3520.5	463.2	78.01	-
FlashSVD V1	332.4	541.6	209.2	139.6	89.91	**0.36**	1213.2	880.8	479.9	77.90	**0.25**
Parameter Ratio: 50%											
Vanilla SVD	249.7	653.5	403.8	150.8	66.88	-	1888.4	1638.7	321.1	37.92	-
FlashSVD V1	249.7	461.6	211.8	**141.6**	66.88	**0.52**	1120.7	870.9	350.6	37.81	**0.53**
FlashSVD V2	249.7	461.6	211.8	192.9	66.88	**0.52**	1120.7	870.9	584.7	37.82	**0.53**
FWSVD	249.7	821.1	571.3	66.5	79.44	-	3734.3	3484.5	427.8	51.06	-
FlashSVD V1	249.7	453.4	203.7	101.5	79.44	**0.36**	1112.5	862.8	**341.4**	50.92	**0.25**
RoBERTa Experiments											
Dense	475.5	753.0	277.5	78.1	94.17	1.00	1605.4	1129.9	217.5	87.59	1.00
Parameter Ratio: 25%											
Vanilla SVD	390.3	883.4	493.1	119.2	49.20	-	2383.8	1993.5	225.0	33.37	-
FlashSVD v1	390.3	634.5	244.2	184.0	49.20	**0.49**	1288.0	897.7	393.1	33.40	**0.45**
FlashSVD v2	390.3	634.5	244.2	238.1	49.20	**0.49**	1288.0	897.7	744.4	33.42	**0.45**
FWSVD	390.3	875.4	485.1	110.5	89.29	-	2374.8	1984.5	220.8	77.21	-
FlashFWSVD	390.3	599.5	209.2	127.4	89.00	**0.43**	1271.1	880.8	401.3	77.46	**0.44**
Parameter Ratio: 50%											
Vanilla SVD	307.6	790.8	483.3	110.3	50.94	-	2264.2	1956.7	247.7	31.16	-
FlashSVD v1	307.6	519.4	211.8	148.8	51.34	**0.44**	1178.5	870.9	287.1	31.10	**0.45**
FlashSVD v2	307.6	519.4	211.8	195.1	51.63	**0.44**	1178.5	870.9	570.8	31.07	**0.45**
FWSVD	307.6	782.9	475.3	103.0	71.09	-	2256.1	1948.5	345.6	37.82	-
FlashFWSVD	307.6	511.3	203.7	105.5	69.49	**0.43**	1170.4	862.8	**279.2**	38.40	**0.44**

Table 2. Main Results. Comparison of FlashSVD and SVD-based methods across multiple compression ratios (25%, 50%) with a batch size of 64. Metrics reported include base memory usage (Base), peak total memory (Peak), transient activation memory (Tran), inference latency (Lat), accuracy (Acc), and parameter compression ratio (Ratio).

vanilla SVD and FWSVD in accuracy at all compression levels, while slashing transient activation memory by up to 75% and reducing peak inference memory to just 1,213 MiB–substantially lower than both the original BERT (1,547 MiB) and naive SVD (1,990 MiB). Notably, naive SVD can paradoxically increase runtime memory despite parameter reduction, highlighting the need for holistic memory optimization. FlashSVD also maintains or improves inference speed: at 50% parameter retention on MNLI, it processes batches 20% faster than baseline SVD, and achieves similar gains on SST-2. These results demonstrate that FlashSVD's rank-aware streaming design not only preserves accuracy and minimizes memory, but can even accelerate inference by eliminating large dense intermediates. In extreme settings, FlashSVD matches the best memory savings, but with higher latency due to reduced parallelism in FFN reconstruction. Among design variants, FlashSVDFFN V1 offers the best balance of memory efficiency and speed, and is recommended for practical deployment.

Decoder Experiment Alongside the encoder-focused evaluations in previous sections, we also extend our analysis to decoder models, evaluating the impact of FlashSVD in inference acceleration for widely-adopted SVD compression methods. On LLaMA-2-7B, FlashSVD achieves the strongest memory reductions during generation (Table 3). Compared to dense KV caching, activation-only memory (peak minus model-only) decreases by 80.3% during prefill and by 50.9% across decoding at 128/256 tokens. Peak memory likewise falls by 38.7% (prefill) and 19.5% (average decode). While ASVD KV alone attains substantial savings (up to 76.0% activation, 33.6% peak), FlashSVD consistently yields superior performance. Practically, these gains mitigate non-KV activations from becoming memory bottlenecks, enabling longer contexts or increased batch sizes under limited VRAM budgets.

Ablation Study

FlashSVDAttn We evaluate FlashSVDAttn against PyTorch's fused SDPA (FlashAttention) across short-to-medium ($M \leq 256$) and long ($M \geq 512$) contexts (Table 4). FlashSVDAttn achieves parity with dense at rank R=32 for M=256 (1.00 $\times$) and surpasses dense by 8% at R=16 (1.08$\times$). For longer, compute-intensive contexts, FlashSV-

Method	Activation-only (GiB)			Act. Reduction (%)		Peak memory (GiB)			Peak Reduction (%)	
	Prefill	Dec@128	Dec@256	Prefill	Decode (avg)	Prefill peak	D@128	D@256	Prefill	Decode (avg)
Dense KV	6.80	3.23	4.28	–	–	15.36	11.79	12.85	–	–
ASVD KV	1.63	1.79	2.39	76.0	44.4	10.20	10.36	10.95	33.6	13.5
+FlashSVD	1.34	1.58	2.11	**80.3**	**50.9**	9.41	9.65	10.18	**38.7**	**19.5**

Table 3. LLaMA-2-7B Experiment Result with activation-aware decoder baseline (ASVD). Activation memory is defined as peak minus model-only baseline; result reported for decoding lengths of 128 and 256 tokens.

FlashSVDAttn (vs. fused SDPA / Dense)			
Rank	M=256	M=512	M=1024
Dense (baseline)	0.74 / 1.00×	2.31 / 1.00×	6.98 / 1.00×
64	0.83 / 0.89×	2.21 / 1.05×	5.65 / 1.24×
48	0.81 / 0.92×	2.10 / **1.10**×	5.41 / **1.29**×
32	0.74 / **1.00**×	1.79 / **1.29**×	4.57 / **1.53**×
16	0.68 / **1.08**×	1.68 / **1.37**×	4.25 / **1.64**×
FlashSVDFFN (vs. dense FFN)			
Version / Rank	M=256	M=512	M=1024
Dense	0.29 / 1.00×	0.68 / 1.00×	1.47 / 1.00×
V1 (Rank 192)	0.54 / 0.53×	0.86 / 0.79×	1.82 / 0.81×
V1 (Rank 96)	0.23 / **1.26**×	0.43 / **1.60**×	0.78 / **1.88**×
V2 (Rank 96)	1.39 / 0.21×	2.72 / **0.25**×	5.25 / **0.28**×

Table 4. Profiling results for FlashSVD kernels. Each cell shows time in ms and speedup (×) vs. the corresponding dense baseline at the same M.

DAttn consistently outperforms dense attention at ranks $R \leq 48$, peaking at 1.64× speedup for R=16, M=1024. These results highlight FlashSVDAttn's capability to deliver substantial speedups in low-rank settings, and was scalable (with more benefits) in long context scenario.

FlashSVDFFN FlashSVDFFN V1 provides an advantageous speed-memory trade-off, especially at lower ranks. At rank 96, it achieves speedups of 1.26× for M=256, 1.60× for M=512, and up to 1.88× at M=1024. FlashSVDFFN V2, designed to fuse the entire FFN into a single kernel to avoid writing intermediate results to HBM, naturally incurs greater computational overhead, leading to expected lower performance. Consequently, V1 emerges as the preferred variant for practical low-rank deployments, effectively leveraging low-rank structures and significantly outperforming dense baselines at reduced computational budgets.

Fine-tuning Unlocks Extreme Low-Rank Compression While recent methods (Yuan and Others 2023; Wang et al. 2024b; Hsu et al. 2022) favor training-free SVD to avoid retraining costs, we demonstrate that carefully fine-tuning low-rank factors achieves substantially greater compression without sacrificing accuracy. Specifically, fine-tuning BERT at 50% parameter retention preserves near-baseline performance while reducing transient memory by 48.2% and peak memory by 29.4%–approaching the dense model's persistent memory footprint (Table 5). This direction represents a compelling avenue for future research.

Model	Acc. (%)	Trans. Mem (MiB)	Peak Mem. (MiB)	Lat. (ms)
Dense	81.41	281.3	699.0	98.8
SVD	80.52	399.8	652.8	128.9
FlashSVD	80.52	**207.8**	**460.8**	**114.8**
SVD (no finetune)	13.50	408.3	658.0	160.1

Table 5. STS-B (Pearson) Performance and 50% Compression Ratio with Finetuning

Conclusion

In this work, we introduce FlashSVD, the first fused, rank-aware inference framework for SVD-compressed transformers. By streaming low-rank projections directly into FlashAttention and FFN kernels, we eliminate large activations and cut peak on-chip memory by up to 71% with no extra computational cost. Across BERT and RoBERTa, FlashSVD matches or exceeds dense throughput under 25-50% compression and achieves up to 1.9× FFN speedups at modest ranks. These results demonstrate that rank-aware tiling makes low-rank SVD a practical, high-performance strategy for memory-constrained transformer deployment.

Limitation. Despite these gains, low-rank approximations inherently introduce some performance degradation, especially at extremely low ranks. While FlashSVD ensures consistency with existing SVD-compressed models, it still incurs the accuracy loss associated with reduced-rank representations relative to dense models. Therefore, exploring fine-tuning strategies tailored for ultra-low-rank regimes remains an important direction–where our hardware-efficient inference support can further amplify the benefits of such model-level optimizations.

Ethical Statement

This work poses no ethical or societal risks. FlashSVD is designed to improve computational efficiency and enable sustainable, on-device deployment of large language models with reduced memory and energy footprints. All experiments use publicly available datasets under appropriate licenses, and no human or personally identifiable data are involved.

Acknowledgements

This work was supported in part by the U.S. National Science Foundation (NSF) under Grants 2328805 and 2112562, and by the Army Research Office (ARO) under Award

W911NF-23-2-0224. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF, ARO, or the U.S. Government.

References

Chang, C.-C.; Lin, W.-C.; Lin, C.-Y.; Chen, C.-Y.; Hu, Y.-F.; Wang, P.-S.; Huang, N.-C.; Ceze, L.; Abdelfattah, M. S.; and Wu, K.-C. 2025. Palu: KV-Cache Compression with Low-Rank Projection. In *The Thirteenth International Conference on Learning Representations*.

Dao, T.; Fu, D. Y.; Ermon, S.; Rudra, A.; and Ré, C. 2022. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. *arXiv preprint*. ArXiv:2205.14135.

Dao, T.; Fu, D. Y.; Ermon, S.; Rudra, A.; and Ré, C. 2023. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. *arXiv preprint*. ArXiv:2307.08691.

Dettmers, T.; Pagnoni, A.; Holtzman, A.; and Zettlemoyer, L. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. *arXiv preprint*. ArXiv:2305.14314.

Eckart, C.; and Young, G. 1936. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3): 211–218.

Han, S.; Mao, H.; and Dally, W. J. 2016. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In *International Conference on Learning Representations (ICLR)*. ArXiv:1510.00149.

Hsu, Y.-C.; Hua, T.; Chang, S.; Lou, Q.; Shen, Y.; and Jin, H. 2022. Language model compression with weighted low-rank factorization. *arXiv preprint arXiv:2207.00112*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. LoRA: Low-Rank Adaptation of Large Language Models. In *ICLR*. ArXiv:2106.09685.

Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; and Kalenichenko, D. 2018. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2704–2713. ArXiv:1712.05877.

Li, S.; Chen, W.; and Zeng, D. 2025. EVODiff: Entropy-aware Variance Optimized Diffusion Inference. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Lin, B.; and Colleagues. 2024. MoDeGPT: Modular Decomposition for Large Language Model Compression. In *ICML*. OpenReview: 8EfxjTCg2k.

Liu, F.; and Team, N. 2024. Star Attention: Efficient LLM Inference over Long Sequences. arXiv preprint. ArXiv:2411.17116.

Pope, R.; Douglas, S.; Chowdhery, A.; Devlin, J.; Bradbury, J.; Heek, J.; Xiao, K.; Agrawal, S.; and Dean, J. 2023. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5: 606–624.

Shah, J.; Bikshandi, G.; Zhang, Y.; Thakkar, V.; Ramani, P.; and Dao, T. 2024. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *Advances in Neural Information Processing Systems*, 37: 68658–68685.

Shi, E.; and Team. 2024. BlockLLM: Multi-Tenant Finer-Grained Serving for Large Language Models. arXiv preprint. ArXiv:2404.18322.

Shoeybi, M.; Patwary, M.; Puri, R.; LeGresley, P.; Casper, J.; and Catanzaro, B. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Wang, Q.; Ke, J.; Tomizuka, M.; Keutzer, K.; and Xu, C. 2025a. Dobi-SVD: Differentiable SVD for LLM Compression and Some New Perspectives. In *ICLR*. OpenReview: kws76i5XB8.

Wang, Q.; Vahidian, S.; Ye, H.; Gu, J.; Zhang, J.; and Chen, Y. 2024a. Coreinfer: Accelerating large language model inference with semantics-inspired adaptive sparse activation. *arXiv preprint arXiv:2410.18311*.

Wang, Q.; Ye, H.; Chung, M.-Y.; Liu, Y.; Lin, Y.; Kuo, M.; Ma, M.; Zhang, J.; and Chen, Y. 2025b. CoreMatching: A Co-adaptive Sparse Inference Framework with Token and Neuron Pruning for Comprehensive Acceleration of Vision-Language Models. *arXiv preprint arXiv:2505.19235*.

Wang, X.; Alam, S.; Wan, Z.; Shen, H.; and Zhang, M. 2025c. SVD-LLM V2: Optimizing Singular Value Truncation for Large Language Model Compression. *arXiv preprint arXiv:2503.12340*.

Wang, X.; Zheng, Y.; Wan, Z.; and Zhang, M. 2024b. Svd-llm: Truncation-aware singular value decomposition for large language model compression. *arXiv preprint arXiv:2403.07378*.

Ye, H.; Gao, Z.; Ma, M.; Wang, Q.; Fu, Y.; Chung, M.-Y.; Lin, Y.; Liu, Z.; Zhang, J.; Zhuo, D.; et al. 2025. KVCOMM: Online Cross-context KV-cache Communication for Efficient LLM-based Multi-agent Systems. *arXiv preprint arXiv:2510.12872*.

Yuan, A.; and Others. 2023. Activation-aware Singular Value Decomposition for Large Language Models. OpenReview preprint. ArXiv:2502.01403.

Zhang, J.; Zhang, Y.; Gu, J.; Dong, J.; Kong, L.; and Yang, X. 2024. Xformer: Hybrid X-Shaped Transformer for Image Denoising. In *ICLR*.

Zhou, Y.; Wang, Y.; Yin, X.; Zhou, S.; and Zhang, A. R. 2025. The Geometry of Reasoning: Flowing Logics in Representation Space. *arXiv preprint arXiv:2510.09782*.