ACCELERATING ATTENTION PROJECTIONS WITH BASIS DECOMPOSITION

Anonymous authors
Paper under double-blind review

ABSTRACT

Attention is a core operation in large language models (LLMs). We present BD Attention (**BDA**), a *lossless algorithmic reformulation* of attention that accelerates its linear projections. BDA is enabled by a simple matrix identity from Basis Decomposition (**BD**), which restructures multi-head projections into a compact form while preserving attention outputs in exact arithmetic. Complementary to I/O-aware system optimizations such as FlashAttention, BDA provides an architecture-agnostic reduction in projection FLOPs. BDA requires only **4s** of offline preparation on DeepSeek-V2-Lite (16B), **with no retraining required**. On modern GPUs, it achieves **34% faster** key/value projections in FP16 and **25% smaller** K/V projection weights. These gains come with a perplexity (PPL) increase of just **0.02%** (FP16) or **0.0004%** (FP32) on DeepSeek-V2-Lite—a negligible effect on model performance. These results position BDA as a theoretically exact method for lossless attention projection acceleration that is complementary to existing engineering-level optimizations. Our code is available at `https://anonymous.4open.science/r/Basis-decomp-57B8`.

1 INTRODUCTION

Attention (Vaswani et al., 2017) has emerged as the fundamental building block in large language models (LLMs) and vision-language models (VLMs), enabling them to scale effectively across diverse tasks. However, the cost of multi-head attention (MHA) in both computation and memory makes it a major efficiency bottleneck. Existing acceleration techniques can be broadly divided into two categories. On the one hand, *lossless system-level optimizations*, such as FlashAttention (Dao et al., 2022), improve I/O efficiency by reordering memory access and fusing kernels, but their gains are hardware-specific and do not reduce the number of arithmetic operations or parameters. On the other hand, *approximate algorithmic approaches*, including linear attention (Katharopoulos et al., 2020; Choromanski et al., 2021), sparse attention (Child et al., 2019; Beltagy et al., 2020), pruning (Ma et al., 2023; Frantar & Alistarh, 2023), and quantization (Frantar et al., 2023; Lin et al., 2024c), reduce complexity or storage but inevitably trade off accuracy and often require retraining or calibration.

In this work, we take a different path and propose **BD Attention (BDA)**, a *lossless algorithmic reformulation* of attention that accelerates its linear projections. BDA is enabled by a simple yet general matrix identity from **Basis Decomposition (BD)** to restructure MHA projections into a compact form, thereby reducing projection parameters and arithmetic operations while preserving attention outputs in exact arithmetic. This algorithmic perspective complements existing I/O-aware optimizations.

We validate BDA across multiple scenarios. On DeepSeek-V2-Lite (16B, FP16), BDA requires only **4s** of offline preparation with no retraining and achieves **34% faster** key/value projections and **25% smaller** K/V projection weights on modern GPUs, with perplexity (PPL) change at the negligible level of **0.02%**. Beyond inference, training experiments show that BDA achieves BLEU scores comparable to MHA without any hyperparameter adjustment. Furthermore, when applied on top of low-rank pruned models, BD reduces memory by **16.5%** and improves throughput by **17.2%**, demonstrating its compatibility with existing compression techniques. The magnitude of this speedup depends on the head-to-embedding dimension ratio, and is most pronounced in MLA-style architectures such as DeepSeek-V2/V3.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Table 1: **Comparison of acceleration techniques for Transformers.**

Method	Lossless	(Re)Train	CPU Speedup	GPU Speedup	# Params
Flash Attention	✓	✗	✗	✓	Same
Linear / Sparse Attention	✗	✓	✓	✓	Same
Pruning / Quantization	✗	Partial	✓	✓	Reduce
BD Attention	✓	✗	✓	✓	Reduce

Overall, BDA establishes a new algorithmic foundation for lossless attention projection acceleration, uniting theoretical guarantees with practical speedups on real hardware.

We highlight two main contributions of this study:

- **Theoretical foundation.** We propose **Basis Decomposition (BD)**, a general matrix decomposition that guarantees hardware-friendly reconstruction with probability 1 under assumptions that are naturally satisfied by neural network weight matrices (Theorem 3.1).
- **Practical acceleration.** We apply BD to attention projections and low-rank layers, showing that it reduces parameters and arithmetic operations without degrading model quality. We demonstrate consistent projection-operator acceleration with both a standard PyTorch implementation on CPU and a fused Triton kernel on GPU, achieving near-theoretical speedups in practice.

2 Related Work

Research on accelerating attention broadly falls into two categories: (i) *lossless* system-level optimizations that preserve exact outputs, and (ii) *approximate* algorithms that trade accuracy for efficiency. Table 1 summarizes representative approaches.

Lossless methods. FlashAttention and its successors (Dao et al., 2022; Dao, 2023; Shah et al., 2024; Kwon et al., 2023) reduce memory traffic by fusing attention kernels and tiling reads/writes to on-chip SRAM, achieving substantial GPU speedups without altering outputs. These approaches are *lossless* but depend heavily on GPU memory hierarchies and I/O architectures. Crucially, FlashAttention and BDA target disjoint steps of attention: FlashAttention optimizes the softmax computation $\mathrm{softmax}(\mathbf{Q}\mathbf{K}^\top)\mathbf{V}$ (Algorithm 2, step 5), whereas BDA reduces FLOPs in the projection step $\mathbf{X} \to \mathbf{K}, \mathbf{V}$ (Algorithm 2, steps 2–3). The two are therefore complementary and can be combined.

Approximate methods. In contrast, many lines of work accelerate attention or reduce memory by introducing approximations.

- **Linear-attention** replaces the quadratic softmax kernel with linear-time kernel approximations (Wang et al., 2020; Katharopoulos et al., 2020; Choromanski et al., 2021; Schlag et al., 2021; Chen et al., 2021).
- **Sparse-attention** imposes predefined or learned sparsity patterns, e.g., Sparse Transformers (Child et al., 2019), Longformer (Beltagy et al., 2020), Reformer (Kitaev et al., 2020), and sketching-based sampling methods (Chen et al., 2022).
- **Pruning** removes parameters based on importance criteria.
  - **Unstructured pruning** targets individual weights (Frantar & Alistarh, 2023; Sun et al., 2024; Zhang et al., 2024), effective in reducing parameters but hard to accelerate on GPUs.
  - **Semi-structured pruning** (e.g., $N:M$ patterns) (Mishra et al., 2021) enables GPU acceleration on NVIDIA's Ampere GPUs but restricts density flexibility.
  - **Structured pruning** removes entire channels/heads (Ma et al., 2023; van der Ouderaa et al., 2024; Ashkboos et al., 2024), more deployment-friendly but often with higher accuracy loss.

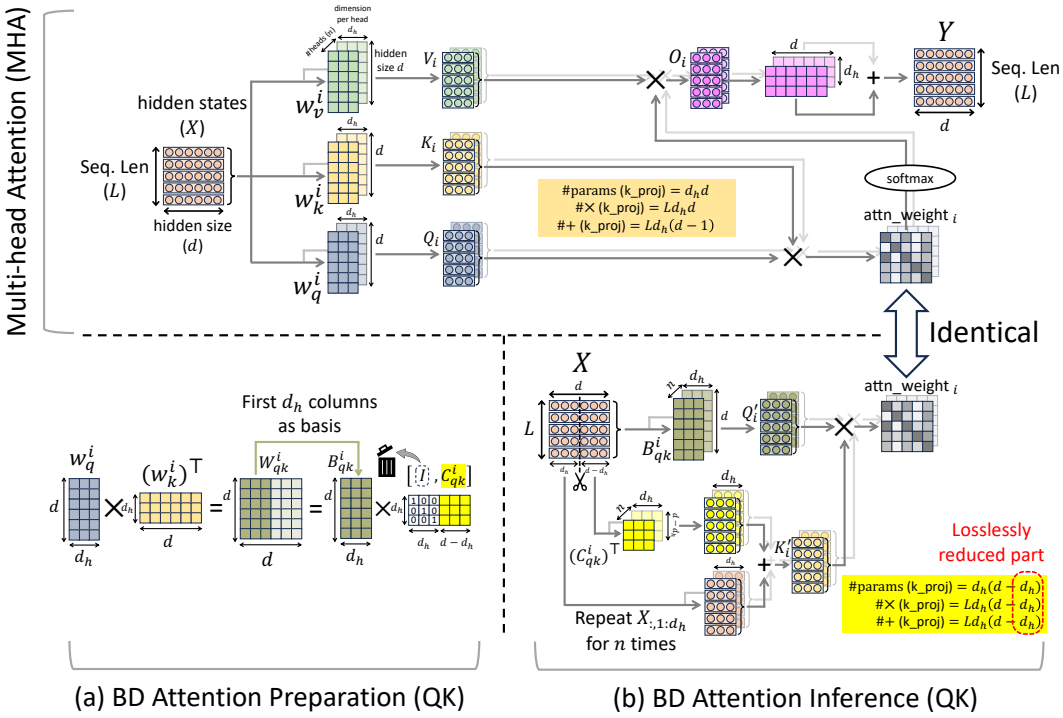


Figure 1: Illustration of BD Attention (**BDA**) using the QK projection as an example (VO is analogous). BDA consists of two stages: (a) **BD Attention Preparation** (Algorithm 5), performed offline *once* during model deployment, where the projection matrices are transformed via Basis Decomposition; (b) **BD Attention Inference** (Algorithm 2) saves d_h/d in both parameters and computation, while preserving exact outputs.

  - **Low-rank pruning** approximates weight matrices by decomposing them into lower-rank components (Yuan et al., 2023; Wang et al., 2024; Zhao et al., 2025), reducing both parameters and compute. While hardware-friendly, it introduces approximation error unless the true rank is low.

- **Quantization** compresses parameters/activations into low-precision formats (Frantar et al., 2023; Lin et al., 2024c; Xiao et al., 2023; Lin et al., 2024b), achieving strong memory savings but inevitably introducing errors at low bit-widths.

These methods can deliver large efficiency gains but are inherently *not lossless*, and usually require careful finetuning to mitigate quality drops.

3 Method

3.1 Basis Decomposition

Given two rectangular matrices $\mathbf{U} \in \mathbb{R}^{m\times r}$ and $\mathbf{V}^\top \in \mathbb{R}^{r\times n}$, where $r < \min(m,n)$, their product matrix $\mathbf{W} = \mathbf{U}\mathbf{V}^\top$ has rank at most r. Our goal is to develop a general decomposition method that expresses $\mathbf{W}$ as $\mathbf{W} = f(\mathbf{M}_1, \ldots, \mathbf{M}_k)$, where the function $f(\mathbf{M}_1, \ldots, \mathbf{M}_k)$ has lower computational cost than the original multiplication $\mathbf{U}\mathbf{V}^\top$.

For a rank-r matrix $\mathbf{W}$, there exist r linearly independent rows. We select these rows to form a basis matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_r]^\top \in \mathbb{R}^{r\times n}$. Thus, $\mathbf{B}$ forms a basis of the row space of $\mathbf{W}$. For any row $\mathbf{w}_i \in \mathbf{W} \setminus \mathbf{B}$, where $\mathbf{w}_i \in \mathbb{R}^n$, it can be written as a linear combination of the basis vectors in $\mathbf{B}$:

$$\mathbf{w}_i = \sum_{j=1}^{r} c_{ij}\mathbf{b}_j, \tag{1}$$

which follows directly from the fact that the row space of $\mathbf{W}$ is spanned by the r basis vectors in $\mathbf{B}$. Collecting all such coefficients c_{ij} forms a coefficient matrix $\mathbf{C} \in \mathbb{R}^{(m-r)\times r}$, where each row contains the weights for reconstructing a non-basis row of $\mathbf{W}$ as a linear combination of the basis vectors in $\mathbf{B}$. With $\mathbf{B}$ and $\mathbf{C}$, the original matrix $\mathbf{W}$ can be fully reconstructed.

This forms an alternative representation of the low-rank matrix $\mathbf{W}$, distinct from the traditional low-rank multiplication $\mathbf{U}\mathbf{V}^\top$. We refer to this representation as **Basis Decomposition (BD)**, encompassing both the decomposition $\mathbf{W} \to (\mathbf{B}, \mathbf{C})$ and its reconstruction $(\mathbf{B}, \mathbf{C}) \to \mathbf{W}$. We use the term *BD* for general reference, and specify *BD decomposition / reconstruction* only when emphasizing the decomposition/reconstruction process. Its relation to classical interpolative decomposition (Voronin & Martinsson, 2017) is discussed in Appendix B.

Memory cost of BD. BD stores two matrices: the basis matrix $\mathbf{B} \in \mathbb{R}^{r\times n}$ and the coefficient matrix $\mathbf{C} \in \mathbb{R}^{(m-r)\times r}$. The total memory cost is $r(m+n-r)$, which is strictly smaller than the full matrix size mn and low rank matrices size $r(m+n)$ for any $r < \min(m,n)$. In contrast, the traditional low-rank representation $\mathbf{W} = \mathbf{U}\mathbf{V}^\top$ requires $r(m+n)$ parameters and is only more compact than the full matrix when $r < \frac{mn}{m+n}$.

Computational cost of BD reconstruction. Reconstructing $\mathbf{W}$ from BD involves computing $\mathbf{C}\mathbf{B}$ and inserting the r basis rows. This requires $2r(m-r)n$ floating-point operations (FLOPs). In comparison, the traditional reconstruction $\mathbf{U}\mathbf{V}^\top$ costs $2rmn$ FLOPs. Therefore, BD reconstruction is computationally more efficient for any $r < \min(m,n)$.

3.2 Selection of Basis

For a rank-r matrix $\mathbf{W}$, not every subset of r rows necessarily forms a basis. The selected rows must be linearly independent, or equivalently, the selected submatrix $\mathbf{B}$ must contain an invertible $r \times r$ sub-block. The following theorem gives a sufficient distributional condition for a square matrix to be nonsingular almost surely.

Theorem 3.1 (Almost Sure Full Rank of Random Matrices)**.** *Let $\mathbf{W}$ be an $r \times r$ real random matrix. Suppose the entries of $\mathbf{W}$ are drawn from a probability measure μ on $\mathbb{R}^{r^2}$ that is absolutely continuous with respect to the Lebesgue measure λ. Then W has full rank ($\operatorname{rank}(W) = r$) with probability 1.*

The proof of Theorem 3.1 is provided in Appendix A. To apply this result to $\mathbf{W} = \mathbf{U}\mathbf{V}^\top$, consider the model in which the entries of each factor have a jointly absolutely continuous distribution. For any row and column index sets $\mathcal{I}, \mathcal{J}$ of size r, we have

$$\mathbf{W}_{\mathcal{I},\mathcal{J}} = \mathbf{U}_{\mathcal{I},:}\big(\mathbf{V}_{\mathcal{J},:}\big)^\top.$$

Both square factors are nonsingular almost surely by Theorem 3.1, and hence so is their product. Since there are finitely many such index pairs, all these submatrices are simultaneously nonsingular with probability one. Consequently, any r rows or columns of $\mathbf{W}$, including the first or last r, form a basis almost surely under this model.

Mandt et al. (2017) show that, under a local Ornstein–Uhlenbeck approximation, constant-learning-rate SGD has a Gaussian stationary distribution over the weights. With nondegenerate stationary covariance, this distribution is jointly absolutely continuous, so the preceding basis-selection result applies to weight matrices $\mathbf{U}$ and $\mathbf{V}$ modeled in this way.

PIFA (Zhao et al., 2025) uses a basis-and-coefficient representation and selects basis rows through pivoted QR factorization (Businger & Golub, 1971). BD does not prescribe a particular basis-selection strategy: any valid row or column basis yields an exact decomposition. We exploit this freedom to select computationally convenient bases and compare their reconstruction residuals in finite precision.

In particular, we find that choosing contiguous rows/columns, such as the first-r or last-r rows/columns, offers significant efficiency advantages. It minimizes the I/O overhead during reconstruction by avoiding scattered memory writes/reads on modern hardware such as GPUs. We adopt this strategy as the standard basis selection method for BD in practice.

Algorithm 1 MHA Inference

input Weight $\mathbf{W}_q \in \mathbb{R}^{d\times nd_h}$, $\mathbf{W}_k \in \mathbb{R}^{d\times nd_h}$, $\mathbf{W}_v \in \mathbb{R}^{d\times nd_h}$, $\mathbf{W}_o \in \mathbb{R}^{nd_h\times d}$; Input $\mathbf{X} \in \mathbb{R}^{L\times d}$

1: $\mathbf{Q} \leftarrow \mathbf{X}\mathbf{W}_q$
2: $\mathbf{K} \leftarrow \mathbf{X}\mathbf{W}_k$
3: $\mathbf{V} \leftarrow \mathbf{X}\mathbf{W}_v$
4: $[\mathbf{Q}_1, \ldots, \mathbf{Q}_n] \leftarrow \mathbf{Q}$, $[\mathbf{K}_1, \ldots, \mathbf{K}_n] \leftarrow \mathbf{K}$, $[\mathbf{V}_1, \ldots, \mathbf{V}_n] \leftarrow \mathbf{V}$
5: $\mathbf{O}_i \leftarrow \text{softmax}(\frac{\mathbf{Q}_i\mathbf{K}_i^\top}{\sqrt{d_h}})\mathbf{V}_i$
6: $\mathbf{Y} \leftarrow [\mathbf{O}_1, \ldots, \mathbf{O}_n]\mathbf{W}_o$

output $\mathbf{Y}$

Algorithm 2 BD Attention Inference

input Weight $\mathbf{B}_{qk} \in \mathbb{R}^{d\times nd_h}$, $\mathbf{C}_{qk} \in \mathbb{R}^{(d-d_h)\times nd_h}$, $\mathbf{C}_{vo} \in \mathbb{R}^{(d-d_h)\times nd_h}$, $\mathbf{B}_{vo} \in \mathbb{R}^{nd_h\times d}$; Input $\mathbf{X} \in \mathbb{R}^{L\times d}$

1: $\mathbf{Q}' \leftarrow \mathbf{X}\mathbf{B}_{qk}$
2: $\mathbf{K}' \leftarrow [\mathbf{X}_{:,\,1:d_h}]^{\times n} + \mathbf{X}_{:,\,d_h:d}\mathbf{C}_{qk}$
3: $\mathbf{V}' \leftarrow [\mathbf{X}_{:,\,1:d_h}]^{\times n} + \mathbf{X}_{:,\,d_h:d}\mathbf{C}_{vo}$
4: $[\mathbf{Q}'_1, \ldots, \mathbf{Q}'_n] \leftarrow \mathbf{Q}'$, $[\mathbf{K}'_1, \ldots, \mathbf{K}'_n] \leftarrow \mathbf{K}'$, $[\mathbf{V}'_1, \ldots, \mathbf{V}'_n] \leftarrow \mathbf{V}'$
5: $\mathbf{O}'_i \leftarrow \text{softmax}(\frac{\mathbf{Q}'_i\mathbf{K}'^\top_i}{\sqrt{d_h}})\mathbf{V}'_i$
6: $\mathbf{Y} \leftarrow [\mathbf{O}'_1, \ldots, \mathbf{O}'_n]\mathbf{B}_{vo}$

output $\mathbf{Y}$

Figure 2: Inference pseudocode comparison between standard MHA and BD Attention. Red operations highlight the structural difference.

Let $\mathbf{I} \in \mathbb{R}^{r\times r}$ denote the identity matrix. The following identities hold for the four types of BD:

$$\begin{aligned} &\text{row-first: } \mathbf{W} \equiv \begin{bmatrix}\mathbf{I}\\ \mathbf{C}\end{bmatrix}\mathbf{B}, \quad \text{row-last: } \mathbf{W} \equiv \begin{bmatrix}\mathbf{C}\\ \mathbf{I}\end{bmatrix}\mathbf{B},\\ &\text{col-first: } \mathbf{W} \equiv \mathbf{B}[\mathbf{I}, \mathbf{C}], \quad \text{col-last: } \mathbf{W} \equiv \mathbf{B}[\mathbf{C}, \mathbf{I}]. \end{aligned} \tag{2}$$

While the theoretical reconstruction is exact, numerical residuals may arise in practice due to finite-precision arithmetic or ill-conditioned basis matrices. To mitigate this, we compare the reconstruction errors from the first- and last-r basis candidates and retain the one with the smaller Frobenius norm residual. The full procedure for *row-based* Basis Decomposition is summarized in Algorithm 3 (BD decomposition) and Algorithm 4 (BD reconstruction), where a subset of rows is selected as the basis. The *column-based* variant can be formulated analogously and is omitted here. An offset-search variant of Residual-min that further improves numerical stability is described in Appendix G.

By losslessly replacing standard low-rank matrix multiplication with a more compact and computationally efficient alternative, BD is broadly applicable to scenarios involving low-rank multiplications. This includes applications such as neural network inference and data compression.

3.3 BD for Multi-Head Attention

Multi-head attention (MHA) is a central component in Transformer-based architectures, widely adopted in large language models (LLMs) (Vaswani et al., 2017; Radford et al., 2018; Brown et al., 2020; Touvron et al., 2023) and vision-language models (VLMs) (Dosovitskiy et al., 2021; Radford et al., 2021; Li et al., 2022). The comparison between BD Attention and MHA is illustrated in Figure 1.

We begin by reviewing the standard multi-head attention (MHA) mechanism. Let d be the input (embedding) dimension, n be the number of attention heads, d_h be the dimension per head, L be the input sequence length, and $\mathbf{X} \in \mathbb{R}^{L\times d}$ be the attention input. MHA produces queries, keys and values ($\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{L\times nd_h}$) by three projection matrices ($\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{d\times nd_h}$):

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_k, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_v \tag{3}$$

$\mathbf{Q}, \mathbf{K}, \mathbf{V}$ are divided into n heads for MHA:

$$[\mathbf{Q}_1, \ldots, \mathbf{Q}_n] = \mathbf{Q}, \quad [\mathbf{K}_1, \ldots, \mathbf{K}_n] = \mathbf{K}, \quad [\mathbf{V}_1, \ldots, \mathbf{V}_n] = \mathbf{V}, \tag{4}$$

$$\mathbf{O}_i = \text{softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^\top}{\sqrt{d_h}}\right)\mathbf{V}_i, \quad \mathbf{Y} = [\mathbf{O}_1, \ldots, \mathbf{O}_n]\mathbf{W}_o. \tag{5}$$

where $\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i \in \mathbb{R}^{L\times d_h}$ represent the respective query, key, and value vectors for the i-th head, and $\mathbf{W}_o \in \mathbb{R}^{nd_h\times d}$ represents the output projection matrix.

We reformulate multi-head attention (MHA).

$$\begin{aligned}\mathbf{Y} &= \sum_{i=1}^{n} \mathbf{O}_i \mathbf{W}_o^i = \sum_{i=1}^{n} \text{softmax}(\frac{\mathbf{Q}_i \mathbf{K}_i^\top}{\sqrt{d_h}}) \mathbf{V}_i \mathbf{W}_o^i \\ &= \sum_{i=1}^{n} \text{softmax}\left(\frac{\mathbf{X}(\mathbf{W}_q^i (\mathbf{W}_k^i)^\top)\mathbf{X}^\top}{\sqrt{d_h}}\right) \mathbf{X}(\mathbf{W}_v^i \mathbf{W}_o^i) \\ \mathbf{W}_q &\to [\mathbf{W}_q^1, \ldots, \mathbf{W}_q^n], \quad \mathbf{W}_k \to [\mathbf{W}_k^1, \ldots, \mathbf{W}_k^n], \\ \mathbf{W}_v &\to [\mathbf{W}_v^1, \ldots, \mathbf{W}_v^n], \quad \mathbf{W}_o \to \begin{bmatrix} \mathbf{W}_o^1 \\ \vdots \\ \mathbf{W}_o^n \end{bmatrix}\end{aligned} \tag{6}$$

where $\mathbf{W}_q^i, \mathbf{W}_k^i, \mathbf{W}_v^i \in \mathbb{R}^{d \times d_h}$ denote the i-th *vertical slice* of the corresponding weight matrices, and $\mathbf{W}_o^i \in \mathbb{R}^{d_h \times d}$ denotes the i-th *horizontal slice* of $\mathbf{W}_o$. Both $(\mathbf{W}_q^i {\mathbf{W}_k^i}^\top)$ and $(\mathbf{W}_v^i \mathbf{W}_o^i)$ are a matrix with shape $d \times d_h$ multiply a matrix with shape $d_h \times d$. As $d_h < d$, this reveals a key insight:

> ***Key Insight***: Each head's QK and VO computation are inherently low-rank matrix multiplications (Elhage et al., 2021).

We can apply Basis Decomposition (BD) **offline during model preparation** to compress these projection weight. Taking QK as an example (VO in Appendix C), we decompose the weight product $(\mathbf{W}_q^i {\mathbf{W}_k^i}^\top)$ using column-based BD. Let $\mathbf{B}_{qk}^i \in \mathbb{R}^{d \times d_h}$ be the first-r columns (last-r is similar) and $\mathbf{C}_{qk}^i \in \mathbb{R}^{d_h \times (d - d_h)}$ the coefficient matrix, we can convert the attention score into expression of BD:

$$\begin{aligned}\text{attn_score}_i &= \mathbf{Q}_i \mathbf{K}_i^\top = \mathbf{X}(\mathbf{W}_q^i (\mathbf{W}_k^i)^\top)\mathbf{X}^\top = \mathbf{X}(\mathbf{B}_{qk}^i [\mathbf{I}, \mathbf{C}_{qk}^i])\mathbf{X}^\top \\ &= (\mathbf{X}\mathbf{B}_{qk}^i)([\mathbf{I}, \mathbf{C}_{qk}^i]\mathbf{X}^\top) = (\mathbf{X}\mathbf{B}_{qk}^i)(\mathbf{X}_{:,\,1:d_h}^\top + \mathbf{C}_{qk}^i \mathbf{X}_{:,\,d_h:d}^\top) \\ &= \mathbf{Q}_i' {\mathbf{K}_i'}^\top \\ \mathbf{Q}_i' &\leftarrow \mathbf{X}\mathbf{B}_{qk}^i, \quad {\mathbf{K}_i'}^\top \leftarrow \mathbf{X}_{:,\,1:d_h}^\top + \mathbf{C}_{qk}^i \mathbf{X}_{:,\,d_h:d}^\top\end{aligned} \tag{7}$$

where $\mathbf{X} \to \left[\mathbf{X}_{:,\,1:d_h}, \mathbf{X}_{:,\,d_h:d}\right]$ partitions $\mathbf{X}$ into its first d_h and the remaining $d - d_h$ columns. By aligning all head's BD to either first-r or last-r, we avoid calculating each head's Q and K projection separately, thereby reducing I/O overhead. The choice between first-r and last-r columns is determined by comparing the average residuals across all heads (Algorithm 5). This alignment allows Equation 7, which computes QK for a single head, to be reformulated to compute QK for all heads simultaneously:

$$\mathbf{Q}' \leftarrow \mathbf{X}\mathbf{B}_{qk}, \quad \mathbf{K}' \leftarrow [\mathbf{X}_{:,\,1:d_h}]^{\times n} + \mathbf{X}_{:,\,d_h:d}\mathbf{C}_{qk}. \tag{8}$$

where

$$\mathbf{B}_{qk} \leftarrow [\mathbf{B}_{qk}^1, \ldots, \mathbf{B}_{qk}^n], \quad \mathbf{C}_{qk} \leftarrow [(\mathbf{C}_{qk}^1)^\top, \ldots, (\mathbf{C}_{qk}^n)^\top]. \tag{9}$$

The operator $[\mathbf{X}_{:,\,1:d_h}]^{\times n}$ denotes repeating the matrix n times along the second dimension. The value-output (VO) projection can be processed analogously using the row-based BD formulation (Appendix C).

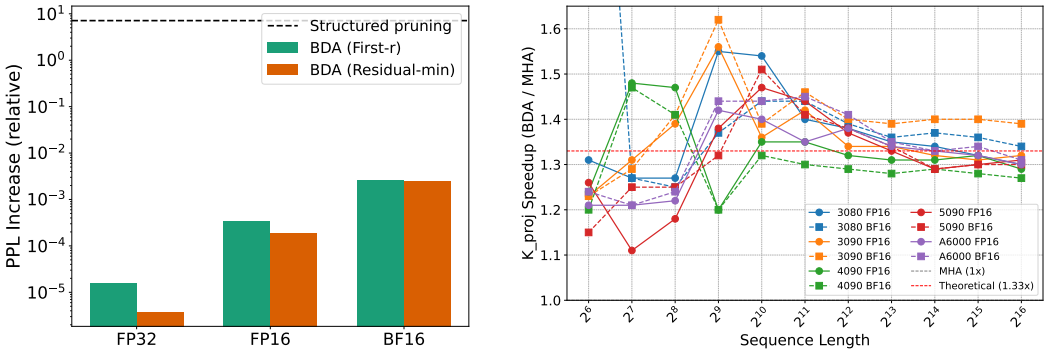
We compare the original MHA inference (Algorithm 1) with the BD-based MHA inference (Algorithm 2). The only differences (highlighted in red) lie in the computation of keys $\mathbf{K}$ and values $\mathbf{V}$, where BD replaces the original projections with smaller matrix multiplications. A discussion of the interaction between positional embeddings and BD is provided in Appendix E.

Table 2 summarizes the projection costs. The attention matrix products and KV-cache dimensions remain unchanged.

Training with BDA. BDA can also be trained directly by treating $\mathbf{B}_{qk}, \mathbf{C}_{qk}, \mathbf{B}_{vo}, \mathbf{C}_{vo}$ as learnable parameters. We initialize these matrices using Xavier initialization and compute gradients through the forward computation in Algorithm 2 using standard automatic differentiation. The identity blocks remain fixed, and no basis reselection or BD recomputation is required during training.

Table 2: **Projection computation and storage costs of MHA and BDA.** Costs include the Q, K, V, and O projections. FLOPs count the dominant matrix-multiplication terms, with two FLOPs per multiply-add.

Quantity	MHA	BDA
Projection parameters	$4ndd_h$	$4ndd_h - 2nd_h^2$
Projection FLOPs	$8Lndd_h$	$8Lndd_h - 4Lnd_h^2$



(a) Perplexity increase (b) Efficiency

Figure 3: **Evaluation of BD Attention (BDA). (a) Perplexity increase**: Perplexity (↓) increase on WikiText2 when replacing all MHA layers of DeepSeek-V2-Lite with BDA. The increase is nearly imperceptible **0.02% (FP16)**, with *Residual-min* performing better. For reference, the dashed line shows the degradation from a structured pruning baseline at the same compression ratio (25% K/V channels). **(b) Efficiency**: Relative speedup of the k_proj operator under FP16 and BF16, measured across multiple NVIDIA architectures (Ampere, Ada, Blackwell). The dashed line at $1.33\times$ marks the theoretical bound. Speedups fluctuate around this line but remain consistent across GPUs, averaging **1.34**$\times$ (FP16) and **1.37**$\times$ (BF16). BDA also reduces parameter and memory usage by **25%**.

4 Experiments

We evaluate BD Attention (BDA) from three perspectives: inference accuracy and efficiency, training evaluation, and integration with low-rank pruning.

4.1 BDA Inference Accuracy

Figure 3a reports the increase in perplexity on WikiText2 when replacing all MHA layers in DeepSeek-V2-Lite (16B) (Liu et al., 2024a) with BDA. Two strategies are compared: *First-r*, which always selects the first r rows, and *Residual-min*, which adaptively selects between the first or last r rows depending on the smaller reconstruction residual.

The perplexity increase is nearly imperceptible **0.0004% (FP32), 0.02% (FP16), 0.2% (BF16)**, with *Residual-min* consistently outperforming *First-r*. A similar advantage is observed at the operator level in per-layer reconstruction errors (see Appendix Table 5), where *Residual-min* achieves up to an order-of-magnitude lower error in FP32.

For reference, we also include a structured pruning baseline that removes 25% of K/V channels at the same compression ratio ($d_h/d = 128/512 = 25\%$). This baseline follows the relative-importance scoring strategy of Zhang et al. (Zhang et al., 2024), where each channel's importance is estimated, summed, and the least important 25% are pruned. Although more recent structured pruning techniques (Ma et al., 2023; van der Ouderaa et al., 2024; Ashkboos et al., 2024) achieve better performance, they typically require access to a calibration dataset, which is beyond the scope of this comparison. Here, structured pruning is reported only as a reference for the scale of perplexity degradation at the same compression ratio.

Table 3: **BD applied to low-rank pruning** on LLaMA2 models (FP16). BD further improves throughput and memory efficiency over low-rank pruning while preserving perplexity. Best throughput and memory are highlighted in **bold**.

Model	Metric	Dense	Low rank 80%	BD (from low-rank)
LLaMA2-7B	Throughput (no kv cache)	338.23	368.90	**422.58**
	Throughput (kv cache)	3726.31	4244.27	**5285.60**
	Memory (GB)	12.55	10.21	**8.52**
	PPL	5.47	7.50	7.50
LLaMA2-13B	Throughput (no kv cache)	181.15	201.50	**238.51**
	Throughput (kv cache)	2345.99	2566.04	**2856.81**
	Memory (GB)	24.36	19.58	**16.35**
	PPL	4.88	6.41	6.42

We further report zero-shot downstream results in Appendix F, which are consistent with the perplexity trends observed here.

4.2 BDA Inference Efficiency

We first recall that BD reduces the FLOPs of the key projection from $Ld_h(2d-1)$ to $Ld_h(2(d-d_h))$, implying a theoretical speedup of $\frac{2d-1}{2(d-d_h)} \approx 1.33\times$ under the DeepSeek configuration ($d = 512, d_h = 128$).

To rigorously evaluate whether this theoretical gain translates into practice, we consider two complementary regimes: **(i)** a *standard PyTorch implementation* without any kernel fusion, and **(ii)** a *fused Triton kernel* evaluated across multiple GPU architectures. Together, these settings demonstrate that BD's efficiency does not rely on implementation-specific I/O optimizations.

CPU (standard PyTorch implementation). We implemented BDA in standard PyTorch and benchmarked on CPU, without any kernel fusion or hardware-specific tuning. BDA achieves acceleration on average **1.17**$\times$ in FP16 and **1.40**$\times$ in BF16. Figure 4 visualizes the speedup across sequence lengths.

GPUs (fused Triton kernel). Because GPUs provide substantially higher peak FLOPs than CPUs, overall runtime becomes more sensitive to memory traffic. A naïve PyTorch implementation of BDA would allocate several intermediate tensors, forcing multiple round-trips to off-chip GPU memory (HBM), which is much slower than on-chip SRAM. To avoid this overhead, we develop a fused Triton kernel that executes BDA in a single load–compute–store pass, matching the memory-access pattern of the cuBLAS matmul used in MHA.

We evaluate this kernel across several NVIDIA architectures — Ampere (A6000, RTX 3080/3090), Ada (RTX 4090), and Blackwell (RTX 5090) — using the **same** kernel configuration without retuning. Across all GPUs, BDA achieves near-theoretical acceleration (Fig. 3b), averaging **1.34**$\times$ (FP16) and **1.37**$\times$ (BF16), while also reducing parameters and memory by **25%**. The consistency across architectures indicates that the performance gains arise from BD's reduced arithmetic cost rather than device-specific tuning.

These measurements concern the projection operator. Appendix H reports FLOPs-based estimates of the K/V projection share and attention-block speedups for three DeepSeek configurations at context lengths of 128 and 1,024 tokens.

Comparison to PIFA-style attention. For completeness, we also implement a *PIFA-style* variant (not described in original paper (Zhao et al., 2025), but constructed by us for comparison): for each head i, *PIFA-style* variant run QR with column pivoting on $\mathbf{W}_Q^i(\mathbf{W}_K^i)^\top$ to select its own basis. Because each head chooses a different basis, this forces per-head slicing and copying of $\mathbf{X}$, causing the memory traffic to increase by number of heads times. Consequently, PIFA-style attention is even slower than baseline MHA (Table 8, 9). In contrast, BD aligns **all** heads to a shared contiguous

Table 4: **Training evaluation of BD Attention (BDA).** BLEU scores (↑) on the IWSLT'14 using the Transformer model. The columns correspond to the **LR scale**, i.e., the multiplicative factor applied to the learning rate of *Noam* schedule (Vaswani et al., 2017). Across all scales, BDA achieves **comparable** BLEU scores to MHA, despite not guaranteeing identical gradients, and requires no hyperparameter tuning. **Bold** numbers indicate the higher BLEU for each LR scale.

	LR Scale=0.5	LR Scale=1	LR Scale=2	LR Scale=4
MHA	24.98	23.98	23.86	24.04
BDA	**25.27**	**25.04**	**23.89**	**24.14**

basis, so $\mathbf{X}$ is sliced only once and reused across heads, dramatically reducing I/O compared with *PIFA-style* variant.

4.3 BD for Low-Rank Pruning

We evaluate BD when applied on top of models already compressed by low-rank pruning (Appendix I).

Table 3 reports results on LLaMA2-7B and LLaMA2-13B under three settings: (i) **Dense**, the original pretrained LLaMA2 model; (ii) **Low-rank (80% density)**, where weights are pruned into a low-rank structure following Zhao et al. (Zhao et al., 2025); and (iii) **BD (from low-rank)**, where the pruned low-rank weights are further transformed using Basis Decomposition (BD). No retraining is performed in any case.

Results show that BD consistently improves efficiency over the low-rank baseline while preserving perplexity. On average, BD increases throughput by **17.21%** and reduces memory usage by **16.52%** compared to low-rank models, while keeping perplexity nearly unchanged. This demonstrates that BD is complementary to existing compression techniques and can serve as a plug-in acceleration step for low-rank pruned models.

4.4 BDA Training Evaluation

While inference is mathematically lossless, training dynamics can still differ from MHA because gradient updates on the BD parameters are not guaranteed to be equivalent to those on the original projections. To evaluate this, we trained Transformer models on the IWSLT'14 English-to-German (Cettolo et al., 2014) using either standard MHA or BDA as the attention module, both under the *Noam* learning-rate schedule (Vaswani et al., 2017). We swept across four learning-rate scales $(0.5, 1, 2, 4)$. As shown in Table 4, despite potential differences in optimization dynamics, the final BLEU scores of BDA are **comparable** to those of MHA across all settings. All hyperparameters are detailed in Appendix D, and were kept identical between MHA and BDA. This highlights that BDA requires no hyperparameter search or tuning, and thus can be *seamlessly integrated* into existing training pipelines without additional cost, while maintaining model quality.

5 Conclusion and Discussion

We presented **BD Attention (BDA)**, a novel *lossless algorithmic reformulation* of multi-head attention. By applying Basis Decomposition (BD), BDA restructures projection matrices into a compact form that eliminates redundant parameters and arithmetic operations while preserving exact outputs. Our experiments confirmed its practical benefits: near-theoretical speedups for attention projections with reduced projection weight storage, training performance comparable to MHA, and additional efficiency gains when applied to low-rank pruned models. Overall, BDA offers a mathematically exact and versatile foundation for attention projection acceleration that complements existing system-level methods. Future work includes evaluating BDA in large-scale language model pretraining to study its impact on training efficiency and model quality.

AI USE STATEMENT

Generative AI tools were used to polish the grammar of the manuscript. They were not used to develop theoretical models or conceptual frameworks, formulate or prove mathematical claims, propose hypotheses, design experiments, implement the proposed methods, generate or process datasets, translate the manuscript, or interpret results. The authors take full responsibility for the final content of this work, including all text, claims, and artifacts.

REFERENCES

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=vXxardq6db.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

P Businger and GH Golub. Linear least squares solutions by householder transformations. *Handbook for automatic computation*, 2:111–118, 1971.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. Report on the 11th IWSLT evaluation campaign. In Marcello Federico, Sebastian Stüker, and François Yvon (eds.), *Proceedings of the 11th International Workshop on Spoken Language Translation: Evaluation Campaign*, pp. 2–17, Lake Tahoe, California, December 4-5 2014. URL https://aclanthology.org/2014.iwslt-evaluation.1.

Jerry Chee, Anil Damle, Christopher M De Sa, et al. Model preserving compression for neural networks. *Advances in neural information processing systems*, 35:38060–38074, 2022.

Yifan Chen, Qi Zeng, Heng Ji, and Yun Yang. Skyformer: Remodel self-attention with gaussian kernel and nystr\" om method. *Advances in Neural Information Processing Systems*, 34:2122–2135, 2021.

Yifan Chen, Qi Zeng, Dilek Hakkani-Tur, Di Jin, Heng Ji, and Yun Yang. Sketching as a tool for understanding and accelerating self-attention for long sequences. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5187–5199, 2022.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457, 2018.

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

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

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pp. 10323–10337. PMLR, 2023.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. OPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tcbBPnfwxS.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

Habib Hajimolahoseini, Walid Ahmed, Mehdi Rezagholizadeh, Vahid Partovinia, and Yang Liu. Strategies for applying low rank decomposition to transformer-based models. In *36th Conference on Neural Information Processing Systems (NeurIPS2022)*, volume 6, 2022.

Andi Han, Jiaxiang Li, Wei Huang, Mingyi Hong, Akiko Takeda, Pratik Jawanpuria, and Bamdev Mishra. SLTrain: a sparse plus low-rank approach for parameter and memory efficient pretraining. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. Language model compression with weighted low-rank factorization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uPv9Y3gmAI5.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Ajay Jaiswal, Lu Yin, Zhenyu Zhang, Shiwei Liu, Jiawei Zhao, Yuandong Tian, and Zhangyang Wang. From galore to welore: How low-rank weights non-uniformly emerge from low-rank gradients. *arXiv preprint arXiv:2407.11239*, 2024.

Siddhartha Rao Kamalakara, Acyr Locatelli, Bharat Venkitesh, Jimmy Ba, Yarin Gal, and Aidan N Gomez. Exploring low rank training of deep neural networks. *arXiv preprint arXiv:2209.13569*, 2022.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Ayush Kaushal, Tejas Vaidhya, and Irina Rish. Lord: Low rank decomposition of monolingual code llms for one-shot compression. *arXiv preprint arXiv:2309.14021*, 2023.

Mikhail Khodak, Neil A. Tenenholtz, Lester Mackey, and Nicolo Fusi. Initialization and regularization of factorized neural layers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=KTlJT1nof6d.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkgNKkHtvB.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pp. 12888–12900. PMLR, 2022.

Yixiao Li, Yifan Yu, Qingru Zhang, Chen Liang, Pengcheng He, Weizhu Chen, and Tuo Zhao. LoSparse: Structured compression of large language models based on low-rank and sparse approximation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 20336–20350. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/li23ap.html.

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. ReloRA: High-rank training through low-rank updates. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=DLJznSp6X3.

Chi-Heng Lin, Shangqian Gao, James Seale Smith, Abhishek Patel, Shikhar Tuli, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. Modegpt: Modular decomposition for large language model compression, 2024a. URL https://arxiv.org/abs/2408.09632.

Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. Duquant: Distributing outliers via dual transformation makes stronger quantized LLMs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL https://openreview.net/forum?id=mp8u2Pcmqz.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. In P. Gibbons, G. Pekhimenko, and C. De Sa (eds.), *Proceedings of Machine Learning and Systems*, volume 6, pp. 87–100, 2024c. URL https://proceedings.mlsys.org/paper_files/paper/2024/file/42a452cbafa9dd64e9ba4aa95cc1ef21-Paper-Conference.pdf.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024b.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: weight-decomposed low-rank adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024c.

Shih-Yang Liu, Maksim Khadkevich, Nai Chit Fung, Charbel Sakr, Chao-Han Huck Yang, Chien-Yi Wang, Saurav Muralidharan, Hongxu Yin, Kwang-Ting Cheng, Jan Kautz, Yu-Chiang Frank Wang, Pavlo Molchanov, and Min-Hung Chen. Eora: Fine-tuning-free compensation for compressed llm with eigenspace low-rank approximation, 2025. URL https://arxiv.org/abs/2410.21271.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems*, 2023.

Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research*, 18(134):1–35, 2017.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. PiSSA: Principal singular values and singular vectors adaptation of large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=6ZBHIEtdP4.

Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378*, 2021.

Wang Qinsi, Jinghan Ke, Masayoshi Tomizuka, Kurt Keutzer, and Chenfeng Xu. Dobi-svd: Differentiable svd for llm compression and some new perspectives. In *The Thirteenth International Conference on Learning Representations*.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Siyu Ren and Kenny Q. Zhu. Low-rank prune-and-factorize for language model compression. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue (eds.), *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 10822–10832, Torino, Italia, May 2024. ELRA and ICCL. URL https://aclanthology.org/2024.lrec-main.945/.

Rajarshi Saha, Naomi Sagan, Varun Srivastava, Andrea Goldsmith, and Mert Pilanci. Compressing large language models using low rank and low precision decomposition. *Advances in Neural Information Processing Systems*, 37:88981–89018, 2024.

Charbel Sakr and Brucek Khailany. ESPACE: Dimensionality reduction of activations for model compression. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=HAcaANQNMK.

Dayana Savostianova, Emanuele Zangrando, Gianluca Ceruti, and Francesco Tudisco. Robust low-rank training via approximate orthonormal constraints. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=NJPSvv0u3R.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pp. 9355–9366. PMLR, 2021.

Steffen Schotthöfer, Emanuele Zangrando, Jonas Kusch, Gianluca Ceruti, and Francesco Tudisco. Low-rank lottery tickets: finding efficient low-rank neural networks via matrix differential equations. *Advances in Neural Information Processing Systems*, 35:20051–20063, 2022.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 68658–68685. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/7ede97c3e082c6df10a8d6103a2eebd2-Paper-Conference.pdf.

Pratyusha Sharma, Jordan T Ash, and Dipendra Misra. The truth is in there: Improving reasoning in language models with layer-selective rank reduction. *arXiv preprint arXiv:2312.13558*, 2023.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2021.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=PxoFut3dWW.

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

Tycho F. A. van der Ouderaa, Markus Nagel, Mart Van Baalen, and Tijmen Blankevoort. The LLM surgeon. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=DYIIRgwg2i.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Sergey Voronin and Per-Gunnar Martinsson. Efficient algorithms for cur and interpolative matrix decompositions. *Advances in Computational Mathematics*, 43(3):495–516, 2017.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/4496bf24afe7fab6f046bf4923da8de6-Paper.pdf.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. Svd-llm: Truncation-aware singular value decomposition for large language model compression. *arXiv preprint arXiv:2403.07378*, 2024.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Zhihang Yuan, Yuzhang Shang, Yue Song, Qiang Wu, Yan Yan, and Guangyu Sun. Asvd: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*, 2023.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Feiyu Zhang, Liangzhi Li, Junhao Chen, Zhouqiang Jiang, Bowen Wang, and Yiming Qian. Increlora: Incremental parameter allocation method for parameter-efficient fine-tuning, 2023a. URL https://arxiv.org/abs/2308.12043.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=lq62uWRJjiY.

Stephen Zhang and Vardan Papyan. OATS: Outlier-aware pruning through sparse and low rank decomposition. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=DLDuVbxORA.

Yingtao Zhang, Haoli Bai, Haokun Lin, Jialin Zhao, Lu Hou, and Carlo Vittorio Cannistraci. Plug-and-play: An efficient post-training pruning method for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Tr0lPx9woF.

Jialin Zhao, Yingtao Zhang, and Carlo Vittorio Cannistraci. Pivoting factorization: A compact meta low-rank representation of sparsity for efficient inference in large language models. *arXiv preprint arXiv:2501.19090*, 2025.

Jiawei Zhao, Yifei Zhang, Beidi Chen, Florian Schäfer, and Anima Anandkumar. Inrank: Incremental low-rank learning. *arXiv preprint arXiv:2306.11250*, 2023.

Algorithm 3 BD Decomposition (Row)

input $\mathbf{W} \leftarrow \mathbf{U}\mathbf{V}^\top$, $\mathbf{W} \in \mathbb{R}^{m\times n}$ with rank r

1: Extract first-r rows: $\mathbf{B}_F \leftarrow$ first r rows of $\mathbf{W}$
2: Extract last-r rows: $\mathbf{B}_L \leftarrow$ last r rows of $\mathbf{W}$
3: Solve coefficients:
$\mathbf{C}_F \leftarrow \text{linsolve}(\mathbf{W}' \setminus \mathbf{B}_F = \mathbf{C}_F\mathbf{B}_F)$
$\mathbf{C}_L \leftarrow \text{linsolve}(\mathbf{W}' \setminus \mathbf{B}_L = \mathbf{C}_L\mathbf{B}_L)$
4: Compute residuals
$R_F \leftarrow \left\|\mathbf{W} - \begin{bmatrix}\mathbf{B}_F \\ \mathbf{C}_F\mathbf{B}_F\end{bmatrix}\right\|_F$
$R_L \leftarrow \left\|\mathbf{W} - \begin{bmatrix}\mathbf{C}_L\mathbf{B}_L \\ \mathbf{B}_L\end{bmatrix}\right\|_F$
5: **Select better candidate**
if $R_F \le R_L$ **then**
tag$\leftarrow$ FIRST, $\mathbf{B} \leftarrow \mathbf{B}_\text{F},\ \mathbf{C} \leftarrow \mathbf{C}_F$
else
tag$\leftarrow$ LAST, $\mathbf{B} \leftarrow \mathbf{B}_L,\ \mathbf{C} \leftarrow \mathbf{C}_L$

output *tag*, basis matrix $\mathbf{B}$, coefficient matrix $\mathbf{C}$

Algorithm 4 BD Reconstruction (Row)

input *tag*, basis matrix $\mathbf{B} \in \mathbb{R}^{r\times n}$, coefficient matrix $\mathbf{C} \in \mathbb{R}^{(m-r)\times r}$

1: **if** *tag* = FIRST **then** {row-first in Equation 2}

$$\mathbf{W} \leftarrow \begin{bmatrix}\mathbf{B} \\ \mathbf{C}\,\mathbf{B}\end{bmatrix}$$

2: **else** {row-last in Equation 2}

$$\mathbf{W} \leftarrow \begin{bmatrix}\mathbf{C}\,\mathbf{B} \\ \mathbf{B}\end{bmatrix}$$

output Reconstructed matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$

Algorithm 5 BD Attention Preparation (QK). R denotes the residual error of basis decomposition.

input $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v, \mathbf{W}_o$ represent query, key, value and output projection matrix; n be the number of attention heads

1: **for** $i = 1, \ldots, n$ **do**
2: *(first-r)* $R_F^i, \mathbf{B}_F^i, \mathbf{C}_F^i \leftarrow \text{BD}_\text{COL}^\text{FIRST}(\mathbf{W}_q^i {\mathbf{W}_k^i}^\top)$
3: *(last-r)* $R_L^i, \mathbf{B}_L^i, \mathbf{C}_L^i \leftarrow \text{BD}_\text{COL}^\text{LAST}(\mathbf{W}_q^i {\mathbf{W}_k^i}^\top)$
4: **end for**
5: Compute mean residuals: $\bar{R}_F \leftarrow \frac{1}{n}\sum_{i=1}^n R_F^i,\ \bar{R}_L \leftarrow \frac{1}{n}\sum_{i=1}^n R_L^i$
6: **Select better candidate**
if $\bar{R}_F \le \bar{R}_L$ **then**
tag$\leftarrow$ FIRST, $\mathbf{B}_{qk}^i \leftarrow \mathbf{B}_F^i,\ \mathbf{C}_{qk}^i \leftarrow \mathbf{C}_F^i,\ i = 1, \ldots, n$
else
tag$\leftarrow$ LAST, $\mathbf{B}_{qk}^i \leftarrow \mathbf{B}_L^i,\ \mathbf{C}_{qk}^i \leftarrow \mathbf{C}_L^i,\ i = 1, \ldots, n$

output *tag*, column basis matrices $\mathbf{B}_{qk}^i$ (replacement of $\mathbf{W}_q^i$), coefficient matrix $\mathbf{C}_{qk}^i$ (replacement of $\mathbf{W}_k^i$), $i = 1, \ldots, n$

A Proof of Almost Sure Full Rank of Random Matrices Theorem

Theorem 3.1 (Almost Sure Full Rank of Random Matrices)**.** *Let* $\mathbf{W}$ *be an* $r \times r$ *real random matrix. Suppose the entries of* $\mathbf{W}$ *are drawn from a probability measure* μ *on* $\mathbb{R}^{r^2}$ *that is absolutely*

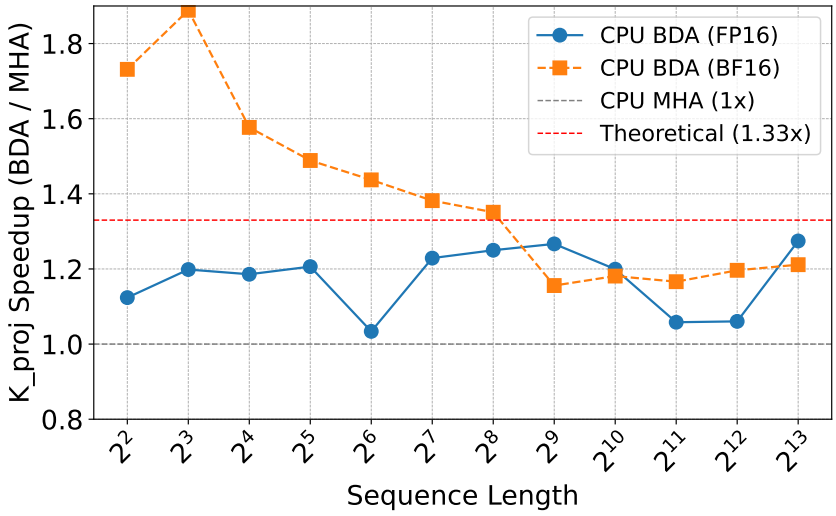


Figure 4: **BD Attention (Standard PyTorch Implementation) CPU Efficiency.**

continuous with respect to the Lebesgue measure λ. Then W has full rank ($\mathrm{rank}(W) = r$) *with probability 1.*

Proof. **Step 1. Define the polynomial map $p(\mathbf{W}) = \det(\mathbf{W})$.**
Let

$$p : \mathbb{R}^{r^2} \to \mathbb{R}, \quad p(\mathbf{W}) = \det(\mathbf{W}),$$

Note that p is a polynomial (in r^2 variables) and it is not the zero polynomial, since, for instance,

$$p(\mathbf{I}_r) = \det(\mathbf{I}_r) = 1,$$

where $\mathbf{I}_r$ denotes the $r \times r$ identity matrix.

Step 2. The zero set of a nontrivial polynomial has measure zero.
Consider

$$Z = \{ x \in \mathbb{R}^{r^2} \mid p(x) = 0 \}.$$

Since p is a nontrivial real polynomial, its zero set Z is a real algebraic variety of dimension $< r^2$, and hence we know

$$\lambda(Z) = 0.$$

(In simple terms, Z is a "lower-dimensional" subset of $\mathbb{R}^{r^2}$.)

Step 3. Absolute continuity of μ and conclusion.
By hypothesis, the measure μ (the distribution of $\mathbf{W}$) is absolutely continuous with respect to the Lebesgue measure λ, i.e. $\mu \ll \lambda$. Hence for any Lebesgue null set A, we have $\mu(A) = 0$. In particular, Z has $\lambda(Z) = 0$, so $\mu(Z) = 0$. But Z exactly corresponds to the event $\{\det(\mathbf{W}) = 0\}$ in the space of all matrix entries. Therefore

$$\Pr(\det(\mathbf{W}) = 0) = \mu(Z) = 0,$$

which implies $\Pr(\det(\mathbf{W}) \neq 0) = 1$, i.e. $\Pr(\mathrm{rank}(\mathbf{W}) = r) = 1$. □

B Relation to Interpolative Decomposition

BD is algebraically a special case of interpolative decomposition (ID) and CUR skeleton factorization (Equation 2) (Voronin & Martinsson, 2017). ID-based representations have also been studied for neural-network compression (Chee et al., 2022). Our application exploits the low-rank structure of the per-head products $\mathbf{W}_q \mathbf{W}_k^\top$ and $\mathbf{W}_v \mathbf{W}_o$, whose ranks are at most d_h. With valid bases,

BD represents these products exactly in exact arithmetic without additional rank truncation. The analysis in Section 3.2 establishes almost-sure basis validity under the stated joint absolute-continuity model. In our implementation, we align contiguous basis indices across heads to support regular memory access and batched computation. Contiguous selection is an implementation choice; the decomposition itself permits any valid row or column basis.

C BD FOR VO IN MHA

Here, we introduce the BD transformation for value and output projection matrices. The right part of Equation 6 can be converted to (using row-first in Equation 2):

$$\mathbf{V}_i\mathbf{W}_o^i = \mathbf{X}(\mathbf{W}_v^i\mathbf{W}_o^i) = \mathbf{X}(\begin{bmatrix}\mathbf{I}\\ \mathbf{C}_{vo}^i\end{bmatrix}\mathbf{B}_{vo}^i) = (\mathbf{X}_{:,\,1:d_h} + \mathbf{X}_{:,\,d_h:d}\mathbf{C}_{vo}^i)\mathbf{B}_{vo}^i \tag{10}$$

where $\mathbf{B}_{vo}^i \in \mathbb{R}^{d_h\times d}$ be the first-d_h row basis and $\mathbf{C}_{vo}^i \in \mathbb{R}^{(d-d_h)\times d_h}$ the coefficient matrix. Similar to Equation 8, we can redefine value and output projection matrices to avoid calculating each head separately:

$$\begin{aligned}
&\mathbf{V}' = [\mathbf{X}_{:,\,1:d_h}]^{\times n} + \mathbf{X}_{:,\,d_h:d}\mathbf{C}_{vo}, \quad \text{where } \mathbf{C}_{vo} := [\mathbf{C}_{vo}^1, \ldots, \mathbf{C}_{vo}^n]\\
&[\mathbf{V}'_1, \ldots, \mathbf{V}'_n] = \mathbf{V}'\\
&\mathbf{O}'_i = \text{softmax}(\frac{\mathbf{Q}'_i{\mathbf{K}'_i}^\top}{\sqrt{d_h}})\mathbf{V}'_i\\
&\mathbf{Y} = [\mathbf{O}'_1, \ldots, \mathbf{O}'_n]\mathbf{B}_{vo}, \quad \text{where } \mathbf{B}_{vo} := \begin{bmatrix}\mathbf{B}_{vo}^1\\ \vdots\\ \mathbf{B}_{vo}^n\end{bmatrix}
\end{aligned} \tag{11}$$

where $[\mathbf{X}_{:,\,1:d_h}]^{\times n}$ means repeat matrix n times along second dimension. The *last* version is similar.

D HYPERPARAMETERS FOR BDA TRAINING

For the IWSLT'14 English-to-German task, we followed the standard Transformer setup with the *Noam* learning-rate schedule (Vaswani et al., 2017). The embedding dimension was set to 512, with a feed-forward dimension of 2048. We used a batch size of 10,240 tokens and trained for 20,000 steps. Dropout was 0.1 and label smoothing was 0.1. The model contained 6 layers with 4 attention heads. For simplicity, we removed positional embeddings inside the MHA module but retained them in the embedding layer (the effect of positional embeddings on BD is discussed in Appendix E). The learning rate scale was varied in $\{0.5, 1, 2, 4\}$ (Section 4.4). Training employed the Adam optimizer with 6,000 warmup steps. Beam search with a beam size of 2 was used for evaluation, and BLEU scores were reported using the checkpoint with the lowest validation perplexity.

All hyperparameters for BDA were kept **identical** to those of MHA to ensure a fair comparison. Moreover, this also shows that BDA naturally matches the training dynamics of MHA, achieving comparable performance without any hyperparameter tuning. As a result, existing LLM training pipelines can migrate from MHA to BDA at essentially no cost.

E EFFECT OF POSITIONAL EMBEDDING ON BD

We briefly discuss how positional embeddings interact with Basis Decomposition (BD).

Embedding-layer positional embedding. Any positional embedding applied at the embedding layer (e.g., learned positional embeddings or sinusoidal embeddings added to input tokens) does not affect BD. Since BD only restructures the projection matrices inside the attention, the addition of position-dependent vectors at the input embedding layer is orthogonal to BD's reformulation.

MHA-internal positional embedding. When positional embeddings are applied inside the multi-head attention (MHA) module, the formulation changes. For example, vanilla Rotary Position Embedding (RoPE) (Su et al., 2021) modifies the attention score computation as

$$\mathbf{X}_n \mathbf{W}_q \, \mathbf{W}_k^\top \mathbf{X}_m^\top \quad \longrightarrow \quad \mathbf{X}_n \mathbf{W}_q \, \mathbf{R}_{n-m} \, \mathbf{W}_k^\top \mathbf{X}_m^\top$$

where $\mathbf{R}_{n-m}$ is a rotation matrix depending on relative positions $n - m$. BD guarantees the exact factorization

$$\mathbf{W}_q \mathbf{W}_k^\top = \mathbf{B}[\mathbf{I}, \mathbf{C}],$$

but in general BD cannot guarantee

$$\mathbf{W}_q \mathbf{R}_{n-m} \mathbf{W}_k^\top = \mathbf{B}\mathbf{R}_{n-m}[\mathbf{I}, \mathbf{C}].$$

Thus, vanilla RoPE breaks the exactness of BD.

Decoupled RoPE as a solution. DeepSeek proposes *Decoupled RoPE* (Liu et al., 2024a), which splits attention channels into RoPE and non-RoPE parts. BD can then be applied to the non-RoPE channels, while RoPE channels remain unchanged. This is also the strategy used in our experiments.

BD under MLA weight absorption. MLA can move the key up-projection to the query side and the value up-projection after aggregation of the cached latent vectors (Liu et al., 2024a). Using our row-vector convention and omitting the head index, let $\mathbf{u} \in \mathbb{R}^{d_q}$ be the input to the query projection and $\mathbf{c}_s \in \mathbb{R}^{d_c}$ a cached KV latent vector. The non-RoPE score is $\mathbf{u}\mathbf{A}\mathbf{c}_s^\top$, where $\mathbf{A} = \mathbf{W}_q \mathbf{W}_k^\top$. To place the identity block on the query-input side, we can use row-based BD, rather than the column-based form used for explicit key projection:

$$\mathbf{A} = \begin{bmatrix} \mathbf{I} \\ \mathbf{C}_{qk} \end{bmatrix} \mathbf{B}_{qk}, \qquad \widetilde{\mathbf{q}} = \mathbf{u}\mathbf{A} = \big(\mathbf{u}_{1:d_h} + \mathbf{u}_{d_h+1:d_q} \mathbf{C}_{qk}\big)\mathbf{B}_{qk}.$$

Here $\mathbf{B}_{qk} \in \mathbb{R}^{d_h \times d_c}$ contains the first d_h rows of $\mathbf{A}$, assumed to form a valid basis, and $\mathbf{C}_{qk} \in \mathbb{R}^{(d_q - d_h) \times d_h}$ contains the reconstruction coefficients. Similarly, with $\mathbf{z} = \sum_s a_s \mathbf{c}_s$ denoting the attention-weighted latent vector, row-based BD of the absorbed value-output product gives

$$\mathbf{D} = \mathbf{W}_v \mathbf{W}_o = \begin{bmatrix} \mathbf{I} \\ \mathbf{C}_{vo} \end{bmatrix} \mathbf{B}_{vo}, \qquad \mathbf{z}\mathbf{D} = \big(\mathbf{z}_{1:d_h} + \mathbf{z}_{d_h+1:d_c} \mathbf{C}_{vo}\big)\mathbf{B}_{vo},$$

where $\mathbf{B}_{vo}$ contains a valid basis formed by the first d_h rows of $\mathbf{D}$; last-row variants follow analogously. These identities preserve the scores and outputs in exact arithmetic while leaving the latent KV cache and decoupled RoPE branch unchanged. The optimization acts on the query and output projections; the latent-space attention products are unchanged.

Model-specific implications.

- **GPT models** use positional embeddings only at the input embedding layer; thus BD is fully lossless for both QK and VO.
- **LLaMA models** adopt vanilla RoPE inside MHA. Since RoPE is applied only to QK, BD remains lossless for VO projections but is not exact for QK.
- **DeepSeek models** employ Decoupled RoPE, which separates RoPE and non-RoPE channels in QK. BD can be applied losslessly to the non-RoPE channels of QK, and to all VO projections.

In summary, the compatibility of BD with positional embeddings depends on how positional information is integrated. Embedding-layer positional encodings pose no issue, while RoPE inside MHA requires modifications (e.g., Decoupled RoPE).

F Zero-shot Evaluation

To provide a more comprehensive evaluation, we additionally report zero-shot downstream performance to complement the perplexity results in the main paper and to assess whether the lossless property of BD is preserved in downstream tasks.

Table 5: **Numerical reconstruction errors of BD** for $\mathbf{W}_q\mathbf{W}_k^\top$ (QK) and $\mathbf{W}_v\mathbf{W}_o$ (VO) under different floating-point formats. Values are averaged across all heads and all layers of the DeepSeek-V2-Lite model. We compare two strategies for selecting the BD basis: (i) always using the first r rows (*First-r*), and (ii) choosing between the first or last r rows depending on which yields the smaller residual (*Residual-min*). We report absolute mean squared error (MSE) and normalized mean squared error (NMSE). The results confirm that BD introduces only **negligible** perturbations to the matrix products, with *Residual-min* consistently outperforming *First-r*, and improving error by at least one order of magnitude in FP32.

		FP32	FP16	BF16
QK MSE	First-r	3.19×10^{-12}	1.09×10^{-7}	7.42×10^{-7}
	Residual-min	3.12×10^{-13}	7.51×10^{-8}	6.51×10^{-7}
QK NMSE	First-r	5.74×10^{-9}	3.20×10^{-4}	2.07×10^{-3}
	Residual-min	7.10×10^{-10}	2.36×10^{-4}	1.88×10^{-3}
VO MSE	First-r	2.45×10^{-12}	1.02×10^{-8}	8.91×10^{-8}
	Residual-min	2.15×10^{-14}	5.97×10^{-9}	8.19×10^{-8}
VO NMSE	First-r	1.26×10^{-7}	2.71×10^{-4}	2.28×10^{-3}
	Residual-min	8.31×10^{-10}	1.61×10^{-4}	2.06×10^{-3}

Table 6: **Perplexity of BD Attention on WikiText2**. All MHA layers in the DeepSeek-V2-Lite model are replaced with BD Attention, and we report perplexity (↓) under different floating-point formats. For BD, we show results using (i) always the first r rows (*First-r*) and (ii) selecting between the first or last r rows based on the smaller residual (*Residual-min*). The last row reports the *relative increase in perplexity (PPL)*, computed as $(\mathrm{PPL}_{\mathrm{BD}} - \mathrm{PPL}_{\mathrm{Original}})/\mathrm{PPL}_{\mathrm{Original}}$. Across all settings, BD introduces only **negligible** changes in model performance, with *Residual-min* consistently yielding smaller increases. Preparation time is very short (a few seconds), making BD efficient for deployment.

		FP32	FP16	BF16
Original PPL		6.306983	6.307075	6.310289
BD PPL	First-r	6.307082	6.309186	6.326459
	Residual-min	6.307007	6.308252	6.325656
PPL Increase (relative)	First-r	0.002%	0.033%	0.256%
	Residual-min	0.0004%	0.019%	0.244%
Preparation Time (s)	First-r	3.56	1.89	2.39
	Residual-min	6.09	4.05	4.05

Experimental Setup. We evaluate zero-shot performance on **DeepSeek-V2-Lite**, comparing the original MHA model (FP16) with its BD-based counterpart (BDA, FP16). All other settings are kept identical between MHA and BDA. We follow the standard zero-shot evaluation protocol and report mean accuracy and standard error, including ARC Easy and Challenge (Clark et al., 2018), SuperGLUE (Wang et al., 2019), and HellaSwag (Zellers et al., 2019). Evaluations are conducted using the LM Evaluation Harness framework (Gao et al., 2023).

Discussion. Across all evaluated tasks, the performance differences between MHA and BDA remain well within one standard error. This behavior is consistent with the theoretical guarantee that BD is a lossless decomposition, as well as with the near-identical FP16 perplexity observed in language modeling. These results confirm that BD preserves downstream zero-shot performance in practice.

G Residual-Min with Offset Search

Theorem 3.1 guarantees that any d_h rows or columns form a valid basis with probability 1, but in finite precision the resulting $d_h \times d_h$ sub-blocks differ in conditioning: ill-conditioned blocks amplify

Table 7: **Zero-shot accuracy** (mean ± std) on DeepSeek-V2-Lite. Performance differences between MHA and BDA are consistently within one standard error across all tasks.

Task	MHA (FP16)	BDA (FP16)
BoolQ	0.8021 ± 0.0070	0.8024 ± 0.0070
MultiRC	0.5598 ± 0.0071	0.5611 ± 0.0071
RTE	0.6065 ± 0.0294	0.6101 ± 0.0294
WIC	0.4984 ± 0.0198	0.4984 ± 0.0198
WSC	0.3654 ± 0.0474	0.3654 ± 0.0474
HellaSwag	0.7779 ± 0.0041	0.7785 ± 0.0041
ARC-Easy	0.7433 ± 0.0090	0.7412 ± 0.0090
ARC-Challenge	0.4590 ± 0.0146	0.4599 ± 0.0146

Table 8: **Throughput comparison on FP16, NVIDIA A6000**. We report throughput in million tokens per second (higher is better) for a single attention operator, comparing MHA, PIFA-style Attention, and BDA across different sequence lengths. The tested matrix shape follows the DeepSeek-V3 configuration (Liu et al., 2024b), with $n = 128$ heads, $d = 512$ (corresponding to d_c in DeepSeek-V3), and $d_h = 128$ (same as d_h in DeepSeek-V3). The last column reports the *relative speedup*, defined as BDA throughput divided by MHA throughput.

Seq. Len	MHA	PIFA-style (per-head QR)	BDA	Speedup
64	1.79	0.99	2.16	1.21×
128	3.13	1.30	3.79	1.21×
256	4.46	1.52	5.43	1.22×
512	4.95	1.51	7.04	1.42×
1024	5.62	1.69	7.87	1.40×
2048	5.95	1.72	8.03	1.35×
4096	5.59	1.72	7.71	1.38×
8192	5.58	1.74	7.47	1.34×
16384	5.51	1.74	7.31	1.33×
32768	5.43	1.74	7.17	1.32×
65536	5.41	1.72	7.06	1.30×

rounding error during basis inversion. The Residual-min strategy in the main paper mitigates this by selecting between the first-d_h and last-d_h candidates based on Frobenius reconstruction residual.

We optionally extend this search to contiguous windows of size d_h whose starting offsets are multiples of 16—a constraint that preserves the memory-coalesced access required for the fused kernel—and select the window with the smallest Frobenius residual. We refer to this variant as **Residual-min (offset search)**.

End-to-end perplexity. On DeepSeek-V2-Lite, replacing all MHA layers with BDA using Residual-min (offset search) reduces the FP16 PPL increase relative to the original model from 2×10^{-4} (Residual-min between {first, last}) to 2×10^{-5}, an order-of-magnitude improvement.

Operator throughput cost. At the operator level (DeepSeek-V3 attention shape, FP16, A6000), Residual-min (offset search) averages $1.27\times$ MHA throughput versus $1.31\times$ for Residual-min {first, last}, i.e. a $\sim 3\%$ throughput cost relative to the standard variant.

H FLOPs-Based Estimates of Attention Speedups

Using the original configurations of DeepSeek-V2-Lite (16B), DeepSeek-V2 (236B), and DeepSeek-V3 (671B), we estimate the computational effect of BD beyond the optimized K/V projections. Table 10 considers single-token decoding with explicitly expanded and cached keys and values. The entries are FLOPs-based estimates, not measured latency or throughput improvements.

Table 9: **Throughput comparison on BF16, NVIDIA A6000**. Setup and notation are the same as Table 8.

Seq. Len	MHA	PIFA-style (per-head QR)	BDA	Speedup
64	1.74	0.98	2.16	1.24×
128	3.13	1.26	3.79	1.21×
256	4.46	1.52	5.56	1.24×
512	4.95	1.50	7.14	1.44×
1024	5.62	1.72	8.06	1.44×
2048	5.62	1.72	8.13	1.45×
4096	5.59	1.73	7.89	1.41×
8192	5.61	1.74	7.60	1.35×
16384	5.52	1.74	7.34	1.33×
32768	5.50	1.74	7.34	1.34×
65536	5.51	1.72	7.22	1.31×

Table 10: **K/V projection shares and attention speedup estimates based on FLOPs.**

Model	Context	K/V share of attention (%)	Attention speedup (%)
V2-Lite 16B	128	14.55	3.77
V2 236B	128	10.86	2.79
V3 671B	128	8.72	2.23
V2-Lite 16B	1,024	11.03	2.84
V2 236B	1,024	8.78	2.24
V3 671B	1,024	7.32	1.87

We count the dominant matrix-multiplication FLOPs, with two FLOPs per multiply-add, and apply BD to both non-RoPE QK and VO products. The attention count includes all projections and attention matrix products. The K/V share is the fraction of baseline attention FLOPs spent on the key and value up-projections. The relative speedup decreases with context length because BD saves a fixed amount of projection computation per token, while the unchanged attention computation grows with context length.

I BD FOR LINEAR LAYER

The linear layer is the most common component in neural networks. Given an input vector $\mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$, a standard linear layer computes the output $\mathbf{y} \in \mathbb{R}^{d_{\text{out}}}$ using a weight matrix $\overline{\mathbf{W}} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$:

$$\mathbf{y} = \mathbf{x}\overline{\mathbf{W}}. \tag{12}$$

To reduce parameter count and computational cost, many recent works adopt low-rank approximations of the weight matrix. A common approach is to factorize $\overline{\mathbf{W}}$ as $\overline{\mathbf{W}} \approx \mathbf{U}\mathbf{V}^\top$, where $\mathbf{U} \in \mathbb{R}^{d_{\text{in}} \times r}$ and $\mathbf{V} \in \mathbb{R}^{d_{\text{out}} \times r}$ with $r < \min(d_{\text{in}}, d_{\text{out}})$. The resulting low-rank layer becomes:

$$\mathbf{y} = (\mathbf{x}\mathbf{U})\mathbf{V}^\top, \tag{13}$$

which reduces both the number of parameters from $d_{\text{in}}d_{\text{out}}$ to $r(d_{\text{in}} + d_{\text{out}})$.

Such low-rank structures appear in various domains in deep learning: (1) **low-rank pruning**, where pretrained weight matrices are compressed post hoc via SVD-like approximations (Hsu et al., 2022; Yuan et al., 2023; Wang et al., 2024; Zhao et al., 2025; Jaiswal et al., 2024; Saha et al., 2024; Kaushal et al., 2023; Sharma et al., 2023; Qinsi et al.; Liu et al., 2025; Sakr & Khailany, 2024; Ren & Zhu, 2024; Lin et al., 2024a; Hajimolahoseini et al., 2022); (2) **low-rank training**, where the weight matrices are parameterized as low-rank products throughout training (Khodak et al., 2021; Schotthöfer et al., 2022; Kamalakara et al., 2022; Zhao et al., 2023; Savostianova et al., 2023); (3) **low-rank + sparse hybridization**, which combines sparsity and low-rank approximations for improved performance (Li et al., 2023; Han et al., 2024; Zhang & Papyan, 2025); and (4) **LoRA-style**

fine-tuning (Hu et al., 2022; Zhang et al., 2023b; Lialin et al., 2024; Meng et al., 2024; Liu et al., 2024c; Zhang et al., 2023a), where a low-rank adaptation is injected into frozen models for efficient parameter updates.

Since Basis Decomposition (BD) operates directly on the product $\mathbf{U}\mathbf{V}^\top$, it can be seamlessly applied to all these cases in a lossless and hardware-efficient manner.

To replace the low-rank linear layer with a BD layer, we decompose the weight product $\mathbf{W} = \mathbf{U}\mathbf{V}^\top$ using column-based BD. Let $\mathbf{B} \in \mathbb{R}^{d_{\text{in}} \times r}$ be the first-r column and $\mathbf{C} \in \mathbb{R}^{r \times (d_{\text{out}} - r)}$ the coefficient matrix. The BD layer computes the output in two steps:

$$\mathbf{h} \leftarrow \mathbf{x}\mathbf{B}, \quad \mathbf{y} \leftarrow [\mathbf{h}, \mathbf{h}\mathbf{C}]. \tag{14}$$

The last-r version is similar. For any $r < \min(d_{\text{in}}, d_{\text{out}})$, BD achieves strictly lower FLOPs and memory cost than the original low-rank layer, reduced by $\frac{r}{d_{\text{in}} + d_{\text{out}}}$ relative to the original.

J Preservation of Query–Key Similarity

The transformed projections $\mathbf{Q}'$ and $\mathbf{K}'$ satisfy $\mathbf{Q}'_i \mathbf{K}'^\top_i = \mathbf{Q}_i \mathbf{K}_i^\top$, which means that every pairwise inner product between queries and keys is exactly preserved. $\mathbf{Q}'_i$ and $\mathbf{K}'_i$ can be regarded as an *inner-product isomorphic* representation of the original $\mathbf{Q}_i, \mathbf{K}_i$ in a d_h-dimensional space. We therefore still denote them as queries and keys ($\mathbf{Q}'_i, \mathbf{K}'_i$) to emphasize that they still maintain the essential property of attention: query–key similarity. Advanced compression methods relying on query–key similarity, such as KV-cache compression, remain fully compatible with BDA. Since the inner products are exactly preserved, these methods can be seamlessly integrated with BDA, enabling it to serve as a general and complementary acceleration framework.