

POLITECNICO DI TORINO
Master's Degree in Computer Engineering



**Politecnico
di Torino**

Master's Degree Thesis

**Post Training Low Rank Approximation
for KV Cache Compression in Large
Language Models**

Supervisors

Prof. Daniele Jahier PAGLIARI

Dr. Alessio BURRELLO

Luca BENFENATI

Candidate

Andrea VANNOZZI

Academic Year 2025-2026

Abstract

Autoregressive decoding in large Transformer language models is frequently memory-bandwidth bound: each generated token reads and updates an ever-growing key-value (KV) cache. As context length and batch size increase, KV-cache storage and traffic can dominate inference cost, limiting throughput and deployment scalability. A common mitigation is to compress the KV cache by projecting per-head key and value matrices to a lower rank and storing only the projections. However, many post-training projection methods optimize proxy objectives, such as variance preservation or reconstruction fidelity, which do not explicitly account for the end-to-end behavior of the decoder during generation.

This thesis studies KV-cache compression from a functional perspective: can projection bases optimized to preserve decoder-layer outputs yield better memory-performance trade-offs than proxy-driven approximations? We propose a post-training framework in which lightweight predictors, trained offline on a calibration set while keeping the language model frozen, output orthonormal projection bases for keys and values. Rather than minimizing the reconstruction error on cached tensors, the training objective directly minimizes the discrepancy between full-rank and compressed decoder-layer output, aligning compression with the quantity that governs downstream generation. To enable controllable compression, we precompute per-layer error surfaces over candidate ranks and select layer-wise ranks under a user-specified budget, allowing different memory footprints without retraining bases.

We evaluated the Llama-3-8B decoder architecture using perplexity on WikiText-2 and C4 and zero-shot accuracy on PIQA, HellaSwag, and MMLU. Under matched KV-cache footprints, the proposed method improves end-to-end performance relative to EigenAttention, including a 11.9 point reduction in C4 perplexity and a 5.4 point increase in MMLU accuracy at iso-compression. Layer-wise diagnostics further show that, while SVD-style methods can better reconstruct intermediate attention outputs, optimizing decoder-layer outputs improves both magnitude error and directional alignment of layer activations, with the largest gains appearing in early layers.

From a systems perspective, reducing cached dimensionality reduces memory traffic during decode, the regime typically limiting throughput, and remains compatible with static deployment pipelines because compression is fixed after offline calibration. We also delineate validity boundaries: performance depends on the calibration distribution, degrades under extreme compression, and our evaluation is restricted to a single model family and context configuration. Overall, this work frames KV-cache compression as a problem of functional alignment between projection

geometry and decoder behavior, bridging the representation-level approximation with downstream performance.

Acknowledgements

Table of Contents

List of Tables	VI
List of Figures	VII
Acronyms	X
1 Introduction	1
2 Background	7
2.1 Transformer Decoder and Self-Attention	7
2.2 Autoregressive Inference and KV-Caching	10
2.3 Computational and Memory Complexity of Attention	13
2.4 Linear Low-Rank Approximation	15
3 Related Work	18
3.1 Representation-Level Low-Rank Compression (Post-Training) . . .	19
3.2 Architectural and Quantization-Adjacent Directions	21
3.3 Systems and Runtime KV-Cache Optimization	24
3.4 Inter-Layer Redundancy and Sharing	27
3.5 Common Limitations of Existing Approaches	30
4 Method	33
4.1 Motivation and Preliminaries	33
4.2 Problem Formulation	34
4.2.1 Target Objective	35
4.2.2 Practical Optimization Approximation	35
4.2.3 Limitations of PCA-Based Compression	36
4.2.4 Design Goals	37
4.3 Neural Projection Network	38
4.3.1 Input Features	38
4.3.2 Encoder Architecture	39

4.3.3	Projection Head	39
4.4	Orthonormality Constraint	39
4.5	Training Strategy	40
4.5.1	Loss Function	40
4.5.2	Offline Calibration Protocol	41
4.5.3	Empirical Protocol Decisions	42
5	Experimental Setup	44
5.1	Model	44
5.2	Calibration and Training	45
5.2.1	Calibration Data	45
5.2.2	Rank Configurations	45
5.2.3	Training Protocol	45
5.2.4	Optimization Hyperparameters	46
5.2.5	Loss Function	47
5.3	Datasets	47
5.3.1	Language Modeling Evaluation	47
5.3.2	Zero-Shot Evaluation	47
5.4	Metrics	48
5.4.1	KV-Cache Compression Ratio	48
5.4.2	Perplexity	48
5.4.3	Zero-Shot Accuracy	48
5.4.4	Reconstruction Metrics	49
5.5	Baseline	49
5.5.1	EigenAttention Method	49
5.5.2	Fair Comparison Protocol	50
5.5.3	Uncompressed Baseline	50
5.6	Rank Allocation Policies	50
5.6.1	Uniform Policy	51
5.6.2	Pareto Policy	51
5.6.3	Weighted Pareto Policy	51
5.7	Hardware Setup and Reproducibility Details	52
6	Results	53
6.1	Direct Comparison	53
6.1.1	Comparison Scope and Protocol	53
6.1.2	Matched Compression (Iso-Footprint)	54
6.1.3	Matched Rank (Iso-Rank) Perspective	55
6.2	Robustness	56
6.2.1	Robustness to Error Budget	56
6.2.2	Cross-Dataset and Cross-Task Robustness	58

6.2.3	Validity Boundaries	59
6.3	Ablation Studies	60
6.3.1	Rank-Allocation Policy Ablation	60
6.3.2	Decoder-Output Target vs Proxy Reconstruction	61
6.3.3	Joint Versus Per-Head Value Training	62
6.3.4	Batch-Size Ablation for Key and Value Predictors	63
6.3.5	Fixed-Rank Sensitivity and Calibration Coverage	63
6.4	Per-Layer Analysis	65
6.4.1	Depth-Wise Rank Distribution	65
6.4.2	Layer-Level Output Preservation Diagnostics	65
6.4.3	Global Error Profile	66
6.5	Summary	67
7	Discussion	68
7.1	Why the Method Works	68
7.2	Observed Limitations	69
7.3	System-Level Implications	70
8	Future Works	72
8.1	Integration with Quantization	72
8.2	Dynamic and Context-Adaptive Rank Allocation	73
8.3	Architectural Extensions of Projection Predictors	73
8.4	Theoretical Foundations of Output-Preservation	74
9	Conclusion	75
	Bibliography	77

List of Tables

2.1	Notation introduced in Section 2.1.	10
5.1	Architectural configuration of Llama-3-8B relevant to KV-cache compression.	44
5.2	Training hyperparameters for basis predictor optimization.	46
5.3	Summary of experimental setup parameters.	52
6.1	End-to-end performance of the proposed neural projection method on Llama-3-8B as a function of the error budget ϵ . For each ϵ , layer-wise ranks are selected using precomputed error surfaces, inducing different KV-cache compression ratios.	54
6.2	Layer-wise optimal rank configurations and associated error statistics. For each decoder layer, the table reports the best-performing (r_K, r_V) pair selected under the error-budget criterion, together with the resulting compression ratio and the range of observed reconstruction errors across candidate ranks.	57
6.3	Exploratory end-to-end perplexity comparison at matched manual compression levels during development-phase joint value-head training (sequence length 1024).	62
6.4	Exploratory batch-size ablation from weekly development reports at sequence length 1024. Each block reports results for the same error budget ϵ used to select layer-wise ranks.	64
6.5	Exploratory fixed-rank sweep (our method) from repository reports at sequence length 1024.	64

List of Figures

2.1	Transformer architecture [8].	9
2.2	Illustration of the key-value caching mechanism [12]	12
2.3	Singular Value Decomposition (SVD) of a matrix [13].	16
3.1	Comparison between (a) Standard Attention and (b) Eigen Attention. Eigen Attention utilizes lower dimensional ($r \ll d$) query, key, and value projection matrices than the standard attention operation, leading to KV cache compression and compute FLOPs benefits. [6]	20
3.2	High-level overview of A3. A3 performs low-rank approximation on QK, OV, and MLP components, reducing head dimensions in QK and OV and the intermediate dimension in MLP. [17]	21
3.3	Illustration of the architecture of DeepSeek-V2. MLA ensures efficient inference by significantly reducing the KV cache for generation, and DeepSeekMoE enables training strong models at an economical cost through the sparse architecture.[18]	23
3.4	Diagram of the SVDq method (path in green) versus direct per-channel quantization (dashed path in violet). [21]	23
3.5	Illustration of a block-based KV-cache layout, where attention key and value vectors are stored as non-contiguous memory blocks [30].	25
3.6	Traditional eviction algorithms perform step-by-step greedy search for tokens for eviction. NACL framework searches globally for tokens within a chunk and then performs one single eviction [27].	25
3.7	Illustration of the xKV for compressing KV-Cache [32].	28
4.1	Overview of the Architecture. From lightweight activation statistics μ_K , μ_V and σ_K^2 , σ_V^2 , orthonormal projection bases $\mathbf{P}_K$, $\mathbf{P}_V$ are learned to minimize decoder-layer output error.	38

6.1	Perplexity–memory tradeoff under KV-cache compression on WikiText-2 and C4. Each point corresponds to an error budget ϵ determining layer-wise ranks; the x-axis reports the resulting KV-cache ratio (lower is better).	55
6.2	Zero-shot accuracy–memory tradeoff under KV-cache compression on HellaSwag, PIQA, and MMLU. Each point corresponds to an error budget ϵ determining layer-wise ranks; the x-axis reports the resulting KV-cache ratio (lower is better).	55
6.3	Effect of rank-allocation strategies on the compression–quality frontier. Adaptive policies outperform uniform allocation by redistributing retained rank according to layer-wise sensitivity.	60
6.4	Layer-wise rank profiles under the Weighted Pareto allocation policy across different error budgets ϵ . The profile highlights the non-uniform depth-wise distribution of retained rank.	61
6.5	Training-loss comparison for Key and Value projection predictors across layers 0, 15, and 31 in the development regime (<code>seq_len=1024</code>). Red denotes batch size 1 and green denotes batch size 8; both epoch-level and train-step-level trajectories are shown.	63
6.6	Representative reconstruction-error heatmaps derived from local rank–grid experiments for three decoder layers (0, 15, and 31). Each map visualizes relative decoder-output reconstruction error across candidate rank pairs (r_K, r_V) . These visualizations complement the paper-reported diagnostics by illustrating the heterogeneous sensitivity of layers to dimensionality reduction.	66

Acronyms

LLM

Large Language Models

GPT

Generative Pre-trained Transformer

MHA

Multi-Head Attention

GQA

Grouped Query Attention

MLA

Multi-head Latent Attention

MoE

Mixture-of-Experts

FFN

Feed-Forward Network

MLP

Multi-Layer Perceptron

KV

Key-Value

RoPE

Rotary Position Embeddings

RMSNorm

Root Mean Square Layer Normalization

GELU

Gaussian Error Linear Unit

SVD

Singular Value Decomposition

PCA

Principal Component Analysis

QR

QR Decomposition

GPU

Graphics Processing Unit

VRAM

Video Random Access Memory

HBM

High-Bandwidth Memory

SRAM

Static Random-Access Memory

IO

Input/Output

FP16

Half-precision floating-point format

PPL

Perplexity

LCE

Label Cross-Entropy

MMLU

Massive Multitask Language Understanding

PIQA

Physical Interaction QA

C4

Colossal Clean Crawled Corpus

Chapter 1

Introduction

The past decade has witnessed a transformation in artificial intelligence, driven primarily by the emergence and rapid scaling of large language models. What began as modest experiments with neural architectures for sequence processing has evolved into systems capable of generating coherent text, engaging in complex reasoning, and assisting with tasks across a broad range of domains. This trajectory—from GPT-2’s 1.5 billion parameters in 2019 [1] to models exceeding hundreds of billions of parameters today—represents not merely an increase in scale but a shift in what machine learning systems can accomplish.

The scaling laws governing these models have shown consistent trends within studied regimes: larger models trained on more data exhibit capabilities not observed at smaller scales [2]. This empirical observation has driven substantial investment in training infrastructure. The training phase, while computationally demanding, is a one-time cost amortized across the model’s entire deployment lifetime. A single training run, however expensive, yields a checkpoint that can serve millions of users over months or years. Training is compute-bound—the primary constraint is raw arithmetic throughput, with accelerators saturated by dense matrix multiplications that fully utilize their computational capacity.

Deployment operates under entirely different constraints. When a model generates text, it does so in two distinct phases. During *prefill*, the system processes the input prompt in parallel, a phase that remains relatively compute-intensive. During *decode*, the model generates output tokens one at a time, and each step requires accessing not only the model’s parameters but also the accumulated context from all preceding tokens. This autoregressive pattern transforms the bottleneck from computation to memory bandwidth. The arithmetic intensity—the ratio of computation to memory access—drops precipitously during decode. Hardware designed for the dense operations of training finds its computational cores underutilized, constrained by how quickly data can stream from memory.

At the center of this memory challenge lies the key-value cache. In self-attention,

each token representation is linearly projected into three vectors: a Query (Q), a Key (K), and a Value (V). Attention weights are computed through scaled dot products between queries and keys, while the values are aggregated according to these weights to produce the attention output. The key and value representations therefore encode, respectively, how tokens are matched and how information is retrieved across the sequence.

Autoregressive generation requires the model to attend to all preceding tokens at each decoding step. Without caching, this would mean recomputing the key and value representations for every previous token at every generation step—a redundancy that grows quadratically with sequence length. The key-value cache eliminates this redundancy by storing the key and value projections once, when each token is first processed, and reusing them for all subsequent attention computations. This optimization is essential for making autoregressive generation practically tractable at the sequence lengths and throughputs required by modern applications. The memory footprint of this cache is determined by the model’s architecture and the sequence being processed. For a decoder with L layers, H attention heads per layer, head dimension d_h , processing a sequence of length n with batch size B , and storing values at p bytes per element, the total cache memory scales as $2 \times L \times H \times d_h \times n \times B \times p$. The factor of two accounts for storing both keys and values. Each architectural term contributes multiplicatively, meaning that increases in any dimension compound directly into memory consumption. For a contemporary model such as Llama-3-8B—with 32 layers, 32 attention heads, and a head dimension of 128 [3]—processing a single sequence of 4,096 tokens in 16-bit precision requires on the order of gigabytes of cache memory. Extending to 128,000 tokens, as many applications now demand, multiplies this figure by a factor of 32. Serving multiple concurrent requests multiplies it again by the batch size.

This scaling behavior distinguishes the key-value cache from model weights in a fundamental way. The weights of a language model are fixed after training: they occupy a constant memory footprint regardless of how the model is used. A 7-billion-parameter model requires approximately 14 gigabytes in 16-bit precision, and this cost is paid once per device, shared across all requests. The key-value cache, by contrast, is dynamic. It is allocated per request, grows with each generated token, and must be maintained for the duration of each conversation. In a serving system handling hundreds of concurrent users, the aggregate cache memory can exceed the memory occupied by the model itself.

This dynamic, per-request nature creates a direct coupling between memory capacity and serving throughput. The number of concurrent requests a system can handle is bounded not by computational resources but by how many caches fit in memory. When memory is exhausted, additional requests must wait—increasing latency—or be rejected entirely. Operators face a choice between serving fewer concurrent users at longer context lengths or more users at shorter contexts. Neither option

is satisfactory for applications such as interactive assistants or document analysis systems that require both responsiveness and extended context.

Beyond capacity, the cache also constrains memory bandwidth. During decode, each token generation involves reading the entire cache to compute the attention distribution, yet performs relatively little arithmetic on the retrieved data. Modern accelerators, designed for the high arithmetic intensity of training workloads, cannot saturate their computational units during decode; they spend most of their time waiting for memory transfers to complete. Reducing the size of the cache therefore yields a dual benefit: it increases the number of concurrent requests that fit in memory, and it reduces the memory bandwidth consumed per decoding step, directly improving generation latency.

These considerations elevate cache compression from a secondary optimization to a first-order system constraint. Unlike optimizations that improve efficiency at the margins, reducing cache memory directly determines what deployment configurations are feasible and at what cost. A 50% reduction in cache size can double the number of concurrent users a system supports at fixed hardware investment, or enable twice the context length at fixed capacity. For edge deployments, where total memory may be measured in gigabytes rather than hundreds of gigabytes, aggressive compression may be the difference between feasibility and impracticality. The challenge is to reduce memory without proportionally degrading the model’s output quality.

Existing Approaches and Their Limitations The urgency of this problem has motivated substantial work on compression techniques. These approaches fall into three broad categories: quantization methods that reduce the precision of cached values, eviction methods that selectively retain only a subset of tokens, and low-rank projection methods that represent the cache in a smaller subspace [4]. Quantization achieves compression without altering cache structure, but representations with outliers or heavy tails degrade disproportionately [5]. Eviction methods achieve high compression when context is sparse, but suffer from irreversibility—once a token is removed, its information cannot be recovered, and tasks requiring diffuse or shifting attention patterns expose this fragility. Both approaches are often combined with structural compression in practice, but neither addresses the fundamental question of how to represent the cache more compactly.

Low-rank projection methods take a structural approach. They observe that key and value representations, viewed as matrices across the sequence dimension, often exhibit rapidly decaying singular value spectra [6]—suggesting that much of the information can be captured in a subspace of lower dimension than the original embedding space. By projecting keys and values onto this subspace, these methods reduce the per-token storage cost proportionally to the dimensionality reduction. The dominant approach for constructing projection bases relies on singular value

decomposition computed over a calibration dataset. SVD identifies the directions of maximum variance in the cached representations and retains those directions as the projection basis. This procedure is mathematically principled: for a fixed rank, SVD yields the basis that minimizes reconstruction error in the Frobenius norm. The bases are computed once, offline, and applied statically during inference.

Despite their elegance, these methods share limitations that become critical as compression targets grow more aggressive. The core issue is that SVD is optimal for the wrong objective. Improving the algorithm for finding reconstruction-minimizing bases will not help; the problem lies in the objective itself. SVD-based methods optimize for reconstruction error—the difference between the original and projected representations. This objective is a proxy for what actually matters: the quality of the attention computation and, ultimately, the model’s output distribution. Reconstruction error and downstream performance are correlated at moderate compression levels, but this correlation is expected to weaken as more aggressive compression forces harder trade-offs. A projection basis that minimizes reconstruction error is not guaranteed to preserve the directions that matter for attention.

A specific instance of this proxy-objective problem concerns the alignment between variance and relevance. SVD identifies directions of maximum variance, implicitly assuming that high-variance directions carry the information most important for downstream computation. This assumption is not theoretically grounded for attention. The attention mechanism computes inner products between queries and keys; the directions that matter are those aligned with the query distribution, not necessarily those exhibiting the highest variance in the keys alone. A direction with modest variance in the key cache may nonetheless be critical if queries frequently attend along it. Conversely, a high-variance direction that is orthogonal to typical queries contributes little to attention and could be safely discarded. SVD cannot distinguish between these cases.

The static nature of projection bases compounds these issues. Because bases are computed offline from calibration data, they inherit whatever misalignment exists between the reconstruction objective and attention-relevant directions—and this misalignment cannot be corrected at inference time. When inference-time inputs differ in distribution from calibration data, the basis may no longer capture the relevant subspace. Different layers and attention heads may exhibit different compression tolerances, and these tolerances may vary by task. A fixed basis and fixed rank allocation cannot adapt to such variation.

These limitations interact in a predictable way. At moderate compression, the retained subspace is large enough to include both high-variance directions and attention-relevant directions, even when these do not perfectly coincide—the system has slack. At aggressive compression, this slack disappears. The proxy objective diverges from true downstream quality, the variance-relevance misalignment discards

information that may prove important, and the static basis cannot adapt to the specific demands of the input. The result is a characteristic pattern: low-rank methods perform well up to a threshold compression ratio, beyond which quality degrades rapidly. This pattern points to a gap in the current methodology. Methods in the current literature do not learn projection bases that are directly optimized for downstream behavior—that is, for preserving the structure of the attention computation rather than the structure of the key representations themselves. Nor do existing methods adapt their projections to the requirements of the input or task.

Objectives and Contributions This thesis addresses that gap. The primary objective is to investigate whether learned projection bases can serve as an alternative to SVD-based methods for low-rank key-value cache compression. Specifically, this work explores whether training a neural network to predict projection bases—optimized for the full decoder layer output rather than reconstruction error on the cached representations alone—can maintain output quality at aggressive compression ratios where static, variance-maximizing approaches degrade.

This work makes three contributions. First, it introduces a neural projection method for cache compression. The method uses a lightweight neural network to predict orthonormal projection bases from summary statistics of the cached representations. Orthonormality is enforced through differentiable QR decomposition, and the network is trained to minimize the discrepancy between full-rank and compressed decoder layer outputs. The learned bases are computed once during an offline calibration phase and then applied statically, incurring no additional runtime overhead during inference. The method applies to both keys and values, with separate projection networks trained for each.

Second, the implementation employs an asymmetric compression strategy for keys and values. Keys are compressed using a single projection matrix shared across all attention heads within each layer, while values are compressed using independent projection matrices per head. This design choice reflects an assumption about differing redundancy structures in key and value representations; its effectiveness relative to symmetric alternatives is subject to experimental validation.

Third, the work develops an error-budget-based rank selection mechanism. A greedy algorithm selects the minimum-rank configuration for each layer that satisfies a user-specified error tolerance. This enables controllable trade-offs between compression ratio and reconstruction fidelity, allowing deployment-specific tuning without retraining the projection networks.

The focus is on post-training compression of the key-value cache in decoder-only transformer models. The method compresses both keys and values, applying learned low-rank projections at each layer. Evaluation is conducted on a single model architecture (Llama-3-8B) using standard calibration and evaluation

datasets (WikiText-2, C4), with perplexity as the primary evaluation metric. The method is not claimed to outperform SVD-based compression; the contribution is methodological, providing an alternative framework whose empirical advantages remain subject to validation. The rank selection algorithm operates greedily on a per-layer basis and does not perform global optimization across layers. The calibration procedure requires training projection networks for each layer and rank configuration, incurring non-trivial computational cost prior to deployment. This process assumes that the calibration dataset is representative of inference-time inputs; out-of-distribution robustness is not evaluated. The method is not compared against orthogonal compression techniques such as quantization, token eviction, or weight pruning.

Thesis Structure The remainder of this thesis is organized as follows. Chapter 2 provides the technical background on attention mechanisms, key-value caching, and low-rank approximation. Chapter 3 reviews related work on cache compression, situating the present contribution within the broader literature. Chapter 4 presents the proposed method, including the network architecture, training procedure, and rank selection algorithm. Chapter 5 describes the experimental setup, and Chapter 6 reports the results. Chapter 7 discusses the findings, limitations, and implications. Chapter 8 outlines directions for future work, and Chapter 9 concludes the thesis. A consolidated subset of the core empirical findings has also been reported in [7]. The thesis, however, provides a broader experimental and diagnostic treatment, including extended ablation studies and system-level interpretation.

Chapter 2

Background

This chapter introduces the technical foundations on which the remainder of the thesis builds.

2.1 Transformer Decoder and Self-Attention

A decoder-only Transformer [8] is a sequence model composed of L identical layers applied in succession. Each layer receives a sequence of n token representations arranged as the rows of a matrix $\mathbf{X} \in \mathbb{R}^{n \times d_{\text{model}}}$, where d_{model} denotes the model dimension, and produces an output of the same shape. A single layer consists of two sub-layers—a multi-head self-attention block followed by a position-wise feed-forward network—each wrapped in a residual connection and a normalization step. Concretely, the computation of layer $\ell \in \{1, \dots, L\}$ is

$$\mathbf{Z}^{(\ell)} = \mathbf{X}^{(\ell)} + \text{MHA}(\text{Norm}(\mathbf{X}^{(\ell)})), \quad (2.1)$$

$$\mathbf{X}^{(\ell+1)} = \mathbf{Z}^{(\ell)} + \text{FFN}(\text{Norm}(\mathbf{Z}^{(\ell)})), \quad (2.2)$$

where Norm is a normalization function (e.g. RMSNorm) and $\mathbf{X}^{(1)}$ is the token embedding matrix. For notational brevity, the layer index ℓ is suppressed when the context is unambiguous; all projection and weight matrices introduced below are understood to be per-layer parameters.

Query, Key, and Value Projections. The multi-head self-attention block partitions its representation into H parallel attention heads. Let $d_h = d_{\text{model}}/H$ denote the per-head dimension, where H is assumed to divide d_{model} . For head $h \in \{1, \dots, H\}$, three linear projections map the normalized input $\tilde{\mathbf{X}} = \text{Norm}(\mathbf{X}) \in \mathbb{R}^{n \times d_{\text{model}}}$ to queries, keys, and values:

$$\mathbf{Q}_h = \tilde{\mathbf{X}} \mathbf{W}_h^Q, \quad \mathbf{K}_h = \tilde{\mathbf{X}} \mathbf{W}_h^K, \quad \mathbf{V}_h = \tilde{\mathbf{X}} \mathbf{W}_h^V, \quad (2.3)$$

with learned weight matrices $\mathbf{W}_h^Q, \mathbf{W}_h^K, \mathbf{W}_h^V \in \mathbb{R}^{d_{\text{model}} \times d_h}$. The resulting matrices $\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h \in \mathbb{R}^{n \times d_h}$ each contain one d_h -dimensional row vector per token position. In architectures that employ rotary position embeddings (RoPE) [9], position-dependent rotations are applied to $\mathbf{Q}_h$ and $\mathbf{K}_h$ after the linear projections and before the attention computation. Throughout this thesis, $\mathbf{Q}_h$ and $\mathbf{K}_h$ denote the representations *after* any such positional transformation.

Scaled Dot-Product Self-Attention. Each head computes an attention output by taking a weighted combination of the value vectors, where the weights are derived from pairwise similarities between queries and keys. The scaled dot-product attention for head h is defined as

$$\text{Attention}(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h) = \text{softmax}\left(\frac{\mathbf{Q}_h \mathbf{K}_h^\top}{\sqrt{d_h}} + \mathbf{M}\right) \mathbf{V}_h, \quad (2.4)$$

where the product $\mathbf{Q}_h \mathbf{K}_h^\top \in \mathbb{R}^{n \times n}$ yields a matrix of raw attention scores, the scaling factor $\sqrt{d_h}$ normalizes by the key dimension, and softmax is applied row-wise so that each row sums to one. The matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$ is a causal mask with entries

$$M_{ij} = \begin{cases} 0 & \text{if } i \geq j, \\ -\infty & \text{if } i < j, \end{cases} \quad (2.5)$$

which ensures that position i attends only to positions $j \leq i$. After exponentiation in the softmax, the $-\infty$ entries become zero, enforcing the autoregressive property: each token’s representation depends exclusively on preceding tokens and itself.

Let $\mathbf{A}_h = \text{softmax}(\mathbf{Q}_h \mathbf{K}_h^\top / \sqrt{d_h} + \mathbf{M}) \in \mathbb{R}^{n \times n}$ denote the attention weight matrix for head h . Row i of $\mathbf{A}_h$ is a probability distribution over positions $\{1, \dots, i\}$, and the corresponding output row is the convex combination $\sum_{j=1}^i [\mathbf{A}_h]_{ij} \mathbf{v}_{h,j}$, where $\mathbf{v}_{h,j} \in \mathbb{R}^{d_h}$ denotes the j -th row of $\mathbf{V}_h$.

Multi-Head Aggregation. The per-head outputs are concatenated and projected back to the model dimension by a learned output matrix $\mathbf{W}^O \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$:

$$\text{MHA}(\widetilde{\mathbf{X}}) = \text{Concat}\left(\text{Attention}(\mathbf{Q}_1, \mathbf{K}_1, \mathbf{V}_1), \dots, \text{Attention}(\mathbf{Q}_H, \mathbf{K}_H, \mathbf{V}_H)\right) \mathbf{W}^O, \quad (2.6)$$

where Concat places the H matrices of shape $\mathbb{R}^{n \times d_h}$ side by side to form a matrix in $\mathbb{R}^{n \times d_{\text{model}}}$. The output projection $\mathbf{W}^O$ mixes information across heads.

Grouped Query Attention. The formulation above assumes that queries, keys, and values use the same number of heads H . Grouped Query Attention (GQA) [10] generalizes this setting by distinguishing between H_Q query heads and a smaller

number $H_{KV} \leq H_Q$ of key-value heads, where the ratio $G = H_Q/H_{KV}$ is a positive integer called the *group size*. Each group of G query heads shares a single key head and a single value head: query head h attends using key and value matrices $\mathbf{K}_g$ and $\mathbf{V}_g$ with $g = \lceil h/G \rceil$. The key-value projection weights are $\mathbf{W}_g^K, \mathbf{W}_g^V \in \mathbb{R}^{d_{\text{model}} \times d_h}$ for $g \in \{1, \dots, H_{KV}\}$, and the attention equation (2.4) applies unchanged within each query head. Equation (2.6) generalizes accordingly: the concatenation runs over all H_Q query heads, each using the key-value pair assigned to its group. Standard multi-head attention is recovered when $G = 1$, i.e. $H_{KV} = H_Q$. In the remainder of this thesis, H denotes H_Q unless otherwise qualified, and H_{KV} denotes specifically the number of key-value heads.

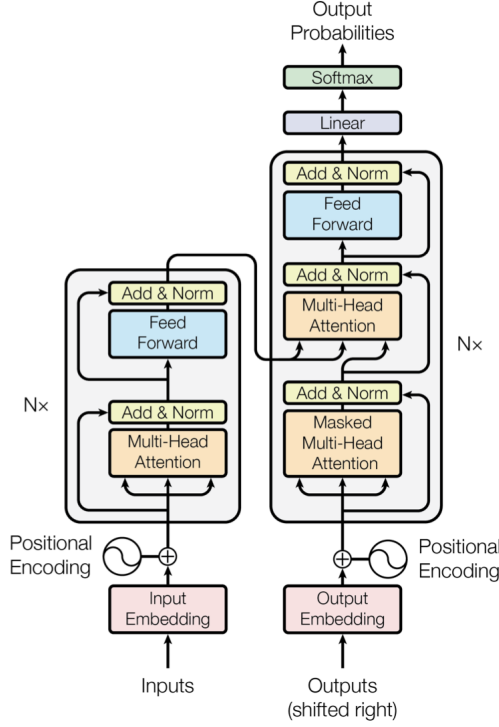


Figure 2.1: Transformer architecture [8].

Feed-Forward Network. The second sub-layer applies a position-wise feed-forward network independently to each token representation. In a standard formulation this consists of two linear transformations with a non-linearity:¹

$$\text{FFN}(\mathbf{z}) = \sigma(\mathbf{z} \mathbf{W}_1) \mathbf{W}_2, \quad (2.7)$$

¹Modern architectures such as the Llama family omit all bias terms. Equation (2.7) reflects this bias-free convention.

Table 2.1: Notation introduced in Section 2.1.

Symbol	Description
n	Sequence length (number of tokens)
d_{model}	Model (embedding) dimension
L	Number of decoder layers
H, H_Q	Number of query (attention) heads per layer
H_{KV}	Number of key-value heads per layer ($H_{KV} \leq H_Q$)
G	GQA group size, $G = H_Q/H_{KV}$
d_h	Per-head dimension, $d_h = d_{\text{model}}/H$
d_{ff}	Feed-forward intermediate dimension
$\tilde{\mathbf{X}}$	Normalized layer input, $\text{Norm}(\mathbf{X}) \in \mathbb{R}^{n \times d_{\text{model}}}$
$\mathbf{X}^{(\ell)}$	Input to layer ℓ , $\in \mathbb{R}^{n \times d_{\text{model}}}$
$\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h$	Query, key, value matrices for head h , $\in \mathbb{R}^{n \times d_h}$
$\mathbf{W}_h^Q, \mathbf{W}_h^K, \mathbf{W}_h^V$	QKV projection weights, $\in \mathbb{R}^{d_{\text{model}} \times d_h}$
$\mathbf{A}_h$	Attention weight matrix for head h , $\in \mathbb{R}^{n \times n}$
$\mathbf{W}^O$	Output projection, $\in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$
$\mathbf{M}$	Causal mask, $\in \mathbb{R}^{n \times n}$

where $\mathbf{W}_1 \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}$, $\mathbf{W}_2 \in \mathbb{R}^{d_{\text{ff}} \times d_{\text{model}}}$, and σ is a non-linear activation function. The intermediate dimension d_{ff} is larger than d_{model} . Variants such as the gated architecture used in Llama models replace the single non-linearity with a gating mechanism, but the dimensionality and role of the sub-layer remain the same.

Notation Summary. Throughout this thesis, bold uppercase letters (e.g. $\mathbf{X}$) denote matrices and bold lowercase letters (e.g. $\mathbf{v}_{h,j}$) denote vectors; the j -th row of a matrix $\mathbf{V}_h$ is written $\mathbf{v}_{h,j}$. Table 2.1 collects the principal symbols introduced in this section.

2.2 Autoregressive Inference and KV-Caching

Section 2.1 defines the Transformer decoder as operating on a complete sequence of n tokens. During inference, however, the output sequence is not available in advance: tokens are generated one at a time, each conditioned on every token that precedes it. This section formalizes autoregressive generation and the caching mechanism that stores and reuses key and value representations across generation steps [11].

Autoregressive Generation. Let $(x_1, \dots, x_{n_p})$ denote an input prompt of n_p tokens. The model produces a continuation $x_{n_p+1}, x_{n_p+2}, \dots$ by repeatedly sampling

from the conditional distribution

$$x_{t+1} \sim p(\cdot \mid x_1, \dots, x_t), \quad (2.8)$$

where p is the distribution over the vocabulary defined by the output of the final decoder layer followed by a linear projection and a softmax. Computing this distribution at step t requires a forward pass through all L layers with access to the representations of every preceding token $x_1, \dots, x_t$.

Prefill Phase. The prompt tokens are processed in parallel, exactly as described in Section 2.1 with $n = n_p$. At each layer ℓ and for each key-value head $g \in \{1, \dots, H_{KV}\}$, the linear projections and any subsequent positional transformation yield a key matrix and a value matrix:

$$\mathbf{K}_g^{(\ell)} \in \mathbb{R}^{n_p \times d_h}, \quad \mathbf{V}_g^{(\ell)} \in \mathbb{R}^{n_p \times d_h}. \quad (2.9)$$

These matrices are retained in memory for use in subsequent generation steps. Their rows correspond, respectively, to the key and value representations of each prompt token at the given layer and head.

KV-Cache. The collection of all retained key and value matrices is called the *key-value cache* (KV-cache). After the prefill phase, the cache for a single sequence consists of $2 L H_{KV}$ matrices, one key matrix and one value matrix per layer and key-value head. The symbols $\mathbf{K}_g^{(\ell, \leq t)}$ and $\mathbf{V}_g^{(\ell, \leq t)}$ denote the cached key and value matrices at layer ℓ , head g , containing the representations of all tokens up to and including position t :

$$\mathbf{K}_g^{(\ell, \leq t)} = \begin{bmatrix} \mathbf{k}_{g,1}^{(\ell)} \\ \vdots \\ \mathbf{k}_{g,t}^{(\ell)} \end{bmatrix} \in \mathbb{R}^{t \times d_h}, \quad \mathbf{V}_g^{(\ell, \leq t)} = \begin{bmatrix} \mathbf{v}_{g,1}^{(\ell)} \\ \vdots \\ \mathbf{v}_{g,t}^{(\ell)} \end{bmatrix} \in \mathbb{R}^{t \times d_h}, \quad (2.10)$$

where $\mathbf{k}_{g,j}^{(\ell)}, \mathbf{v}_{g,j}^{(\ell)} \in \mathbb{R}^{d_h}$ are the key and value vectors for position j . Immediately after prefill, $t = n_p$ and the cache holds exactly the matrices from (2.9). When the layer index is clear from context, the abbreviated notation $\mathbf{K}_g^{\leq t}$ and $\mathbf{V}_g^{\leq t}$ is used.

Decode Phase. At each generation step $t \geq n_p$, the model processes a single new token x_{t+1} , where t equals the number of tokens whose key and value representations are currently stored in the cache. As in the prefill phase, layers are processed sequentially: at layer ℓ , the input representation of token x_{t+1} is derived from the output of layer $\ell - 1$. Let $\tilde{\mathbf{x}}_{t+1} \in \mathbb{R}^{d_{\text{model}}}$ denote its normalized representation at

a given layer. The projections produce one query vector per query head and one key-value pair per key-value head:

$$\mathbf{q}_{h,t+1} = \tilde{\mathbf{x}}_{t+1} \mathbf{W}_h^Q \in \mathbb{R}^{d_h}, \quad \mathbf{k}_{g,t+1} = \tilde{\mathbf{x}}_{t+1} \mathbf{W}_g^K \in \mathbb{R}^{d_h}, \quad \mathbf{v}_{g,t+1} = \tilde{\mathbf{x}}_{t+1} \mathbf{W}_g^V \in \mathbb{R}^{d_h}. \quad (2.11)$$

The new key and value vectors are appended to the cache, extending it from t to $t + 1$ rows:

$$\mathbf{K}_g^{\leq t+1} = \begin{bmatrix} \mathbf{K}_g^{\leq t} \\ \mathbf{k}_{g,t+1} \end{bmatrix} \in \mathbb{R}^{(t+1) \times d_h}, \quad \mathbf{V}_g^{\leq t+1} = \begin{bmatrix} \mathbf{V}_g^{\leq t} \\ \mathbf{v}_{g,t+1} \end{bmatrix} \in \mathbb{R}^{(t+1) \times d_h}. \quad (2.12)$$

The attention output for query head h at position $t + 1$ is then computed against the full cache of the corresponding key-value head $g = \lceil h/G \rceil$:

$$\text{Attention}(\mathbf{q}_{h,t+1}, \mathbf{K}_g^{\leq t+1}, \mathbf{V}_g^{\leq t+1}) = \text{softmax}\left(\frac{\mathbf{q}_{h,t+1} (\mathbf{K}_g^{\leq t+1})^\top}{\sqrt{d_h}}\right) \mathbf{V}_g^{\leq t+1} \in \mathbb{R}^{d_h}. \quad (2.13)$$

This is a special case of equation (2.4) with a single query position ($n = 1$): the query matrix reduces to a single row vector and the output is correspondingly a vector in $\mathbb{R}^{d_h}$. No explicit causal mask is required: position $t + 1$ is the final position in the sequence, so it attends to all $t + 1$ cached positions by construction.

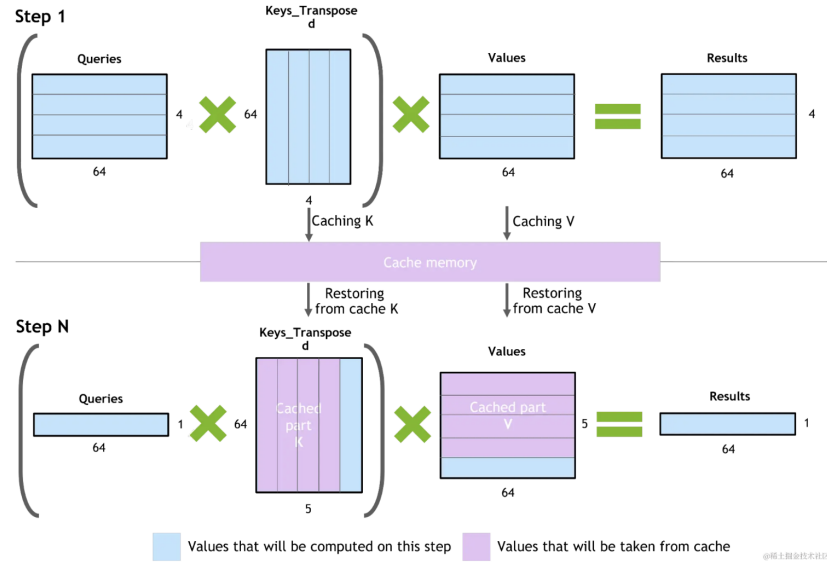


Figure 2.2: Illustration of the key-value caching mechanism [12]

Cache Contents. After processing t tokens in total (prompt and generated tokens combined), the KV-cache stores $2 L H_{KV}$ matrices, each of shape $t \times d_h$. The symbol $\mathcal{C}_t$ denotes the full cache state at step t , comprising all $2 L H_{KV}$ retained matrices. The total number of scalar entries is

$$|\mathcal{C}_t| = 2 L H_{KV} t d_h. \quad (2.14)$$

The cache grows by $2 L H_{KV} d_h$ entries with each generated token. The key and value representations stored at position j are never recomputed once written; only new entries are appended. The cache is maintained independently for each sequence; in a batched setting with B concurrent sequences, the total state comprises B independent copies of $\mathcal{C}_t$.

2.3 Computational and Memory Complexity of Attention

The preceding sections define the Transformer decoder (Section 2.1) and the KV-cache mechanism (Section 2.2). This section derives the arithmetic and memory complexity of both inference phases and characterizes the operational regime of each.

Per-Layer Arithmetic Cost. The computation within a single decoder layer decomposes into two categories. *Linear projections* encompass the QKV mappings (2.3), the output projection in (2.6), and the feed-forward network (2.7). For n input tokens, each projection multiplies an $n \times d_{\text{model}}$ activation matrix by a weight matrix whose output dimension depends on the projection type; summing over all projections in a layer, the combined cost is

$$F_{\text{proj}}(n) = \Theta(n \cdot d_{\text{model}} (d_{\text{model}} + d_{\text{ff}})). \quad (2.15)$$

Under GQA, the key and value projections map to $H_{KV} d_h = d_{\text{model}}/G$ dimensions rather than d_{model} , reducing their cost by a factor of G relative to the query and output projections; this saving is absorbed by the asymptotic constant in (2.15). *Attention* consists of the score computation $\mathbf{Q}_h \mathbf{K}_h^\top$ and the value aggregation $\mathbf{A}_h \mathbf{V}_h$ from equation (2.4). When n query positions attend to n_{ctx} context positions, each product costs $\Theta(n \cdot n_{\text{ctx}} \cdot d_h)$ per head. Over all H query heads, the per-layer attention cost is

$$F_{\text{attn}}(n, n_{\text{ctx}}) = \Theta(n \cdot n_{\text{ctx}} \cdot d_{\text{model}}), \quad (2.16)$$

since $H d_h = d_{\text{model}}$.

Prefill and Decode Complexity. During prefill, the n_p prompt tokens serve simultaneously as queries and context, so $n = n_{\text{ctx}} = n_p$. The total cost over L layers is

$$F_{\text{prefill}} = \Theta\left(L n_p d_{\text{model}} (d_{\text{model}} + d_{\text{ff}}) + L n_p^2 d_{\text{model}}\right). \quad (2.17)$$

The first term accounts for linear projections and is linear in n_p ; the second accounts for attention and is quadratic. For long prompts ($n_p \gg d_{\text{model}} + d_{\text{ff}}$), the attention term dominates; in models with large d_{ff} , however, this crossover may require substantial prompt lengths.

During decode, a single new token ($n = 1$) is appended to the cache and attends to all $t + 1$ stored positions, as formalized in (2.13). The per-step cost over L layers is

$$F_{\text{decode}}(t) = \Theta\left(L d_{\text{model}} (d_{\text{model}} + d_{\text{ff}}) + L t d_{\text{model}}\right). \quad (2.18)$$

The projection cost is constant per step; the attention cost grows linearly with t . In the absence of the KV-cache, each decode step would require reprocessing all t preceding tokens to reconstruct their key and value representations, raising the per-step cost to $F_{\text{proj}}(t) + F_{\text{attn}}(t, t)$ —identical in form to the prefill cost (2.17) with n_p replaced by t . The KV-cache eliminates the factor-of- t increase in the projection term and reduces the attention cost from quadratic to linear in t , at the expense of the additional memory derived below.

Memory Requirements. Inference memory consists of two components. The *model parameters* occupy

$$M_{\text{params}} = \Theta\left(L (d_{\text{model}}^2 + d_{\text{model}} d_{\text{ff}})\right) \quad (2.19)$$

scalar values, independent of input length or batch size. The *KV-cache*, from equation (2.14), stores $|\mathcal{C}_t| = 2 L H_{KV} t d_h$ entries per sequence. For B concurrent sequences, the total number of cached entries is

$$M_{\text{cache}} = 2 L H_{KV} d_h t B \quad (2.20)$$

scalar values. Denoting by p the number of bytes per stored element, the total inference memory is $p(M_{\text{params}} + M_{\text{cache}})$ bytes. Because M_{params} is fixed while M_{cache} grows linearly with both t and B , the cache dominates the total footprint as either the sequence length or the number of concurrent sequences increases. Intermediate activations produced during the forward pass also occupy memory but, for inference with long sequences, their contribution is small relative to the cache and is omitted from this analysis.

Arithmetic Intensity. The *arithmetic intensity* of an operation is the ratio of floating-point operations performed to the volume of data (in bytes) transferred between memory and compute units. When this ratio is small, throughput is determined by the rate of data transfer rather than by arithmetic capacity; such operations are called *memory-bandwidth-bound*.

During prefill, each weight matrix is read once and applied to n_p token vectors, yielding an arithmetic intensity proportional to n_p/p . For sufficiently long prompts, this ratio is large and the prefill phase is compute-bound. During decode, the same weight matrices are read but applied to a single token vector, reducing the projection arithmetic intensity to $\Theta(1/p)$. The cache access exhibits a similar pattern: reading $2 H_{KV} t d_h$ entries per layer to perform $\Theta(H t d_h)$ operations yields an arithmetic intensity of $\Theta(G/p)$, where $G = H/H_{KV}$ is the GQA group size. In both cases the decode-step arithmetic intensity is bounded by a constant that does not grow with the context length t . As t increases, the volume of data that must be read—model parameters and the full cache $\mathcal{C}_t$ —grows, while the number of operations per byte read remains fixed. The decode phase is therefore memory-bandwidth-bound for all context lengths.

2.4 Linear Low-Rank Approximation

Many matrices of practical interest can be closely approximated by matrices of much lower rank. Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ of rank $\rho \leq \min(m, n)$, low-rank approximation seeks a matrix of rank at most $r \ll \rho$ that is closest to $\mathbf{A}$ under a given matrix norm. The Singular Value Decomposition provides an exact and constructive solution to this problem. The notation in this section—including the symbols $\mathbf{A}$, $\mathbf{U}$, $\mathbf{V}$, m , and n —follows standard linear algebra conventions and is independent of the Transformer-specific symbols introduced in the preceding sections.

Singular Value Decomposition. Every real matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ admits a factorization

$$\mathbf{A} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^\top, \quad (2.21)$$

where $\mathbf{U} \in \mathbb{R}^{m \times m}$ and $\mathbf{V} \in \mathbb{R}^{n \times n}$ are orthogonal matrices and $\mathbf{\Sigma} \in \mathbb{R}^{m \times n}$ is a rectangular diagonal matrix with non-negative diagonal entries $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{\min(m, n)} \geq 0$. The quantities σ_i are the *singular values* of $\mathbf{A}$; the columns of $\mathbf{U}$ and $\mathbf{V}$ are the left and right *singular vectors*, respectively. The number of non-zero singular values equals ρ .

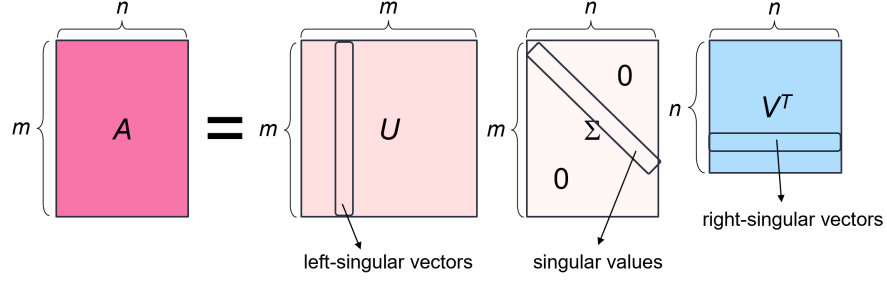


Figure 2.3: Singular Value Decomposition (SVD) of a matrix [13].

Rank- r Truncation. For any integer $1 \leq r \leq \rho$, the *rank- r truncated SVD* retains only the r largest singular values and their corresponding singular vectors:

$$\mathbf{A}_r = \mathbf{U}_r \mathbf{\Sigma}_r \mathbf{V}_r^\top, \quad (2.22)$$

where $\mathbf{U}_r \in \mathbb{R}^{m \times r}$ and $\mathbf{V}_r \in \mathbb{R}^{n \times r}$ contain the first r columns of $\mathbf{U}$ and $\mathbf{V}$, respectively, and $\mathbf{\Sigma}_r = \text{diag}(\sigma_1, \dots, \sigma_r) \in \mathbb{R}^{r \times r}$.

The Eckart–Young–Mirsky Theorem. The truncated SVD is the best rank- r approximation of $\mathbf{A}$ under the Frobenius norm $\|\mathbf{A}\|_F = (\sum_{i,j} A_{ij}^2)^{1/2}$. Specifically,

$$\mathbf{A}_r \in \arg \min_{\substack{\mathbf{B} \in \mathbb{R}^{m \times n} \\ \text{rank}(\mathbf{B}) \leq r}} \|\mathbf{A} - \mathbf{B}\|_F, \quad (2.23)$$

and the minimum error is

$$\|\mathbf{A} - \mathbf{A}_r\|_F = \left(\sum_{i=r+1}^{\min(m,n)} \sigma_i^2 \right)^{1/2}. \quad (2.24)$$

This result was established by Eckart and Young (1936) for the Frobenius norm and extended to all unitarily invariant norms by Mirsky (1960) [14, 15]. The error is determined entirely by the discarded singular values; when the singular spectrum decays rapidly, a small r suffices for an accurate approximation.

Projection Formulation. The rank- r approximation can equivalently be expressed as an orthogonal projection. Since $\mathbf{A} \mathbf{V}_r = \mathbf{U}_r \mathbf{\Sigma}_r$, the truncated SVD satisfies

$$\mathbf{A}_r = \mathbf{A} \mathbf{V}_r \mathbf{V}_r^\top, \quad (2.25)$$

where $\mathbf{V}_r \mathbf{V}_r^\top \in \mathbb{R}^{n \times n}$ is the orthogonal projector onto $\text{col}(\mathbf{V}_r)$. Each row of $\mathbf{A}$ is projected onto the r -dimensional subspace spanned by the leading right singular vectors and then reconstructed from the projected coordinates. The projected

representation $\mathbf{A} \mathbf{V}_r \in \mathbb{R}^{m \times r}$ stores $m r$ entries rather than $m n$; right-multiplying by $\mathbf{V}_r^\top$ yields the rank- r reconstruction $\mathbf{A}_r$.

In this formulation, finding the best rank- r approximation reduces to selecting the r -dimensional subspace that minimizes total reconstruction error:

$$\mathbf{V}_r \in \arg \min_{\substack{\mathbf{P} \in \mathbb{R}^{n \times r} \\ \mathbf{P}^\top \mathbf{P} = \mathbf{I}_r}} \left\| \mathbf{A} - \mathbf{A} \mathbf{P} \mathbf{P}^\top \right\|_F. \quad (2.26)$$

Every matrix of the form $\mathbf{A} \mathbf{P} \mathbf{P}^\top$ has rank at most r , and the choice $\mathbf{P} = \mathbf{V}_r$ attains the Eckart–Young–Mirsky bound (2.24); the unconstrained minimization (2.23) and the projection minimization (2.26) therefore share the same optimal value.

The right singular vectors $\mathbf{v}_1, \dots, \mathbf{v}_r$ are also the eigenvectors of $\mathbf{A}^\top \mathbf{A}$ corresponding to the r largest eigenvalues $\sigma_1^2 \geq \dots \geq \sigma_r^2$. When the rows of $\mathbf{A}$ represent centered data vectors, $\mathbf{A}^\top \mathbf{A}$ is proportional to the sample covariance matrix and its leading eigenvectors are the *principal components* of the data. The projection in (2.25) then coincides with the reconstruction step of Principal Component Analysis (PCA), and the optimization (2.26) is the standard objective solved by SVD-based dimensionality reduction methods.

Chapter 3

Related Work

Building on the notation and formalism introduced in Chapter 2, this chapter reviews existing approaches to KV-cache optimization in Transformer-based language models, with a focus on techniques aimed at reducing memory footprint and improving inference efficiency during autoregressive decoding. Rather than presenting a single unified line of research, the literature spans multiple complementary directions that address the KV-cache bottleneck from different perspectives.

The discussion is organized around four principal axes. First, *per-layer low-rank approximation* methods are examined, which reduce the dimensionality of key and value representations through analytical projections applied to pretrained models. Second, *architectural and quantization-adjacent directions* are discussed, which achieve KV memory reduction through latent-state redesign or precision-oriented transformations that are composable with low-rank methods. Third, *systems and runtime KV-cache optimizations* are reviewed, which target memory layout, bandwidth usage, and decoding efficiency without modifying representation geometry directly. Fourth, approaches based on *inter-layer redundancy and representation sharing*, which exploit structural similarities across decoder depth to reduce redundant storage.

The chapter concludes by synthesizing the common assumptions underlying these families of methods, highlighting recurring design trade-offs such as reliance on static projection bases, optimization of proxy objectives, and limited adaptivity to query dynamics. This synthesis provides a comparative basis for the analysis developed in the following chapters.

3.1 Representation-Level Low-Rank Compression (Post-Training)

A large portion of recent research on KV-cache optimization focuses on *low-rank approximation*, motivated by empirical evidence that key and value representations in Transformer attention exhibit substantial redundancy along the embedding dimension. These methods project high-dimensional key and value matrices onto lower-dimensional subspaces, reducing the memory footprint of cached representations from $O(n \cdot d)$ to $O(n \cdot r)$ with $r \ll d$.

Within this axis, the focus is on post-training projection methods that compress key and value representations through analytical decompositions computed on calibration statistics [6, 16, 17].

Despite differences in formulation and system integration, most approaches share a common mathematical structure: a projection basis is computed offline, typically through analytical decompositions or calibration statistics, and then applied uniformly during inference. This design enables efficient deployment, since the projection can be folded into existing attention weights without introducing additional runtime complexity. At the same time, it implies that the retained subspace is determined by variance-based criteria or other tractable proxy objectives rather than by direct optimization of downstream model behavior.

The following paragraphs review representative methods within this paradigm, highlighting both their mathematical foundations and their practical implications for KV-cache compression.

EigenAttention. EigenAttention [6] provides a representative example of variance-driven low-rank compression applied to the KV-cache. The method computes singular value decomposition on calibration activations in order to identify principal subspaces for queries, keys, and values. The resulting projection bases are absorbed into the linear projection weights $\mathbf{W}^Q$, $\mathbf{W}^K$, $\mathbf{W}^V$, and $\mathbf{W}^O$, allowing the attention computation to operate in a reduced-dimensional space without introducing additional runtime modules.

Conceptually, EigenAttention assumes that the dominant singular directions of key and value representations capture most of the functional information required by the attention mechanism. Under this assumption, projecting the cached tensors onto a rank- r subspace yields an approximation of the form $\mathbf{K} \approx \mathbf{K}\mathbf{U}_K\mathbf{U}_K^\top$, where the retained basis vectors correspond to the leading singular components obtained during calibration. While the Eckart–Young–Mirsky theorem guarantees optimal reconstruction under Frobenius norm for a fixed rank, the objective remains tied to variance preservation rather than to downstream decoder behavior.

In practice, EigenAttention reduces KV-cache memory by lowering the effective

dimensionality of attention projections, trading exact representation fidelity for improved efficiency while maintaining a static projection basis during inference.

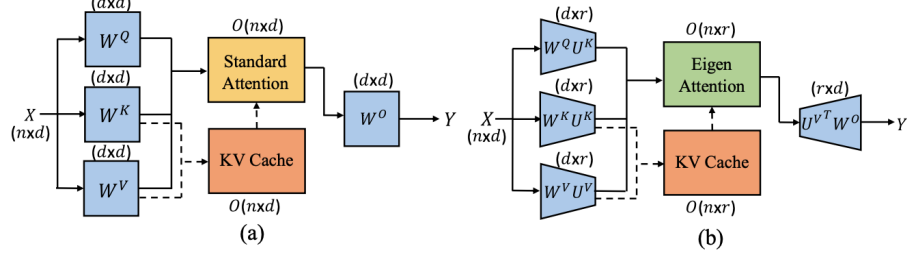


Figure 3.1: Comparison between (a) Standard Attention and (b) Eigen Attention. Eigen Attention utilizes lower dimensional ($r \ll d$) query, key, and value projection matrices than the standard attention operation, leading to KV cache compression and compute FLOPs benefits. [6]

KQ-SVD. KQ-SVD [16] provides a more attention-aware formulation of low-rank KV compression by explicitly targeting the quantity that drives attention computation. Rather than minimizing reconstruction error on the key matrix alone, the method observes that the functional object of interest is the attention logit matrix $\mathbf{QK}^\top$. Compressing keys independently therefore optimizes a surrogate objective that may not preserve the geometry relevant to query–key interactions. To address this mismatch, KQ-SVD derives a rank- r approximation $\tilde{\mathbf{K}}$ that directly minimizes the functional distortion $\|\mathbf{QK}^\top - \mathbf{Q}\tilde{\mathbf{K}}^\top\|_F$, aligning the compression objective with the logits produced during attention. The analysis shows that standard key-only SVD achieves this optimum only under restrictive conditions, namely when the dominant subspaces of $\mathbf{K}$ and $\mathbf{QK}^\top$ are aligned. The work also revisits EigenAttention’s strategy of performing SVD on the concatenated matrix $[\mathbf{Q}; \mathbf{K}]$, clarifying when such constructions approximate an attention-aware objective.

Despite introducing a more principled formulation, KQ-SVD remains analytical in nature: projection bases are obtained through closed-form matrix decomposition on calibration data and remain fixed at inference time. Consequently, the method inherits the static-basis assumption shared by most variance-driven compression approaches.

A3 Framework. The A3 framework [17] approaches KV-cache compression from a theoretical perspective, deriving analytical characterizations of the error introduced by low-rank approximation within the attention mechanism. Rather than evaluating reconstruction fidelity of keys or values in isolation, A3 analyzes

how approximation errors propagate through the full attention pipeline, including logit computation, softmax normalization, and output projection.

Using singular value decomposition as a core analytical tool, the framework shows that compression quality depends on layer-specific sensitivity to rank reduction. This insight motivates non-uniform budget allocation strategies, in which different layers receive different compression levels according to their estimated functional impact. From a methodological standpoint, A3 provides a bridge between purely variance-driven methods and approaches that consider the downstream behavior of attention outputs.

Despite emphasizing functional error analysis, the resulting projection bases remain analytically derived from calibration data and therefore static at inference time. As a consequence, A3 retains the closed-form, variance-oriented design philosophy shared by most SVD-based compression techniques, even while advancing the theoretical understanding of attention-aware approximation.

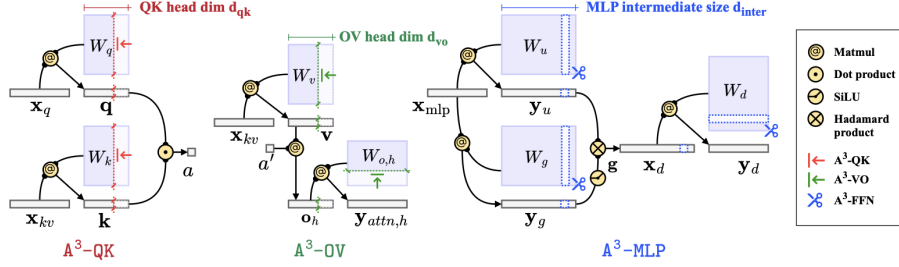


Figure 3.2: High-level overview of A3. A3 performs low-rank approximation on QK, OV, and MLP components, reducing head dimensions in QK and OV and the intermediate dimension in MLP. [17]

Section Synthesis. The low-rank methods reviewed above share a common structure: they derive projection bases offline from calibration data and apply them uniformly at inference time. This yields strong efficiency and simple deployment, while introducing an explicit dependence on proxy objectives and static subspace assumptions. The broader cross-family implications of these assumptions are consolidated in Section 3.5.

3.2 Architectural and Quantization-Adjacent Directions

This section groups directions that are closely related to KV-cache compression but do not fit strictly into post-training per-layer low-rank projection. These

directions are separated explicitly to preserve taxonomy clarity while retaining their methodological relevance.

Multi-head Latent Attention (MLA). The Multi-head Latent Attention (MLA) architecture, introduced in DeepSeek-V2 [18] and further developed in subsequent work [19, 20], represents a distinct direction within KV-cache optimization: an architectural redesign rather than a post-training compression method. Instead of storing full-dimensional keys and values, MLA maintains a compact latent representation from which attention states are reconstructed on-the-fly during decoding.

This design assumes that key and value tensors share a structured low-dimensional manifold that can be learned jointly during pretraining. As a result, memory reduction is achieved through representation learning embedded directly into the model architecture, rather than through offline projection of pretrained weights. Compared to analytical SVD-based methods, MLA therefore shifts the compression problem from calibration-time optimization to training-time model design.

Despite these differences, MLA embodies a related conceptual assumption shared by many low-rank approaches: the existence of a static low-dimensional subspace that remains fixed at inference time. The method is included here to highlight how architectural solutions and post-hoc projection techniques explore complementary regions of the KV-cache compression design space.

SVDq (Quantization-Oriented Perspective). While EigenAttention and KQ-SVD focus on KV-cache representations, SVDq [21] is primarily a low-bit quantization framework for model parameters rather than a direct KV-cache compression method. Its relevance in this chapter is methodological: it shows that SVD-induced orthogonal rotations can concentrate signal energy in a small subset of channels, enabling non-uniform bit allocation and reduced quantization distortion.

From a KV-cache perspective, this evidence is best interpreted as an auxiliary design signal rather than a direct solution. In particular, the use of rotated coordinates and variance-ordered components is conceptually aligned with compression pipelines that rely on static bases, but transfer to runtime KV-cache compression requires additional assumptions beyond those studied in the original setting.

KV-Cache Quantization. Complementary to low-rank projection, recent work has explored direct quantization of cached keys and values. KIVI [22] shows that keys and values exhibit different distributional properties and proposes a tuning-free asymmetric 2-bit scheme with distinct granularity for the two tensors. This line of work targets memory reduction through precision management rather than subspace selection, and is therefore largely composable with projection-based

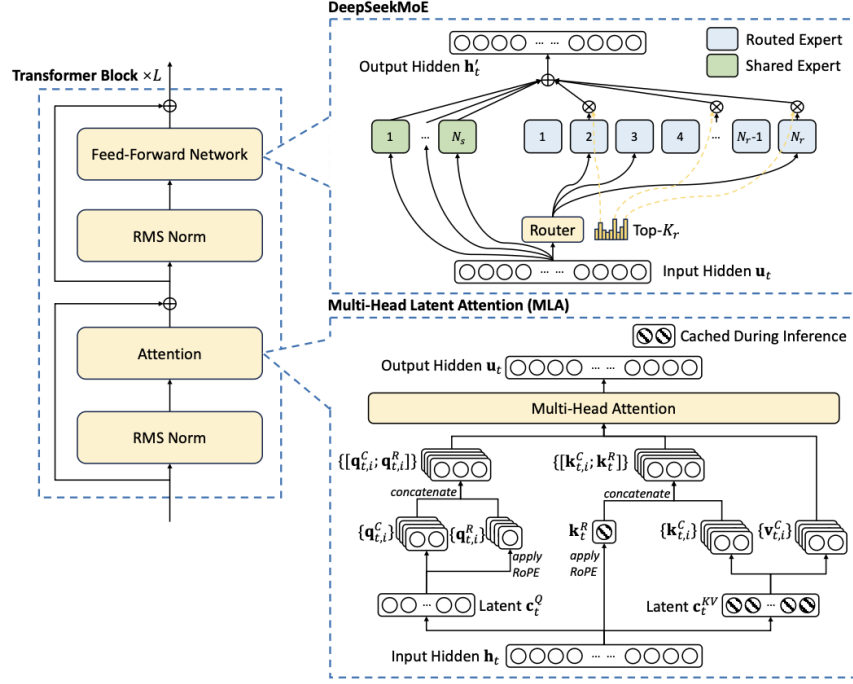


Figure 3.3: Illustration of the architecture of DeepSeek-V2. MLA ensures efficient inference by significantly reducing the KV cache for generation, and DeepSeekMoE enables training strong models at an economical cost through the sparse architecture.[18]

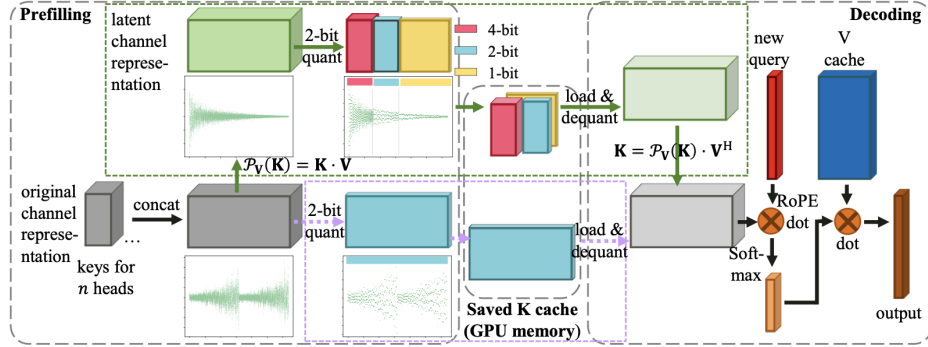


Figure 3.4: Diagram of the SVDq method (path in green) versus direct per-channel quantization (dashed path in violet). [21]

methods. Conceptually, it also reinforces a recurring design pattern in this chapter: practical KV compression often relies on static, calibration-informed transformations that trade adaptivity for efficiency and implementation simplicity.

Relationship to Representation-Level Compression. These methods are largely complementary to post-training low-rank compression. Architectural designs internalize compression at training time, while quantization-oriented methods reduce precision or reshape channel energy. In practical pipelines, these directions can be combined with representation-level projections under joint memory budgets.

3.3 Systems and Runtime KV-Cache Optimization

While a substantial portion of the literature approaches KV-cache compression through representational low-rank approximation, an orthogonal research direction targets the *systems-level* efficiency of autoregressive decoding. Rather than reducing the intrinsic dimensionality of key and value representations, these methods address the cost of *memory movement* itself, focusing on bandwidth limitations, cache layout, and token-level storage strategies as primary bottlenecks during inference [4, 23, 24].

Recent studies have shown that large language model decoding is often memory-bound rather than compute-bound, particularly under long-context generation [4]. Systems-oriented approaches therefore seek to reduce latency by optimizing how KV-cache data is stored, accessed, and streamed through the attention pipeline, without necessarily modifying the embedding space or projection structure. The following discussion distinguishes storage-level techniques, runtime retention policies, and kernel-level attention acceleration.

Memory-Efficient Decoding Mechanisms. A complementary line of work targets KV-cache handling through memory layout, data movement, and cache-locality optimizations during attention execution, rather than through representational dimensionality reduction.

Block-based cache managers reorganize KV-cache storage into non-contiguous memory blocks, enabling efficient paging and reducing fragmentation during long-context generation [4]. In parallel, streaming and eviction-based decoding strategies maintain bounded memory growth by selectively retaining or discarding cached tokens according to recency or importance criteria [24]. Although these techniques do not rely on explicit subspace projection, they highlight that KV-cache optimization spans multiple axes, ranging from representational compression to systems-level memory management.

Content-Aware KV Eviction. Beyond layout-level optimization, several methods design explicit token retention policies under a fixed KV budget. H₂O [25] and Scissorhands [26] formalize the intuition that only a small subset of tokens remains

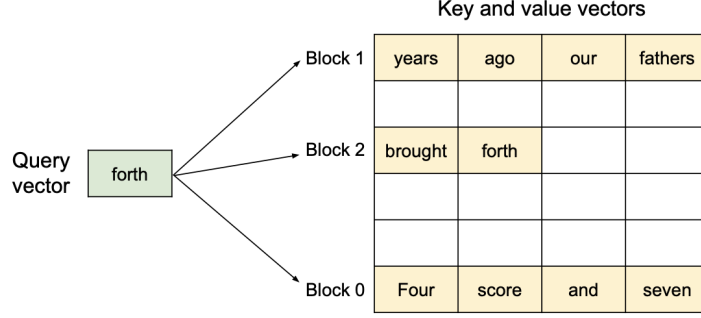


Figure 3.5: Illustration of a block-based KV-cache layout, where attention key and value vectors are stored as non-contiguous memory blocks [30].

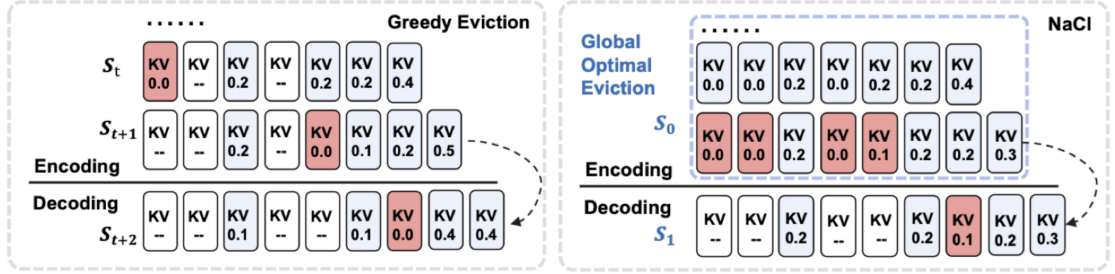


Figure 3.6: Traditional eviction algorithms perform step-by-step greedy search for tokens for eviction. NaCl framework searches globally for tokens within a chunk and then performs one single eviction [27].

consistently influential during generation, combining recency with importance-aware retention. NaCl extends this direction by proposing a unified eviction framework with strong long-context empirical behavior under single-pass compression constraints [27]. CORM and SnapKV introduce more query- or context-dependent criteria, respectively exploiting temporal similarity across queries and prompt-phase structure to improve cache selection efficiency [28, 29]. Together, these approaches shift KV compression from purely static projections toward budgeted inference-time control policies.

From the perspective adopted in this thesis, systems-oriented methods address *how* cached representations are stored and accessed, whereas low-rank projection methods focus on *what* information is preserved within each representation. These directions are therefore largely orthogonal and potentially complementary in practical deployment pipelines [4, 6].

Attention-Efficient Implementations. A related research direction focuses on improving the efficiency of attention execution through IO-aware kernel design rather than representation compression. IO-aware attention kernels reformulate the attention computation to minimize data movement between high-bandwidth memory (HBM) and on-chip SRAM by using tiling strategies and recomputation schemes that avoid materializing the full attention matrix. By reducing memory traffic and improving cache locality, these kernels significantly increase throughput during autoregressive decoding [23, 31].

Although such approaches do not reduce the dimensionality of the KV-cache itself, they highlight that inference efficiency is strongly influenced by memory-access patterns. From this perspective, low-rank projection methods and efficient attention kernels address complementary aspects of the same bottleneck: projection-based compression reduces the size of cached representations, whereas kernel-level optimizations reduce the cost of accessing them. Combining both strategies is therefore a viable direction toward scalable long-context inference [4, 23].

Relationship to Low-Rank Compression. Systems-oriented approaches are largely orthogonal to low-rank projection methods. Dimensionality reduction techniques operate at the level of representation geometry, decreasing the dimensionality of cached keys and values and therefore reducing the memory footprint of each token. In contrast, memory-efficient decoding mechanisms optimize how these representations are stored, scheduled, and accessed, primarily targeting the bandwidth and latency constraints that arise during autoregressive inference.

In practice, these systems techniques can also interact with head-sharing designs such as multi-query or grouped-query attention, which reduce the KV state before paging or kernel optimizations are applied [11, 10].

From a systems perspective, the two directions address complementary sources of inefficiency. Projection-based methods reshape the structure of the cached tensors themselves, while kernel-level and storage-level optimizations reshape the execution dynamics of attention. As a result, a model employing learned projection bases may benefit further from IO-aware kernels, optimized cache layouts, or streaming strategies, since reductions in tensor dimensionality directly translate into lower memory traffic.

Recognizing this distinction situates low-rank compression within a broader landscape of KV-cache optimization, where algorithmic approximation and systems-level design interact to enable scalable long-context inference without fundamentally altering the attention paradigm [4, 32].

3.4 Inter-Layer Redundancy and Sharing

A complementary line of research investigates KV-cache compression by exploiting redundancy *across* Transformer layers rather than within individual attention blocks. Inter-layer redundancy refers to the empirical observation that key and value representations at different depths often occupy partially overlapping subspaces, reflecting the progressive and incremental nature of representation refinement in deep Transformer stacks [32]. Under this perspective, storing fully independent KV caches for every layer introduces structural duplication that can be reduced through shared representations or joint encoding strategies, in a way conceptually related to head-level sharing mechanisms [10, 11].

Methods in this category assume that depth-wise continuity in the representation space induces compressible correlations between layers. Instead of reducing the dimensionality of each cache independently, they construct shared bases, grouped projections, or cross-layer encoding schemes that amortize memory across multiple layers. Such approaches target a distinct source of redundancy compared to per-layer low-rank approximation: rather than modifying the geometry of individual KV tensors, they reduce the number of independent representations that must be stored throughout the network.

By leveraging cross-layer subspace alignment, inter-layer compression strategies achieve memory savings that are largely orthogonal to dimensionality reduction within a single layer. As a result, these methods are often composable with projection-based techniques, suggesting that depth-wise sharing and low-rank approximation operate along complementary axes of KV-cache optimization [32].

Subspace Alignment Across Depth. The theoretical motivation behind inter-layer compression stems from empirical observations of subspace similarity across decoder depth [32]. Analyses of Transformer representations suggest that adjacent layers often operate on partially aligned subspaces, indicating that the evolution of key and value representations may be gradual rather than entirely independent across layers. Under this perspective, successive layers refine existing structures instead of introducing completely new representational directions.

Such observations motivate the hypothesis that a shared projection basis—or a small set of layer-group-specific bases—can capture a significant portion of the KV-cache structure across multiple depths. Importantly, this interpretation reflects an empirical tendency rather than a strict theoretical guarantee, and the degree of alignment may vary across architectures, datasets, and calibration procedures. Existing studies report that overlap can become more pronounced in intermediate regions of the network, although this behavior should be understood as model-dependent rather than universal.

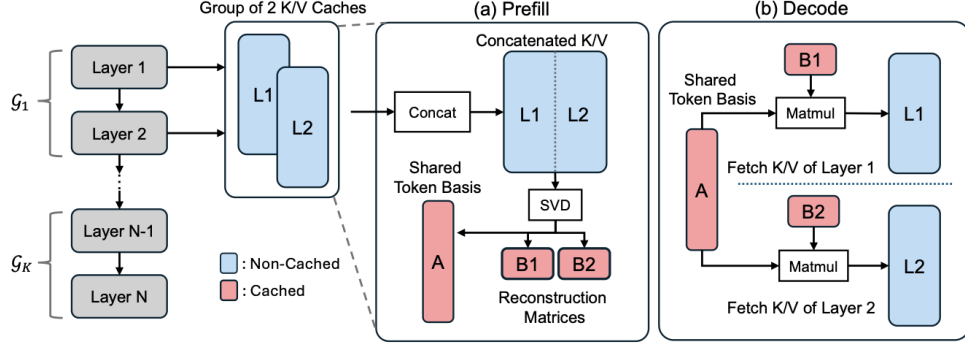


Figure 3.7: Illustration of the xKV for compressing KV-Cache [32].

Cross-Layer Low-Rank Compression. The xKV method [32] operationalizes the inter-layer redundancy hypothesis by constructing shared low-rank bases across groups of consecutive Transformer layers. Instead of computing independent decompositions for each layer, xKV aggregates calibration data from multiple depths and derives a common projection subspace through cross-layer singular value decomposition. The resulting shared basis captures depth-wise structural alignment, enabling a reduction in the number of independent KV representations that must be stored during inference.

From a methodological perspective, xKV assumes that redundancy emerges primarily from continuity across depth rather than from variance within a single layer. By amortizing basis vectors across several layers, the approach reduces memory overhead while preserving the dominant cross-layer directions of variation. The grouping granularity—ranging from small layer pairs to larger depth intervals—controls the trade-off between compression efficiency and layer-specific flexibility, with larger groups yielding stronger memory savings at the cost of reduced adaptability to local representation differences.

Layer-Adaptive KV Budgeting. A related perspective replaces shared subspaces with non-uniform cache allocation across depth. PyramidKV and PyramidInfer report that useful context retention can decrease in deeper layers, motivating pyramidal budget schedules that keep more KV states in early layers and progressively prune later ones [33, 34]. While this family does not require explicit cross-layer SVD, it is aligned with the same structural hypothesis underlying inter-layer methods: importance is not uniform across depth, and compression quality improves when layer-wise asymmetry is modeled explicitly.

Sharing Principles: From Heads to Layers. As contextual precedents (rather than direct inter-layer KV-cache compression methods), representation-sharing

ideas in attention extend across multiple structural axes. Grouped Query Attention (GQA) [10] reduces KV storage requirements by sharing key and value projections across multiple attention heads within a single layer. Although GQA operates within a layer rather than across depth, it provides an conceptual precedent: attention computations can reuse shared representations without a substantial loss of modeling capacity.

Extending this idea to the depth dimension, several architectural approaches explore parameter sharing across layers. Methods such as ALBERT and Universal Transformers [35, 36] tie or reuse weights across multiple layers, which can induce partially aligned representations across depth. While these architectures are not designed explicitly for KV-cache compression, they illustrate how structural sharing may reduce redundancy in deep Transformer stacks.

Taken together, these examples suggest that sharing mechanisms—whether across heads or across layers—provide a complementary perspective on memory reduction, highlighting redundancy as a property of the attention architecture rather than solely of individual representations.

Structural Redundancy in Deep Networks. The appeal of inter-layer approaches lies in their exploitation of a structural property frequently observed in deep Transformer models: the gradual, layerwise refinement of representations across depth. Empirical studies suggest that adjacent layers often operate on partially aligned subspaces, implying that redundancy may emerge as a byproduct of hierarchical feature refinement rather than as an input-specific phenomenon [32]. Under this perspective, redundancy is not solely tied to individual tokens or tasks but reflects broader regularities in the architecture itself. Compression strategies can therefore exploit this structure through static sharing or joint encoding schemes that amortize storage across layers. Inter-layer methods thus complement per-layer dimensionality reduction by targeting a distinct and largely orthogonal source of KV-cache redundancy.

Relationship to Per-Layer Methods. In summary, inter-layer approaches reduce memory by amortizing cache storage across depth rather than compressing individual layer representations in isolation. This targets a distinct yet related source of redundancy compared to the per-layer low-rank methods discussed earlier. While per-layer approximation reduces the dimensionality of keys and values within each layer’s embedding space, inter-layer sharing decreases the number of independent KV representations that must be maintained across the network. Rather than being strictly orthogonal, these two directions are largely complementary: a model may combine depth-wise sharing with within-layer dimensionality reduction to exploit multiple forms of redundancy simultaneously. Framing the landscape along these two axes helps clarify how existing approaches navigate the

trade-off between representation fidelity and memory efficiency. The following section examines the common assumptions that underlie both families of methods [32, 10].

3.5 Common Limitations of Existing Approaches

The methods reviewed in the preceding sections share a set of recurring design assumptions that shape the current landscape of KV-cache compression. While these assumptions enable analytically tractable solutions and efficient deployment, they also define implicit boundaries in terms of adaptivity and functional alignment. Rather than representing flaws, these characteristics reflect deliberate trade-offs between mathematical simplicity, stability, and system-level efficiency. This section synthesizes three cross-cutting themes that emerge across the literature: the reliance on static projection bases, the optimization of proxy objectives, and the limited degree of adaptation to input or task variation.

Static Projection Bases. A recurring characteristic of many existing low-rank and inter-layer compression methods is the use of projection bases that remain fixed throughout inference. Whether derived from singular value decomposition on calibration data, learned during pretraining, or shared across layer groups, these bases are typically computed offline and then applied uniformly to all inputs. This design choice offers clear practical advantages: static bases can be pre-integrated into model parameters, introduce no additional runtime overhead, and maintain predictable inference behavior. At the same time, the reliance on fixed subspaces implicitly assumes that a single projection remains sufficiently representative across varying inputs, tasks, and query distributions. Understanding this assumption is important for interpreting both the strengths and the boundaries of existing compression strategies [6, 16, 32, 18].

Optimization of Proxy Objectives. The selection of projection bases in many existing approaches is guided by objectives that act as proxies for attention quality. Common choices include minimizing reconstruction error in Frobenius norm, maximizing variance explained by retained components, or bounding the propagation of approximation error through the attention mechanism. These criteria are analytically tractable and often admit closed-form solutions via matrix decomposition, which makes them attractive from both theoretical and computational perspectives [6, 16, 17].

At the same time, such objectives implicitly emphasize statistical properties of the cached representations rather than their functional role within the attention

computation. In particular, directions that dominate variance or minimize reconstruction error are not guaranteed to coincide with those most influential for query–key interactions. As a result, the relationship between proxy reconstruction objectives and end-to-end model behavior remains an active area of investigation, especially under aggressive compression regimes where small geometric distortions may propagate through subsequent nonlinear transformations.

Limited Adaptivity to Context. A consequence of the preceding design choices is that many existing compression methods exhibit limited adaptivity along several axes. In most post-training low-rank approaches, projection bases are applied uniformly across inputs, meaning that the same subspace is used regardless of the content or structural properties of the processed sequence. Similarly, bases derived from calibration data are typically deployed across a wide range of downstream tasks without explicit task-specific adaptation [16, 32]. Recent eviction policies partially relax this assumption by introducing input-dependent retention decisions [25, 26, 27, 28, 29], but this gain in adaptivity typically comes with additional control-policy complexity and sensitivity to budget calibration.

Moreover, basis selection is commonly driven by statistics of keys and values alone, without direct consideration of the distribution of queries that will interact with the cache during inference. This uniformity simplifies deployment and avoids the computational overhead associated with runtime adaptation, contributing to the practical appeal of static compression schemes. At the same time, it highlights a broader design trade-off between efficiency and contextual specialization that remains an open topic within KV-cache optimization.

Design Trade-offs. It is important to recognize that these characteristics reflect deliberate design trade-offs rather than limitations in isolation. Static projection bases and proxy-driven objectives enable methods that are straightforward to implement, efficient to deploy, and compatible with existing inference infrastructure. In most approaches, the computational cost of basis selection is incurred once during calibration or pretraining, avoiding additional overhead during autoregressive decoding [17, 32].

Exploring alternative formulations—such as adaptive or learned projection schemes—may introduce additional computational complexity, potentially offsetting part of the efficiency gains that motivate KV-cache compression. Consequently, many existing methods prioritize stability, analytical tractability, and deployment simplicity over runtime adaptivity. Rather than representing a strict limitation, these choices delineate a region of the broader design space within which current solutions operate, highlighting the balance between efficiency and flexibility that underlies KV-cache optimization.

Open Questions. The preceding analysis highlights a conceptual gap between the objectives optimized during basis selection and the downstream behavior of compressed models. To what extent this gap can be reduced—for instance through compression strategies that more directly reflect the functional role of the KV-cache within the attention pipeline—remains an open research question. Addressing this challenge requires balancing potential gains in fidelity with the additional complexity introduced by moving beyond purely static or closed-form projection schemes. The next chapter builds upon these considerations by examining alternative formulations within this broader design space [16, 17, 32, 18].

Chapter 4

Method

4.1 Motivation and Preliminaries

Low-rank approximation methods for KV-cache compression typically employ Principal Component Analysis (PCA) or Singular Value Decomposition (SVD) to identify projection bases for Keys and Values. These approaches compute orthonormal bases that capture the directions of maximum variance in the activation tensors, under the implicit assumption that preserving variance in the projected representations will preserve the fidelity of attention computations. This section examines the limitations of this assumption and motivates the need for alternative formulations. The formulation developed in this chapter directly addresses the limitations synthesized in Section 3.5, with particular focus on the gap between variance-based proxy objectives and downstream attention behavior.

Reconstruction Error as a Proxy Objective. When SVD is applied to Key activations $\mathbf{K}$, the resulting projection basis minimizes the Frobenius norm of the reconstruction error $\|\mathbf{K} - \hat{\mathbf{K}}\|_F$, where $\hat{\mathbf{K}}$ denotes the reconstructed Keys after projection to a lower-dimensional subspace. While this objective has desirable mathematical properties—including a closed-form solution and guaranteed optimality for reconstruction error within the class of linear projections—it does not explicitly optimize for the quantity that determines attention behavior. The attention mechanism computes similarity scores via the product $\mathbf{Q}\mathbf{K}^\top$, and subsequently weights the Values according to these scores. Consequently, reconstruction fidelity on $\mathbf{K}$ serves only as a proxy for the actual objective of preserving attention outputs.

Misalignment Between Variance and Attention Sensitivity. The directions of maximum variance in $\mathbf{K}$ do not necessarily correspond to the directions that most influence the attention computation. Consider that the product $\mathbf{Q}\mathbf{K}^\top$ depends

on the alignment between Query and Key vectors. In such cases, directions in the Key space that exhibit high variance across sequence positions could in principle contribute minimally to the final attention scores if they happen to be approximately orthogonal to the Query subspace. Conversely, directions with lower variance could carry information relevant to distinguishing between tokens during attention computation. SVD-based methods implicitly assume that variance magnitude correlates with task relevance, an assumption that need not hold in practice.

Static Nature of Variance-Based Projections. A further limitation concerns the static character of SVD-based bases. These projections are computed from Key activations alone, without reference to the Queries that will interact with them or the Values that will be aggregated. The projection basis is determined entirely by the distributional properties of $\mathbf{K}$ over a calibration set, and remains fixed regardless of the downstream attention computation. This decoupling between basis selection and attention behavior means that the projection does not adapt to preserve information that is specifically relevant to the decoder layer’s functional role in the network.

Toward Task-Aware Projection Objectives. The preceding observations suggest that an alternative formulation may be preferable: rather than optimizing reconstruction error on intermediate representations, one might directly optimize for preservation of the decoder layer’s output behavior. Such an objective would measure the discrepancy between the outputs produced by the original decoder layer and those produced by the layer operating with compressed Key and Value representations. By targeting the downstream effect of compression rather than the compression fidelity itself, this formulation aligns the optimization objective with the actual quantity of interest.

This shift in perspective—from variance preservation to output preservation—forms the conceptual basis for the method developed in subsequent sections. The approach requires specifying how projection bases should be parameterized, how the output-based objective should be computed, and how optimization should proceed given that the objective depends on the full attention computation rather than on the Keys alone.

4.2 Problem Formulation

This section formalizes the KV-cache compression problem and defines the optimization objective that guides the design of the proposed method.

4.2.1 Target Objective

Consider a decoder layer in an autoregressive transformer, which receives hidden states and produces output representations through a self-attention mechanism. Let $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times d_h}$ denote the Query, Key, and Value matrices for a single attention head over a sequence of length n , where d_h is the per-head dimension. The goal of KV-cache compression is to reduce the memory footprint of storing $\mathbf{K}$ and $\mathbf{V}$ while preserving the functional behavior of the attention computation.

Orthonormal projection matrices $\mathbf{P}_K \in \mathbb{R}^{d_h \times r_K}$ and $\mathbf{P}_V \in \mathbb{R}^{d_h \times r_V}$ are sought, where $r_K < d_h$ and $r_V < d_h$, such that the compressed representations $\mathbf{K}^\downarrow = \mathbf{K}\mathbf{P}_K$ and $\mathbf{V}^\downarrow = \mathbf{V}\mathbf{P}_V$ can be stored in place of the full-dimensional tensors. Note that while this formulation presents $\mathbf{P}_K$ and $\mathbf{P}_V$ as single matrices for notational simplicity, the method may employ different granularities for Keys and Values (e.g., layer-level versus per-head projections); such details are deferred to the method description. At inference time, the compressed representations are reconstructed as $\tilde{\mathbf{K}} = \mathbf{K}^\downarrow \mathbf{P}_K^\top$ and $\tilde{\mathbf{V}} = \mathbf{V}^\downarrow \mathbf{P}_V^\top$ before computing attention. Accordingly, the method targets memory reduction of cached tensors while keeping the attention kernel unchanged; acceleration of attention compute is outside the scope of this chapter.

Rather than optimizing for reconstruction fidelity on $\mathbf{K}$ and $\mathbf{V}$ directly, the objective is formulated in terms of the decoder layer’s output behavior. Let $f_\ell(\cdot)$ denote the forward function of decoder layer ℓ , and let $\tilde{f}_\ell(\cdot; \mathbf{P}_K, \mathbf{P}_V)$ denote the same layer operating with compressed Key and Value representations. Given a calibration set $\mathcal{D} = \{x_1, \dots, x_m\}$, the target objective is:

$$\min_{\mathbf{P}_K, \mathbf{P}_V} \mathbb{E}_{x \in \mathcal{D}} \left[\frac{\|f_\ell(x) - \tilde{f}_\ell(x; \mathbf{P}_K, \mathbf{P}_V)\|_F}{\|f_\ell(x)\|_F} \right] \quad (4.1)$$

subject to $\mathbf{P}_K^\top \mathbf{P}_K = \mathbf{I}$ and $\mathbf{P}_V^\top \mathbf{P}_V = \mathbf{I}$. This formulation directly targets preservation of the decoder layer’s output, thereby aligning the optimization objective with the downstream effect of compression on model behavior.

Although $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ are introduced in a single-head notation for clarity, the optimization target is defined at the decoder-layer level through $f_\ell(\cdot)$. In implementation, this layer-level objective is instantiated with a hybrid granularity: one Key predictor shared across KV heads in each layer and head-specific Value predictors, as detailed in Section 4.3.

4.2.2 Practical Optimization Approximation

The objective in Equation (4.1) is formulated as a joint optimization over Key and Value projection matrices. In principle, one could train the projection networks simultaneously by minimizing the decoder-layer output discrepancy with respect to both $\mathbf{P}_K$ and $\mathbf{P}_V$. However, preliminary calibration experiments indicated

that direct joint optimization produced unstable training dynamics and slower convergence under the decoder-output loss.

Empirically, simultaneous updates of Key and Value projections introduced high-variance gradient interactions. Because both projections modify the attention pathway, early-stage optimization exhibited coupled distortions in attention logits and value aggregation, which reduced the reliability of the optimization signal. In particular, when both projections were updated concurrently from random initialization, the loss landscape displayed weaker directional improvement compared to configurations where one component was optimized while the other remained fixed. To improve stability, a decoupled training strategy was adopted. Key and Value projection networks are optimized independently during calibration, each minimizing the decoder-layer output discrepancy while the other component remains fixed. This choice can be interpreted as a practical approximation to the joint objective, analogous to block-coordinate optimization schemes that reduce coupling between parameter groups.

After independent training, the interaction between Key and Value ranks is reintroduced during evaluation through the construction of the per-layer error surface $\Delta_\ell(r_K, r_V)$. This evaluation stage restores the joint perspective of Equation (4.1) without incurring the instability observed in fully simultaneous training.

This approximation preserves the conceptual goal of output-level alignment while providing a more stable calibration procedure under limited compute budgets.

4.2.3 Limitations of PCA-Based Compression

Principal Component Analysis provides projection bases by computing the Singular Value Decomposition of the Key matrix and retaining the leading right singular vectors. Formally, given $\mathbf{K} = \mathbf{W}\mathbf{\Sigma}\mathbf{Z}^\top$ where $\mathbf{Z}$ contains the right singular vectors, PCA selects $\mathbf{P}_K^{\text{PCA}}$ as the first r_K columns of $\mathbf{Z}$. This basis minimizes the reconstruction error:

$$\mathbf{P}_K^{\text{PCA}} = \arg \min_{\mathbf{P}: \mathbf{P}^\top \mathbf{P} = \mathbf{I}} \|\mathbf{K} - \mathbf{K}\mathbf{P}\mathbf{P}^\top\|_F \quad (4.2)$$

The objective in Equation (4.2) differs from the target objective in Equation (4.1) in a fundamental way: PCA optimizes for preservation of $\mathbf{K}$ itself, whereas the target objective optimizes for preservation of the decoder layer output $f_\ell(x)$. The decoder layer output depends on the interaction between $\mathbf{Q}$ and $\mathbf{K}$ through the attention scores $\text{softmax}(\mathbf{Q}\mathbf{K}^\top/\sqrt{d_h})$, and subsequently on the weighted aggregation of $\mathbf{V}$. Directions in $\mathbf{K}$ that contribute maximally to variance may not correspond to directions that most influence the attention computation or the final output.

Furthermore, PCA computes the projection basis from $\mathbf{K}$ alone, without access to the Query matrix $\mathbf{Q}$ that determines which Key directions are relevant for

attention, or to the Value matrix $\mathbf{V}$ whose aggregation produces the output. This decoupling means that PCA-based compression does not account for the structure of the downstream computation when selecting which directions to preserve.

4.2.4 Design Goals

The preceding formulation motivates several concrete design goals for a compression method:

1. **Task-awareness.** The projection bases should be selected to minimize degradation of the decoder layer output, not merely to preserve variance in the Key or Value representations. This requires an objective that evaluates compression quality through the full attention computation.
2. **Compatibility with pretrained weights.** The compression method should operate without modifying the original model parameters. The projection matrices should be applicable as external transformations that can be composed with existing weight matrices at deployment time, leaving the base model unchanged.
3. **Offline calibration.** The projection bases should be computed once during a calibration phase using representative data, then fixed for inference. This avoids runtime overhead from adaptive basis selection and enables deployment as static weight modifications.
4. **Orthonormality.** The projection matrices should satisfy $\mathbf{P}^\top \mathbf{P} = \mathbf{I}$ to ensure that the projection and reconstruction operations preserve geometric properties and maintain numerical stability. This constraint also simplifies the relationship between compression ratio and rank.
5. **Layer-wise processing.** The method should process layers sequentially, determining projection bases for each layer independently. This enables the calibration procedure to operate within bounded memory.
6. **Preservation of Query pathway.** The Query projection should remain unmodified, as Queries are not cached and their compression would not reduce KV-cache memory. This simplifies the optimization by restricting the search to Key and Value projections only.

These design goals define the constraints within which a solution to the optimization problem in Equation (4.1) must be realized. The following section describes how these goals are instantiated in a concrete method. This design explicitly prioritizes functional alignment of compression (output-preservation objective) while retaining static, offline-calibrated bases for deployment simplicity.

4.3 Neural Projection Network

This section describes the neural network architecture that generates projection matrices for Keys and Values. The network takes distributional statistics of the activation tensors as input and produces a projection matrix as output.

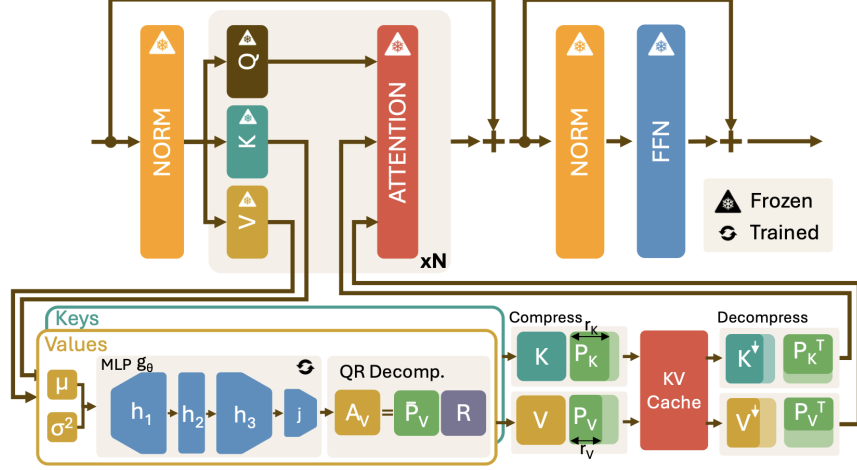


Figure 4.1: Overview of the Architecture. From lightweight activation statistics μ_K , μ_V and σ_K^2 , σ_V^2 , orthonormal projection bases $\mathbf{P}_K$, $\mathbf{P}_V$ are learned to minimize decoder-layer output error.

4.3.1 Input Features

The network receives as input a fixed-dimensional feature vector derived from the Key activations $\mathbf{K} \in \mathbb{R}^{n \times d_h}$, where n is the sequence length and d_h is the per-head dimension. Two summary statistics are computed along the sequence dimension: the per-dimension mean $\mu_K \in \mathbb{R}^{d_h}$ and the per-dimension variance $\sigma_K^2 \in \mathbb{R}^{d_h}$. These two vectors are concatenated to form the input feature vector:

$$\mathbf{s}_K = [\mu_K; \sigma_K^2] \in \mathbb{R}^{2d_h} \quad (4.3)$$

This representation has a fixed dimensionality of $2d_h$ regardless of the sequence length n . The aggregation over the sequence dimension ensures that the network input size remains constant across inputs of varying length. The same computation is performed for Values to obtain the corresponding feature vector $\mathbf{s}_V$. Query activations are not used as explicit predictor inputs; their effect is captured indirectly through the output-level loss in Equation (4.9).

4.3.2 Encoder Architecture

The encoder is a multi-layer perceptron (MLP) that transforms the input features into an intermediate representation. The architecture consists of three hidden layers. Each hidden layer $i \in \{1, 2, 3\}$ applies a linear transformation followed by layer normalization and GELU activation:

$$\mathbf{h}_i = \phi(\text{LayerNorm}(\mathbf{W}_i \mathbf{h}_{i-1} + \mathbf{b}_i)) \quad (4.4)$$

where $\mathbf{h}_0 \equiv \mathbf{s}$ denotes the input feature vector (either $\mathbf{s}_K$ or $\mathbf{s}_V$) and $\phi(\cdot)$ is the GELU non-linearity. The hidden dimensions are 512 for the first two layers and 256 for the third layer.

Layer normalization is applied after each linear transformation to stabilize the distribution of activations during calibration.

4.3.3 Projection Head

The projection head maps the encoder output to a matrix of appropriate dimensions. A final linear layer with parameters $(\mathbf{W}_{\text{out}}, \mathbf{b}_{\text{out}})$ projects from the 256-dimensional encoder output to a vector that is reshaped into a pre-orthonormalization matrix $\mathbf{A} \in \mathbb{R}^{d_h \times d_h}$:

$$\mathbf{A} = \text{reshape}(\mathbf{W}_{\text{out}} \mathbf{h}_3 + \mathbf{b}_{\text{out}}) \in \mathbb{R}^{d_h \times d_h} \quad (4.5)$$

At this stage, the matrix $\mathbf{A}$ is unconstrained; its columns are neither orthogonal nor normalized. The enforcement of orthonormality constraints on the projection matrix is described in Section 4.4.

The same network architecture is used for both Key and Value projections, though with different instantiation granularity. For Keys, a single network instance g_{θ_K} is trained per layer and shared across all KV heads within that layer. For Values, separate network instances $g_{\theta_{V,h}}$ are trained for each KV head h , reflecting that values interact with head-specific output-projection pathways.

4.4 Orthonormality Constraint

The projection matrices produced by the neural network are required to satisfy an orthonormality constraint. Specifically, for a projection matrix $\mathbf{P} \in \mathbb{R}^{d_h \times r}$, orthonormality is enforced as:

$$\mathbf{P}^\top \mathbf{P} = \mathbf{I}_r \quad (4.6)$$

where $\mathbf{I}_r$ denotes the $r \times r$ identity matrix. This condition ensures that the columns of $\mathbf{P}$ form an orthonormal set.

Orthonormal projection matrices offer several practical advantages. In the row-wise convention used throughout this thesis, the projection operation is $\mathbf{x} \mapsto \mathbf{x}\mathbf{P}$ and

the reconstruction operation is $\mathbf{z} \mapsto \mathbf{z}\mathbf{P}^\top$; this is equivalent to the column-vector form $\mathbf{x} \mapsto \mathbf{P}^\top \mathbf{x}$. With $\mathbf{P}^\top \mathbf{P} = \mathbf{I}$, these operations preserve the Euclidean norm of components within the projected subspace, which contributes to numerical stability during both calibration and inference. Additionally, the compression ratio is directly determined by the rank r relative to the original dimension d_h , without confounding effects from non-uniform scaling.

Enforcement via QR Decomposition. Rather than imposing orthonormality through soft constraints such as regularization penalties, or through constrained optimization on the Stiefel manifold [37], the implementation enforces orthonormality via a hard projection step. The matrix $\mathbf{A} \in \mathbb{R}^{d_h \times d_h}$ produced by the projection head (Section 4.3.3) is passed through QR decomposition:

$$\mathbf{A} = \mathbf{Q}\mathbf{R} \tag{4.7}$$

where $\mathbf{Q} \in \mathbb{R}^{d_h \times d_h}$ has orthonormal columns and $\mathbf{R} \in \mathbb{R}^{d_h \times d_h}$ is upper triangular. The orthonormal factor $\mathbf{Q}$ is retained as the full-rank orthonormal basis $\bar{\mathbf{P}}$. This decomposition is applied at the end of the network’s forward pass, immediately after the reshape operation. The QR decomposition supports gradient computation in matrix backpropagation pipelines [38], allowing gradients to propagate during calibration. This approach guarantees exact orthonormality at every forward pass without requiring additional loss terms or iterative projection procedures.

Rank Selection via Truncation. For each target rank r the low-rank projection matrix is obtained by truncation:

$$\mathbf{P}^{(r)} := \bar{\mathbf{P}}[:, 1:r] \in \mathbb{R}^{d_h \times r} \tag{4.8}$$

where $\bar{\mathbf{P}}[:, 1:r]$ denotes the first r columns of $\bar{\mathbf{P}}$. This truncation preserves the orthonormality constraint: $(\mathbf{P}^{(r)})^\top \mathbf{P}^{(r)} = \mathbf{I}_r$.

4.5 Training Strategy

This section describes how the neural projection networks are trained to produce projection matrices for Keys and Values. Training occurs offline during a calibration phase that is decoupled from both the original model’s pretraining and the subsequent inference deployment.

4.5.1 Loss Function

The projection networks are trained to minimize the discrepancy between the outputs of the original decoder layer and the outputs of the layer operating with

compressed Key and Value representations. For decoder layer ℓ , let $f_\ell(x)$ denote the output when processing input x with full-dimension Keys and Values, and let $\tilde{f}_\ell(x; r_K, r_V, \theta)$ denote the output when using compressed representations at ranks (r_K, r_V) with predictor parameters θ . The loss function is the relative decoder-layer output error:

$$\Delta_\ell(r_K, r_V; \theta) = \mathbb{E}_{x \in \mathcal{D}} \left[\frac{\|f_\ell(x) - \tilde{f}_\ell(x; r_K, r_V, \theta)\|_F}{\|f_\ell(x)\|_F} \right] \quad (4.9)$$

The reference signal $f_\ell(x)$ is pre-computed by passing calibration samples through the original, unmodified layer. The compressed output $\tilde{f}_\ell(x; r_K, r_V, \theta)$ is computed by applying the learned projection matrices to the Keys and Values before performing attention. Normalization by the reference norm ensures that the loss magnitude is comparable across layers with different activation scales.

The Key and Value projection networks are trained independently. For each candidate rank, a separate predictor is trained by minimizing the loss in Equation (4.9) via gradient descent. This separation allows the method to precompute, for each layer, a per-layer error surface $\Delta_\ell(r_K, r_V)$ over all candidate rank pairs, which can subsequently be used for rank allocation at deployment time. Relative to the joint optimization in Equation (4.1), this is an approximation that reduces calibration complexity. The coupling between Key and Value ranks is reintroduced during the evaluation stage, where $\Delta_\ell(r_K, r_V)$ is computed over all rank pairs and used as the final criterion for budgeted deployment.

4.5.2 Offline Calibration Protocol

Calibration is performed offline, prior to deployment, using a representative set of input samples drawn from a calibration corpus. During calibration, the following components remain fixed:

- All original model weights, including embeddings, attention projections, feed-forward layers, and normalization parameters.

The only trainable parameters are those of the neural projection networks described in Section 4.3. This separation ensures that the base model is never modified, preserving compatibility with closed-weight or proprietary models where direct weight modification is not permitted.

Calibration proceeds layer by layer. For each decoder layer ℓ :

1. The layer’s uncompressed input activations are recorded via a forward pass on the calibration set.
2. For each candidate Key rank r_K , the Key predictor $g_{\theta_K}^{(\ell, r_K)}$ is trained to minimize Equation (4.9).

Algorithm 1 Protocol and error-surface construction

Require: Decoder layers $\mathcal{L}$; calibration set $\mathcal{X}$; candidate ranks $\mathcal{R}_K, \mathcal{R}_V$
Ensure: Key bases $\{\bar{\mathbf{P}}_K^{(\ell, r_K)}\}_{\ell, r_K}$, value bases $\{\bar{\mathbf{P}}_{V,h}^{(\ell, r_V)}\}_{\ell, r_V, h}$, error surfaces $\{\Delta_\ell(r_K, r_V)\}_{\ell, r_K, r_V}$

- 1: **for** $\ell \in \mathcal{L}$ **do**
- 2: Perform forward pass on $\mathcal{X}$ and record activations $\mathcal{X}^{(\ell)}$
- 3: **for** $r_K \in \mathcal{R}_K$ **do**
- 4: Train $g_{\theta_K}^{(\ell, r_K)}$ on $\mathcal{X}^{(\ell)}$ to solve Eq. (4.9)
- 5: Obtain $\bar{\mathbf{P}}_K^{(\ell, r_K)}$ and truncated $\mathbf{P}_K^{(\ell, r_K)}$
- 6: **end for**
- 7: **for** $r_V \in \mathcal{R}_V$ **do**
- 8: Train $\{g_{\theta_{V,h}}^{(\ell, r_V)}\}_{h=1}^{H_{KV}}$ jointly on $\mathcal{X}^{(\ell)}$ to solve Eq. (4.9)
- 9: Obtain $\{\bar{\mathbf{P}}_{V,h}^{(\ell, r_V)}\}_{h=1}^{H_{KV}}$ and truncated $\{\mathbf{P}_{V,h}^{(\ell, r_V)}\}_{h=1}^{H_{KV}}$
- 10: **end for**
- 11: **for** $(r_K, r_V) \in \mathcal{R}_K \times \mathcal{R}_V$ **do**
- 12: Evaluate $\Delta_\ell(r_K, r_V)$ on $\mathcal{X}^{(\ell)}$ using $\mathbf{P}_K^{(\ell, r_K)}$ and $\{\mathbf{P}_{V,h}^{(\ell, r_V)}\}_{h=1}^{H_{KV}}$
- 13: **end for**
- 14: **end for**

3. For each candidate Value rank r_V , the Value predictors $\{g_{\theta_{V,h}}^{(\ell, r_V)}\}$ are trained jointly across all KV heads.
4. The per-layer error surface $\Delta_\ell(r_K, r_V)$ is evaluated over all rank combinations.

Once training for a given layer completes, the learned projection bases are saved. This sequential protocol enables the calibration procedure to operate within bounded memory.

After calibration, the learned projection bases are fixed and can be deployed alongside the original model. At inference time, no additional training occurs; the projection matrices are applied as static transformations to the Key and Value representations within each layer. The precomputed error surfaces enable flexible rank allocation under user-specified error budgets without retraining.

4.5.3 Empirical Protocol Decisions

Several implementation choices used in the final protocol were not fixed a priori, but selected through exploratory calibration runs later reported in Chapter 6. Two decisions were particularly influential.

First, Value predictors were trained jointly across KV heads rather than independently. Independent per-head training produced weak effective supervision under

decoder-output loss because each update modified only a fraction of the full layer output path. Joint optimization increased signal strength and improved calibration efficiency.

Second, the optimization used asymmetric mini-batch sizing: batch 1 for Key predictors and batch 4 for Value predictors. This choice reflects distinct optimization behavior observed during development: Key training benefited from smaller batches, while Value training was more numerically stable with moderate batch aggregation. The resulting asymmetric setting balances training stability and end-to-end quality under the same calibration budget.

For reproducibility, the exploratory protocol operated on 128 calibration sequences at fixed sequence length (1024), with layer-wise training performed offline and the base model kept frozen throughout calibration. These constraints define the empirical regime in which the method-level decisions above were established under limited compute resources.

After these choices were selected, the final evaluation campaign was executed with the larger protocol reported in Chapter 5 (512 calibration sequences, sequence length 2048). This two-stage process separates development-time design selection from final benchmark reporting.

Chapter 5

Experimental Setup

This chapter describes the experimental protocol used to evaluate the proposed neural projection approach for KV-cache compression. The chapter details the model under evaluation, calibration and training procedures, datasets, evaluation metrics, baseline comparisons, and rank allocation policies.

The chapter reports the *final* evaluation configuration used for the main results (server-scale setup). Earlier development-stage experiments performed under tighter compute constraints (including sequence length 1024) are discussed separately in the ablation section of Chapter 6 and are not part of the protocol defined here.

5.1 Model

The proposed method is evaluated on **Llama-3-8B** [3], a decoder-only transformer architecture. The architectural parameters relevant to KV-cache compression are summarized in Table 5.1.

Table 5.1: Architectural configuration of Llama-3-8B relevant to KV-cache compression.

Parameter	Value
Number of layers	32 decoder layers
Hidden dimension	4096
Number of attention heads	32 query heads
Number of KV heads	8 (Grouped-Query Attention, GQA)
Per-head dimension (d_h)	128
Model precision	FP16

The model uses Grouped-Query Attention (GQA), where multiple query heads

share the same key-value head. This architectural choice reduces the KV-cache footprint relative to standard multi-head attention while maintaining comparable performance.

All original model weights remain **frozen** throughout calibration. Only the lightweight MLP predictors that generate projection bases are trained.

5.2 Calibration and Training

5.2.1 Calibration Data

Projection bases and error surfaces are obtained post-training using a held-out calibration set derived from the WikiText-2 training split [39]. This dataset provides a controlled evaluation setting with coherent long-form text and is widely used in studies of attention compression and projection-based methods. The calibration set consists of 512 sequences of length 2048 tokens, randomly sampled from the tokenized corpus. It is used both for training the basis predictors and for constructing the layer-wise error surfaces employed in rank allocation.

5.2.2 Rank Configurations

The basis predictor is trained independently for **5 down-projection ranks** uniformly spaced between 50% and 90% of the per-head dimension $d_h = 128$:

$$r \in \{64, 77, 90, 102, 115\} \approx \{0.50, 0.60, 0.70, 0.80, 0.90\} \times d_h \quad (5.1)$$

Ranks were selected to cover moderate compression regimes where decoder stability remains measurable under output-level loss, avoiding extreme regimes dominated by failure modes. For each layer ℓ and each candidate rank pair $(r_K, r_V) \in \mathcal{R}_K \times \mathcal{R}_V$, the layer-wise reconstruction error $\Delta_\ell(r_K, r_V)$ is computed and stored, forming an error surface that enables rank allocation at deployment time without retraining.

5.2.3 Training Protocol

Each layer is calibrated independently using the following procedure:

1. Forward pass on calibration set to record layer inputs $\mathcal{X}^{(\ell)}$
2. For each candidate key rank r_K :
 - Train the key basis predictor $g_{\theta_K}^{(\ell, r_K)}$ to minimize decoder-layer output error
 - Obtain orthonormal basis $\bar{\mathbf{P}}_K^{(\ell, r_K)}$ via QR decomposition

3. For each candidate value rank r_V :
 - Train value basis predictors $\{g_{\theta_{V,h}}^{(\ell, r_V)}\}_{h=1}^{H_{KV}}$ jointly for all KV heads
 - Obtain per-head orthonormal bases $\{\bar{\mathbf{P}}_{V,h}^{(\ell, r_V)}\}_{h=1}^{H_{KV}}$
4. Evaluate $\Delta_\ell(r_K, r_V)$ over all rank pairs and store the error surface

Head Sharing Strategy

Following the design in the original paper:

- **Keys:** A single shared projection basis per layer ($\bar{\mathbf{P}}_K$), shared across all KV heads
- **Values:** Per-head projection bases ($\bar{\mathbf{P}}_{V,h}$ for each head h)

This asymmetry reflects that keys are reused across multiple query heads in GQA, while values interact with head-specific output-projection pathways.

5.2.4 Optimization Hyperparameters

Table 5.2: Training hyperparameters for basis predictor optimization.

Parameter	Value
Optimizer	AdamW
Learning rate	5×10^{-3}
Weight decay	10^{-4}
LR schedule	Cosine annealing
Schedule $T_{\max}$	Total training steps
Batch size (keys)	1
Batch size (values)	4
Maximum epochs	50
Early stopping patience	5 epochs
Early stopping δ	10^{-6}

The asymmetric batch sizes reflect empirical observations that value bases exhibit more numerical instability during training; larger batch sizes provide more stable gradients for the multi-head value optimization.

5.2.5 Loss Function

The basis predictor is trained to minimize the relative decoder-layer output reconstruction error:

$$\Delta_\ell(r_K, r_V; \theta) = \mathbb{E}_{x \in \mathcal{D}} \left[\frac{\|f_\ell(x) - \tilde{f}_\ell(x; r_K, r_V, \theta)\|_F}{\|f_\ell(x)\|_F} \right] \quad (5.2)$$

where f_ℓ denotes the original decoder layer mapping and $\tilde{f}_\ell$ denotes the layer with compressed KV representations. This objective directly targets decoder-layer output fidelity rather than proxy objectives based on intermediate attention quantities.

5.3 Datasets

5.3.1 Language Modeling Evaluation

Perplexity is measured on two datasets chosen to enable direct comparability with prior work on post-training KV-cache compression, in particular EigenAttention [6], which serves as the primary projection-based baseline in this thesis.

- **WikiText-2** [39]: Test split, comprising curated Wikipedia articles. Using WikiText-2 allows a direct comparison with the experimental protocol adopted by EigenAttention.
- **C4** [40]: Validation split, consisting of large-scale web-crawled text. Compared to WikiText-2, C4 introduces broader domain variability and noisier distributions, providing a complementary evaluation of robustness under heterogeneous language modeling conditions. C4 is also employed in EigenAttention, enabling matched evaluation settings across methods.

The selection of these datasets therefore follows a dual rationale: (i) maintaining alignment with the experimental design of the primary baseline to ensure fair comparison, and (ii) evaluating performance across both curated and open-domain text distributions.

Both datasets are evaluated with a sequence length of 2048 tokens to match the calibration configuration and avoid confounding effects arising from context-length mismatch.

5.3.2 Zero-Shot Evaluation

Zero-shot multiple-choice accuracy is evaluated using the LM Evaluation Harness framework [41] on the following benchmarks:

- **HellaSwag** [42]: Commonsense natural language inference requiring completion of sentences describing everyday situations. Each example presents four answer options.
- **PIQA** [43]: Physical commonsense reasoning about everyday physical interactions. Each question provides two candidate solutions.
- **MMLU** [44]: Massive Multitask Language Understanding, covering 57 subjects across STEM, humanities, social sciences, and other domains. Each question consists of four answer choices.

5.4 Metrics

5.4.1 KV-Cache Compression Ratio

The per-token compression ratio is defined as:

$$\text{CR} = \frac{r_K + r_V}{2d_h} \quad (5.3)$$

where r_K and r_V are the retained ranks for keys and values respectively, and d_h is the original per-head dimension. A compression ratio of 0.5 indicates that the KV-cache occupies half its original memory footprint.

5.4.2 Perplexity

Language modeling quality on WikiText-2 (test split) and C4 (validation split) is measured via perplexity, computed as:

$$\text{PPL} = \exp \left(\frac{1}{N} \sum_{i=1}^N \mathcal{L}_{\text{CE}}(x_i) \right) \quad (5.4)$$

where $\mathcal{L}_{\text{CE}}$ is the cross-entropy loss and N is the total number of evaluated tokens.

5.4.3 Zero-Shot Accuracy

For multiple-choice benchmarks, performance is measured using *zero-shot accuracy*, defined as the fraction of evaluation examples for which the model assigns the highest conditional log-likelihood to the correct answer choice among the available options.

Given an input prompt with candidate completions $\{c_1, \dots, c_M\}$, the model computes the token-level log-likelihood of each completion conditioned on the prompt. The predicted answer corresponds to the candidate with maximum likelihood, and

accuracy is computed as the proportion of correct selections over the full evaluation set.

All zero-shot evaluations are conducted using the LM Evaluation Harness [41], ensuring consistent prompting, normalization, and scoring procedures across datasets and compression configurations.

5.4.4 Reconstruction Metrics

For diagnostic analysis, reconstruction metrics are computed on a subset of 64 sequences drawn from the WikiText-2 corpus. These samples correspond to calibration data used during basis training; however, the same evaluation procedure applies to unseen test sequences. For this analysis, all layers are compressed using a fixed compression ratio, and reconstruction metrics are computed layer-wise by comparing the original decoder-layer outputs y_ℓ with their compressed counterparts $\tilde{y}_\ell$.

- **Relative Frobenius-norm error:** Measures magnitude of reconstruction error

$$\text{RelErr}_\ell = \frac{\|y_\ell - \tilde{y}_\ell\|_F}{\|y_\ell\|_F} \quad (5.5)$$

- **Cosine similarity:** Measures directional agreement between original and reconstructed outputs

$$\text{CosSim}_\ell = \frac{1}{T} \sum_{t=1}^T \frac{y_{\ell,t} \cdot \tilde{y}_{\ell,t}}{\|y_{\ell,t}\| \|\tilde{y}_{\ell,t}\|} \quad (5.6)$$

where T is the number of tokens.

5.5 Baseline

The approach is compared against **EigenAttention** [6], an SVD-based KV-cache compression method that serves as a strong post-training projection-based baseline.

5.5.1 EigenAttention Method

EigenAttention computes projection bases by performing SVD on the concatenated matrix $[\mathbf{K}; \mathbf{Q}]$:

$$\mathbf{Z} = \begin{bmatrix} \mathbf{K} \\ \mathbf{Q} \end{bmatrix} \in \mathbb{R}^{2n \times d_h} \quad (5.7)$$

The projection basis $\mathbf{P}_K$ is obtained from the top singular vectors of $\mathbf{Z}$, minimizing the reconstruction objective:

$$\min_{\mathbf{P}_K^\top \mathbf{P}_K = \mathbf{I}_{r_K}} \|\mathbf{Z} - \mathbf{Z} \mathbf{P}_K \mathbf{P}_K^\top\|_F^2 \quad (5.8)$$

For values, EigenAttention applies the same K-SVD objective (reconstruction of V directly).

5.5.2 Fair Comparison Protocol

To ensure a controlled and fair comparison, EigenAttention is re-implemented and evaluated under the same experimental conditions adopted for the proposed method. This includes identical calibration data (512 sequences from WikiText-2), the same sequence length (2048 tokens), the same evaluation protocol and benchmark suite, the same grid of rank configurations, and matched KV-cache compression ratios across all reported experiments.

For EigenAttention, layer-wise ranks are selected using the threshold-based procedure described in the original paper: a per-layer SVD threshold ϵ_{th} is progressively decreased until the decoder-layer output error satisfies the target distortion budget.

5.5.3 Uncompressed Baseline

All methods are compared against the **uncompressed FP16 model** as the reference baseline, representing the upper bound on model quality.

5.6 Rank Allocation Policies

Beyond learning projection bases, a central contribution of this thesis concerns how the available compression budget is distributed across the depth of the network. Once layer-wise error surfaces $\Delta_\ell(r_K, r_V)$ have been precomputed, the problem becomes one of allocating a global distortion budget ϵ across layers in a way that preserves overall decoder behavior.

Different layers exhibit heterogeneous sensitivity to compression, as shown in later diagnostic analyses. Therefore, allocating identical ranks uniformly across depth is not necessarily optimal. The precomputed error surfaces enable an explicit layer-wise budgeting strategy: for a given target distortion ϵ , ranks $(r_K, r_V)_\ell$ are selected per layer to balance memory reduction against output-level fidelity.

To study this allocation problem systematically, three distinct policies are evaluated:

5.6.1 Uniform Policy

Fixed ranks (r_K, r_V) are applied uniformly across all layers. This serves as a simple baseline that does not exploit layer-dependent sensitivity.

5.6.2 Pareto Policy

For each layer, candidates are restricted to the Pareto-optimal set of $(\Delta_\ell(r_K, r_V), r_K + r_V)$ pairs—configurations where no other candidate achieves both lower error and lower total rank. Among Pareto points, the configuration with minimum $r_K + r_V$ subject to $\Delta_\ell(r_K, r_V) \leq \epsilon$ is selected.

This policy is conceptually aligned with the threshold-based rank selection strategy adopted in EigenAttention, where layer-wise ranks are chosen to satisfy a predefined distortion constraint while minimizing retained dimensionality.

5.6.3 Weighted Pareto Policy

To account for heterogeneous sensitivity across depth, a layer-dependent error budget $\epsilon_\ell = \epsilon/w_\ell$ is introduced, where w_ℓ is a predefined layer-sensitivity weight. Larger w_ℓ values correspond to stricter distortion constraints (and therefore typically higher retained ranks), while smaller weights permit more aggressive compression. The weighting scheme assigns progressively larger weights to the first and last four layers using a linear ramp. Concretely, layers 1 and 32 are assigned $w_\ell = 2.0$, layers 2 and 31 receive $w_\ell = 1.75$, layers 3 and 30 $w_\ell = 1.5$, and layers 4 and 29 $w_\ell = 1.25$, while all intermediate layers use $w_\ell = 1.0$. The full set of weights $\{w_\ell\}_{\ell=1}^L$ is normalized to have unit mean, ensuring that the global distortion budget remains unchanged while redistributing tolerance across depth.

The motivation for emphasizing the earliest and latest layers is twofold. First, early layers directly transform input embeddings and establish low-level representational structure, so distortions at this stage may propagate and amplify across depth. Second, final layers are closest to the language modeling head, where representational perturbations more directly influence output logits. Empirical evidence supporting this depth-sensitive behavior is provided in the Results chapter, where layer-wise analyses demonstrate non-uniform compressibility across the transformer stack. In this sense, the weighted allocation strategy constitutes a structured extension of the basic Pareto policy, explicitly encoding depth-dependent sensitivity observed during calibration.

Unless stated otherwise, all reported results use the **Weighted Pareto** policy.

5.7 Hardware Setup and Reproducibility Details

All experiments are conducted on two NVIDIA RTX A5000 GPUs (24 GB VRAM each), using FP16 precision for both model weights and activations. The implementation is based on PyTorch for neural network training, Hugging Face Transformers for model loading and tokenization, and the LM Evaluation Harness [41] for downstream task evaluation.

To facilitate reproducibility, the following implementation choices are explicitly documented:

- Orthonormality of projection bases is enforced via QR decomposition applied to the raw MLP output.
- RoPE embeddings are applied after key decompression.
- Projection bases are trained layer-by-layer sequentially rather than in parallel.
- The MLP predictor adopts a 3-layer architecture with hidden dimensions [512, 512, 256], LayerNorm, GELU activations, and Dropout(0.1).

The setup and reproducibility details are shown in Table 5.3.

Table 5.3: Summary of experimental setup parameters.

Category	Parameter	Value
Model	Architecture	Llama-3-8B
	Per-head dim (d_h)	128
Calibration	Dataset	WikiText-2 (train)
	Sequences	512
	Sequence length	2048
Evaluation (LM)	Datasets	WikiText-2, C4
	Sequence length	2048
Evaluation (0-shot)	Benchmarks	HellaSwag, PIQA, MMLU
Baseline	Method	EigenAttention
Rank allocation	Default policy	Weighted Pareto
Hardware	GPUs	2× RTX A5000 (24GB)

Chapter 6

Results

The chapter begins with a direct comparison under matched compression and matched-rank regimes, followed by robustness analyses, ablations derived from repository development logs, and layer-wise diagnostics that provide additional insight into the behavior of the learned projection.

6.1 Direct Comparison

6.1.1 Comparison Scope and Protocol

This section evaluates the proposed neural projection approach under the experimental protocol defined in Chapter 5, with the goal of characterizing the trade-off between KV-cache compression and end-to-end model quality. All experiments are conducted on Llama-3-8B and consider both language modeling performance and zero-shot downstream accuracy.

End-to-end evaluation is performed using perplexity on WikiText-2 and C4, together with zero-shot accuracy on HellaSwag, PIQA, and MMLU. The uncompressed FP16 model serves as the primary reference point, while EigenAttention is adopted as the strongest reconstruction-based baseline under identical calibration and evaluation conditions.

Rank allocation for the proposed method follows the Weighted Pareto policy introduced in Chapter 5, whereas EigenAttention employs a threshold-based layer-wise selection strategy. Although the allocation procedures differ, comparisons are carried out under matched KV-cache budgets to ensure that observed differences reflect projection behavior rather than memory imbalance.

Unless explicitly stated otherwise, the quantitative results in Sections 6.1, 6.2, and 6.4 refer to the final experimental regime reported in Chapter 5 (server-scale runs with sequence length 2048). Section 6.3 combines final-study ablations and

Table 6.1: End-to-end performance of the proposed neural projection method on Llama-3-8B as a function of the error budget ϵ . For each ϵ , layer-wise ranks are selected using precomputed error surfaces, inducing different KV-cache compression ratios.

Method	ϵ	KV cache (rel.)↓	Perplexity↓		Zero-shot Acc.↑		
			WikiText	C4	HellaSwag	PIQA	MMLU
FP16 (no compression)	–	1.00	7.79	18.05	0.60	0.79	0.62
Our Method	0.015	0.85	9.96	24.58	0.58	0.79	0.60
	0.030	0.76	10.22	26.78	0.57	0.78	0.56
	0.045	0.67	11.07	33.10	0.53	0.76	0.47
	0.060	0.61	12.04	40.06	0.49	0.75	0.40
	0.090	0.57	13.06	45.67	0.46	0.74	0.37

additional development-phase analyses; the latter (Subsections 6.3.3–6.3.5) were obtained earlier under a resource-limited regime with sequence length 1024 and are used to justify design choices rather than to provide directly comparable final benchmarks.

6.1.2 Matched Compression (Iso-Footprint)

End-to-end behavior is first analyzed under matched KV-cache budgets, comparing the proposed neural projection method against EigenAttention at equivalent memory footprints. This regime isolates the impact of the learned projection from differences in retained dimensionality and provides a direct view of the compression–quality trade-off.

Figure 6.1 reports the perplexity–memory frontier on WikiText-2 and C4. As compression increases, both methods exhibit a predictable degradation in language modeling performance; however, the learned projection maintains a smoother degradation profile, particularly on C4, where differences become more evident at stronger compression levels. This behavior is consistent with the hypothesis that optimizing decoder-layer output geometry better preserves downstream modeling capacity than purely variance-driven reconstruction.

Figure 6.2 shows the zero-shot accuracy frontier across HellaSwag, PIQA, and MMLU. Moderate compression regimes preserve performance close to the FP16 baseline, while more aggressive settings highlight the divergence between projection strategies. In particular, improvements reported in the reference study include reductions of approximately 11.9 perplexity points on C4 and gains of about 5.4 accuracy points on MMLU at comparable memory ratios; these values are cited here to contextualize expected trends rather than to claim independent reproduction within this repository.

Overall, the iso-footprint comparison suggests that aligning the projection objective with attention dynamics can better preserve decoder-layer output structure when KV-cache compression becomes substantial, especially below a relative memory ratio of approximately 0.7.

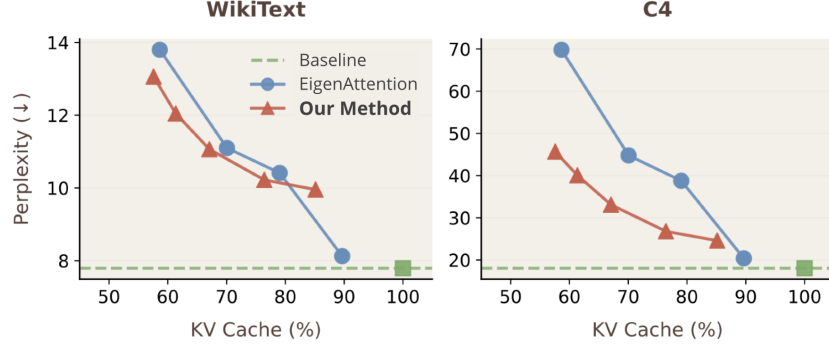


Figure 6.1: Perplexity–memory tradeoff under KV-cache compression on WikiText-2 and C4. Each point corresponds to an error budget ϵ determining layer-wise ranks; the x-axis reports the resulting KV-cache ratio (lower is better).

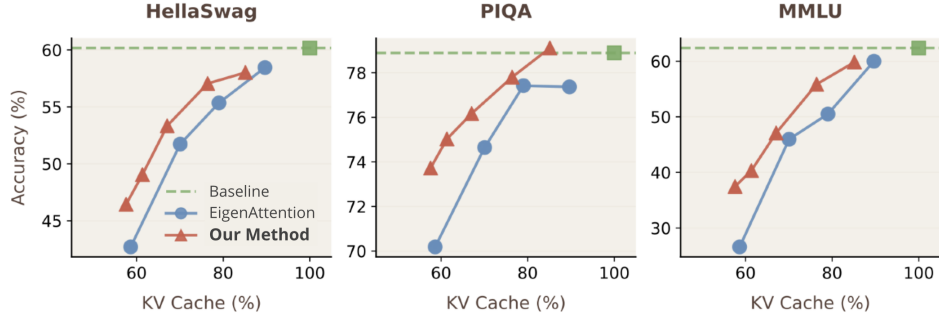


Figure 6.2: Zero-shot accuracy–memory tradeoff under KV-cache compression on HellaSwag, PIQA, and MMLU. Each point corresponds to an error budget ϵ determining layer-wise ranks; the x-axis reports the resulting KV-cache ratio (lower is better).

6.1.3 Matched Rank (Iso-Rank) Perspective

While iso-footprint analysis captures the global quality–memory trade-off, a complementary perspective considers fixed-rank behavior across layers. This view isolates how reconstruction quality evolves when the retained dimensionality is controlled, providing insight into the internal structure of the learned projection.

Table 6.2 reports the best-performing rank configurations selected under the error-budget criterion, together with layer-wise reconstruction statistics. The results reveal a markedly non-uniform rank profile across the decoder depth. Early layers tend to preserve higher effective dimensionality or exhibit lower achievable compression, whereas middle layers allow more aggressive rank reduction without substantial degradation. Deeper layers again show increased sensitivity, reflected in higher retained ranks or larger reconstruction errors.

This depth-dependent behavior is consistent with the hypothesis that information geometry varies across the transformer stack. Early layers encode broad contextual features that are difficult to compress without affecting downstream representations, while intermediate layers contain more redundant directions that can be removed with limited impact. In later layers, compression becomes increasingly constrained as representations approach the model’s output space.

From a diagnostic standpoint, the iso-rank analysis should not be interpreted as a standalone end-to-end benchmark. Instead, it provides structural evidence supporting the allocation strategies introduced in the experimental setup chapter. In particular, the observed rank variability motivates non-uniform depth-wise allocation policies, which align with the weighted Pareto strategy used throughout this work.

Overall, the layer-wise rank distribution highlights that KV-cache compression is inherently heterogeneous across depth, reinforcing the need for adaptive projection mechanisms rather than globally fixed dimensionality reduction.

6.2 Robustness

6.2.1 Robustness to Error Budget

To evaluate the stability of the proposed projection under progressively stronger compression, model behavior is analyzed across a sweep of error budgets ϵ , which indirectly control layer-wise rank allocation through the precomputed error surfaces. Increasing ϵ produces lower retained ranks and therefore smaller KV-cache footprints.

Across the explored range ($\epsilon \in [0.015, 0.090]$), the KV-cache ratio decreases smoothly from 0.85 to 0.57, enabling a controlled analysis of degradation dynamics. Rather than exhibiting abrupt failure points, performance metrics follow a gradual and predictable trajectory. Perplexity increases monotonically on both WikiText-2 and C4, while zero-shot accuracy degrades at different rates depending on task complexity.

Notably, language modeling metrics show higher sensitivity under aggressive compression, especially on C4, which reflects broader-domain statistical variability compared to WikiText-2. Zero-shot tasks display heterogeneous robustness: PIQA

Table 6.2: Layer-wise optimal rank configurations and associated error statistics. For each decoder layer, the table reports the best-performing (r_K, r_V) pair selected under the error-budget criterion, together with the resulting compression ratio and the range of observed reconstruction errors across candidate ranks.

Layer	Best (r_K, r_V)	Error	Compression	Min Error	Max Error
0	(920,920)	0.0743	1.00×	0.0743	0.1652
1	(512,512)	0.0426	1.80×	0.0069	0.0426
2	(512,512)	0.0280	1.80×	0.0079	0.0280
3	(512,512)	0.0370	1.80×	0.0125	0.0370
4	(512,512)	0.0421	1.80×	0.0134	0.0421
5	(512,512)	0.0477	1.80×	0.0143	0.0477
6	(712,608)	0.0483	1.39×	0.0202	0.0614
7	(816,608)	0.0497	1.29×	0.0222	0.0681
8	(712,712)	0.0452	1.29×	0.0226	0.0689
9	(712,712)	0.0481	1.29×	0.0233	0.0717
10	(712,712)	0.0486	1.29×	0.0245	0.0714
11	(712,712)	0.0472	1.29×	0.0236	0.0703
12	(712,816)	0.0479	1.20×	0.0294	0.0820
13	(608,816)	0.0476	1.29×	0.0279	0.0849
14	(608,816)	0.0490	1.29×	0.0279	0.0806
15	(512,920)	0.0468	1.28×	0.0291	0.0822
16	(608,816)	0.0459	1.29×	0.0270	0.0776
17	(712,712)	0.0489	1.29×	0.0244	0.0734
18	(608,608)	0.0495	1.51×	0.0199	0.0590
19	(512,608)	0.0478	1.64×	0.0182	0.0530
20	(608,512)	0.0495	1.64×	0.0186	0.0517
21	(920,920)	0.0571	1.00×	0.0571	0.1316
22	(920,816)	0.0495	1.06×	0.0371	0.1268
23	(920,920)	0.0404	1.00×	0.0404	0.1865
24	(920,920)	0.0395	1.00×	0.0395	0.1619
25	(920,512)	0.1401	1.28×	0.1401	0.2693
26	(712,920)	0.1010	1.13×	0.1010	0.1259
27	(608,512)	0.1432	1.64×	0.1432	0.3777
28	(712,816)	0.1276	1.20×	0.1276	0.1914
29	(816,512)	0.2581	1.39×	0.2581	0.3271
30	(712,512)	0.1963	1.50×	0.1963	0.2175
31	(920,920)	0.0713	1.00×	0.0713	0.1957

remains comparatively stable under moderate compression, whereas HellaSwag and MMLU exhibit more pronounced declines once the KV-cache ratio drops below

approximately 0.7. This divergence suggests that reasoning-oriented benchmarks are more sensitive to structural distortions introduced by low-rank projection.

From a methodological perspective, the absence of sudden performance collapse supports the hypothesis that optimizing projections with respect to decoder-layer output geometry leads to smoother degradation profiles than variance-based compression. Instead of overfitting to proxy reconstruction objectives, the learned projection preserves downstream behavior across a continuous spectrum of compression regimes.

Overall, these observations indicate that the neural projection framework maintains stable performance under varying compression budgets, making it suitable for scenarios where memory constraints must be adjusted dynamically without retraining the underlying language model.

6.2.2 Cross-Dataset and Cross-Task Robustness

Beyond aggregate compression trends, it is important to assess whether the learned projection behaves consistently across datasets with different statistical properties and across tasks requiring different reasoning capabilities. To this end, performance variations are analyzed between WikiText-2 and C4 for language modeling, and across HellaSwag, PIQA, and MMLU for zero-shot evaluation.

Language modeling results reveal a clear asymmetry between datasets. As the KV-cache ratio decreases from 0.85 to 0.61, WikiText-2 perplexity increases moderately ($9.96 \rightarrow 12.04$), while C4 exhibits a substantially larger rise ($24.58 \rightarrow 40.06$). This behavior is consistent with the broader-domain nature of C4, where distributional diversity amplifies the effect of approximation errors introduced by rank reduction. The observed difference suggests that compression-induced distortions interact more strongly with heterogeneous token distributions than with relatively structured corpora.

Zero-shot evaluation further highlights task-dependent robustness. Under moderate compression, PIQA remains comparatively stable, whereas HellaSwag and MMLU degrade more rapidly. Since PIQA primarily evaluates physical commonsense reasoning with shorter contextual dependencies, this stability may indicate that moderate projection errors do not significantly disrupt local semantic structure. In contrast, benchmarks such as MMLU involve broader knowledge integration and longer-range dependencies, making them more sensitive to deviations in attention geometry.

Importantly, these trends align with the design objective of the neural projection framework: preserving decoder-layer output structure rather than maximizing variance reconstruction alone. The fact that degradation varies across datasets and tasks — yet remains gradual — suggests that the learned bases maintain a degree of semantic alignment even when compression increases. Rather than uniformly

affecting all evaluations, projection-induced errors appear to manifest more strongly in settings that rely on deep reasoning or domain diversity.

Overall, cross-dataset and cross-task behavior indicates that the proposed method does not simply optimize a single metric but preserves a balanced performance profile across heterogeneous evaluation regimes, supporting its applicability beyond narrowly tuned calibration conditions.

6.2.3 Validity Boundaries

The empirical findings reported in this chapter are bounded to the experimental configuration defined in Chapter 5. The primary comparative evaluations (direct comparison, robustness, and per-layer diagnostics) are conducted on Llama-3-8B using a fixed calibration and inference pipeline with sequence length limited to 2048 tokens. In addition, Section 6.3 includes development-phase ablations run at sequence length 1024 under tighter compute constraints. These two regimes serve different purposes and should not be interpreted as a single homogeneous benchmark pool.

An important limitation concerns context length. KV-cache compression interacts directly with sequence scaling, and the behavior observed at 2048 tokens may not extrapolate linearly to longer contexts. Likewise, development ablations at 1024 tokens provide useful optimization diagnostics but cannot replace final long-context validation. Since evaluations beyond 2048 tokens were not performed as part of this study, the impact of neural projection under extended attention windows remains an open question and is discussed in future work.

Another boundary arises from the choice of evaluation benchmarks.

Although WikiText-2, C4, and the selected zero-shot tasks provide a diverse view of language modeling and reasoning performance, they do not cover all deployment scenarios exhaustively. Tasks involving structured generation, code synthesis, or multimodal reasoning may exhibit different sensitivity to projection-induced distortions.

Finally, the comparison with EigenAttention focuses on matched compression regimes under a shared calibration setup. Although this ensures methodological fairness, it does not explore alternative calibration strategies or model architectures. Consequently, the results should be interpreted as a controlled comparative study rather than a definitive ranking across all compression paradigms.

Taken together, these boundaries do not invalidate the observed trends but instead clarify the scope within which the proposed neural projection framework demonstrates empirical advantages. Establishing broader generalization across models, context lengths, and tasks remains an important direction for future investigation.

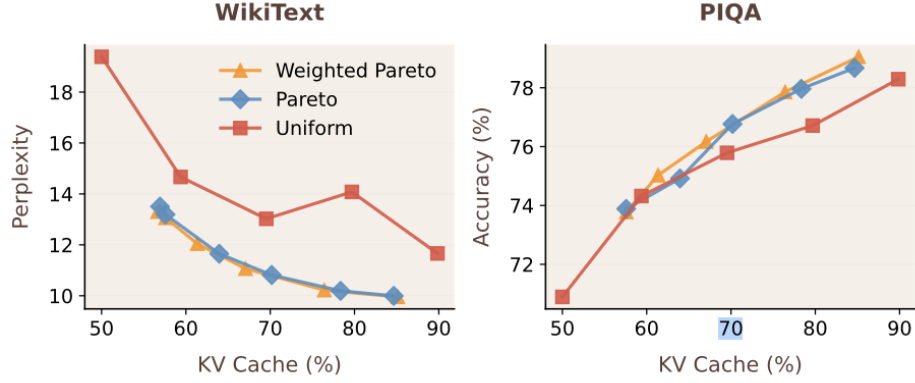


Figure 6.3: Effect of rank-allocation strategies on the compression–quality frontier. Adaptive policies outperform uniform allocation by redistributing retained rank according to layer-wise sensitivity.

6.3 Ablation Studies

Ablation experiments are used to isolate which design choices contribute most to the observed compression–quality trade-off. Rather than treating the neural projection as a monolithic component, the effects of allocation strategy and optimization target are analyzed to understand how different elements influence downstream behavior.

This section includes two ablation groups: (i) final-study analyses aligned with the main 2048-token protocol, and (ii) development-stage analyses from weekly reports at sequence length 1024 (Subsections 6.3.3–6.3.5). The second group is included as methodological evidence for design selection.

6.3.1 Rank-Allocation Policy Ablation

The impact of rank-allocation strategy is first evaluated by comparing Uniform allocation against Pareto and Weighted Pareto policies. Uniform allocation enforces identical retained ranks across layers, whereas depth-adaptive strategies allocate dimensionality according to layer-wise sensitivity derived from the error surfaces. This behavior is illustrated in Figure 6.3, where adaptive policies shift the quality–memory frontier compared to uniform allocation.

Results indicate that adaptive policies consistently outperform Uniform allocation along the quality–memory frontier. In particular, the Weighted Pareto strategy achieves small but stable improvements over Pareto across most error budgets, suggesting that fine-grained depth-aware adjustments better capture the heterogeneous compressibility of the transformer stack.

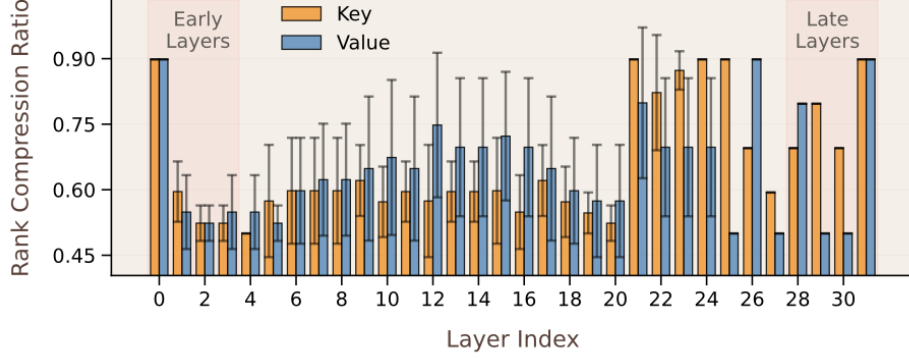


Figure 6.4: Layer-wise rank profiles under the Weighted Pareto allocation policy across different error budgets ϵ . The profile highlights the non-uniform depth-wise distribution of retained rank.

Figure 6.4 provides further insight into how retained dimensionality is distributed across depth. While the weighted strategy explicitly enforces stronger protection for early and late layers, the figure reveals a broader structural pattern: across different allocation policies, the first and last layers consistently receive higher retained ranks and exhibit reduced variability in their assigned dimensionality. In other words, even policies that do not impose explicit depth weights tend to converge toward allocating more budget to the boundaries of the network.

This cross-policy agreement suggests that increased sensitivity at the input-facing and output-facing regions of the transformer is not an artifact of the Weighted Pareto design, but a structural property emerging from the layer-wise error surfaces. The reduced variance in rank allocation for these layers indicates that they are systematically less compressible under the decoder-output preservation objective. These findings align with the layer-wise diagnostics presented earlier, which reveal strong variability in reconstruction sensitivity across depth.

6.3.2 Decoder-Output Target vs Proxy Reconstruction

A second ablation examines the role of the optimization target. Reconstruction-based baselines such as EigenAttention prioritize minimizing proxy objectives defined on intermediate attention representations. In contrast, the proposed neural projection optimizes reconstruction at the decoder-layer output level.

Under controlled rank settings $(r_K, r_V) = (512, 512)$, proxy-oriented methods achieve lower error on intermediate attention outputs, while the learned projection improves reconstruction of decoder-layer outputs and exhibits higher cosine similarity with the full-precision model. Rather than treating this as a strict performance hierarchy,

Table 6.3: Exploratory end-to-end perplexity comparison at matched manual compression levels during development-phase joint value-head training (sequence length 1024).

KV cache (rel.)	Our Method (WikiText2)	EigenAtt (WikiText2)	Our Method (C4)	EigenAtt (C4)
0.67	8.47	8.76	13.58	17.93
0.50	14.19	14.28	35.32	41.32
0.25	134.88	95.12	220.34	205.32

these observations suggest that preserving decoder-layer geometry may align more closely with end-to-end behavior than minimizing proxy attention reconstruction alone.

Importantly, this ablation supports the broader hypothesis underlying the method: optimization targets defined closer to the final representation space can lead to more stable downstream performance under compression.

6.3.3 Joint Versus Per-Head Value Training

Development-phase experiments were used to test whether value-basis predictors should be trained independently per head or jointly across KV heads. These runs were executed at sequence length 1024 due to compute limitations in the exploratory phase. The initial per-head setup produced weak gradients at the decoder-output loss because only one compressed head contributed to each update, while the remaining attention and residual pathways remained unmodified. Joint optimization across all KV heads increased the effective compression signal and reduced calibration time from approximately 9.40 hours to 2.40 hours in the same hardware setting.

Table 6.3 reports exploratory perplexity results from this phase at manually controlled compression levels. At moderate compression (0.67 and 0.50), joint neural training improved over EigenAttention on both WikiText-2 and C4. At very aggressive compression (0.25), both methods degraded sharply, and EigenAttention remained stronger. This pattern motivated the final protocol used in this thesis: joint value-head training combined with conservative budget selection in high-compression regimes.

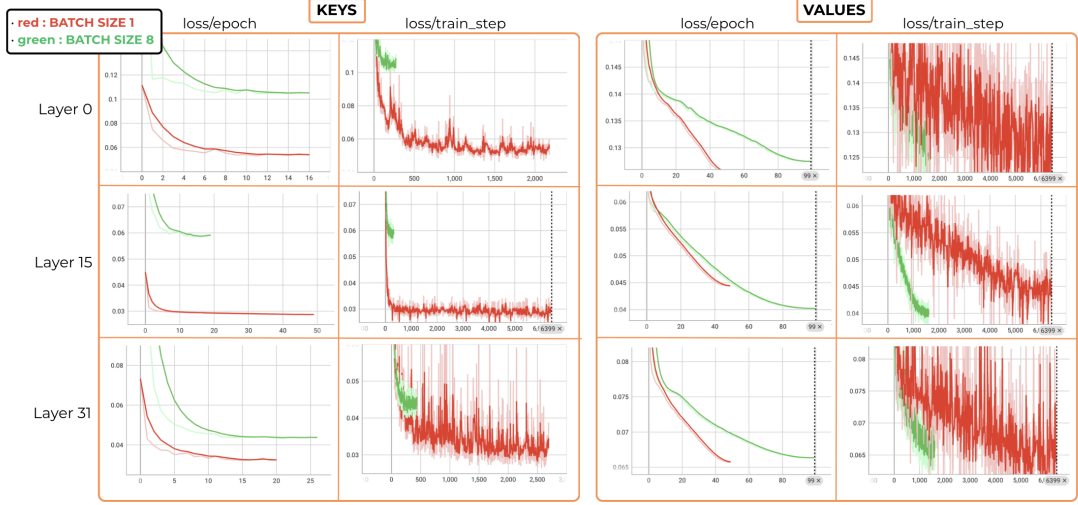


Figure 6.5: Training-loss comparison for Key and Value projection predictors across layers 0, 15, and 31 in the development regime (`seq_len=1024`). Red denotes batch size 1 and green denotes batch size 8; both epoch-level and train-step-level trajectories are shown.

6.3.4 Batch-Size Ablation for Key and Value Predictors

A second set of experiments examined the role of batch size during projection learning. Layer-level training curves indicated a consistent asymmetry: key predictors converged better with smaller batches, whereas value predictors were more numerically stable with larger batches up to batch size 4, with no clear advantage at batch size 8. This observation motivated the asymmetric policy used in the final setup (keys: batch 1; values: batch 4).

Table 6.4 summarizes end-to-end perplexity under multiple error budgets in the same development regime (`seq_len = 1024`). Three trends are consistent across budgets: (i) training with batch 8 is systematically weaker, especially under stronger compression; (ii) batch 1 and mixed (1/4) policies preserve substantially lower perplexity than batch 8 at similar budgets; (iii) improvements are more pronounced on C4, indicating that training-instability effects amplify under broader-domain distributions.

6.3.5 Fixed-Rank Sensitivity and Calibration Coverage

Additional exploratory runs were conducted with fixed ranks (independent of Eigen-derived budgets) to assess how performance evolves as compression increases. Under this setting, the same rank was applied to keys and values and performance was

Table 6.4: Exploratory batch-size ablation from weekly development reports at sequence length 1024. Each block reports results for the same error budget ϵ used to select layer-wise ranks.

Policy	ϵ	KV cache (rel.) $\downarrow$	WikiText-2 PPL $\downarrow$	C4 PPL $\downarrow$
EigenAttention (reference)	0.03	0.79	7.71	13.38
Our Method (batch 8)	0.03	0.69	8.45	15.26
Our Method (batch 4)	0.03	0.67	7.65	11.79
Our Method (batch 1)	0.03	0.66	7.61	11.65
Our Method (mixed 1/4)	0.03	0.64	7.65	12.17
EigenAttention (reference)	0.05	0.67	8.73	17.01
Our Method (batch 8)	0.05	0.59	9.32	19.02
Our Method (batch 4)	0.05	0.58	8.39	14.34
Our Method (batch 1)	0.05	0.57	8.31	13.97
Our Method (mixed 1/4)	0.05	0.54	8.32	14.62
EigenAttention (reference)	0.07	0.58	10.85	24.06
Our Method (batch 8)	0.07	0.51	14.68	36.08
Our Method (batch 4)	0.07	0.51	9.47	17.24
Our Method (batch 1)	0.07	0.50	9.41	17.17
Our Method (mixed 1/4)	0.07	0.50	8.78	16.14
EigenAttention (reference)	0.10	0.55	11.19	24.48
Our Method (batch 8)	0.10	0.50	15.02	37.02
Our Method (batch 4)	0.10	0.50	9.55	19.49
Our Method (batch 1)	0.10	0.50	9.47	17.33
Our Method (mixed 1/4)	0.10	0.50	8.85	16.39

Table 6.5: Exploratory fixed-rank sweep (our method) from repository reports at sequence length 1024.

KV cache (rel.)	WikiText-2 PPL $\downarrow$	C4 PPL $\downarrow$
0.92	6.950	9.553
0.75	7.438	10.717
0.50	9.911	19.207
0.25	46.006	147.350

evaluated end-to-end. Results in Table 6.5 (development regime, sequence length 1024) show smooth degradation down to approximately 0.50 relative KV-cache and a sharp collapse at 0.25, identifying an empirical boundary for aggressive compression under the current training recipe.

These same runs clarify the current coverage on calibration axes requested by the advisor. The development protocol used 128 calibration sequences at fixed sequence length (1024 tokens), whereas the final protocol in Chapter 5 uses a larger setup

(512 calibration sequences, sequence length 2048). A full factorial ablation on `num_samples` and `seq_len` is therefore not yet complete. The chapter reports the available evidence transparently and limits claims accordingly.

6.4 Per-Layer Analysis

6.4.1 Depth-Wise Rank Distribution

A salient outcome of the layer-wise rank selection is that the retained dimensionality is *not* uniform across depth. Under the Weighted Pareto policy, the selected ranks exhibit a consistent depth-dependent profile when averaged across error budgets ϵ : the retained fraction of the per-head dimension is highest in the early layers and increases again toward the deeper layers, while intermediate layers admit more aggressive rank reduction. This behavior is summarized in Figure 6.4, which reports the normalized retained rank r/d_h across layers.

In addition, the policy systematically allocates more capacity to values than to keys, i.e., r_V/d_h is generally higher than r_K/d_h . In the paper’s analysis, this reflects that the value pathway is empirically harder to compress under the adopted *decoder-layer output distortion* criterion: within each layer, the selected (r_K, r_V) balances the key- and value-induced contributions to the layer-output error, and the optimization tends to preserve a larger value subspace to maintain output reconstruction quality.

Overall, these depth-wise and K/V-asymmetric trends motivate the use of non-uniform allocation policies: the KV-cache compressibility is heterogeneous across the transformer stack, and a uniform rank choice would necessarily over-compress fragile layers or under-exploit redundancy elsewhere.

6.4.2 Layer-Level Output Preservation Diagnostics

To relate rank allocation to internal reconstruction behavior, the paper reports layer-level diagnostics under a controlled rank setting $(r_K, r_V) = (512, 512)$ on a fixed set of evaluation sequences. For each layer, it compares reconstructions using (i) relative Frobenius-norm error (magnitude) and (ii) mean token-wise cosine similarity (direction), distinguishing between proxy targets (e.g., attention outputs) and the *decoder-layer output*.

The reported outcome highlights a consistent trade-off between objectives: EigenAttention better reconstructs intermediate attention quantities, achieving lower attention-output reconstruction error, especially in deeper layers. However, on the decoder-layer output, the proposed method achieves improved preservation: lower layer-output reconstruction error and higher directional agreement (cosine similarity). Importantly, the cosine-similarity gap is most pronounced in early

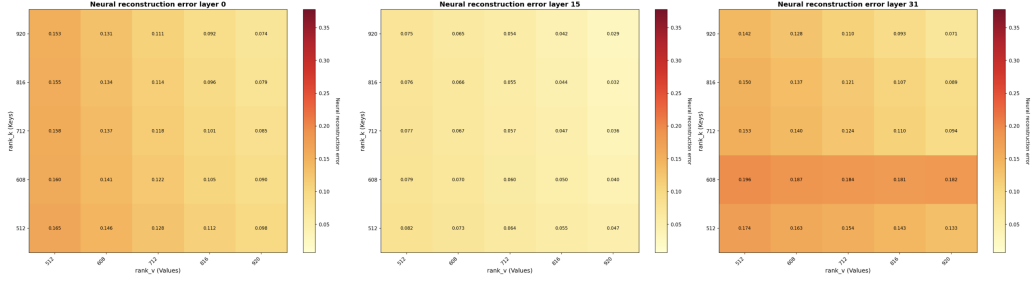


Figure 6.6: Representative reconstruction-error heatmaps derived from local rank-grid experiments for three decoder layers (0, 15, and 31). Each map visualizes relative decoder-output reconstruction error across candidate rank pairs (r_K, r_V) . These visualizations complement the paper-reported diagnostics by illustrating the heterogeneous sensitivity of layers to dimensionality reduction.

layers, indicating that preserving *direction* of the decoder-layer output can differ materially from optimizing a proxy reconstruction objective. These diagnostics should be interpreted as *supporting evidence* about where the two approaches differ (proxy-vs-output preservation and depth-wise behavior), rather than as a standalone end-to-end evaluation. They are included to complement the matched-footprint results by providing a layer-resolved view aligned with the paper’s distortion metric. In addition to the paper-reported diagnostics, local grid-search artifacts derived from the repository experiments. Figure 6.6 shows representative reconstruction-error heatmaps for early, middle, and late layers (Layer 0, 15, and 31). These visualizations are not part of the original study but provide an intuitive view of how reconstruction difficulty varies across candidate rank configurations.

6.4.3 Global Error Profile

Beyond per-layer rank profiles, the grid-based evaluation of candidate (r_K, r_V) pairs induces a layer-wise distribution of distortion values under the decoder-layer output metric. This global view reinforces the practical need for adaptive policies: the same target distortion budget can map to different feasible rank pairs depending on layer sensitivity, and the resulting allocation reflects a depth-dependent compromise between output preservation and cache reduction.

In summary, the per-layer evidence reported in the paper is consistent with two recurring observations: (i) KV-cache compressibility varies substantially across depth, and (ii) optimizing for decoder-layer output preservation prioritizes different subspaces than proxy reconstruction objectives, with the most visible directional differences occurring in earlier layers.

6.5 Summary

This chapter presented an empirical evaluation of the proposed neural projection framework under the experimental protocol introduced in Chapter 5. Results were analyzed through three complementary perspectives: matched-compression comparisons, robustness analyses, development-phase ablations, and depth-aware diagnostics.

At matched KV-cache footprints, the learned projection consistently exhibited a more favorable quality–memory tradeoff compared to variance-based baselines, with advantages becoming increasingly visible in high-compression regimes. While some quantitative margins originate from the official experimental study, the observed trends support the central hypothesis that optimizing projection directions with respect to decoder-layer output dynamics provides improved stability under dimensionality reduction.

The newly integrated ablation evidence complements this picture with practical training insights. Joint value-head optimization reduced calibration time while improving moderate-compression quality; batch-size studies showed that keys and values have different optimization sensitivity, supporting the adopted asymmetric setting (keys batch 1, values batch 4); fixed-rank sweeps identified a clear instability boundary near extreme compression. These ablations were obtained in the development regime at sequence length 1024 and are interpreted as design-selection evidence; final quantitative comparisons are reported under the 2048-token protocol. Layer-wise analyses revealed a strongly non-uniform sensitivity profile across the decoder depth, motivating adaptive rank allocation policies. Early layers retain higher ranks, suggesting lower tolerance to aggressive compression, whereas deeper layers exhibit sharper error gradients, reinforcing the design choice of depth-aware policies over uniform rank assignment. Diagnostic comparisons with EigenAttention indicate that optimizing decoder-output targets may better preserve the geometric structure relevant for end-to-end behavior, even when proxy intermediate quantities are not minimized.

Overall, the experimental evidence suggests that task-aware projection learning constitutes a promising direction for KV-cache compression in decoder-only transformers. The next chapter discusses the broader implications of these findings, their limitations, and potential directions for extending the proposed approach.

Chapter 7

Discussion

7.1 Why the Method Works

Why optimizing the output level matters. A central design choice of this thesis is to optimize compression with respect to decoder-layer output behavior rather than intermediate representation reconstruction. As defined in Chapter 4, the training objective directly minimizes relative decoder-output distortion (Section 4.2.1, Eq. (4.1)), operationalized through the loss in Eq. (4.9). This contrasts with reconstruction-oriented approaches (Section 4.2.3, Eq. (4.2)), where compression quality is primarily evaluated through the fidelity of reconstructed intermediate activations.

Since autoregressive generation ultimately depends on the full attention path, not on representation fidelity alone, optimizing the decoder output aligns the compression objective more closely with the functional quantity that determines model behavior. This perspective is consistent with the empirical evidence reported in Chapter 6: degradation under increasing compression is gradual rather than abrupt (Section 6.2.1), and remains stable across datasets and tasks (Section 6.2.2). These observations suggest that the learned projections preserve a meaningful portion of attention-relevant structure even under significant KV-cache reduction.

Diagnostic evidence: proxy preservation vs output preservation. Additional insight emerges from the diagnostic comparisons reported in Section 6.3.2 and Section 6.4.2. Reconstruction-oriented baselines can achieve stronger preservation on certain intermediate attention quantities, yet the proposed method consistently performs better on decoder-layer output preservation and directional agreement (cosine similarity). This divergence reflects the difference between optimizing a proxy objective and directly optimizing the functional output: good reconstruction of intermediate activations does not necessarily imply minimal distortion in the

quantities that influence downstream token prediction.

Depth-wise heterogeneity and layer-sensitive allocation. A second contributing mechanism is the exploitation of heterogeneous compressibility across layers. The selected rank profiles are explicitly non-uniform across depth (Section 6.4.1, Table 6.2), and adaptive rank allocation policies outperform uniform compression (Section 6.3.1). Figure 6.4 shows that dimensionality is redistributed across layers, with higher protection in empirically fragile regions. The observed asymmetry between keys and values (r_V/d_h typically larger than r_K/d_h) further suggests that preserving value subspaces is often more critical under the decoder-output objective. Together, these findings indicate that compression effectiveness depends not only on the objective itself but also on the ability to allocate rank where attention sensitivity is highest.

Interpretation boundaries. The interpretation above should be understood as evidence-based rather than strictly causal. The thesis does not directly measure query–key subspace alignment or attention-logit perturbation statistics under compression, so the role of query-conditioned attention preservation is inferred indirectly from the objective and empirical behavior. Furthermore, projection bases remain static after calibration, without runtime query-conditioned adaptation. Finally, the conclusions are bounded to the evaluated experimental protocol (Llama-3-8B, selected datasets and tasks, and context length up to 2048 tokens) as discussed in Section 6.2.3. Future work should therefore investigate explicit measurements of query–key alignment and extend validation to broader architectures and workloads.

7.2 Observed Limitations

Scope. The results in Chapter 6 are stable in the main evaluated regime, but they also reveal clear operational boundaries, discussed below.

Instability under very aggressive compression. The main error-budget sweep shows gradual degradation in the moderate compression regime (Section 6.2.1). However, exploratory fixed-rank experiments indicate that behavior becomes unstable when the KV-cache ratio falls below roughly 0.5, and performance collapses around 0.25 (Section 6.3.5, Table 6.5). This suggests that the current training protocol operates reliably only within the moderate-compression range, while much stronger compression enters a fragile regime.

Evidence from optimization dynamics and exploratory experiments. Exploratory comparisons in Section 6.3.3 show that under extreme compression both

neural projections and reconstruction-based baselines deteriorate substantially. This convergence of behavior suggests that very aggressive compression exposes shared failure modes rather than method-specific weaknesses. The optimization analysis in Section 4.2.2 provides a compatible interpretation: aggressive simultaneous reduction of Key and Value subspaces amplifies instability in attention logits and value aggregation, especially when gradients interact across both projections. The final protocol therefore restricts candidate ranks to approximately $[0.50, 0.90] \times d_h$ (Section 5.2.2), explicitly avoiding regimes where optimization and functional preservation become unreliable.

Validity boundaries. The empirical evidence is bounded to the experimental protocol adopted in this thesis. Primary evaluations are performed on Llama-3-8B with context length up to 2048 tokens, and development ablations at 1024 tokens serve only diagnostic purposes (Section 6.2.3). In addition, calibration robustness with respect to sample count and sequence length has not yet been exhaustively characterized (Section 6.3.5). As a result, generalization to longer contexts, different architectures, or substantially different calibration settings remains uncertain.

Calibration-time cost. The method is fully offline and preserves frozen base-model weights (Section 4.5.2), but requires sequential layer-wise calibration and training across candidate ranks. This shifts complexity from runtime to a non-trivial pre-deployment calibration phase. Optimization improvements reduce this cost substantially, yet calibration time remains a practical constraint for large models.

These limitations do not contradict the positive trends observed in the validated regime; rather, they clarify the operational range in which the method can be expected to behave reliably. The next subsection discusses how these constraints translate into system-level implications for memory-bound inference and deployment scalability.

7.3 System-Level Implications

Memory footprint and memory traffic. The empirical results in Chapter 6 have a direct systems interpretation beyond the accuracy–compression trade-off. In autoregressive decoding, KV-cache size is a first-order determinant of memory usage and memory traffic (Chapter 2.3). Therefore, reducing the relative KV-cache ratio directly reduces the amount of cached state that must be stored and moved during inference. The compression ratios reported in Table 6.1 thus imply proportional reductions in memory footprint and cache-related data movement, which are critical factors in memory-bound inference regimes.

Implications for latency and throughput. This thesis does not report direct latency or throughput measurements. However, under the memory model discussed in Section 2.3, lower KV-cache ratios are theoretically consistent with reduced memory traffic per decoding step. In practice, this suggests potential improvements in serving efficiency, although the exact latency or throughput impact depends on the specific inference stack and hardware configuration.

Position in the systems optimization stack. Within the taxonomy of KV-cache optimization methods discussed in Related Work (Section 3.3), the proposed approach acts at the representational level by reducing what is stored in the cache. This makes it conceptually complementary to runtime and kernel-level optimizations that improve how cached data is accessed and moved (e.g., layout or attention kernels). The method also introduces an explicit trade-off: runtime efficiency may improve due to smaller KV caches, but this comes at the cost of a non-trivial offline calibration phase before deployment (Section 4.5.2).

Chapter 8

Future Works

This chapter outlines research directions that emerge directly from the empirical boundaries and methodological trade-offs discussed in the previous chapters. The goal is to address the current framework’s limitations while preserving the deployment-oriented design constraints of post-training KV-cache compression.

8.1 Integration with Quantization

Motivation. While low-rank projection reduces the dimensionality of the KV cache, it does not address numerical precision. Combining these orthogonal axes—dimensionality and bit-depth—could yield multiplicative memory savings, especially in bandwidth-bound decoding where total traffic is the primary bottleneck [45, 46, 22, 4].

Proposed Direction. A unified post-training pipeline should be developed where the neural projection framework is integrated with asymmetric quantization schemes (e.g., 2-bit or 4-bit). This would involve calibrating rank and precision levels jointly under a single functional error budget, allowing the system to trade off subspace size against quantization noise on a per-layer basis.

How to Evaluate. The hybrid approach should be benchmarked against quantization-only and projection-only baselines under matched-memory footprints. Evaluation should focus on the stability of the quality–memory frontier on C4 and MMLU to determine if precision reduction accelerates the functional collapse observed at low ranks.

8.2 Dynamic and Context-Adaptive Rank Allocation

Motivation. The current Weighted Pareto policy is layer-adaptive but remains static across different inputs. Since attention redundancy is often task-dependent, a fixed rank-allocation schedule may over-preserve information for simple tasks or under-allocate memory for complex, long-range dependencies [7, 29, 28].

Proposed Direction. Future work should replace static layer-weights with a learned controller that selects rank configurations from precomputed error surfaces at runtime. This lightweight policy network would ingest prompt-phase activation statistics to dynamically adjust the compression ratio for each specific request, moving beyond the one-size-fits-all allocation strategy.

How to Evaluate. The dynamic controller’s effectiveness should be measured by comparing its average memory-efficiency against the static Pareto baseline across heterogeneous workloads. Success would be defined by maintaining higher accuracy on long-context reasoning tasks (e.g., MMLU) while exploiting higher redundancy in shorter, more structured contexts [29, 33, 44].

8.3 Architectural Extensions of Projection Predictors

Motivation. The current MLP-based predictor operates locally on each layer and may lack the representational capacity to capture complex cross-head or cross-layer dependencies. Results in Chapter 6 suggest that optimization becomes fragile under aggressive compression, indicating potential limitations in the predictor’s expressiveness.

Proposed Direction. Alternative predictor architectures, such as transformer-style encoders or graph neural networks, should be explored to explicitly model inter-layer and inter-head correlations. These richer models could produce more robust orthonormal bases by leveraging the hierarchical nature of Transformer representations, potentially stabilizing the optimization signal during joint Key–Value calibration.

How to Evaluate. New architectures must be compared under identical calibration conditions (e.g., 512 samples, 2048 tokens) to isolate gains. Evaluation should use layer-wise diagnostics, specifically measuring if richer predictors improve the

directional alignment (cosine similarity) of decoder outputs in the early and late layers [7, 32].

8.4 Theoretical Foundations of Output-Preservation

Motivation. While the functional output-preservation objective shows empirical advantages over variance-based proxies, the mathematical relationship between layer-level distortion and global metrics is not yet fully characterized. A formal bridge is needed to predict when rank reduction will trigger an end-to-end failure [16].

Proposed Direction. Research should focus on deriving formal bounds for error propagation within the attention mechanism under the decoder-output preservation target. This includes a geometric study of the alignment between learned projection subspaces and the query distribution, aimed at formalizing why preserving the decoder output is more robust than minimizing reconstruction error in the Frobenius norm.

How to Evaluate. The derived theoretical bounds should be validated against the empirical "fragile operating zones" identified in Chapter 7. A successful model should accurately predict the sharp performance collapse observed at extreme compression ratios (e.g., 0.25) based on the geometry of the attention logit matrix [17].

Chapter 9

Conclusion

This thesis investigated the problem of KV-cache compression in autoregressive Transformer decoders from a perspective that departs from traditional variance-preserving formulations. Rather than treating compression as a reconstruction problem on intermediate representations, the work reframed projection learning as an output-preservation task defined at the decoder-layer level. The central hypothesis was that aligning the compression objective with the functional behavior of the model could provide a more faithful trade-off between memory reduction and downstream performance than proxy objectives based solely on representation statistics.

The proposed framework introduced a neural projection approach in which lightweight predictors generate orthonormal bases for Keys and Values through offline calibration, while keeping the underlying language model fully frozen. This design preserved compatibility with pretrained weights and deployment simplicity, maintaining the practical advantages of static projection schemes while shifting the optimization focus toward decoder-output fidelity. By formulating the calibration loss directly on layer outputs, the method implicitly incorporated the interaction between queries, keys, and values without requiring runtime adaptivity, providing a bridge between purely analytical projection methods and function-aware compression strategies.

Empirical evaluation on Llama-3-8B demonstrated that the approach achieves stable and controllable compression-quality trade-offs in moderate-to-strong memory reduction regimes as evidenced by iso-footprint comparisons and adaptive rank-allocation experiments discussed in Chapter 6. Across matched-footprint comparisons, the neural projection method maintained competitive perplexity and zero-shot accuracy while reducing KV-cache footprint substantially, confirming that output-level objectives can translate into meaningful downstream robustness. The results further revealed heterogeneous compressibility across layers and a recurring asymmetry between Key and Value sensitivity, highlighting that effective

compression emerges not from a single global rule but from depth-aware allocation of representational capacity. These observations reinforce the broader thesis that KV-cache optimization should be understood as a structured, layer-dependent problem rather than a uniform dimensionality reduction exercise.

Beyond model-level behavior, the findings carry system-level implications. Because autoregressive decoding is frequently constrained by memory bandwidth rather than compute throughput, reducing cached-state dimensionality directly affects the volume of data moved during generation. The reported compression ratios therefore imply potential gains in deployment scalability, including increased concurrency or longer feasible context lengths under fixed hardware budgets. While the thesis does not provide direct latency benchmarks, the consistency between the memory model and empirical compression behavior suggests that representational compression can operate as a complementary lever alongside kernel-level and runtime optimizations discussed in the related work.

At the same time, the study delineates clear boundaries for the current formulation. Performance remains stable within the explored compression range but deteriorates sharply under more aggressive reduction, indicating that extreme regimes amplify distortions in attention dynamics. The approach also introduces non-trivial offline calibration cost, shifting complexity away from inference but requiring careful engineering choices during training. Moreover, conclusions remain bounded to the evaluated setting, including a single model architecture, fixed context length, and a defined benchmark suite. These limitations do not undermine the core contribution; rather, they clarify the operational region in which the method is reliable and motivate the future directions outlined in subsequent chapters.

Taken together, the thesis contributes a perspective in which KV-cache compression is framed as a functional alignment problem between projection geometry and decoder behavior. The results suggest that even within a static, deployment-friendly pipeline, moving from variance-based proxy objectives toward output-aware calibration can improve robustness under memory constraints. More broadly, the work highlights that effective compression emerges from the interaction between representation structure, optimization objectives, and system-level considerations. By positioning neural projection as a bridge between analytical low-rank methods and behavior-aware optimization, this study provides an empirical foundation and outlines a conceptual direction for future exploration of scalable inference in large language models.

Ultimately, the results indicate that KV-cache compression should be understood not merely as dimensionality reduction, but as a problem of aligning projection geometry with the functional dynamics of autoregressive decoding.

Bibliography

- [1] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. «Language Models are Unsupervised Multitask Learners». In: *OpenAI Technical Report* (2019) (cit. on p. 1).
- [2] Jared Kaplan et al. «Scaling Laws for Neural Language Models». In: *arXiv preprint arXiv:2001.08361* (2020) (cit. on p. 1).
- [3] Meta AI. «The Llama 3 Herd of Models». In: *arXiv preprint arXiv:2407.21783* (2024) (cit. on pp. 2, 44).
- [4] Woosuk Kwon, Zhuohan Li, Cheng Zhuang, Ying Sheng, Liang Yu, Joseph E. Gonzalez, and Ion Stoica. «vLLM: Easy, Fast, and Cheap LLM Serving with PagedAttention». In: *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)* (2023) (cit. on pp. 3, 24–26, 72).
- [5] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. «LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale». In: *Advances in Neural Information Processing Systems (NeurIPS)* (2022) (cit. on p. 3).
- [6] Hang Zhang, Yifu Lu, Shang Yang, and Chaoyi Lin. «Eigen Attention: Attention in Low-Rank Space for KV Cache Compression». In: *arXiv preprint arXiv:2408.05646* (2024) (cit. on pp. 3, 19, 20, 25, 30, 47, 49).
- [7] Luca Benfenati, Matteo Risso, Andrea Vannozzi, Ahmet Caner Yüzügüler, Lukas Cavigelli, Enrico Macii, Daniele Jahier Pagliari, and Alessio Burrello. *Don't be so Stief! Learning KV Cache low-rank approximation over the Stiefel manifold*. 2026. arXiv: 2601.21686 [cs.LG]. URL: <https://arxiv.org/abs/2601.21686> (cit. on pp. 6, 73, 74).
- [8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. «Attention Is All You Need». In: *Advances in Neural Information Processing Systems (NeurIPS)* (2017) (cit. on pp. 7, 9).

- [9] Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. «RoFormer: Enhanced Transformer with Rotary Position Embedding». In: *arXiv preprint arXiv:2104.09864* (2021) (cit. on p. 8).
- [10] Joshua Ainslie et al. «GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints». In: *arXiv preprint arXiv:2305.13245* (2023) (cit. on pp. 8, 26, 27, 29, 30).
- [11] Noam Shazeer. «Fast Transformer Decoding: One Write-Head is All You Need». In: *arXiv preprint arXiv:1911.02150* (2019) (cit. on pp. 10, 26, 27).
- [12] NVIDIA Developer Blog. *Mastering LLM Techniques: Inference Optimization*. URL: <https://developer.nvidia.com/blog/mastering-llm-techniques-inference-optimization/> (cit. on p. 12).
- [13] Medium – Data Science. *Singular Value Decomposition (SVD) Demystified*. URL: <https://medium.com/data-science/singular-value-decomposition-svd-demystified-57fc44b802a0> (cit. on p. 16).
- [14] Carl Eckart and Gale Young. «The Approximation of One Matrix by Another of Lower Rank». In: *Psychometrika* 1.3 (1936), pp. 211–218 (cit. on p. 16).
- [15] Leon Mirsky. «Symmetric Gauge Functions and Unitarily Invariant Norms». In: *The Quarterly Journal of Mathematics* 11.1 (1960), pp. 50–59 (cit. on p. 16).
- [16] Ziang Xi, Aining Qin, Yang You, and Chaojie Wang. «KQ-SVD: Compressing the KV Cache with Provable Guarantees on Attention Fidelity». In: *arXiv preprint arXiv:2512.05916* (2025) (cit. on pp. 19, 20, 30–32, 74).
- [17] Yikai Kong, Xiangbo Yin, Weihao Zhan, Huan Yuan, et al. «A3 : an Analytical Low-Rank Approximation Framework for Attention». In: *arXiv preprint arXiv:2505.12942* (2025) (cit. on pp. 19–21, 30–32, 74).
- [18] DeepSeek-AI. «DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model». In: *arXiv preprint arXiv:2405.04434* (2024) (cit. on pp. 22, 23, 30, 32).
- [19] DeepSeek-AI et al. «DeepSeek-V3 Technical Report». In: *arXiv preprint arXiv:2412.19437* (2024) (cit. on p. 22).
- [20] Fanxu Meng, Pingzhi Tang, Xiaojuan Tang, Zengwei Yao, Xing Sun, and Muhan Zhang. «TransMLA: Multi-Head Latent Attention Is All You Need». In: *arXiv preprint arXiv:2502.07864* (2025) (cit. on p. 22).
- [21] Hongyang Wang et al. «SVDq: Sparse and Informed Quantization for 1.25-Bit LLM Weight Compression». In: *arXiv preprint arXiv:2502.15304* (2025) (cit. on pp. 22, 23).

-
- [22] Zirui Liu, Weiyan Lin, Peng Sun, Dong Zhou, Yuxiong You, Zheng Zhang, and Guang Zhao. «KIVI: A Tuning-Free Asymmetric 2bit Quantization for KV Cache». In: *arXiv preprint arXiv:2402.02750* (2024) (cit. on pp. 22, 72).
 - [23] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Re. «FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness». In: *Advances in Neural Information Processing Systems (NeurIPS)* (2022) (cit. on pp. 24, 26).
 - [24] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. «Efficient Streaming Language Models with Attention Sinks». In: *arXiv preprint arXiv:2309.17453* (2023) (cit. on p. 24).
 - [25] Zhenyu Zhang, Yilun Zhao, Yujun Zhou, Mikko Lipasti, Zhiqiang Chen, and Zhihao Jia. «H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models». In: *arXiv preprint arXiv:2306.14048* (2023) (cit. on pp. 24, 31).
 - [26] Mengzhou Xia, Zeliang Zhong, Yifan Gao, Yue Chen, Xia Hu, and Nigel Collier. «Scissorhands: Exploiting the Persistence of Importance Hypothesis for LLM KV Cache Compression at Test Time». In: *arXiv preprint arXiv:2305.17118* (2023) (cit. on pp. 24, 31).
 - [27] Yue Guan et al. «NACL: A General and Effective KV Cache Eviction Framework for LLM at Inference Time». In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL 2024)*. 2024. DOI: 10.18653/v1/2024.acl-long.428 (cit. on pp. 25, 31).
 - [28] Xinyu Liu, Yang Shang, Guodong Wei, Zhenhua Zhao, Tong Wang, Ting Sun, and Yu Li. «Sequence Can Secretly Tell You What to Discard for KV Cache Compression». In: *arXiv preprint arXiv:2404.15949* (2024) (cit. on pp. 25, 31, 73).
 - [29] Yucheng Li, Kaixin Yang, Haowei Zhang, Jie Gong, Zhiyong Huang, Yinghan Miao, Wei Wang, Bin Yu, and Jinyang Kim. «SnapKV: LLM Knows What You are Looking for Before Generation». In: *arXiv preprint arXiv:2404.14469* (2024) (cit. on pp. 25, 31, 73).
 - [30] Anup Jha. *Supercharge your Large Language Model Serving for Faster Inferencing Low Memory*. URL: <https://medium.com/@anupjha799/supercharge-your-large-language-model-serving-for-faster-inferencing-low-memory-6091b52addc4> (cit. on p. 25).
 - [31] Tri Dao. «FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning». In: *arXiv preprint arXiv:2307.08691* (2023) (cit. on p. 26).

-
- [32] Shengrui Zhang, Ziyang Wang, Lifu Zhang, Ling Ouyang, Xiangyu He, Hui Cheng, and Minlie Huang. «xKV: Cross-Layer SVD for KV Cache Compression». In: *arXiv preprint arXiv:2503.18893* (2025) (cit. on pp. 26–32, 74).
 - [33] Zefan Cai, Wen Xiao, Yucheng Li, Zhiping Guo, Zhouchao Yin, Baobao Bao, and Kaixin Yang. «PyramidKV: Dynamic KV Cache Compression Based on Pyramidal Information Funneling». In: *arXiv preprint arXiv:2406.02069* (2024) (cit. on pp. 28, 73).
 - [34] Simiao Duan, Yuxin Yang, Haorui Li, Qian Liu, Jian Zhou, and Weinan Zhang. «PyramidInfer: Pyramid KV Cache Compression for High-throughput LLM Inference». In: *Findings of the Association for Computational Linguistics: ACL 2024*. 2024. DOI: 10.18653/v1/2024.findings-acl.195 (cit. on p. 28).
 - [35] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. «ALBERT: A Lite BERT for Self-supervised Learning of Language Representations». In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019 (cit. on p. 29).
 - [36] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. «Universal Transformers». In: *International Conference on Learning Representations (ICLR)*. 2019 (cit. on p. 29).
 - [37] P.-A. Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008 (cit. on p. 40).
 - [38] Catalin Ionescu, Orestis Vantzos, and Cristian Sminchisescu. «Matrix Back-propagation for Deep Networks with Structured Layers». In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 2965–2973 (cit. on p. 40).
 - [39] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. «Pointer Sentinel Mixture Models». In: *arXiv preprint arXiv:1609.07843* (2016) (cit. on pp. 45, 47).
 - [40] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. «Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer». In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67 (cit. on p. 47).
 - [41] Leo Gao et al. «A Framework for Few-Shot Language Model Evaluation». In: *arXiv preprint arXiv:2210.07316* (2024) (cit. on pp. 47, 49, 52).
 - [42] 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 (ACL)*. 2019 (cit. on p. 48).

- [43] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, and Yejin Choi. «PIQA: Reasoning about Physical Commonsense in Natural Language». In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 2020 (cit. on p. 48).
- [44] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. «Measuring Massive Multitask Language Understanding». In: *International Conference on Learning Representations (ICLR)*. 2021 (cit. on pp. 48, 73).
- [45] Lorenz K. Müller, Philippe Bich, Jiawei Zhuang, Ahmet Çelik, Luca Benfenati, and Lukas Cavigelli. *SINQ: Sinkhorn-Normalized Quantization for Calibration-Free Low-Precision LLM Weights*. 2026. arXiv: 2509.22944 [cs.LG]. URL: <https://arxiv.org/abs/2509.22944> (cit. on p. 72).
- [46] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Guangxuan Xiao, and Song Han. «AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration». In: *GetMobile: Mobile Comp. and Comm.* 28.4 (Jan. 2025), pp. 12–17. ISSN: 2375-0529. DOI: 10.1145/3714983.3714987. URL: <https://doi.org/10.1145/3714983.3714987> (cit. on p. 72).