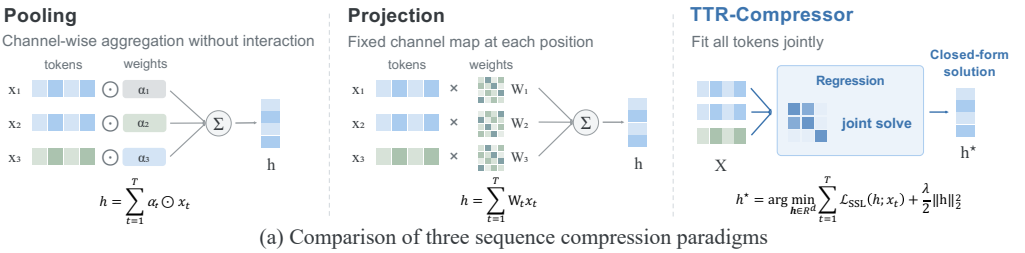
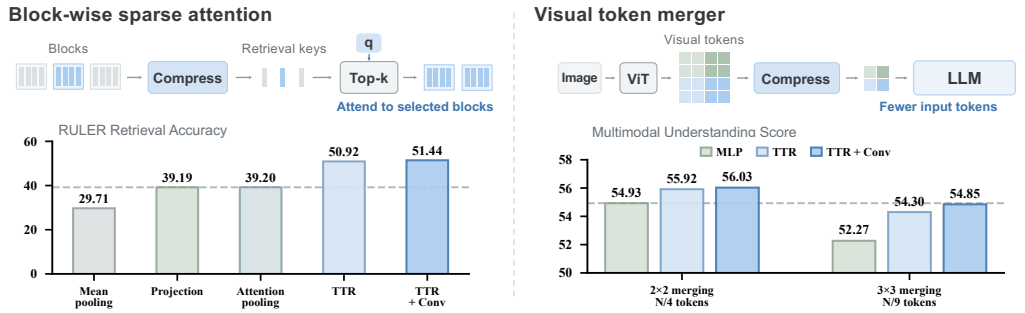
RETHINKING SEQUENCE COMPRESSION WITH TEST-TIME REGRESSION

Anonymous authors
Paper under double-blind review



(a) Comparison of three sequence compression paradigms



(b) Applications and empirical comparison

Figure 1: **Overview of TTR-Compressor.** (a) Unlike pooling and projection, which process tokens independently, TTR-Compressor fits all tokens jointly and thereby captures the interplay among them. (b) TTR-Compressor outperforms existing methods on both block-wise sparse attention and visual token merging.

ABSTRACT

Sequence compression is a fundamental operation and a critical information bottleneck in efficient language and multimodal models, and yet prevailing approaches based on pooling or learned projections remain underperforming. We revisit sequence compression through the lens of test-time regression: input tokens define a regression problem whose fitted coefficients form the compressed representation. Importantly, the regression problem is learned end-to-end via backpropagation, while its coefficients are computed efficiently in closed form during the forward pass. This leads to TTR-Compressor, a general alternative to conventional pooling and projection-based compression. We evaluate TTR-Compressor as the compressor for block-wise sparse attention and the visual token merger for multimodal large language models. On block-wise sparse attention, TTR-Compressor improves RULER performance over attention-based compressors by a large margin. For visual token merging, it consistently outperforms the standard MLP merger and matches its performance with 55.6% fewer visual tokens. These results establish test-time regression as an effective primitive for sequence compression in efficient language and multimodal models.

1 INTRODUCTION

Sequence compression is a ubiquitous operation in modern efficient large model architectures (Qwen Team, 2026b; DeepSeek-AI, 2026), yet remains surprisingly underexplored. For example, block-wise sparse attention compresses each token block into a summary vector used to score and select

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

relevant blocks (Jiang et al., 2024; Lu et al., 2025; Yuan et al., 2025), while multimodal large language models (MLLMs) merge neighboring visual features into compact tokens before feeding them into the language backbone (Wang et al., 2025; Bai et al., 2025; Team et al., 2026a). In both settings, the compressor acts as a critical information bottleneck: once information is discarded, it cannot be recovered downstream. One would therefore expect compression to be a carefully designed architectural component. In practice, however, most models rely on simple defaults, such as parameter-free pooling, learned projections over concatenated tokens, or at best, attention-weighted pooling.

We argue that these default compressors share a fundamental structural limitation: *they aggregate tokens using weights that do not account for inter-token relationships*. Mean pooling uses uniform weights, linear projection uses fixed position-dependent weights, and attention-weighted pooling scores each token independently. As a result, redundant tokens can reinforce one another, while distinctive information is diluted. From a statistical perspective, these methods primarily exploit *first-order* token statistics while ignoring *second-order* inter-token correlations, the very structure that reveals which information is redundant and which is distinctive.

This paper introduces a fundamentally different compression strategy. Rather than asking *which weighted average* best summarizes a set of tokens, we let the tokens define a regression problem and take its *fitted coefficients* as the summary. Specifically, our proposed TTR-Compressor, constructs a self-supervised ridge-regression problem directly from the input tokens, requiring no external labels, and uses its closed-form solution as the compressed representation. This formulation creates a natural division: the outer training loop meta-learns *which* regression problem produces task-useful representations, while the inner closed-form solution determines *how* that problem should be fitted to each input. The key distinction from conventional pooling and projection arises from the Gram matrix in the regression solution. By explicitly capturing inter-token correlations, TTR-Compressor fits correlated tokens jointly, so redundant evidence has diminishing rather than purely additive influence. This input-adaptive behavior cannot be reproduced by a fixed projection or additive token combination. Moreover, learning the regularizer removes the need for ridge hyperparameter tuning in large-scale pretraining. Because the compression window is typically smaller than the channel dimension (*i.e.*, $T < d$), we solve the dual problem with an efficient $T \times T$ Cholesky factorization. Thus, TTR-Compressor incurs minimal overhead and introduces only $2d$ learnable parameters per compressor, making it an effective drop-in replacement for conventional compression methods.

We evaluate TTR-Compressor on two applications that impose fundamentally different requirements on compression. In *block-wise sparse attention*, the compressed summary must act as a faithful *retrieval key*. In this setting, TTR-Compressor consistently outperforms traditional baselines on RULER (Hsieh et al., 2024) retrieval tasks, surpassing sophisticated attention-based compressors (Hu et al., 2026) by a considerable margin. In *visual token merging*, by contrast, the compressed summary must preserve a faithful *representation*. Here, TTR-Compressor consistently outperforms the widely used MLP merger across diverse compression ratios, and achieves comparable performance with 55.6% fewer visual tokens. Our contributions are summarized below:

- **A novel compression framework.** We introduce a test-time regression framework for sequence compression, reframing token summarization as an instance-adaptive optimization problem.
- **A practical operator.** We derive an efficient dual closed-form solution and eliminate hyperparameter tuning, making TTR-Compressor fully practical and scalable for large-scale pretraining.
- **Empirical effectiveness.** As a drop-in component in large models, TTR-Compressor consistently outperforms conventional baselines in both block-wise sparse attention and visual token merging.

2 Related Work

Sequence compression. Compressing a sequence of tokens into a single vector is a fundamental operation across modern architectures, with applications ranging from multimodal token reduction (Bolya et al., 2022; Li et al., 2023; Yao et al., 2024; Cha et al., 2024; Chu et al., 2024) to block-wise sparse attention (Jiang et al., 2024; Lu et al., 2025; Gao et al., 2024; Lai et al., 2025). Existing compression operators largely fall into three paradigms: (1) *parameter-free pooling*, such as mean pooling; (2) *concatenation followed by learned projection*; and (3) *attention-based compression*, which summarizes tokens via attention-weighted averaging (Mohtashami & Jaggi, 2023; Yang et al., 2025b). A fundamental limitation of these families is that they compute token combinations without explicitly accounting for the inter-token correlations. In contrast, TTR-Compressor

derives its representation as the solution to a linear system whose Gram matrix directly encodes these correlations, providing an expressive, drop-in replacement for the operators above.

Test-time regression and test-time training. The idea of adapting part of a network on each input at inference time dates back to fast-weight programmers (Schmidhuber, 1992; Schlag et al., 2021) and was revived as test-time training (TTT), first for robustness under distribution shift (Sun et al., 2020) and then as sequence-modeling layers whose hidden states are small models updated by gradient descent on a self-supervised loss (Sun et al., 2024; Behrouz et al., 2026b;a; Han et al., 2026). In-Place TTT (Feng et al., 2026) further extends this paradigm by directly adapting the existing MLP projection weights of pretrained Transformers. A parallel line of work reinterprets linear attention and its descendants (e.g., DeltaNet (Yang et al., 2024), Gated DeltaNet (Yang et al., 2025c), Longhorn (Liu et al., 2025), Titans (Behrouz et al., 2026b)) as online regression: each layer maintains a key-to-value regressor that is refined with one gradient or delta-rule step per token and read out by the query. MesaNet (von Oswald et al., 2026) pushes the inner optimization to convergence, solving the problem at every time step with conjugate gradients, while LaCT (Zhang et al., 2026) updates fast weights over large, unordered chunks to improve hardware utilization.

TTR-Compressor differs from these methods in two fundamental ways. First, because compression operates independently on small, disjoint blocks without cross-block dependencies, the regression admits an efficient, *exact closed-form solution* via a single $T \times T$ dual solve. This avoids both iterative gradient-based inner optimization (as in TTT, ViT^3, In-Place TTT, and MesaNet) and single-step approximations (such as delta-rule updates). Second, rather than using test-time regression to produce input-dependent model weights (Bertinetto et al., 2018; Sun et al., 2020) or a fast-weight memory that is subsequently queried (Schlag et al., 2021; Yang et al., 2025c; Behrouz et al., 2026b), we use regression to directly yield a *representation* of a token block. Specifically, the fitted coefficients themselves serve as the compressed representation (e.g., a visual token or a retrieval key).

3 Joint Sequence Compression via Test-time Regression

We introduce TTR-Compressor, a sequence compression operator based on *test-time regression*, as shown in Figure 2(a). Given a sequence of token representations, TTR-Compressor constructs a regression problem from the tokens and uses its fitted coefficients as the compressed representation. We first present the regression formulation of compression and specify its construction, then derive an efficient solution and describe how TTR-Compressor is learned end-to-end within a model.

3.1 Compression as Test-time Regression

We formulate sequence compression as ridge regression (Hoerl & Kennard, 1970): the tokens in a sequence provide regression observations, and the fitted regression coefficients serve as a compact representation of that sequence. The regression objective encourages this representation to capture the relationships among tokens, while regularization controls the magnitude of the fitted coefficients.

Given a sequence $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T]^\top \in \mathbb{R}^{T \times d}$, our goal is to compress its T token representations into a single vector $\boldsymbol{h} \in \mathbb{R}^d$. We associate each token with a regression key $\boldsymbol{k}_t \in \mathbb{R}^d$ and a scalar target $v_t \in \mathbb{R}$, and define the compressed representation as:

$$\boldsymbol{h}^\star = \arg\min_{\boldsymbol{h} \in \mathbb{R}^d} \frac{1}{2} \sum_{t=1}^{T} \omega_t \big(\boldsymbol{k}_t^\top \boldsymbol{h} - v_t\big)^2 + \frac{\lambda}{2} \|\boldsymbol{h}\|_2^2, \tag{1}$$

where $\omega_t \geq 0$ determines the contribution of each token and $\lambda > 0$ controls regularization. The vector $\boldsymbol{h}^\star$ summarizes the sequence by fitting the key-target relationships across its tokens. The regression problem is constructed and solved for every input sequence, during both training and inference. Here, *test-time* refers to fitting these input-specific coefficients; the learned model parameters remain fixed at inference.

3.2 Constructing the Regression Problem

Keys and Targets. We construct regression keys by normalizing the input features and generate scalar targets using a learned projection:

$$\boldsymbol{k}_t = \frac{\boldsymbol{x}_t}{\|\boldsymbol{x}_t\|_2}, \qquad v_t = \boldsymbol{a}^\top \boldsymbol{x}_t, \tag{2}$$

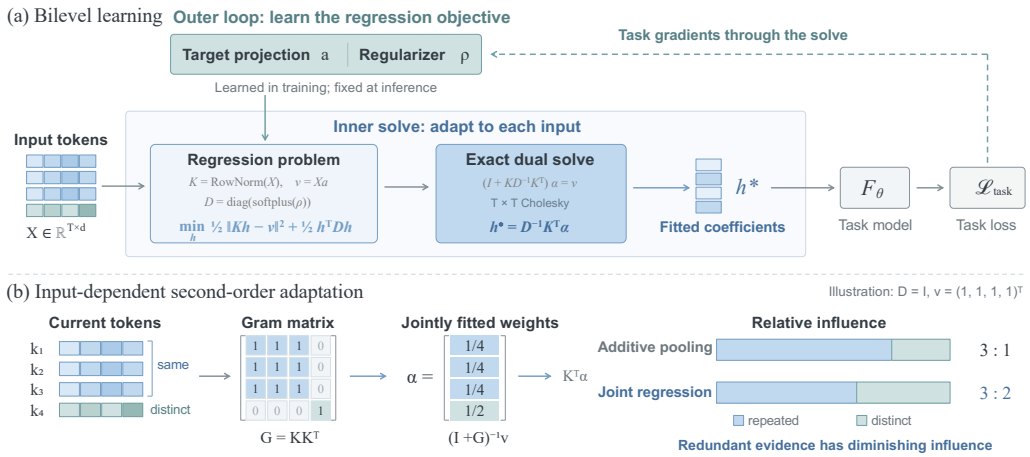


Figure 2: **Overview of TTR-Compressor.** (a) Bilevel learning: The inner solve formulates sequence compression as test-time ridge regression with an exact dual Cholesky solve to obtain the representation $\boldsymbol{h}^*$, while the regression parameters are learned end-to-end via task gradients. (b) Input-dependent second-order adaptation: By jointly coupling tokens through the Gram matrix, TTR naturally diminishes the relative influence of redundant tokens compared to additive pooling. For intuitive illustration, we set $\boldsymbol{D} = \boldsymbol{I}$ and uniform targets $\boldsymbol{v} = \mathbf{1}$; as typically learned in practice, a smaller regularizer $\boldsymbol{D}$ further amplifies this deduplication effect, yielding an even higher relative share for distinct tokens.

where $\boldsymbol{a} \in \mathbb{R}^d$ is shared across tokens and sequences within a compressor. Key normalization ensures that $\boldsymbol{K}^\top \boldsymbol{K}$ measures directional (semantic) similarity alone. Crucially, this normalization is asymmetric: targets retain original token magnitudes ($v_t = \boldsymbol{a}^\top \boldsymbol{x}_t$). If targets were also normalized ($v_t = \boldsymbol{a}^\top \boldsymbol{k}_t$), the fitting term would reduce to $\frac{1}{2}\|\boldsymbol{K}(\boldsymbol{h} - \boldsymbol{a})\|_2^2$, trivially collapsing $\boldsymbol{h}^\star$ toward the sequence-invariant static vector $\boldsymbol{a}$. By preserving token norms in the targets, no constant vector can fit varying magnitudes, forcing $\boldsymbol{h}^\star$ to dynamically integrate both token semantics and their relative importance. The targets are derived entirely from the input, so fitting $\boldsymbol{h}^\star$ requires no additional labels. Optionally, targets can be further enriched by first applying a local convolution (1D for language and 2D for vision) to aggregate neighboring context before the projection, *i.e.*, $v_t = \boldsymbol{a}^\top \mathrm{Conv}(\boldsymbol{X})$, where $\mathrm{Conv}(\cdot)$ denotes the convolution operator and $\boldsymbol{X}$ represents the local context window of n neighboring tokens around position t.

Uniform Token Contributions. We use uniform weights $\omega_t = 1$, assigning the same coefficient to every token's regression error. Let $\boldsymbol{K} = [\boldsymbol{k}_1, \ldots, \boldsymbol{k}_T]^\top \in \mathbb{R}^{T \times d}$ and $\boldsymbol{v} = [v_1, \ldots, v_T]^\top \in \mathbb{R}^T$. Setting the gradient of Eq. 1 to zero gives:

$$(\boldsymbol{K}^\top \boldsymbol{K} + \lambda \boldsymbol{I}_d)\boldsymbol{h}^\star = \boldsymbol{K}^\top \boldsymbol{v}, \qquad \boldsymbol{h}^\star = (\boldsymbol{K}^\top \boldsymbol{K} + \lambda \boldsymbol{I}_d)^{-1} \boldsymbol{K}^\top \boldsymbol{v}. \tag{3}$$

Since $\lambda > 0$, the objective is strictly convex and has a unique solution. The solution combines the key-target statistic $\boldsymbol{K}^\top \boldsymbol{v}$ with the sequence-dependent feature geometry $\boldsymbol{K}^\top \boldsymbol{K}$.

Learnable Diagonal Regularization. In conventional ridge regression, λ is a tunable hyperparameter. Such tuning is expensive during large-model pretraining and complicates scaling to different model sizes. To avoid this, we replace $\lambda \boldsymbol{I}_d$ with a learnable positive diagonal matrix, yielding a generalized ridge regression objective (Tikhonov & Arsenin, 1977):

$$\boldsymbol{D} = \mathrm{diag}(\delta_1, \ldots, \delta_d), \qquad \delta_j = \mathrm{softplus}(\rho_j), \tag{4}$$

where $\boldsymbol{\rho} \in \mathbb{R}^d$ is trainable. The resulting objective and its solution are:

$$\boldsymbol{h}^\star = \arg\min_{\boldsymbol{h}} \frac{1}{2}\|\boldsymbol{K}\boldsymbol{h} - \boldsymbol{v}\|_2^2 + \frac{1}{2}\boldsymbol{h}^\top \boldsymbol{D}\boldsymbol{h}, \tag{5}$$

$$= (\boldsymbol{K}^\top \boldsymbol{K} + \boldsymbol{D})^{-1} \boldsymbol{K}^\top \boldsymbol{v}. \tag{6}$$

3.3 Efficient Dual-form Solution

Directly evaluating Eq. 3 requires solving a $d \times d$ system. When the number of tokens being compressed is typically smaller than the feature dimension, we instead use the equivalent dual

form (Saunders et al., 1998; Jordan et al., 2006):

$$\boldsymbol{h}^\star = \boldsymbol{K}^\top(\boldsymbol{K}\boldsymbol{K}^\top + \lambda \boldsymbol{I}_T)^{-1}\boldsymbol{v}. \tag{7}$$

This replaces the feature-space system with a $T \times T$ token-space system. Its entries depend on pairwise similarities between regression keys, allowing the fitted representation to account for correlations among tokens.

We evaluate Eq. 7 by solving $(\boldsymbol{K}\boldsymbol{K}^\top + \lambda \boldsymbol{I}_T)\boldsymbol{\alpha} = \boldsymbol{v}$ and computing $\boldsymbol{h}^\star = \boldsymbol{K}^\top\boldsymbol{\alpha}$, without explicitly forming an inverse. The resulting computational cost is $\mathcal{O}(T^2d + T^3)$, compared with $\mathcal{O}(Td^2 + d^3)$ for the direct primal formulation. The dual form is therefore particularly suitable for compressing short sequences of high-dimensional features.

Obtaining an Efficient Dual Solution. With the learnable diagonal regularization introduced in Section 3.2, $\lambda \boldsymbol{I}_d$ is replaced by $\boldsymbol{D}$. To obtain an efficient dual solution, we rescale the keys as $\widetilde{\boldsymbol{K}} = \boldsymbol{K}\boldsymbol{D}^{-1/2}$, yielding the final form of TTR-Compressor:

$$\boxed{\mathrm{TTR}(\boldsymbol{X}) = \boldsymbol{D}^{-1/2}\widetilde{\boldsymbol{K}}^\top\left(\boldsymbol{I}_T + \widetilde{\boldsymbol{K}}\widetilde{\boldsymbol{K}}^\top\right)^{-1}\boldsymbol{v}} \tag{8}$$

Because $\boldsymbol{D}$ is diagonal, rescaling the keys costs only $\mathcal{O}(Td)$, preserving the dual formulation's overall complexity. In practice, we compute a Cholesky factorization $\boldsymbol{L}\boldsymbol{L}^\top = \boldsymbol{I}_T + \widetilde{\boldsymbol{K}}\widetilde{\boldsymbol{K}}^\top$, solve $\boldsymbol{L}\boldsymbol{L}^\top\boldsymbol{\alpha} = \boldsymbol{v}$ through triangular solutions, and return $\boldsymbol{D}^{-1/2}\widetilde{\boldsymbol{K}}^\top\boldsymbol{\alpha}$.

3.4 End-to-end Training

The target projection $\boldsymbol{a}$ and diagonal parameters $\boldsymbol{\rho}$ are optimized end-to-end through the differentiable closed-form solution using the original model's training objective. Each compressor maintains its own parameters, which are shared across all input sequences. At inference time, these learned parameters remain fixed, while $\boldsymbol{h}^\star$ is recomputed independently for each input sequence. TTR-Compressor therefore combines a learned regression objective with an exact, input-adaptive solution, without requiring iterative gradient-based optimization at test time.

4 Understanding TTR-Compressor

The learning process of TTR-Compressor can be decomposed into two distinct mechanisms: outer-loop training learns which regression problem is useful for compression, while the inner solution adapts the compressed representation to the current input's statistics. We examine them separately.

4.1 Task-Supervised Learning of the Compression Objective

The regression objective is not prescribed independently of the task. Its targets $v_t = \boldsymbol{a}^\top\boldsymbol{x}_t$ and regularization matrix $\boldsymbol{D}$ are learned through the downstream use of the compressed representation. Writing $\boldsymbol{h}^\star_{\boldsymbol{a},\boldsymbol{\rho}}$ for the regression solution, the training objective takes the form:

$$\min_{\theta,\boldsymbol{a},\boldsymbol{\rho}} \mathbb{E}_{(x,y)}\left[\mathcal{L}_{\text{task}}\left(F_\theta\left(\boldsymbol{h}^\star_{\boldsymbol{a},\boldsymbol{\rho}}(\boldsymbol{X}_\theta(x))\right), y\right)\right], \tag{9}$$

where $\boldsymbol{X}_\theta$ produces the input representations and F_θ denotes the subsequent computation. Thus, the outer objective evaluates the usefulness of $\boldsymbol{h}^\star$ for the final task, rather than leveraging its learnability to merely minimize the inner regression error.

Learning the Targets and Regularization. This coupling can be seen directly from the gradients. Let $\boldsymbol{A} = \boldsymbol{K}^\top\boldsymbol{K} + \boldsymbol{D}$, $\boldsymbol{g} = \nabla_{\boldsymbol{h}^\star}\mathcal{L}_{\text{task}}$, and $\boldsymbol{s} = \boldsymbol{A}^{-1}\boldsymbol{g}$. Differentiating the regression solution gives:

$$\nabla_{\boldsymbol{a}}\mathcal{L}_{\text{task}} = \boldsymbol{X}^\top\boldsymbol{K}\boldsymbol{s}, \qquad \nabla_{\boldsymbol{\rho}}\mathcal{L}_{\text{task}} = -\operatorname{sigmoid}(\boldsymbol{\rho}) \odot \boldsymbol{h}^\star \odot \boldsymbol{s}, \tag{10}$$

where $\odot$ denotes element-wise multiplication. The downstream gradient therefore shapes both the scalar targets derived from tokens via $\boldsymbol{a}$ and the coordinate-wise penalties imposed through $\boldsymbol{\rho}$. Consequently, the scalar targets require no predefined semantics; instead, they serve as learned measurements whose fitted coefficients yield informative representations for downstream layers. In this sense, the regression provides a structural inductive bias for compression, while outer-loop optimization steers this structural bias towards better handling the downstream task.

4.2 Input-Dependent Second-Order Adaptation

Beyond learning the regression objective, the inner solve adapts the compressed representation to the correlation structure of each input. For isotropic regularization, the dual-form solution is

$$\boldsymbol{h}^\star = \boldsymbol{K}^\top (\boldsymbol{K}\boldsymbol{K}^\top + \lambda \boldsymbol{I})^{-1} \boldsymbol{v}. \tag{11}$$

Let $\boldsymbol{G} = \boldsymbol{K}\boldsymbol{K}^\top \in \mathbb{R}^{T\times T}$. Since the keys are normalized, each entry $G_{ij} = \boldsymbol{k}_i^\top \boldsymbol{k}_j$ directly measures the similarity between tokens i and j. Thus, $\boldsymbol{G}$ captures the input-dependent second-order correlation structure of the token group.

The compressed representation can equivalently be written as

$$\boldsymbol{h}^\star = \boldsymbol{K}^\top \boldsymbol{\alpha}, \qquad \boldsymbol{\alpha} = (\boldsymbol{G} + \lambda \boldsymbol{I})^{-1} \boldsymbol{v}. \tag{12}$$

Unlike pooling and projection, where token contributions are aggregated independently, the effective weights $\boldsymbol{\alpha}$ are determined jointly by the target responses and all pairwise token correlations. Highly similar tokens are explicitly coupled through the Gram matrix, so redundant evidence exhibits diminishing rather than additive influence. Conversely, information not already explained by strongly correlated tokens can retain a larger relative influence when supported by the regression targets, as shown in Figure 2 (b). TTR therefore adapts its compression rule to the second-order structure of every input rather than applying a fixed or independently weighted combination of tokens.

Learning the inter-token geometry. With the learnable diagonal regularizer $\boldsymbol{D}$, this interpretation extends to the rescaled keys $\widetilde{\boldsymbol{K}} = \boldsymbol{K}\boldsymbol{D}^{-1/2}$. The corresponding token Gram matrix becomes

$$\widetilde{\boldsymbol{G}} = \widetilde{\boldsymbol{K}}\widetilde{\boldsymbol{K}}^\top = \boldsymbol{K}\boldsymbol{D}^{-1}\boldsymbol{K}^\top. \tag{13}$$

Training therefore learns which feature dimensions should matter more when measuring token similarity. The outer loop learns a task-adapted correlation geometry, while the inner solve adapts the compressed representation to the correlation structure of each individual input.

5 Applications and Experimental Results

We evaluate TTR-Compressor in two settings: block-wise sparse attention for efficient language modeling, where compressed representations guide context selection, and visual token merging for efficient multimodal understanding, where they serve as the visual input to the language backbone.

5.1 Block-wise Sparse Attention

Architecture and Baselines. Block-wise sparse attention partitions the context into blocks, scores their compressed representations against each query, and attends to tokens in selected blocks alongside a local window. The compressor therefore determines which distant information remains accessible: an uninformative summary can prevent a relevant block from being selected, regardless of the quality of its individual token representations.

We adopt HiLS-Attention (Hu et al., 2026) as a base framework and vary only the compressor. HiLS-Attention has three effective architectural components: hierarchical softmax attention, which combines within-block attention with learned block-level weights; low-rank query calibration, which adapts queries for scoring compressed representations; and HoPE, which applies RoPE to only a subset of feature dimensions to support length extrapolation. We compare four compressors: mean pooling, concat & projection, the attention-weighted pooling used in the original HiLS-Attention, and TTR-Compressor. Full attention and sliding-window attention serve as additional references.

Training and Evaluation. We train 345M-parameter language models on the FineWeb dataset (Penedo et al., 2024) with an 8K context length. Training runs for 30,000 optimization steps with a global batch size of 128 sequences, corresponding to approximately 30B tokens, and uses AdamW with a learning rate of 3×10^{-4}. All sparse attention variants use 64-token blocks, select up to 32 distant blocks per query, and retain a 512-token local window.

Evaluation covers two complementary aspects. First, since queries use the compressed representations to retrieve relevant blocks, retrieval accuracy directly reflects compression quality. Following (Hatamizadeh et al., 2026), we evaluate on the single-needle (S-NIAH) and multi-key (MK-NIAH) retrieval tasks from RULER (Hsieh et al., 2024), which require locating relevant information

Model	S-NIAH-1				S-NIAH-2				S-NIAH-3				MK-NIAH-1				Avg.
	4K	8K	16K	32K	4K	8K	16K	32K	4K	8K	16K	32K	4K	8K	16K	32K	
Full Attention	0.834	0.428	0	0	0.850	0.412	0	0	0.572	0.470	0	0	0.374	0.272	0	0	0.2632
Sliding Window Attention	0.116	0.050	0.044	0.016	0.148	0.048	0.034	0.030	0.112	0.094	0.030	0.034	0.126	0.080	0.026	0.028	0.0635
Mean Pooling	0.938	0.536	0.470	0.270	0.840	0.258	0.042	0.050	0.292	0.282	0.040	0.042	0.366	0.230	0.042	0.056	0.2971
Concat & Projection	0.892	0.840	0.792	0.582	0.764	0.232	0.124	0.068	0.512	0.324	0.174	0.076	0.476	0.278	0.084	0.052	0.3919
Attention-Weighted Pooling	0.944	0.914	0.876	0.740	0.926	0.274	0.136	0.060	0.340	0.082	0.026	0.026	0.474	0.258	0.124	0.072	0.3920
TTR-Compressor	0.996	0.996	0.992	0.650	**0.982**	**0.566**	**0.288**	**0.126**	0.770	0.376	**0.188**	0.066	0.528	**0.302**	**0.212**	**0.110**	0.5092
TTR-Compressor (w/ Conv)	**0.998**	**1.000**	**0.994**	**0.988**	0.960	0.454	0.128	0.102	**0.834**	**0.482**	0.130	**0.096**	**0.542**	0.298	0.116	0.108	**0.5144**

Table 1: Accuracy on Single Needle-In-A-Haystack (S-NIAH) and Multi-Key Needle-In-A-Haystack (MK-NIAH) tasks from RULER across varied context lengths. Best values for each context length are **bolded**; second-best values are underlined.

Model	ARC-c	ARC-e	BoolQ	COPA	Hella.	LMB.	OBQA	PIQA	SciQ	Wino.	Avg.
	acc ↑	acc ↑	acc ↑	acc ↑	acc ↑	acc ↑	acc ↑	acc ↑	acc ↑	acc ↑	acc ↑
Full Attention	24.74	42.09	60.15	68.00	39.85	35.71	29.60	67.74	62.70	51.07	48.16
Sliding Window Attention	24.91	41.96	55.02	69.00	39.53	35.47	29.00	67.14	63.00	50.75	47.58
Mean Pooling	24.66	**42.38**	55.99	66.00	39.62	35.88	28.40	**69.31**	**65.50**	**53.04**	48.08
Concat & Projection	24.83	42.30	58.65	68.00	39.58	**36.62**	30.00	67.74	62.20	51.78	**48.17**
Attention-Weighted Pooling	**25.00**	41.75	**61.22**	65.00	39.61	35.61	29.40	68.93	63.80	50.51	48.08
TTR-Compressor	23.29	41.58	57.95	**71.00**	39.41	36.27	**30.20**	**69.31**	63.00	49.57	48.16
TTR-Compressor (w/ Conv)	24.15	41.33	59.94	70.00	**39.63**	35.36	**30.20**	67.85	62.80	49.96	48.12

Table 2: Performance comparison on language modeling and zero-shot common-sense reasoning. The evaluation context lengths are too short to activate sparse attention, so the compressor is not involved during inference; these results therefore only reflect the common-sense reasoning ability acquired during language model pretraining. Note that Concat & Projection has significantly more parameters than the others. Best values are **bolded**; second-best values are underlined.

among distractors, as the most direct and demanding test of the compressor. We evaluate at 4K–32K context lengths, including extrapolation beyond the 8K training length. Additionally, LAMBADA accuracy and the zero-shot benchmarks in Table 2 assess contextual prediction, commonsense reasoning, and question answering.

Results. As shown in Table 1, the choice of compressor largely determines retrieval quality. TTR-Compressor outperforms all baselines by a large margin, and its advantage persists when extrapolating to 16K and 32K. We attribute this to its second-order modeling: pooling and projection add up tokens independently, so the redundant context that fills most of a block drowns out the few informative tokens. The Gram matrix in TTR-Compressor fits correlated tokens jointly and reduces the influence of redundant evidence, so the resulting key keeps the distinctive content that queries need to find. The convolutional variant brings a further gain, reaching 12.24 points over the strongest baseline, as convolution makes the regression targets more informative and thus improves compression quality. On the benchmarks in Table 2, the evaluation contexts are too short to activate sparse attention, so these results only reflect the common-sense reasoning ability acquired during pretraining. Concat & Projection has substantially more parameters than the other compressors, which may benefit pretraining. TTR-Compressor performs well and on par with full attention.

5.2 Visual Token Merger

Architecture and Baselines. High-resolution images produce many visual tokens, making their direct processing by a language backbone expensive. Modern multimodal models therefore commonly use a spatial merger to reduce the visual sequence length. A widely adopted design concatenates each 2×2 group of neighboring visual features and maps it to one token through a two-layer MLP (Wang et al., 2025; Bai et al., 2025; Qwen Team, 2026a; Team et al., 2026b). Longer multimodal contexts and agentic workflows with repeated visual observations further motivate stronger compression, such as 3×3 merging (Team et al., 2026a; DeepSeek-AI, 2026).

We compare TTR-Compressor with the standard MLP merger. TTR-Compressor fits one regression per spatial group to compress its four or nine visual features into a single vector, which a two-layer MLP then projects to the language model's hidden dimension. Its convolutional variant applies a 3×3 depthwise convolution to the feature map before constructing the regression targets.

Training and Evaluation. We use the ViT of Qwen3-VL (Bai et al., 2025) as the vision encoder and Qwen3-4B-Instruct (Yang et al., 2025a) as the language model. Training consists of two stages.

Method	General VQA			Knowledge Reasoning		Document Understanding			Spatial Under.		Avg.
	MMBench	MMStar	RWQA	MMMU	AI2D	OCRBench	InfoVQA	ChartQA	MMSI	ERQA	
2×2 *Token Compression*											
MLP	76.63	49.93	64.44	**46.22**	75.49	62.80	41.72	70.64	26.20	35.25	54.93
TTR-Compressor	**76.89**	**53.94**	63.40	45.44	**76.94**	62.90	43.74	**73.04**	**26.90**	**36.00**	55.92
TTR-Compressor (w/ Conv)	75.86	52.26	**66.93**	45.67	76.65	**63.80**	**44.33**	72.52	26.30	**36.00**	**56.03**
3×3 *Token Compression*											
MLP	74.74	50.89	64.31	44.44	74.68	55.90	35.16	63.60	26.00	33.00	52.27
TTR-Compressor	76.46	**52.40**	65.62	45.89	**75.23**	58.70	38.27	**67.16**	26.80	**36.50**	54.30
TTR-Compressor (w/ Conv)	**76.63**	51.16	**65.88**	**47.44**	75.16	**60.00**	**39.70**	**67.16**	**28.90**	**36.50**	**54.85**

Table 3: Evaluation across multimodal benchmarks under different token compression ratios. Results are grouped by 2×2 and 3×3 token compression. Best values within each compression group are **bolded**; second-best values are underlined.

The first stage trains only the merger modules on LLaVA-558K (Liu et al., 2023) for one epoch with a learning rate of 10^{-3}. The second stage jointly trains the merger modules and language backbone on 1.3M instruction samples mixing image (Liu et al., 2024a), multi-image (Li et al., 2024), and video (Zhang et al., 2024) data for one epoch with a learning rate of 2×10^{-5}. We evaluate on benchmarks spanning 4 categories: general visual question answering (MMBench (Liu et al., 2024b), MMStar (Chen et al., 2024), and RealWorldQA (xAI, 2024)), knowledge-intensive multimodal reasoning (MMMU (Yue et al., 2024) and AI2D (Kembhavi et al., 2016)), text, infographic, and chart understanding (OCRBench (Liu et al., 2024c), InfoVQA (Tito et al., 2021), and ChartQA (Masry et al., 2022)), and spatial reasoning (MMSI (Yang et al., 2026) and ERQA (Team et al., 2025)).

Results. As shown in Table 3, TTR-Compressor outperforms the MLP merger under both compression ratios. Under 2×2 merging, it raises the average from 54.93 to 55.92, and its convolutional variant further improves it to 56.03. The advantage grows under stronger 3×3 merging. The gains are most pronounced on text-rich benchmarks, where fine-grained details are easily lost during merging: under 3×3 merging, the convolutional variant improves over the MLP merger by 4.10 on OCRBench, 4.54 on InfoVQA, and 3.56 on ChartQA. On average, it outperforms the MLP merger by 2.58 points. Notably, TTR-Compressor (w/ Conv) with 3×3 merging achieves an average of 54.85, close to the 54.93 of the MLP merger with 2×2 merging, while using fewer than half as many visual tokens. We attribute the strong performance of TTR-Compressor to its second-order modeling. The MLP merger concatenates the features in each group and maps them with fixed weights, so the mixing does not depend on how the features relate to each other. When most features in a group are redundant, such as background patches, they dominate the merged token and wash out fine-grained details. In contrast, TTR-Compressor fits correlated features jointly and reduces the influence of redundant ones, so the merged token keeps the distinctive content within the group.

5.3 Efficiency Analysis

Theoretical Complexity and Parameter Efficiency. For a sequence $X \in \mathbb{R}^{T \times d}$ compressed into $h \in \mathbb{R}^{d}$, Table 4 summarizes the asymptotic complexity and parameter count across different paradigms. Under the targeted regime where $T \ll d$ (e.g., $T \in \{4, 9, 64\}$ vs. $d \geq 1024$), TTR-Compressor fundamentally avoids the $\mathcal{O}(d^2)$ overhead and delivers strong, data-dependent expressiveness with only $2d$ parameters.

Paradigm	Complexity	Params
Mean Pooling	$\mathcal{O}(Td)$	0
Linear Projection	$\mathcal{O}(Td^2)$	Td^2
Attention (Reusing)	$\mathcal{O}(Td)$	d
Attention (Standalone)	$\mathcal{O}(Td^2)$	$\sim 4d^2$
TTR-Compressor	$\mathcal{O}(T^2d)$	$2d$

Table 4: Complexity comparison of different token compression methods ($T \ll d$).

Empirical Training Efficiency. Following the setups in Sections 5.1 and 5.2, Figure 3 benchmarks the actual wall-clock training efficiency. For block-wise sparse attention ($8 \times$ H100 GPUs), while mean pooling is the most efficient, its representational capacity is inherently limited; other compression variants exhibit comparable training times. For visual token merging ($2 \times 8 \times$ H100 GPUs), TTR-

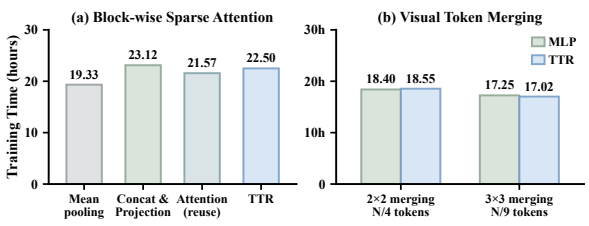


Figure 3: Wall-clock training time comparisons.

Model	S-NIAH-1				S-NIAH-2				S-NIAH-3				MK-NIAH-1				Avg.
	4K	8K	16K	32K	4K	8K	16K	32K	4K	8K	16K	32K	4K	8K	16K	32K	
TTR-Compressor	**0.996**	**0.996**	**0.992**	**0.650**	**0.982**	**0.566**	**0.288**	**0.126**	**0.770**	0.376	**0.188**	0.066	**0.528**	**0.302**	**0.212**	0.110	**0.5092**
w/o normalize $\mathbf{k}$	0.010	0.000	0.002	0.000	0.938	0.254	0.136	0.104	0.764	**0.592**	0.186	**0.092**	0.500	0.258	0.098	**0.116**	0.2531

Table 5: Ablation study on key normalization in TTR-Compressor. We compare the full model with a variant without normalizing $\mathbf{k}$ (w/o normalize $\mathbf{k}$). Best values for each context length are **bolded**.

Model	S-NIAH-1				S-NIAH-2				S-NIAH-3				MK-NIAH-1				Avg.
	4K	8K	16K	32K	4K	8K	16K	32K	4K	8K	16K	32K	4K	8K	16K	32K	
$\lambda = 0.0001$	0.986	0.896	0.826	0.628	0.888	0.348	0.134	0.054	0.654	0.528	0.086	0.108	0.384	0.242	0.118	0.102	0.4364
$\lambda = 0.001$	0.998	0.996	0.990	0.948	**0.998**	0.448	0.186	0.112	0.686	0.420	0.186	0.072	0.466	0.274	0.174	0.082	0.5023
$\lambda = 0.01$	**1.000**	**1.000**	**1.000**	**1.000**	0.936	0.486	0.148	0.076	0.692	0.374	0.042	0.022	0.484	0.288	0.146	0.068	0.4851
$\lambda = 0.1$	0.990	0.986	0.982	0.978	0.932	0.312	0.054	0.068	0.728	0.424	0.088	0.078	0.470	0.252	0.112	0.074	0.4705
$\lambda = 1$	0.872	0.552	0.826	0.794	0.896	0.204	0.062	0.072	0.772	0.496	0.184	0.088	0.526	0.278	0.090	0.068	0.4238
Learnable Scalar	0.988	0.986	0.970	0.806	0.938	0.368	0.104	0.104	**0.788**	**0.536**	**0.212**	**0.112**	**0.554**	0.274	0.132	0.082	0.4971
Learnable Diagonal	0.996	0.996	0.992	0.650	0.982	**0.566**	**0.288**	**0.126**	0.770	0.376	0.188	0.066	0.528	**0.302**	**0.212**	**0.110**	**0.5092**

Table 6: Ablation study on the regularization parameter λ in TTR-Compressor. Best values for each context length are **bolded**; second-best values are underlined.

Compressor achieves efficiency on par with the widely used MLP. Notably, while marginally slower under 2×2 merging, TTR-Compressor outpaces MLP under 3×3 merging, highlighting its superior efficiency and scalability at higher compression ratios.

5.4 Ablation Study and Analysis

Criticality of Key Normalization. As shown in Table 5, omitting key normalization ($\mathbf{k}_t = \mathbf{x}_t / \|\mathbf{x}_t\|_2$) precipitates a collapse across retrieval tasks. We attribute this to two reasons. First, from an optimization perspective, since scalar targets are linearly generated via $v_t = \mathbf{a}^\top \mathbf{x}_t$, using raw keys ($\mathbf{k}_t = \mathbf{x}_t$) makes the static weight $\mathbf{a}$ a trivial exact fit ($\mathbf{k}_t^\top \mathbf{a} \equiv v_t$). Consequently, the regularized solution $\mathbf{h}^\star$ is biased toward the fixed vector $\mathbf{a}$, which weakens the instance adaptivity of test-time regression and pushes it toward an input-invariant projection. Second, from a geometric and numerical viewpoint, outlier tokens with large norms dominate the Gram matrix $\mathbf{K}\mathbf{K}^\top$, causing correlation modeling to reflect token norms rather than semantic similarity. Normalization resolves both issues at once: it breaks the trivial alignment with static parameters and prevents large-norm tokens from dominating the Gram matrix.

From Static Heuristics to Anisotropic Regularization. Table 6 compares fixed regularization scalars against learnable formulations. All fixed settings outperform the baseline. Moderate values around $\lambda = 10^{-2}$ perform well, while performance decreases when λ is too small (0.4364 at $\lambda = 10^{-4}$) or too large (0.4238 at $\lambda = 1$). Notably, a *Learnable Scalar* already achieves a competitive average accuracy of 0.4971, demonstrating that the outer-loop task objective can reliably discover effective regularization strengths end-to-end. Extending this to a coordinate-wise *Learnable Diagonal* ($\mathbf{D} = \mathrm{diag}(\mathrm{softplus}(\boldsymbol{\rho}))$) achieves the overall best performance. This progression shows the benefit of learning a separate regularization strength for each channel: the model penalizes less useful feature directions more strongly while preserving task-critical ones.

6 Concluding Remarks

In this work, we revisit sequence compression, a pervasive yet often overlooked bottleneck in modern architectures, through the lens of test-time regression. Instead of relying on fixed pooling or learned projections, we formulate token compression as an instance-adaptive ridge regression problem, allowing the compressed representation to be solved directly from each input sequence. By exploiting inter-token correlations through second-order interactions, TTR-Compressor offers a more expressive way to preserve sequence information while remaining lightweight and parameter-efficient. Its dual-form solution further makes this formulation scalable in practice. Empirically, TTR-Compressor performs strongly across both block-wise sparse attention and multimodal token reduction, improving long-context retrieval while matching standard visual mergers with significantly fewer tokens. More broadly, our results suggest that sequence compression need not be viewed as static aggregation, but as an input-dependent inference problem that adapts to the structure of each sequence. We expect this perspective establishes test-time regression as a simple, expressive, and scalable primitive for efficient foundation models.

AI USE STATEMENT

In this work, we used generative AI tools for assisting with translation. We have not used generative AI tools for other tasks with required disclosure, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for creating or modifying scientific figures and editing the research paper to improve readability. We have reviewed all AI-assisted work. For example, LLM-polished text and LLM-modified figures were verified and checked by authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, the experimental configurations and model setups are detailed in Section 5 and Section C. Comprehensive implementation details, including hyperparameters, training protocols, and dataset preprocessing, are documented in Appendix D.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025. 2, 7

Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. Nested learning: The illusion of deep learning architectures. *Advances in Neural Information Processing Systems*, 38:46968–47002, 2026a. 3

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *Advances in Neural Information Processing Systems*, 38:113506–113543, 2026b. 3

Luca Bertinetto, Joao F Henriques, Philip HS Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. *arXiv preprint arXiv:1805.08136*, 2018. 3

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*, 2022. 2

Junbum Cha, Wooyoung Kang, Jonghwan Mun, and Byungseok Roh. Honeybee: Locality-enhanced projector for multimodal llm. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13817–13827. IEEE, 2024. 2

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, et al. Are we on the right way for evaluating large vision-language models? *Advances in Neural Information Processing Systems*, 37:27056–27087, 2024. 8

Yunfei Chu, Jin Xu, Qian Yang, Haojie Wei, Xipin Wei, Zhifang Guo, Yichong Leng, Yuanjun Lv, Jinzheng He, Junyang Lin, et al. Qwen2-audio technical report. *arXiv preprint arXiv:2407.10759*, 2024. 2

DeepSeek-AI. Deepseek-v4.1-flash: Pushing the limits of kv cache compression, 2026. 1, 7

Guhao Feng, Shengjie Luo, Kai Hua, Ge Zhang, Wenhao Huang, Di He, and Tianle Cai. In-place test-time training. In *International Conference on Learning Representations*, volume 2026, pp. 114452–114472, 2026. 3

Yizhao Gao, Zhichen Zeng, Dayou Du, Shijie Cao, Peiyuan Zhou, Jiaxing Qi, Junjie Lai, Hayden Kwok-Hay So, Ting Cao, Fan Yang, et al. Seerattention: Learning intrinsic sparse attention in your llms. *arXiv preprint arXiv:2410.13276*, 2024. 2

Dongchen Han, Yining Li, Tianyu Li, Zixuan Cao, Ziming Wang, Jun Song, Yu Cheng, Bo Zheng, and Gao Huang. Vit3: Unlocking test-time training in vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 51–61, 2026. 3

Ali Hatamizadeh, Yejin Choi, and Jan Kautz. Gated deltanet-2: Decoupling erase and write in linear attention. *arXiv preprint arXiv:2605.22791*, 2026. 6

Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. 3

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024. 2, 6

Xiang Hu, Xinyu Wei, Hao Gu, Minshen Zhang, Tian Liang, Huayang Li, Lei Zhu, Yan Wang, Sirui Han, Yushi Bai, et al. Hierarchical sparse attention done right: Toward infinite context modeling. *arXiv preprint arXiv:2607.02980*, 2026. 2, 6, 18

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, et al. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *Advances in Neural Information Processing Systems*, 37:52481–52515, 2024. 2

M Jordan, J Kleinberg, and Bernhard Schoelkopf. Information science and statistics. *No Title*, 2006. 5

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *European conference on computer vision*, pp. 235–251. Springer, 2016. 8

Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. Flexprefill: A context-aware sparse attention mechanism for efficient long-sequence inference. *arXiv preprint arXiv:2502.20766*, 2025. 2

Feng Li, Renrui Zhang, Hao Zhang, Yuanhan Zhang, Bo Li, Wei Li, Zejun Ma, and Chunyuan Li. Llava-next: Tackling multi-image, video, and 3d in large multimodal models, June 2024. URL https://llava-vl.github.io/blog/2024-06-16-llava-next-interleave/. 8

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pp. 19730–19742. PmLR, 2023. 2

Bo Liu, Rui Wang, Lemeng Wu, Yihao Feng, Peter Stone, and Qiang Liu. Longhorn: State space models are amortized online learners. In *International Conference on Learning Representations*, volume 2025, pp. 95419–95434, 2025. 3

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning, 2023. 8

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024a. URL https://llava-vl.github.io/blog/2024-01-30-llava-next/. 8

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *European conference on computer vision*, pp. 216–233. Springer, 2024b. 8

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. Ocrbench: on the hidden mystery of ocr in large multimodal models. *Science China Information Sciences*, 67(12):220102, 2024c. 8

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, Yuxin Wu, Yutao Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. Moba: Mixture of block attention for long-context llms. *arXiv preprint arXiv:2502.13189*, 2025. 2

Ahmed Masry, Jia Qing Tan, Shafiq Joty, Enamul Hoque, et al. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the association for computational linguistics: ACL 2022*, pp. 2263–2279, 2022. 8

Amirkeivan Mohtashami and Martin Jaggi. Landmark attention: Random-access infinite context length for transformers. *arXiv preprint arXiv:2305.16300*, 2023. 2

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024. 6

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5. 7

Qwen Team. Qwen3.8-Flash-Next: A new architecture, towards ultimate cost-efficiency, August 2026b. URL https://qwen.ai/blog?id=qwen3.8-flash-next. 1

Craig Saunders, Alexander Gammerman, and Volodya Vovk. Ridge regression learning algorithm in dual variables. 1998. 5

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pp. 9355–9366. PMLR, 2021. 3

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992. doi: 10.1162/neco.1992.4.1.131. 3

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pp. 9229–9248. PMLR, 2020. 3

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024. 3

Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025. 8

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026a. 2, 7

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026b. 7

Andrey N. Tikhonov and Vasiliy Y. Arsenin. *Solutions of ill-posed problems*. V. H. Winston & Sons, Washington, D.C.: John Wiley & Sons, New York, 1977. Translated from the Russian, Preface by translation editor Fritz John, Scripta Series in Mathematics. 4

Rubèn Tito, Minesh Mathew, CV Jawahar, Ernest Valveny, and Dimosthenis Karatzas. Icdar 2021 competition on document visual question answering. In *International conference on document analysis and recognition*, pp. 635–649. Springer, 2021. 8

Johannes von Oswald, Nino Scherrer, Seijin Kobayashi, Luca Versari, Songlin Yang, Maximilian Schlegel, Kaitlin Maile, Yanick Schimpf, Oliver Sieberling, Alexander Meulemans, et al. Mesanet: Sequence modeling by locally optimal test-time training. In *International Conference on Learning Representations*, volume 2026, pp. 26397–26447, 2026. 3

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025. 2, 7

xAI. Grok-1.5 vision preview. https://x.ai/news/grok-1.5v, April 2024. 8

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a. 7

Sihan Yang, Runsen Xu, Chenhang Cui, Tai Wang, Dahua Lin, and Jiangmiao Pang. Vflowopt: A token pruning framework for lmms with visual information flow-guided optimization. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 23924–23934. IEEE, 2025b. 2

Sihan Yang, Runsen Xu, Yiman Xie, Sizhe Yang, Mo Li, Jingli Lin, Chenming Zhu, Xiaochen Chen, Haodong Duan, Xiangyu Yue, et al. Mmsi-bench: A benchmark for multi-image spatial intelligence. In *International Conference on Learning Representations*, volume 2026, pp. 157051–157088, 2026. 8

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *Advances in neural information processing systems*, 37:115491–115522, 2024. 3

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *International Conference on Learning Representations*, volume 2025, pp. 29687–29707, 2025c. 3

Linli Yao, Lei Li, Shuhuai Ren, Lean Wang, Yuanxin Liu, Xu Sun, and Lu Hou. Deco: Decoupling token compression from semantic abstraction in multimodal large language models. *arXiv preprint arXiv:2405.20985*, 2024. 2

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23078–23097, 2025. 2

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9556–9567, 2024. 8

Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William Freeman, and Hao Tan. Test-time training done right. In *International Conference on Learning Representations*, volume 2026, pp. 157604–157638, 2026. 3

Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. Llava-video: Video instruction tuning with synthetic data. *arXiv preprint arXiv:2410.02713*, 2024. 8

Appendix

Table of Contents

A Derivation of the Dual-form Solutions **15**

B Redundancy-Aware Weighting of TTR-Compressor **16**

C Implementation Details **18**

D Experimental Details **18**

E Inference Latency of Block-wise Sparse Attention **18**

A Derivation of the Dual-form Solutions

We first prove the equivalence between the primal and dual forms of ridge regression (Eq. 7). We then show that, with learnable diagonal regularization, rescaling the keys yields the dual solution in Eq. 8.

Equivalence of the Primal and Dual Forms.

Lemma 1 (Push-through identity). *For any $\boldsymbol{K} \in \mathbb{R}^{T\times d}$ and $\lambda > 0$,*

$$(\boldsymbol{K}^\top \boldsymbol{K} + \lambda \boldsymbol{I}_d)^{-1}\boldsymbol{K}^\top = \boldsymbol{K}^\top(\boldsymbol{K}\boldsymbol{K}^\top + \lambda \boldsymbol{I}_T)^{-1}. \tag{14}$$

Proof. Both $\boldsymbol{K}^\top\boldsymbol{K}$ and $\boldsymbol{K}\boldsymbol{K}^\top$ are positive semidefinite. Hence, for $\lambda > 0$, the matrices $\boldsymbol{K}^\top\boldsymbol{K} + \lambda\boldsymbol{I}_d$ and $\boldsymbol{K}\boldsymbol{K}^\top + \lambda\boldsymbol{I}_T$ are symmetric positive definite and therefore invertible. Expanding both sides directly gives

$$(\boldsymbol{K}^\top\boldsymbol{K} + \lambda\boldsymbol{I}_d)\,\boldsymbol{K}^\top = \boldsymbol{K}^\top\boldsymbol{K}\boldsymbol{K}^\top + \lambda\boldsymbol{K}^\top = \boldsymbol{K}^\top(\boldsymbol{K}\boldsymbol{K}^\top + \lambda\boldsymbol{I}_T). \tag{15}$$

Left-multiplying by $(\boldsymbol{K}^\top\boldsymbol{K}+\lambda\boldsymbol{I}_d)^{-1}$ and right-multiplying by $(\boldsymbol{K}\boldsymbol{K}^\top+\lambda\boldsymbol{I}_T)^{-1}$ yields Eq. 14. □

Right-multiplying Eq. 14 by $\boldsymbol{v}$ turns the primal solution in Eq. 3 into the dual form in Eq. 7:

$$\boldsymbol{h}^\star = (\boldsymbol{K}^\top\boldsymbol{K} + \lambda\boldsymbol{I}_d)^{-1}\boldsymbol{K}^\top\boldsymbol{v} = \boldsymbol{K}^\top(\boldsymbol{K}\boldsymbol{K}^\top + \lambda\boldsymbol{I}_T)^{-1}\boldsymbol{v}. \tag{16}$$

Therefore, $\boldsymbol{h}^\star$ lies in the row space of $\boldsymbol{K}$. Its dual coefficients $\boldsymbol{\alpha} = (\boldsymbol{K}\boldsymbol{K}^\top + \lambda\boldsymbol{I}_T)^{-1}\boldsymbol{v} \in \mathbb{R}^T$ are obtained by solving a single $T \times T$ system.

Dual Solution under Diagonal Regularization.

Proposition 1. *Let $\boldsymbol{D} = \mathrm{diag}(\delta_1, \ldots, \delta_d)$ with $\delta_j > 0$, and let $\widetilde{\boldsymbol{K}} = \boldsymbol{K}\boldsymbol{D}^{-1/2}$. Then the solution of Eq. 5 satisfies*

$$(\boldsymbol{K}^\top\boldsymbol{K} + \boldsymbol{D})^{-1}\boldsymbol{K}^\top\boldsymbol{v} = \boldsymbol{D}^{-1/2}\widetilde{\boldsymbol{K}}^\top\big(\boldsymbol{I}_T + \widetilde{\boldsymbol{K}}\widetilde{\boldsymbol{K}}^\top\big)^{-1}\boldsymbol{v}. \tag{17}$$

Proof. Since $\boldsymbol{D}$ is diagonal with positive entries, $\boldsymbol{D}^{\pm 1/2}$ are well defined and $\boldsymbol{D}^{1/2}\boldsymbol{D}^{-1/2} = \boldsymbol{I}_d$. Using $\boldsymbol{K} = \widetilde{\boldsymbol{K}}\boldsymbol{D}^{1/2}$, we factorize

$$\boldsymbol{K}^\top\boldsymbol{K} + \boldsymbol{D} = \boldsymbol{D}^{1/2}\big(\widetilde{\boldsymbol{K}}^\top\widetilde{\boldsymbol{K}} + \boldsymbol{I}_d\big)\boldsymbol{D}^{1/2}. \tag{18}$$

Inverting this product and using $\boldsymbol{K}^\top = \boldsymbol{D}^{1/2}\widetilde{\boldsymbol{K}}^\top$ gives

$$(\boldsymbol{K}^\top\boldsymbol{K}+\boldsymbol{D})^{-1}\boldsymbol{K}^\top\boldsymbol{v} = \boldsymbol{D}^{-1/2}\big(\widetilde{\boldsymbol{K}}^\top\widetilde{\boldsymbol{K}}+\boldsymbol{I}_d\big)^{-1}\boldsymbol{D}^{-1/2}\boldsymbol{D}^{1/2}\widetilde{\boldsymbol{K}}^\top\boldsymbol{v} = \boldsymbol{D}^{-1/2}\big(\widetilde{\boldsymbol{K}}^\top\widetilde{\boldsymbol{K}}+\boldsymbol{I}_d\big)^{-1}\widetilde{\boldsymbol{K}}^\top\boldsymbol{v}. \tag{19}$$

Applying Lemma 1 with $\boldsymbol{K} \leftarrow \widetilde{\boldsymbol{K}}$ and $\lambda = 1$ gives Eq. 17. □

Equivalently, the change of variables $\boldsymbol{g} = \boldsymbol{D}^{1/2}\boldsymbol{h}$ turns Eq. 5 into a standard ridge problem, $\frac{1}{2}\|\widetilde{\boldsymbol{K}}\boldsymbol{g} - \boldsymbol{v}\|_2^2 + \frac{1}{2}\|\boldsymbol{g}\|_2^2$. Its dual solution is then mapped back through $\boldsymbol{h}^\star = \boldsymbol{D}^{-1/2}\boldsymbol{g}^\star$.

Two further properties follow:

- The matrix $\boldsymbol{I}_T + \widetilde{\boldsymbol{K}}\widetilde{\boldsymbol{K}}^\top$ is symmetric positive definite, so its Cholesky factorization exists and is unique.
- Because $\boldsymbol{D}$ is diagonal, forming $\widetilde{\boldsymbol{K}}$ and applying $\boldsymbol{D}^{-1/2}$ each cost only $\mathcal{O}(Td)$ and $\mathcal{O}(d)$. The overall complexity therefore remains $\mathcal{O}(T^2d + T^3)$.

Setting $\boldsymbol{D} = \lambda\boldsymbol{I}_d$ recovers Eq. 7.

B REDUNDANCY-AWARE WEIGHTING OF TTR-COMPRESSOR

We formalize the claim in Section 4.2 that redundant tokens have diminishing influence, while information not explained by correlated tokens keeps a larger relative influence. We take isotropic regularization $\lambda > 0$ and treat the targets $\boldsymbol{v}$ as arbitrary, so the results hold for any learned projection $\boldsymbol{a}$. The extension to the learnable $\boldsymbol{D}$ is given at the end. Throughout, let $\boldsymbol{A} = \boldsymbol{K}^\top \boldsymbol{K} + \lambda \boldsymbol{I}_d$, and let $\boldsymbol{\alpha} = (\boldsymbol{G} + \lambda \boldsymbol{I}_T)^{-1} \boldsymbol{v}$, so that $\boldsymbol{h}^\star = \boldsymbol{K}^\top \boldsymbol{\alpha}$.

Effective Weights as Unexplained Residuals.

Proposition 2 (Residual form). *For any keys $\boldsymbol{K}$ and targets $\boldsymbol{v}$, the effective weights satisfy*

$$\boldsymbol{\alpha} = \frac{1}{\lambda}\big(\boldsymbol{v} - \boldsymbol{K}\boldsymbol{h}^\star\big). \tag{20}$$

Proof. From $(\boldsymbol{G} + \lambda \boldsymbol{I}_T)\boldsymbol{\alpha} = \boldsymbol{v}$ we have $\lambda \boldsymbol{\alpha} = \boldsymbol{v} - \boldsymbol{K}\boldsymbol{K}^\top \boldsymbol{\alpha} = \boldsymbol{v} - \boldsymbol{K}\boldsymbol{h}^\star$. □

Thus, a token is weighted by the part of its target that the jointly fitted representation leaves unexplained. By contrast, additive pooling assigns weights $\boldsymbol{v}$ regardless of how the other tokens are arranged. To separate the role of the *other* tokens, let $\boldsymbol{K}_{-t}$ and $\boldsymbol{v}_{-t}$ denote the keys and targets with token t removed. Define

$$\boldsymbol{A}_{-t} = \boldsymbol{K}_{-t}^\top \boldsymbol{K}_{-t} + \lambda \boldsymbol{I}_d, \qquad \boldsymbol{h}_{-t} = \boldsymbol{A}_{-t}^{-1} \boldsymbol{K}_{-t}^\top \boldsymbol{v}_{-t}, \tag{21}$$

where $\boldsymbol{h}_{-t}$ is the solution fitted without token t.

Proposition 3 (Leave-one-out form). *Let $r_t = v_t - \boldsymbol{k}_t^\top \boldsymbol{h}_{-t}$ be the residual of token t under the other tokens, and let $q_t = \boldsymbol{k}_t^\top \boldsymbol{A}_{-t}^{-1} \boldsymbol{k}_t \geq 0$. Then*

$$\alpha_t = \frac{r_t}{\lambda(1 + q_t)}, \qquad \boldsymbol{h}^\star = \boldsymbol{h}_{-t} + \frac{r_t}{1 + q_t} \boldsymbol{A}_{-t}^{-1} \boldsymbol{k}_t. \tag{22}$$

Proof. Write $\boldsymbol{A} = \boldsymbol{A}_{-t} + \boldsymbol{k}_t \boldsymbol{k}_t^\top$ and $\boldsymbol{K}^\top \boldsymbol{v} = \boldsymbol{K}_{-t}^\top \boldsymbol{v}_{-t} + v_t \boldsymbol{k}_t$. Substituting into $\boldsymbol{A}\boldsymbol{h}^\star = \boldsymbol{K}^\top \boldsymbol{v}$ gives

$$\boldsymbol{h}^\star = \boldsymbol{h}_{-t} + \big(v_t - \boldsymbol{k}_t^\top \boldsymbol{h}^\star\big) \boldsymbol{A}_{-t}^{-1} \boldsymbol{k}_t. \tag{23}$$

Taking the inner product with $\boldsymbol{k}_t$ gives $\boldsymbol{k}_t^\top \boldsymbol{h}^\star = \boldsymbol{k}_t^\top \boldsymbol{h}_{-t} + (v_t - \boldsymbol{k}_t^\top \boldsymbol{h}^\star)\, q_t$, and hence $v_t - \boldsymbol{k}_t^\top \boldsymbol{h}^\star = r_t/(1 + q_t)$. Substituting this into the display above and into Eq. 20 yields Eq. 22. □

Proposition 3 shows that token t changes the representation only through r_t, the part of its target that the remaining tokens cannot predict. In particular, if the other tokens fully explain token t ($r_t = 0$), then $\alpha_t = 0$ and $\boldsymbol{h}^\star = \boldsymbol{h}_{-t}$, so the token has no influence. Additive pooling instead always adds $v_t \boldsymbol{k}_t$.

Saturation under Repeated Evidence. We now quantify this effect for groups of correlated tokens, generalizing the example in Figure 2(b).

Proposition 4 (Group saturation). *Suppose the tokens form C groups. Group c contains m_c tokens that share the key $\boldsymbol{u}_c$, with $\|\boldsymbol{u}_c\|_2 = 1$ and $\boldsymbol{u}_c^\top \boldsymbol{u}_{c'} = 0$ for $c \neq c'$. Let S_c be the sum of the targets in group c. Then*

$$\boldsymbol{h}^\star = \sum_{c=1}^{C} \frac{S_c}{\lambda + m_c} \boldsymbol{u}_c, \qquad \textit{whereas} \qquad \boldsymbol{K}^\top \boldsymbol{v} = \sum_{c=1}^{C} S_c \boldsymbol{u}_c. \tag{24}$$

Proof. Under this assumption $\boldsymbol{G}$ is block diagonal, and the block for group c is $\boldsymbol{J}_{m_c} = \boldsymbol{1}\boldsymbol{1}^\top$. Since $\boldsymbol{J}_m^2 = m\boldsymbol{J}_m$, one can verify that $(\boldsymbol{J}_m + \lambda \boldsymbol{I}_m)^{-1} = \lambda^{-1}\big(\boldsymbol{I}_m - \boldsymbol{J}_m/(\lambda + m)\big)$. Hence the weights $\boldsymbol{\alpha}_c$ of group c sum to

$$\boldsymbol{1}^\top \boldsymbol{\alpha}_c = \frac{1}{\lambda}\Big(S_c - \frac{m_c S_c}{\lambda + m_c}\Big) = \frac{S_c}{\lambda + m_c}. \tag{25}$$

Therefore $\boldsymbol{h}^\star = \sum_c \boldsymbol{u}_c\, \boldsymbol{1}^\top \boldsymbol{\alpha}_c$, which gives Eq. 24. □

Corollary 1 (Diminishing influence of redundancy)**.** *Assume every token has the same target $v > 0$, so that $S_c = m_c v$. Compare a group of m repeated tokens with a single distinct token. Their relative influence is*

$$\frac{m/(\lambda+m)}{1/(\lambda+1)} = \frac{m(\lambda+1)}{\lambda+m} < m \quad \textit{for } m > 1. \tag{26}$$

This ratio is bounded by $\lambda + 1$ for every m, whereas the corresponding ratio for additive pooling is m. Moreover, the n-th repeated token increases the group's coefficient by

$$\frac{nv}{\lambda+n} - \frac{(n-1)v}{\lambda+n-1} = \frac{\lambda v}{(\lambda+n)(\lambda+n-1)} = \mathcal{O}(n^{-2}), \tag{27}$$

which decays with n, so repeated evidence has diminishing returns.

The ridge strength controls the transition between two regimes:

- As $\lambda \to \infty$, $\lambda \boldsymbol{h}^\star \to \boldsymbol{K}^\top \boldsymbol{v}$, which is additive pooling.
- As $\lambda \to 0$, each group contributes S_c/m_c, its mean target, regardless of how many tokens it contains.

Setting $\lambda = 1$, $m = 3$ recovers Figure 2(b): the ratio is 3:2 and $\boldsymbol{\alpha} = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{2})^\top$.

Extension to Learnable Diagonal Regularization. All results carry over to learnable diagonal regularization. Using the change of variables in Proposition 1, set $\boldsymbol{\alpha} = (\boldsymbol{I}_T + \widetilde{\boldsymbol{K}}\widetilde{\boldsymbol{K}}^\top)^{-1}\boldsymbol{v}$, so that $\boldsymbol{h}^\star = \boldsymbol{D}^{-1}\boldsymbol{K}^\top\boldsymbol{\alpha}$. Since $\boldsymbol{K}\boldsymbol{h}^\star = \widetilde{\boldsymbol{K}}\widetilde{\boldsymbol{K}}^\top\boldsymbol{\alpha}$, the argument of Proposition 2 gives $\boldsymbol{\alpha} = \boldsymbol{v} - \boldsymbol{K}\boldsymbol{h}^\star$. Proposition 3 then holds with $\lambda = 1$ and $\boldsymbol{A}_{-t} = \boldsymbol{K}_{-t}^\top\boldsymbol{K}_{-t} + \boldsymbol{D}$. Proposition 4 holds when the groups are orthogonal under the learned metric ($\boldsymbol{u}_c^\top \boldsymbol{D}^{-1}\boldsymbol{u}_{c'} = 0$ for $c \neq c'$), with λ replaced by the group-specific $\lambda_c = (\boldsymbol{u}_c^\top \boldsymbol{D}^{-1}\boldsymbol{u}_c)^{-1}$ and $\boldsymbol{u}_c$ replaced by $\lambda_c \boldsymbol{D}^{-1}\boldsymbol{u}_c$. The learned $\boldsymbol{D}$ therefore decides which feature directions count as redundant, while the diminishing-influence mechanism itself is unchanged.

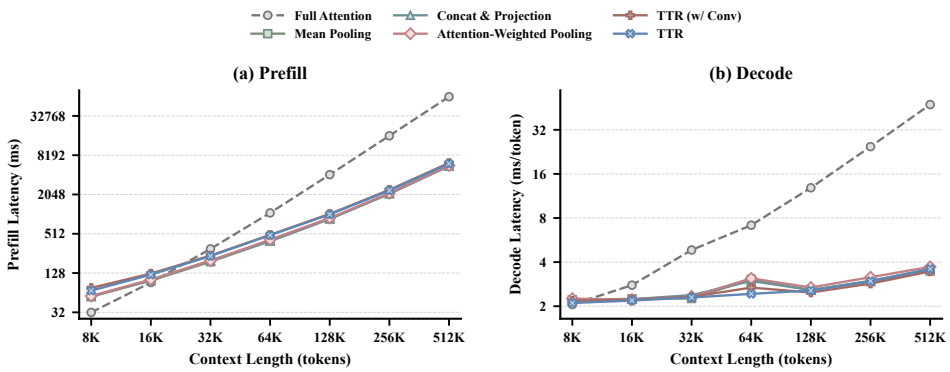


Figure 4: Prefill and decode latency across context lengths.

C Implementation Details

Numerical Precision. Convolution and target projection use the input precision (BF16), while regression keys and targets are cast to FP32 before the dual solve. Diagonal regularization is evaluated in FP32. We solve the positive-definite dual system using Cholesky factorization without explicitly forming an inverse, and cast the compressed representation back to the input dtype.

Local Target Construction. Only the regression targets incorporate convolutional context; keys are computed from the original inputs. For language, we apply a non-causal depthwise 1D convolution with kernel size 5, independently within each compression block, using two-sided zero padding of width 2. The convolution has neither bias nor activation and is initialized to the identity. For vision, we use a 3×3 depthwise convolution. Each channel has its own spatial kernel, with no cross-channel mixing before the scalar target projection.

The base compressor has $2d$ effective trainable parameters: d for the target projection and d for diagonal regularization. A bias-free depthwise convolution with s kernel elements adds sd parameters and $\mathcal{O}(sTd)$ computation. Thus, the language and vision variants require $7d$ and $11d$ parameters, respectively. Their total computational complexity remains $\mathcal{O}(T^2 d)$ for fixed kernel sizes and $T \ll d$.

D Experimental Details

Block-wise Sparse Attention. HiLS-Attention (Hu et al., 2026) argues that small models provide a more diagnostic setting for RULER's predominantly "find-and-copy" tasks, as larger models may compensate for retrieval inaccuracies through greater information bandwidth. We therefore follow their setup and use 345M-parameter models to evaluate compression quality. All models use 16 Transformer layers, a hidden dimension of 1,024, an FFN dimension of 4,096, and grouped-query attention with 16 query heads and 2 key–value heads. We use rank-64 query calibration. Training uses AdamW weight decay of 0.01 and gradient clipping at a norm of 1.0; normalization parameters, biases, and embeddings are excluded from weight decay. The learning rate warms up over the first 0.7% of training and remains constant thereafter.

Visual Token Merger. We use BF16 training with DeepSpeed ZeRO-1, gradient checkpointing, and a maximum sequence length of 16,384 tokens. The two stages use per-device batch sizes of 4 and 1, with gradient accumulation over 4 and 8 steps, respectively. Both stages apply a 3% learning-rate warmup followed by cosine decay, with minimum learning rates of 10^{-5} and 0 and weight decay of 0.01 and 0.02, respectively. For TTR-Compressor, the regression systems are solved in FP32.

E Inference Latency of Block-wise Sparse Attention

As shown in Figure 4, all compression methods exhibit similar prefill and decode latency and substantially outperform full attention for context lengths beyond 16K tokens.