Multi-Spot Transformer: An Efficient Transformer with Linear Complexity

Anonymous ACL submission

Abstract

Transformers suffer from quadratic complexity due to dense token interactions. Existing sparse and compression methods exploit token sparsity only in limited ways, leading to suboptimal performance. We analyze token interactions in standard attention and unify existing sparsity-based approaches. First, we show that **Softmax acts as a relevance filter**, amplifying correlated tokens and suppressing irrelevant ones. Based on this, we propose **Multi-Scale Compression (MSC)**, which adaptively groups token relationships by relevance, applying different compression levels to preserve key information while reducing noise. MSC achieves $O(N \log N)$ complexity, and with kernel functions that emulate Softmax's filtering, it yields a linear $O(N)$ model (named **MST**). Our method achieves performance comparable to state-of-the-art efficient attention mechanisms in natural language processing and computer vision tasks, and provides a superior approximation of full attention mechanisms.

1 Introduction

The Transformer model (Vaswani et al., 2017; Devlin et al., 2018; Brown et al., 2020; Dosovitskiy et al., 2021) excels at capturing long-range dependencies via self-attention and non-linear layers. However, its quadratic complexity in sequence length hinders scalability. Many approaches have been proposed to reduce this complexity, including exploiting sparsity (Child et al., 2019; Radosavovic et al., 2020; Kitaev et al., 2020; Han et al., 2021) or reordering operations after replacing Softmax (Shen et al., 2021; Yun et al., 2021; Yuan and Wang, 2021). Yet these methods often overlook the role of Softmax and the intrinsic correlation between tokens, limiting their performance and efficiency. In this paper, we first analyze Softmax as a correlation filter, then develop a multi-scale compression strategy based on token correlations. By fully exploiting the sparsity of token relationships, our method achieves superior performance and efficiency.

Replacing Softmax for linear complexity. Several works replace Softmax with kernel functions to achieve linear complexity, rewriting attention as Sim(Q,K)$V = f_1(Q)f_2(K)^\top V$ and reordering operations to shift complexity from $O(N^2d)$ to $O(Nd^2)$ (e.g., Linear Attention (Katharopoulos et al., 2020), Performer (Choromanski et al., 2022), Random Feature Attention (Peng et al., 2021), Flatten Transformer (Han et al., 2023)). Subsequent enhancements introduce gating or Delta rules (e.g., GLA (Yang et al., 2024a), DeltaNet (Yang et al., 2024b)). However, lacking a deep understanding of Softmax's role often leads to suboptimal performance. This paper provides a thorough analysis of Softmax's filtering effect and proposes a flexible improvement that maintains performance across scenarios.

Exploiting sparsity in Attention. Existing sparsity-based methods fall into three categories: (1) **Static region-based** methods partition sequences into fixed regions (e.g., Longformer (Beltagy et al., 2020), Swin Transformer (Liu et al., 2021a)) or combine multiple fixed strategies (e.g., BigBird (Zaheer et al., 2020b)). While efficient, they lack flexibility and risk omitting critical information. (2) **Dynamic routing-based** methods adaptively select relevant KV regions for each Q based on similarity (e.g., Routing Transformer (Roy et al., 2021), Biformer (Zhu et al., 2023a), MoBA (Lu et al., 2025), NSA (Yuan et al., 2025)). However, they still face challenges such as insufficient selected tokens, suboptimal selection, or computational overhead. (3) **Compression-based** methods compress KV sequences via quantization or hidden states (e.g., Transformer-VQ (Lingle, 2024), Linformer (Wang et al., 2020), ABC (Peng et al., 2022), GSA (Zhang et al., 2024)), but suffer from information loss.

We argue these limitations stem from an insufficient understanding of the correlation structure in Attention. We provide a comprehensive analysis of how Softmax shapes this correlation, guiding the design of more efficient Attention mechanisms. Based on these insights, we propose **Relevance-based Multi-Scale Compressed Attention** together with a **kernel function that emulates Softmax's filtering effect**. We decompose Attention into three components: (1) Softmax as a relevance filter, (2) the dot product between Q and K as a fundamental similarity measure, and (3) their joint effect on sparsification. Our multi-scale compression strategy effectively exploits this sparsity. For completeness, we also analyze convex interpolation for V (see Appendix H and I). Our theoretical framework yields algorithms with $O(N^{4/3})$, $O(N \log N)$, and $O(N)$ complexity, adaptable to various applications.

2 Background and Preliminaries

2.1 Linear Attention

Methods in this category abstract the softmax function in attention, `Softmax`(Q, K), into a similarity function `Sim`(Q, K) and approximate it using kernel functions. The output of such attention mechanisms is given by:

$$\text{output} = \text{Sim}(Q, K)V = \phi_1(Q)\phi_2(K)^\top V. \quad (1)$$

Due to the symmetry property of similarity functions, it is common to set $\phi_1 = \phi_2 = \phi$. Various linear attention mechanisms employ different choices for $\phi(x)$, such as:

(a) Linear Attention: $\phi(x) = \text{elu}(x) + 1$. (b) Performer: $\phi(x) = \exp(\omega^\top x - \frac{\|x\|^2}{2})$ with an additional averaging operation. (c) Flatten Transformer: $\phi(x) = \frac{\|x\|}{\|x\|^p} \cdot x^p$.

Recent linear attention methods have introduced various enhancements, such as gating mechanisms and delta rule methods. A representative example is the GLA mechanism:

$$\text{output} = (\phi_1(Q) \cdot B) \left(\frac{\phi_2(K)}{B}\right)^\top V, \quad (2)$$

where $B = \text{cmuprod}(\sigma(X))$. The t-th row of B is given by:

$$b_t = \prod_{i=1}^{t} \sigma(x_i), \quad (3)$$

where $x_i \in \mathbb{R}^{1\times d}$ represents the i-th row vector of the input tensor X.

Although existing linear attention methods achieve promising performance, we argue that they lack a deeper understanding of the role of the softmax function, which limits their ability to achieve optimal effectiveness.

2.2 Compressed Attention

This class of methods compresses keys and values into fixed-size representations $\bar{K}, \bar{V}$ to reduce complexity:

$$\bar{K} = C_1(K), \bar{V} = C_2(V), \text{output} = \text{Softmax}(Q\bar{K}^\top)\bar{V}.$$

For example, Linformer uses learned projection matrices $E, F \in \mathbb{R}^{k\times n}$: $C(K) = EK,\ C(V) = FV$. However, the projection matrices scale with sequence length n, limiting scalability. ABC replaces projections with softmax-weighted transformations:

$$C(K) = \text{Softmax}(W_{kc}X^\top)K,$$

where $X \in \mathbb{R}^{n\times d}$ and learnable $W_{kc} \in \mathbb{R}^{k\times d}$ acts as a pseudo-query. GSA introduces a gating mechanism to bias toward nearby tokens:

$$K_t = \text{diag}(\sigma(x_t))K_{t-1} + (1 - \sigma(x_t))^\top k_t, \quad (4)$$

with hidden state $K_t \in \mathbb{R}^{k\times d}$ and current input/key $x_t, k_t \in \mathbb{R}^{1\times d}$. While compressed attention reduces complexity, the loss of fine-grained information can hurt performance on precise tasks.

2.3 Fixed-Pattern Sparse Attention

This category of methods restricts queries (Q) to fixed or sequence-independent key (K) positions, which significantly reduces the computational cost. The general formulation is:

$$\tilde{K} = \text{Loc}(K, \text{idx} = i_k), \quad \tilde{V} = \text{Loc}(V, \text{idx} = i_k),$$

$$\text{output} = \text{Softmax}(Q\tilde{K})\tilde{V}, \quad (5)$$

where $\text{Loc}(\cdot)$ represents a selection function, and i_k denotes the selected indices. In fixed-pattern approaches, i_k is independent of Q and K:

(a) Sliding Window: $i_k \in (i - \frac{s}{2}, i + \frac{s}{2})$. (b) Global Tokens: $i_k \in (0, s)$, commonly used in attention sink mechanisms. (c) Swin Transformer: Alternates between windowed attention $i_k \in (i - \frac{s}{2}, i + \frac{s}{2})$ and shifted window attention $i_k \in ((i - \frac{s}{2} + l) \mod n, (i + \frac{s}{2} + l) \mod n)$. (d) BigBird: Incorporates sliding window, global tokens, and randomly selected keys and values: $i_k \in (r_1, r_2, ..., r_m)$.

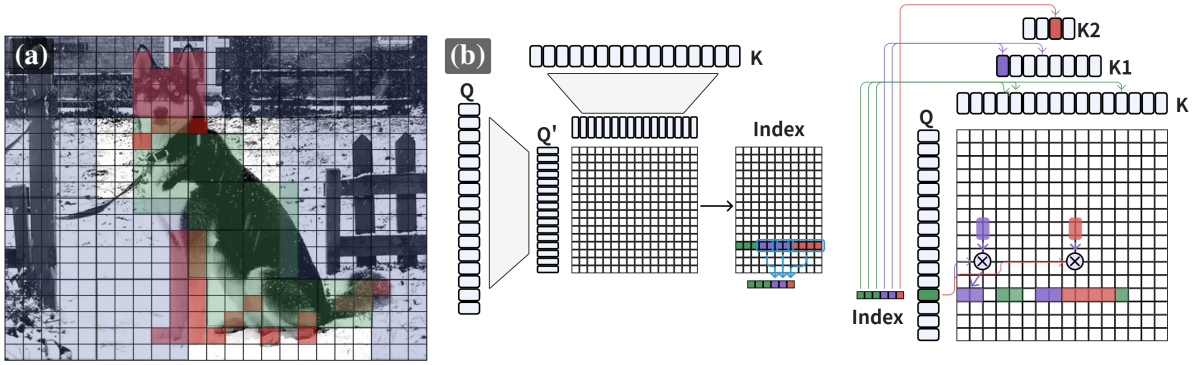

Figure 1: Multi-scale compression (MSC) scheme. (a) Multi-scale compressed token selection mechanism: red regions denote highly relevant uncompressed tokens, green regions correspond to moderately compressed correlated tokens, and blue regions represent aggressively compressed low-correlation context. (b) Multi-scale compression pipeline. Left: compressed chunk routing followed by top-k selection. Right: KV retrieval and multi-scale attention computation.

While these approaches offer computational efficiency, their fixed selection strategies may lead to information loss in certain scenarios, resulting in suboptimal performance.

2.4 Dynamically Routed Sparse Attention

Unlike fixed-pattern approaches, these methods dynamically determine key (K) positions based on the importance of different regions, thereby achieving improved performance. The general formulation is:

$$\tilde{K} = \mathrm{Loc}(K, i_k(Q, K)), \tilde{V} = \mathrm{Loc}(V, i_k(Q, K)),$$

$$\text{output} = \mathrm{Softmax}(Q\tilde{K})\tilde{V}, \tag{6}$$

where the selected indices $i_k(Q, K)$ depend on Q and K. For instance, Biformer selects keys dynamically using:

$$i_k(Q, K) = \arg\max_{\text{top-}K}(\bar{Q}\bar{K}^\top), \tag{7}$$

where $\bar{Q} \in \mathbb{R}^{n/s \times d}$ and $\bar{K} \in \mathbb{R}^{n/s \times d}$ are compressed segment representations of Q and K, respectively. By leveraging dynamic selection mechanisms, these approaches achieve more flexible and adaptive attention computation, leading to improved effectiveness over fixed-pattern sparse attention methods.

3 Method

We analyse the aforementioned Attention formula in three parts from inside to outside: (1) using the dot product between q tokens and k tokens as a distance metric to represent their similarity, serving as the initial regression weights; (2) the outer Softmax as a filtering operation that scales the similarity indices, diminishing the weights of unrelated tokens near to zero, resulting in a sparser relationship that relies only on the most similar tokens for regression; (3) the outermost layer involves a convex combination with v.

3.1 Softmax as a Correlation Filter

We formalize the role of Softmax in attention mechanisms through the following proposition and its proof. The analysis reveals that Softmax does more than simply normalize; it actively filters token correlations, selectively amplifying relevant ones while suppressing irrelevant or negative interactions.

Proposition 3.1 (Softmax as a Correlation Filtering Operator)**.** *Let* $\mathbf{d}_i = \{d_{ij}\}_{j \le i}$ *denote the dot-product similarities between query token* i *and all previous tokens, and let the corresponding Softmax attention distribution be*

$$p_{ij} = \frac{e^{d_{ij}}}{\sum_{k \le i} e^{d_{ik}}}.$$

Then, the Softmax operator possesses the following correlation filtering properties:

1. ***Exponential suppression of negatively correlated tokens.*** *For any* $d_{ij}^- < 0$ *and* $d_{ik}^+ > 0$,

$$\frac{p_{ij}^-}{p_{ik}^+} = e^{d_{ij}^- - d_{ik}^+},$$

which decays exponentially as the correlation gap increases.

2. ***Exponential amplification of positively correlated tokens.*** *For two positive dot products satisfying* $d_2 \ge d_1 > 0$*, define the original similarity ratio and the corresponding*

attention-score ratio by applying the exponential kernel as

$$r_d = \frac{d_2}{d_1} = 1 + \frac{\Delta d}{d_1}, r_s = \frac{e^{d_2}}{e^{d_1}} = e^{\Delta d},$$

where $\Delta d = d_2 - d_1 \geq 0$. *Consequently, the Softmax scaling ratio*

$$s = \frac{r_s}{r_d} = \frac{e^{\Delta d}}{1 + \frac{\Delta d}{d_1}}$$

grows super-linearly with respect to Δd, *implying that Softmax amplifies relatively more relevant tokens.*

3. ***Limited contribution of low-magnitude similarities.*** *When* $d_1 < 1$, *even though the relative scaling factor may increase for small* Δd, *the normalized attention contribution remains bounded due to the small absolute magnitude of* e^{d_1}.

Therefore, Softmax can be interpreted as a nonlinear correlation filtering mechanism that selectively amplifies highly relevant correlations while suppressing weak or negatively correlated interactions.

The above proposition is discussed in many articles and is quite straightforward; for the sake of completeness, we have provided a proof in the appendix C. Proposition 3.1 establishes that Softmax acts as a nonlinear filter that simultaneously suppresses negative correlations and amplifies positive ones, with the amplification being exponential in the difference of dot products. This insight motivates our later design of alternative kernel functions that can mimic this filtering effect while enabling linear complexity.

The theoretical analysis of softmax-like filtering is as follows and the proof is provided in the appendix D.

Proposition 3.2 (Softmax-like Filtering via Separable Power Functions)**.** *Let* $\psi : \mathbb{R} \to \mathbb{R}^+$ *be a positive activation function, and let* $\phi : \mathbb{R}^+ \to \mathbb{R}^+$ *satisfy* $\phi'(x) \geq cx^{1/n}$ *for all* $x > 0$, *with constants* $c > 0$ *and* $n > 0$. *Define the kernelized dot product as* $\kappa(q, k) = \phi(\psi(q))\,\phi(\psi(k))$. *Then the operation* $\kappa(q, k)$ *can replace the standard dot product* qk *in an attention mechanism while preserving the essential correlation-filtering characteristics of Softmax. Specifically, for two pairs* (q_1, k_1) *and* (q_2, k_2) *with* $q_1k_1 > q_2k_2 > 0$, *the ratio* $\frac{\kappa(q_1,k_1)}{\kappa(q_2,k_2)}$ *grows super-linearly compared to* $\frac{q_1k_1}{q_2k_2}$, *analogous to the exponential amplification of positive correlations observed in Softmax.*

Moreover, a sufficient condition for this filtering behavior is that for $x \geq 1$, $\phi(x) \geq cx^{1+1/n}$, *and for* $0 < x < 1$, $\phi(x) \geq cx^{1-1/n}$, *with the same* c, n.

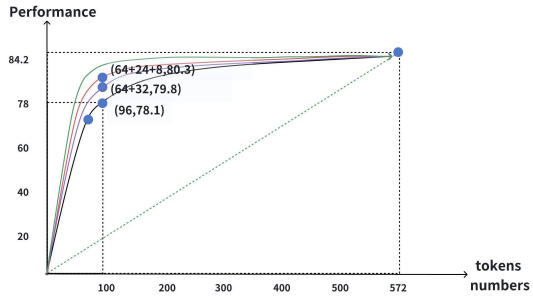

Figure 2: Performance of ViT with correlation-aware multi-scale attention (no fine-tuning). Our method matches or exceeds full attention across tasks.

3.2 Multi-Scale Compression

The sparsity introduced by the Softmax function in correlation filtering suggests that not all pairwise interactions are equally important, motivating the use of efficient attention mechanisms that preserve semantic fidelity.

Existing sparse attention falls into three categories: (1) **Fixed-pattern** (sliding window, global tokens) – restricts attention to predefined regions but cannot adapt to irregular structures. (2) **Routing-based** (top-k selection) – dynamically selects relevant tokens but often discards secondary context. (3) **Compression-based** – aggregates tokens into coarse representations, losing fine-grained detail.

Consider a visual reasoning task (Fig. 1(a)) requiring understanding of a husky's posture and environment. Fixed-pattern methods miss elongated leg structures spanning non-contiguous regions; routing-based methods may omit contextual cues; compression loses fine details. The key insight: different components need different granularities – fine for discriminative features, medium for object parts, coarse for background. We propose a *relevance-aware multi-scale compression mechanism* that unifies these paradigms. It adaptively assigns tokens to multiple scales based on correlation strength with the query: highly relevant tokens retain fine granularity while less relevant ones are progressively compressed. Below we formalize the mechanism and its theoretical properties.

3.2.1 Formulation and Unified Framework

Definition 3.1 (Multi-Scale Compression Operator)**.** *Let $\{k_j\}_{j=1}^n$ and $\{v_j\}_{j=1}^n$ denote the key and value sequences, respectively. For a scale parameter $p \in \mathbb{N}$, the compression operator $\mathcal{C}_p$ aggregates consecutive tokens into representative embeddings:*

$$\bar{k}_{p,s} = \mathcal{C}_p(\{k_j\}_{j=s\cdot p}^{(s+1)\cdot p-1}) = \frac{1}{p}\sum_{j=s\cdot p}^{(s+1)\cdot p-1} k_j, \quad (8)$$

$$\bar{v}_{p,s} = \mathcal{C}_p(\{v_j\}_{j=s\cdot p}^{(s+1)\cdot p-1}) = \frac{1}{p}\sum_{j=s\cdot p}^{(s+1)\cdot p-1} v_j, \quad (9)$$

where $s \in \{0, 1, \ldots, \lfloor n/p \rfloor - 1\}$ indexes the compressed blocks.

Proposition 3.3 (Multi-Scale Relevance-Aware Attention)**.** *Let $\{p_m\}_{m=1}^M$ denote a hierarchy of compression scales with $p_1 < p_2 < \cdots < p_M$. Assume tokens are sorted in descending order of relevance to query q_i, i.e., $q_i k_j^\top \geq q_i k_{j+1}^\top$. The multi-scale attention output is given by:*

$$\mathcal{A}_{MS}(q_i) = \sum_{m=1}^{M}\sum_{s=0}^{K_m-1} \alpha_{m,s}\cdot \bar{v}_{p_m,s}, \quad (10)$$

where the attention weights are computed as:

$$\begin{aligned} \alpha_{m,s} &= \frac{p_m \exp(q_i \bar{k}_{p_m,s}^\top)}{Z_{MS}}, \\ Z_{MS} &= \sum_{m'=1}^{M}\sum_{s'=0}^{K_{m'}-1} p_{m'} \exp(q_i \bar{k}_{p_{m'},s'}^\top), \end{aligned} \quad (11)$$

and the block indices are offset by $t_m = \sum_{n=1}^{m-1} p_n K_n$ to ensure non-overlapping partitions across scales. In the actual implementation, we optimize p_n as a learnable parameter.

3.2.2 Gated Multi-Scale Formulation

To enable dynamic adaptation and theoretical unification, we introduce a learnable scale-gating vector $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_M]^\top \in [0,1]^M$, where each γ_m controls the activation strength of the m-th compression branch. The gated multi-scale attention operator is defined as:

$$\mathcal{A}_{\text{MS}}(q_i) = \sum_{m=1}^{M}\sum_{s=0}^{K_m-1} \alpha_{m,s}\cdot \bar{v}_{p_m,s}, \quad (12)$$

where the normalized attention weights are computed as:

$$\alpha_{m,s} = \frac{\gamma_m \cdot p_m \cdot \exp(q_i \bar{k}_{p_m,s}^\top)}{\sum_{m'=1}^{M}\gamma_{m'}\sum_{s'=0}^{K_{m'}-1} p_{m'}\exp(q_i \bar{k}_{p_{m'},s'}^\top)}.$$

Here, p_m serves as a volume-conservation scaling factor, and the gating coefficients γ_m can be parameterized as context-aware functions $\gamma_m(q_i)$ or fixed structural priors. This formulation naturally supports soft interpolation across scales and hard routing to specific branches.

Proposition 3.4 (Unified Sparse & Compressive Attention Framework)**.** *Equipped with the scale-gating mechanism, the multi-scale framework subsumes existing sparse and compressive attention paradigms as special cases or asymptotic limits:*

(i) ***Sliding Window Attention:*** *Under spatial locality priors, the optimal routing policy converges to a contiguous window selection.*

(ii) ***Global Token Attention:*** *With structural positional biases, the selection mechanism asymptotically retains a fixed set of global tokens while compressing local context.*

(iii) ***Top-k Routing:*** *Via constructive hard-gating on the finest scale ($p_1 = 1$), the framework exactly recovers dynamic token selection.*

(iv) ***Block-Sparse Attention:*** *Fixed block activation patterns recover structured sparse attention with aggregated representations.*

(v) ***Compression-Based Attention:*** *By deactivating routing and activating a single coarse scale ($\gamma_c = 1, \gamma_{m\neq c} = 0$) with $p_c \gg 1$, the framework reduces to uniform token compression/pooling mechanisms.*

We have included the above proof in Appendix E. We now establish rigorous error bounds for the multi-scale approximation. For notational simplicity, we absorb the scaling factor $1/\sqrt{d}$ into the dot products $d_{ij} = q_i k_j^\top$.

Proposition 3.5 (First-Order Approximation Error)**.** *Let $\mathcal{A}_{full}^{(l)}$ and $\mathcal{A}_{MS}^{(l)}$ denote the full and multi-scale attention outputs restricted to the l-th scale partition, respectively. The approximation error satisfies:*

$$\begin{aligned} \varepsilon_l &= \|\mathcal{A}_{full}^{(l)} - \mathcal{A}_{MS}^{(l)}\|_2 \\ &\leq \frac{C_l}{p_l}\left(\sum_{j\in\mathcal{B}_l} |d_{ij} - \bar{d}_{il}|\right)\left(\sum_{j\in\mathcal{B}_l} \|v_j - \bar{v}_l\|_2\right), \end{aligned}$$

where $\mathcal{B}_l$ is the index set for scale p_l, $\bar{v}_l$ is the compressed value vector, and C_l is a constant depending on the attention normalization. The right-hand side is dominated by second-order residuals, rendering the overall error numerically negligible.

For the proof, please refer to Appendix F. The multi-scale compression framework thus provides not only a computationally efficient attention mechanism but also a unified theoretical lens for understanding Transformer representations, context learning, and data efficiency.

3.3 Sub-quadratic Complexity Models

The above analysis can be extended to construct high-performance and efficient attention mechanisms through appropriate compositions.

Complexity Case 1: Multi-scale Compression Attention Algorithm 1 describes the basic multi-scale compression attention. Following a configuration similar to BiFormer, we partition Q and K into $N^{2/3}$ blocks of size $N^{1/3}$. After compression, routing constructs a relevance matrix of size $N^{2/3} \times N^{2/3}$, resulting in $O(N^{4/3})$ complexity. Subsequently, we select a constant number C of KV regions for each Q region and further select C_2 regions from the remaining less-relevant regions for compression to a size of S_2; subsequent scale compression proceeds in a similar manner. Given that each selected KV region has a length of $CN^{1/3}$, the computational complexity for computing attention within each region is $O(CN^{2/3})$. Multiplying by the total number of $N^{2/3}$ regions, the overall complexity of this method is $O(CN^{4/3})$.

Complexity Case 2: Hierarchical Routing for Lower Complexity (MST) Algorithm 2 implements hierarchical routing. First, we compress Q into a fixed number of M_1 tokens, leading to a region size of N/M_1. Instead of compressing K, we perform direct routing, resulting in a first-stage routing metric matrix with complexity $O(M_1N)$. For each region, we select either N/M_1 or N/M_2 KV tokens, and the remaining part is maximally compressed. Next, we recursively divide each of the N/M_1 large query regions into M_2 smaller subregions of size $N/(M_1M_2)$. Each query subregion is then routed to the previously selected N/M_2 KV tokens, and the remaining KV tokens undergo small-scale compression, again achieving routing complexity $O(M_1N)$. This process continues recursively, selecting $N/(M_2M_3)$ KV tokens at each step while compressing the remaining parts accordingly. The recursion leads to a final routing complexity of $O(MN \log_M N)$. We terminate when the query region size falls below the embedding dimension d and the number of selected KV tokens is less than $d \log_M N$, ensuring the final attention computation remains $O(MN \log_M N)$.

Complexity Case 3: Kernel-based Attention with Filtering (MSKT) Finally, we consider replacing the Softmax function with kernel functions that achieve equivalent filtering capabilities. Directly applying this replacement results in a computational complexity of $O(N)$. However, for extremely long sequences, noise significantly outweighs useful information for each token, necessitating highly effective filtering to maintain performance. To address this, we first apply explicit routing to filter out some noise before computing kernel-based attention. The routing process is similar to that in Case 2, where Q is compressed into a fixed number of tokens before routing. However, we use a fixed number C of routing iterations instead of $\log_M N$, leading to a routing complexity of $O(CMN)$. Since kernel-based attention maintains a computational complexity of $O(N)$, the overall complexity of this model remains $O(N)$ while achieving superior efficiency. By integrating routing with kernel-based filtering, we achieve an attention mechanism with linear complexity that remains robust for long sequences.

4 Experiments

4.1 Experimental Setup

We evaluate the proposed Multi-Scale Attention with hierarchical routing (MST) framework on both language and vision tasks to verify: (1) the correlation-filtering role of Softmax, (2) the effectiveness of multi-scale compression, (3) and the universality of the proposed sparse attention formulation.

For language modeling, we conduct autoregressive training on FineWeb-edu and evaluate zero-shot reasoning performance on standard commonsense benchmarks. For vision experiments, we evaluate ImageNet-1K classification and downstream dense prediction transfer on COCO. The MST configuration uses a three-level scale, with compression factors of 1, 8, and 64. To improve efficiency, the KV sequence is also compressed at a factor of 16. The number of tokens selected is 16, 16, and 16, respectively, resulting in a total of over 8k tokens.

For the language model, we use AdamW with a learning rate of $4e-4$ and a cosine annealing mechanism. The training length is 2048, and each batch contains 0.5 million tokens. Visual models are trained using AdamW for 300 epochs with weight

Table 1: **Analysis of correlation filtering and multi-scale compression mechanisms.** "C1: top64" denotes selecting the top-64 fine-grained tokens, while "C2:36" denotes selecting 36 compressed regions. Results in the form "80.3/83.9" indicate performance before and after fine-tuning.

Architecture	Configuration	Top-1 (%)
ViT-B/16 Multi-Scale Variants	Full Attention	84.1
	Fine-scale only (C1:64)	73.9
	Fine-scale only (C1:96)	78.1
	Two-scale (C1:64 + C2:36)	79.8
	Three-scale (64/24/8)	80.3/83.9
	Three-scale (96/32/16)	83.1
MST-T Filtering Variants	Top-k routing only	81.4
	Linear normalization	73.0/81.2
	Cubic filtering kernel	79.1/81.4
	Polynomial filtering kernel	79.4/81.4
	Kernelized (MSKT)	-/81.0

decay 0.05 and lr $1e-3$, and standard augmentation settings following Swin Transformer. Model configurations are kept approximately aligned with prior efficient attention baselines for fair comparison.

4.1.1 Experimental Validation

Softmax's primary role. We empirically verify the correlation filtering effect of Softmax (Proposition 3.1).

To isolate the filtering effect, we replace Softmax with several variants on ImageNet using ViT-B/16 and MST-T (Table 1). **Top-k routing** alone achieves 81.4%, close to BiFormer, indicating that a major part of Softmax's benefit comes from selecting relevant tokens. **Linear normalization** ($\frac{d_{ij}}{\sum |d_{ij}|}$) drops to 73.0% but recovers to 81.2% after fine-tuning, showing that exponential amplification is not strictly necessary after adaptation. **Cubic scaling** ($\frac{d_{ij}^3}{\sum |d_{ij}^3|}$) recovers 79.1% before fine-tuning and 81.4% after, confirming that a higher-order polynomial can mimic the exponential filter. **Polynomial kernel** behaves similarly. The linear kernel substitution $f(Q)(f(K)V)$ (replacing $e^{d_{ij}} v_j$ with $q_i(k_j^\top v_j)$) reaches 81.0% after fine-tuning, slightly below the cubic case due to normalization constraints; the gap is eliminated when training from scratch.

These findings confirm Softmax's primary role as a correlation filter that can be approximated by top-k selection or higher-order kernels, motivating our multi-scale compression design.

Multi-Scale Compression Validation. We verify the multi-scale compression mechanism on ImageNet using ViT-B/16 (84.1% baseline, Table 1). Following the hierarchical routing scheme (Case 2), queries are partitioned into 16 segments of 36 tokens each. Key selection proceeds in scales: at scale $C_1 = 1$ (no compression) we select 64 or 96 fine-grained tokens; at $C_2 = 4$ we select 32 or 16 compressed blocks; at $C_3 = 32$ we select 8 blocks. For a comparable number of selected tokens, the three-scale configuration consistently outperforms two-scale, which in turn outperforms single-scale fine-only routing. After 20 epochs of fine-tuning, the three-scale (64/24/8) variant recovers 83.9% accuracy, nearly matching full attention. The impact of Softmax filtering is analyzed in Section 4.1.1.

4.2 Language Modeling Results

Table 2 presents the main language modeling results. Across both 340M and 1.3B parameter scales, MST consistently achieves the strongest average performance among efficient attention baselines.

The gains are particularly evident on long-context reasoning and commonsense benchmarks such as LAMBADA, HellaSwag, and ARC. These results suggest that the proposed multi-scale formulation preserves critical long-range dependencies more effectively than existing sparse routing or block-compression approaches.

4.3 Visual tasks

Table 3 shows that MST achieves highly competitive ImageNet performance across different model scales. Compared with prior sparse attention approaches, MST consistently provides stronger accuracy-efficiency tradeoffs.

4.4 Inference Efficiency Analysis

To comprehensively evaluate the computational efficiency of MST, we measure end-to-end inference latency across varying sequence lengths on a single H800. Table 4 reports the average inference time (in milliseconds) for FlashAttention, NSA (from FLA triton operator), and our proposed MST. For the NSA, hyperparameters such as the partition size were set to the default values in the FLA library, while the MST used the settings mentioned earlier.

These efficiency characteristics make MST particularly well-suited for long-document understanding, multi-turn dialogue modeling, and retrieval-augmented generation tasks where context windows routinely exceed 100K tokens. The modest overhead at short sequences is a favorable trade-off for the substantial gains in long-context scenarios,

Table 2: **Performance comparison of efficient attention methods across language model scales.** MST consistently outperforms existing efficient attention baselines under comparable training budgets and parameter counts.

Model	Tokens	LAMBADA	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	SIQA	BoolQ	Avg
340M Parameters										
Transformer++-340M	15B	30.32	65.01	35.21	51.12	55.68	26.45	36.32	59.08	44.93
NSA-370M	15B	30.45	65.94	35.96	52.01	54.56	27.22	**37.32**	59.79	45.41
MoBA-340M	15B	30.01	65.51	36.15	51.97	55.21	26.89	37.14	60.09	45.38
MST-340M	15B	**31.42**	**66.46**	**36.24**	**52.38**	**56.70**	**27.56**	37.24	**60.12**	**46.02**
1.3B Parameters										
Transformer++-1.3B	40B	42.54	69.18	49.52	53.01	67.33	33.68	39.96	59.69	51.86
NSA-1.3B	40B	42.76	68.96	50.33	53.24	68.21	33.26	**41.22**	**62.11**	52.49
MoBA-1.3B	40B	42.50	69.24	50.12	53.92	68.14	33.76	40.52	61.26	52.38
MST-1.3B	40B	**43.69**	**69.78**	**50.89**	**54.04**	**68.89**	**34.05**	40.66	61.54	**52.94**

Table 3: **ImageNet-1K classification comparison.** Our multi-scale attention achieves competitive or superior performance across different model scales while preserving efficient sparse computation.

Model	Params (M)	Top-1 (%)
Tiny-scale Models		
BiFormer-T	13	81.4
SMT-T	12	82.0
MST-T (Top-k Routing)	13	81.4
MST-T (Kernelized)	13	82.0
MST-T (Multi-Scale)	13	**82.1**
Small-scale Models		
Swin-T	29	81.3
CSWin-T	23	82.7
SG-Former-S	23	83.2
BiFormer-S	26	**83.8**
MST-S (Multi-Scale)	26	**83.8**
Base-scale Models		
NAT-S	51	83.7
GC-ViT-S	51	84.3
BiFormer-B	57	84.3
MViTv2-B	52	84.4
MST-B (Multi-Scale)	54	**84.7**

Table 4: Inference latency (ms) across sequence lengths on H800 GPU. MST demonstrates superior asymptotic scaling for long-context modeling.

Seq. Len.	FlashAttention	NSA	MST (Ours)	MSKT (Ours)
8,192	47.69	57.81	67.66	76.72
16,384	117.51	123.55	146.27	145.89
32,768	331.92	284.45	365.71	270.15
65,536	1057.79	761.73	762.67	576.34
131,072	3801.54	2388.78	1759.15	1204.66
262,144	14420.34	8121.86	3987.24	2459.96

Table 5: **COCO2017 transfer results.** We replace the global attention module in SAM with MST and fine-tune for five epochs.

Model	Mean IoU	Mean F1
Original SAM	0.7978	0.8749
SAM + MST	0.7902	0.8710

Table 6: **Ablation study on ImageNet.**

Configuration	Top-1 (%)
Single-scale routing	81.4
+ Multi-scale	82.1
+ Kernel filtering	82.0
Kernel only	81.7

enabling practical deployment of high-quality attention mechanisms on commodity hardware.

4.5 Ablation Study

The ablation results validate the importance of both hierarchical compression and correlation-aware filtering. Introducing multi-scale compression yields the largest improvement, confirming that preserving information at multiple semantic resolutions is critical for approximating full attention.

5 Conclusion

This paper deconstructs the Transformer formulation into three fundamental components: (1) a multi-scale selection mechanism for the q-k inner product, replacing the inefficient global similarity computation; (2) an analysis of the Softmax operation as a filtering mechanism, enabling more flexible and efficient alternatives; and (3) a polynomial interpolation of v to enhance expressiveness. Building on these insights, we propose a highly flexible model that achieves computational complexities of $O(N^{\frac{4}{3}})$, $O(N \log N)$, and $O(N)$ under specific conditions, leveraging multi-scale token selection and separable kernel functions to improve efficiency and performance.

Limitations

The limitations of the method described in this paper lie in its excessive complexity, which makes it difficult to implement efficient hardware operators. Additionally, while we believe that the analytical framework we propose could also be applied to improving FFN layers—specifically, to develop more efficient and high-performance MOE methods—this paper does not provide an in-depth analysis or implementation of such approaches. Finally, our unification of the current state-of-the-art methods is somewhat limited, as we provide only an intuitive unification, an analysis of the relative merits of the methods, and an error analysis of Softmax Attention. However, the original Attention mechanism itself has limitations. Although our method achieves optimal error performance compared to it, we have not fundamentally addressed its shortcomings or identified directions for improvement. Future research requires us to delve deeper, explore the limitations of the current architecture, and realize a true next-generation architecture.

References

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

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

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

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. 2022. Rethinking attention with performers. *Preprint*, arXiv:2009.14794.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. *Preprint*, arXiv:2010.11929.

Abhimanyu Dubey, Filip Radenovic, and Dhruv Mahajan. 2022. Scalable interpretability via polynomials. *Preprint*, arXiv:2205.14108.

Feng-Lei Fan, Mengzhou Li, Fei Wang, Rongjie Lai, and Ge Wang. 2023. On expressivity and trainability of quadratic networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.*, 23(1).

Dongchen Han, Xuran Pan, Yizeng Han, Shiji Song, and Gao Huang. 2023. Flatten transformer: Vision transformer using focused linear attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5961–5971.

Kai Han and 1 others. 2021. Cswin transformer: A general vision transformer backbone with cross-shaped windows. *arXiv preprint arXiv:2107.00652*.

Arthur Jacot, Franck Gabriel, and Clement Hongler. 2018. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *Proceedings of the International Conference on Learning Representations*.

Lucas Dax Lingle. 2024. Transformer-VQ: Linear-time transformers via vector quantization. In *The Twelfth International Conference on Learning Representations*.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021a. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021b. Swin transformer: Hierarchical vision transformer

using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, and 1 others. 2025. Moba: Mixture of block attention for long-context llms. *arXiv preprint arXiv:2502.13189*.

Hao Peng, Jungo Kasai, Nikolaos Pappas, Dani Yogatama, Zhaofeng Wu, Lingpeng Kong, Roy Schwartz, and Noah A Smith. 2022. Abc: Attention with bounded-memory control. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A. Smith, and Lingpeng Kong. 2021. Random feature attention. *Preprint*, arXiv:2103.02143.

Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. 2020. Data-efficient image transformer. In *Advances in Neural Information Processing Systems*, pages 14167–14179.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68.

Zenglin Shen, Yanzhao Zhou, Ruixuan Bai, and Jiwei Li. 2021. Kernelized self-attention for generating long sequences. In *arXiv preprint arXiv:2106.13254*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

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

Yongtao Wu, Zhenyu Zhu, Fanghui Liu, Grigorios Chrysos, and Volkan Cevher. 2022. Extrapolation and spectral bias of neural nets with hadamard product: a polynomial net study. In *Advances in Neural Information Processing Systems*, volume 35, pages 26980–26993. Curran Associates, Inc.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. 2024a. Gated linear attention transformers with hardware-efficient training. In *International Conference on Machine Learning*, pages 56501–56523. PMLR.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. 2024b. Parallelizing linear transformers with the delta rule over sequence length. In *Advances in Neural Information Processing Systems*, volume 37, pages 115491–115522. Curran Associates, Inc.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, YX Wei, Lean Wang, Zhiping Xiao, and 1 others. 2025. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*.

Li Yuan and Yunhe Wang. 2021. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *arXiv preprint arXiv:2101.11986*.

Hyemin Yun, Woojeong Han, and Jungwook Kim. 2021. Global context transformer: The power of the word embeddings. In *arXiv preprint arXiv:2106.09681*.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020a. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297. Curran Associates, Inc.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and 1 others. 2020b. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297.

Yu Zhang, Songlin Yang, Rui-Jie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, Bolun Wang, Freda Shi, Bailin Wang, Wei Bi, Peng Zhou, and Guohong Fu. 2024. Gated slot attention for efficient linear-time sequence modeling. In *Advances in Neural Information Processing Systems*, volume 37, pages 116870–116898. Curran Associates, Inc.

Lei Zhu, Xinjiang Wang, Zhanghan Ke, Wayne Zhang, and Rynson WH Lau. 2023a. Biformer: Vision transformer with bi-level routing attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10323–10333.

Lei Zhu, Xinjiang Wang, Zhanghan Ke, Wayne Zhang, and Rynson W.H. Lau. 2023b. Biformer: Vision transformer with bi-level routing attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10323–10333.

Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. 2021. Deformable detr: Deformable transformers for end-to-end object detection. *Preprint*, arXiv:2010.04159.

A algorithm

Here we describe the pseudocode for the algorithm in 3.3.

Algorithm 1 Multi-scale Compression Attention (Basic)

Require: $Q, K, V \in \mathbb{R}^{B\times H\times N\times D}$; block size $b = N^{1/3}$; number of selected regions C; compression scales $S_2, S_3, \ldots$ with corresponding selection counts $C_2, C_3, \ldots$

Ensure: Output $O \in \mathbb{R}^{B\times H\times N\times D}$

Partition Q, K into blocks of size b, obtaining $n = N/b = N^{2/3}$ blocks.
Compute block-wise representatives (e.g., average) for Q and K, obtaining $Q_{blk}, K_{blk} \in \mathbb{R}^{B\times H\times n\times D}$.
Compute relevance matrix $R = \mathrm{sim}(Q_{blk}, K_{blk})$ $\triangleright\ O(BHn^2) = O(N^{4/3})$
For each Q block, select top-C K blocks according to R.
For each remaining scale $i = 2, 3, \ldots$**:**
  Compress the not-yet-selected K blocks into S_i tokens (e.g., average groups of S_i).
  Select C_i compressed tokens per Q block based on relevance.
Gather selected K, V blocks (or compressed tokens) and compute standard attention per Q block.
Merge outputs from all scales by weighted combination (weights proportional to original relevance).
return O

Algorithm 2 Hierarchical Routing for Lower Complexity (MST)

Require: $Q, K, V \in \mathbb{R}^{B\times H\times N\times D}$; hierarchy parameters $M_1, M_2, \ldots$; maximum recursion depth L; final selection size $d \log_M N$; embedding dimension d

Ensure: Output $O \in \mathbb{R}^{B\times H\times N\times D}$

Compress Q into M_1 tokens (e.g., average pooling) $\rightarrow Q^{(1)} \in \mathbb{R}^{B\times H\times M_1\times D}$. Each token corresponds to region size N/M_1.
Compute first-stage relevance $R^{(1)} = \mathrm{sim}(Q^{(1)}, K)$ $\triangleright\ O(M_1 N)$
For each Q region, select $\frac{N}{M_1}$ or $\frac{N}{M_2}$ KV tokens. Maximally compress all remaining KV tokens.

for $l = 2$ to L **do**
  Subdivide each Q region from previous level into M_l subregions of size $N/(M_1 \cdots M_l)$.
  For each subregion, route to previously selected $\frac{N}{M_{l-1}}$ KV tokens (or compressed tokens) $\triangleright\ O(M_1 N)$ per level
  Compress the remaining (not selected) KV tokens to a small-scale representation.
  Store selected KV indices for this level.
end for
For each leaf Q region (size $< d$), retrieve corresponding KV tokens (selected at final level) and compute standard attention.
Merge results from all levels using learned weights.
return O

B Related Work

Work related to section 2.1: For instance, Swin Transformer(Liu et al., 2021b) first performs self-attention within a local region, exchanges information between different regions using shift windows, and gradually merges regions to expand the receptive field. Deformable DETR(Zhu et al., 2021) utilizes reference points, computes a series of offsets about these points, and obtains corresponding sampling points as K tokens for interaction with Q. Biformer(Zhu et al., 2023b) partitions Q into specific regions, selects several KV regions related to each Q region using region-averaged tokens, and performs self-attention with the partitioned Q regions and their respective selected KV tokens. This method achieves a complexity of $N^{4/3}$. BigBird(Zaheer et al., 2020a) and Longformer(Beltagy et al., 2020) uses window attention to capture local characteristics, global tokens for global information. While BigBird randomly selects r tokens to mix information from other parts of the sequence, Longformer uses Dilated Sliding Window to capture information at moderate distances. Switch Transformer(Fedus et al., 2022) employs sparse routing to achieve an efficient expert mixing model. However, these models with fixed or random attention strategies for specific regions lack sufficient flexibility and may not efficiently capture crucial information, resulting in potential performance losses. On the same computational and storage resources, introducing a selection mechanism offers greater flexibility to combine relevant regions, enhancing the model's perception of various local or global information.

Work related to section 2.2: Many papers seek to replace softmax for lower computational complexity, abstracting the output as sim(Q,K)/C V.

Algorithm 3 Kernel-based Attention with Filtering (MSKT)

Require: $Q, K, V \in \mathbb{R}^{B\times H\times N\times D}$; kernel function $\kappa(q,k) = \phi(\psi(q))\phi(\psi(k))$ satisfying Proposition 3.2; routing iterations C; compression factor M (fixed number of query tokens)

Ensure: Output $O \in \mathbb{R}^{B\times H\times N\times D}$

Explicit routing for noise filtering
Compress Q into M tokens $\bar{Q} \in \mathbb{R}^{B\times H\times M\times D}$ (e.g., average pooling).
for $t = 1$ to C **do**
  Compute relevance $R^{(t)} = \text{sim}(\bar{Q}^{(t-1)}, K)$ ▷ using raw K, not compressed
  For each compressed query, select top-k relevant K indices (constant k).
  Update $\bar{Q}$ based on selected tokens (or keep fixed).
end for
Kernel-based attention with filtering
Apply kernel function: $\Phi_Q = \phi(\psi(Q))$, $\Phi_K = \phi(\psi(K))$ ▷ $O(ND)$
Compute output using linear attention order: $\tilde{O} = \Phi_Q(\Phi_K^\top V)$ ▷ $O(ND)$ after reordering
(Optional) Re-apply routing-based selection on $\tilde{O}$ to refine.
return $\tilde{O}$

They use separable operations like $f_1(Q)f_2(K)^T$ to achieve an equivalent of sim(Q,K), enabling the exchange of orders for linear complexity, i.e., $f1(Q)(f2(K)^TV)$, which reduces the original complexity from $O(N^2d)$ to $O(Nd^2)$. Linear attention(Katharopoulos et al., 2020) uses separable kernel functions to replace softmax, swapping orders for linear complexity. Random Feature Attention(Peng et al., 2021) and Performer(Choromanski et al., 2022) utilizes different random feature method to approximate the Gaussian kernel function and softmax. Linformer(Wang et al., 2020) modifies the KV token matrix with low-rank characteristics and applies two softmax layers to achieve linearity. Flatten Transformer(Han et al., 2023) employs positive features and polynomial transformations to pull tokens in similar directions in the embedding with similar positive features. However, these papers lack a deep understanding of softmax's essential effect, which may result in performance losses in certain situations. Therefore, this paper conducts an in-depth analysis of the role of Softmax and utilizes its filtering effect as an equivalent replacement to flexibly improve Attention operations, ensuring adaptability to various scenarios while maintaining performance.

Work related to section 2.3: Finally, as the ultimate output in Transformers involves a linear fit for V, this linear operation further limits its expressive capacity. Leveraging the theories of polynomial neural networks, this paper introduces polynomial fitting to enhance expressiveness and offer a more interpretable improvement to the model. Since all functions can be approximated by piecewise polynomial functions, many studies have investigated artificial neural networks (ANNs) from the perspectives of polynomials and kernel functions. (Jacot et al., 2018) establishes a connection between ANNs and kernel methods, introducing a new kernel function called the Neural Tangent Kernel to describe the evolution of the ANN training process. Furthermore, (Wu et al., 2022) derives finite-width Neural Tangent Kernel formulas for neural networks with Hadamard products (NNs-Hp) and explains the advantages of polynomial neural networks (PNNs) over standard neural networks in extrapolation and spectral deviation. (Fan et al., 2023) uses spline interpolation, and the Scalable Polynomial Additive Model (SPAM)(Dubey et al., 2022) models high-order feature interactions to achieve an interpretable high-performance system.

C Softmax filter

Here is the proof of Proposition 3.1.

Proof. We analyze the three properties separately.

1. Exponential suppression of negatively correlated tokens. Consider a negatively correlated token with similarity $d_{ij}^- < 0$ and a positively correlated token with similarity $d_{ik}^+ > 0$. Their normalized attention weights are

$$p_{ij}^- = \frac{e^{d_{ij}^-}}{\sum_{t\le i} e^{d_{it}}}, \qquad p_{ik}^+ = \frac{e^{d_{ik}^+}}{\sum_{t\le i} e^{d_{it}}}.$$

Taking the ratio yields

$$\frac{p_{ij}^-}{p_{ik}^+} = e^{d_{ij}^- - d_{ik}^+}.$$

Since $d_{ij}^- - d_{ik}^+ < 0$, the ratio decreases exponentially as the correlation gap widens. Hence, negatively correlated tokens contribute negligibly to the final attention distribution.

2. Exponential amplification of positively correlated tokens. Let $d_2 = d_1 + \Delta d$ with $\Delta d \geq 0$. Before applying Softmax, the relative similarity ratio is

$$r_d = \frac{d_2}{d_1} = 1 + \frac{\Delta d}{d_1}.$$

After exponentiation,

$$r_s = \frac{e^{d_2}}{e^{d_1}} = e^{\Delta d}.$$

Thus, the Softmax scaling ratio is

$$s = \frac{e^{\Delta d}}{1 + \frac{\Delta d}{d_1}}.$$

Using the Taylor expansion $e^{\Delta d} = 1 + \Delta d + \sum_{n=2}^{\infty} \frac{(\Delta d)^n}{n!}$, we obtain

$$e^{\Delta d} - \left(1 + \frac{\Delta d}{d_1}\right) = \left(1 - \frac{1}{d_1}\right)\Delta d + \sum_{n=2}^{\infty} \frac{(\Delta d)^n}{n!}.$$

The higher-order polynomial terms imply that the exponential kernel enlarges the separation between relevant tokens in a super-linear manner. Consequently, highly correlated tokens dominate the resulting attention distribution after normalization.

Moreover, the absolute score difference after exponentiation is

$$\Delta p = e^{d_2} - e^{d_1} = e^{d_1}\left(e^{\Delta d} - 1\right).$$

Applying the mean value theorem to e^x gives $e^{\Delta d} - 1 = e^{\theta(\Delta d)}\Delta d$ for some $\theta(\Delta d) \in (0, \Delta d)$, so

$$\Delta p = e^{d_1 + \theta(\Delta d)}\Delta d.$$

Hence, the amplification strength itself grows exponentially with the underlying similarity magnitude.

3. Limited contribution of low-magnitude similarities. When $d_1 < 1$, the denominator $1 + \frac{\Delta d}{d_1}$ can become relatively large, which might increase the relative scaling factor for small similarities. However, the corresponding exponential term e^{d_1} remains close to 1, implying that the absolute attention contribution remains limited after normalization:

$$p_{ij} = \frac{e^{d_{ij}}}{\sum_{k \leq i} e^{d_{ik}}}.$$

Therefore, although Softmax may mildly up-weight small positive similarities, their overall influence is still dominated by strongly correlated tokens with substantially larger exponential responses. □

D Filtering Equivalence

Firstly, each token is composed of features from various columns, where a larger value indicates a more prominent and pronounced presence of that feature, while a smaller value signifies the absence of the feature. Of course, the numerical values themselves might have inherent meanings, displaying different effects based on their positive or negative magnitudes. When modeling similarity or distance through inner products, it fundamentally involves comparing features. If both tokens share certain features and lack others, it will positively influence the inner product. However, when comparing features, one token may have positive values and the other negative, which could negatively impact the final inner product. In Flatten Transformer(Han et al., 2023), using ReLU selectively with positive values filters out the need to check whether the original dot products are greater than zero. This allows for directly stretching x_1x_2 based on the use of $f(x_1)f(x_2)$, achieving a filtering effect similar to softmax. However, relying solely on positive features may lead to misjudgments of similarity. For instance, if the original dot products are small but using only positive values results in a large inner product, as in the case of $[2, -2, 3, -5, 5]$ and $[3, 2, -1, 5, 6]$. Nevertheless, from another perspective, ReLU can be viewed as a feature activation, akin to learning to activate specific features for similarity judgments.

This paper currently focuses on finding a function for the QK operation and filtering effect that is equivalent to Softmax, under the assumption of feature activation. Initially, an activation function is applied to activate the features of input Q and V. If the features activated by the activation function do not guarantee positivity, further processing is performed to ensure that the activated features are represented in a positive manner. Subsequently, these features are used for modeling similarity through inner product operations. Representing features as positive values facilitates the identification of simple functions to achieve a non-linear filtering effect similar to Softmax, akin to the approach used in Flatten Transformer, which is extended in this paper.

Proof. We analyze the required stretching effect. Softmax essentially makes the ratio of attention scores $e^{q_1k_1}/e^{q_2k_2} = e^{q_1k_1 - q_2k_2}$ much larger than the original ratio $q_1k_1/(q_2k_2)$ when $q_1k_1 > q_2k_2$. To mimic this, we require that the function

$F(q,k) = \frac{\phi(\psi(q))\phi(\psi(k))}{qk}$ be positive and increasing in each argument when the other is fixed. Since ψ is positive, we absorb it into ϕ by redefining $\tilde{\phi} = \phi \circ \psi$, and assume $\tilde{\phi}$ satisfies the same derivative condition. For simplicity, denote ϕ directly.

Take partial derivative with respect to q (the case for k is symmetric):

$$\frac{\partial F(q,k)}{\partial q} = \frac{\phi'(q)\phi(k)qk - \phi(q)\phi(k)k}{(qk)^2} = \frac{\phi(k)}{q^2 k}\big(\phi'(q)q - \phi(q)\big).$$

Since $\phi(k) > 0$ and $q^2 k > 0$, the sign is determined by $\phi'(q)q - \phi(q)$. Requiring F to increase with q (for fixed $k > 0$) gives $\phi'(q)q > \phi(q)$, i.e. $\frac{d}{dq}(\ln \phi(q) - \ln q) > 0$. Hence $\ln(\phi(q)/q)$ is strictly increasing, so $\phi(q)/q > c_0$ for some constant $c_0 > 0$. To ensure that the growth is not merely linear (which would produce constant ratio), we impose a polynomial lower bound: there exists $n > 0$ such that $\phi'(q) \geq cq^{1/n}$.

Integrating $\phi'(q) \geq cq^{1/n}$ from 1 to q (for $q \geq 1$):

$$\phi(q) - \phi(1) \geq c\int_1^q t^{1/n} dt = c\frac{n}{n+1}(q^{(n+1)/n} - 1).$$

Thus for sufficiently large q, $\phi(q) \geq c' q^{1+1/n}$ with $c' = cn/(n+1)$. For $0 < q < 1$, we integrate from q to 1: $\phi(1) - \phi(q) \geq c\int_q^1 t^{1/n} dt$, leading to $\phi(q) \leq \phi(1) - c\frac{n}{n+1}(1 - q^{(n+1)/n})$. However, to guarantee $\phi(q)/q$ increasing as q decreases, a symmetric lower bound $\phi(q) \geq cq^{1-1/n}$ is appropriate.

Now check the ratio condition for two different products. Suppose $q_1 k_1 > q_2 k_2 > 0$ and without loss assume $q_1 \geq q_2$, $k_1 \geq k_2$ (if one is larger and the other smaller, the analysis is analogous). Then using the lower bounds:

$$\frac{\phi(q_1)\phi(k_1)}{\phi(q_2)\phi(k_2)} \geq \frac{c^2 q_1^{1+1/n} k_1^{1+1/n}}{c^2 q_2^{1+1/n} k_2^{1+1/n}} = \left(\frac{q_1 k_1}{q_2 k_2}\right)^{1+1/n}.$$

Since $1 + 1/n > 1$, this ratio grows super-linearly in the product ratio, mimicking the exponential amplification. For the mixed case $q_1 > 1$, $k_1 < 1$, we have bounds $\phi(q_1) \geq cq_1^{1+1/n}$ and $\phi(k_1) \geq ck_1^{1-1/n}$, giving $\phi(q_1)\phi(k_1) \geq c^2 q_1^{1+1/n} k_1^{1-1/n}$. The ratio becomes

$$\frac{\phi(q_1)\phi(k_1)}{\phi(q_2)\phi(k_2)} \geq \left(\frac{q_1}{q_2}\right)^{1+1/n}\left(\frac{k_1}{k_2}\right)^{1-1/n}.$$

When $q_1 k_1 > q_2 k_2$, it can be shown that this lower bound exceeds $(q_1 k_1)/(q_2 k_2)$ provided n is finite, thus preserving the stretching property. Therefore the sufficient condition holds. □

E Unification analysis of Sparse Attention

Proof of Proposition 3.4. We establish each case by analyzing the asymptotic behavior of the gating vector $\boldsymbol{\gamma}$ and the routing policy $\pi(q_i)$.

(i) Sliding Window Attention via Optimality Convergence. Consider the constrained selection objective:

$$\Omega^*(q_i) = \arg\max_{\Omega \subseteq \{1,\ldots,n\}, |\Omega| \leq w} \sum_{j \in \Omega} \exp(q_i k_j^\top). \tag{13}$$

Under the standard spatial locality prior, relevance scores decay monotonically with distance: $q_i k_j^\top \approx \phi(|i-j|)$ with $\phi' < 0$. By the rearrangement inequality, the optimum is uniquely achieved at $\Omega^* = \{j : |i-j| \leq \lfloor w/2 \rfloor\}$. During training, the learnable policy $\pi(q_i)$ asymptotically converges to this configuration. Setting $\gamma_1 = 1$ for the finest scale and masking indices outside Ω^* yields:

$$\mathcal{A}_{\text{MS}}(q_i) \xrightarrow{\text{opt.}} \sum_{j \in \Omega^*} \frac{\exp(q_i k_j^\top)}{\sum_{j' \in \Omega^*} \exp(q_i k_{j'}^\top)} v_j, \tag{14}$$

which precisely recovers sliding window attention.

(ii) Global Token Attention via Structural Prior Convergence. Augment relevance scores with learnable positional biases: $s_{ij} = q_i k_j^\top + b_j$. Tokens corresponding to structural anchors (e.g., [CLS], initial patches) acquire persistent high biases $b_j \to \infty$ during training. The constrained optimization enforces $\mathcal{G} \subset \Omega^*$ for all queries, where $\mathcal{G}$ denotes the global token set. With $\gamma_1 = 1$ for global tokens and $\gamma_2 = 1$ for compressed local blocks ($p_2 = w$), the attention decomposes as:

$$\mathcal{A}_{\text{MS}}(q_i) \xrightarrow{\text{opt.}} \sum_{j \in \mathcal{G}} \alpha_{1,j} v_j + \sum_{s=0}^{\lfloor (n-|\mathcal{G}|)/w \rfloor} \alpha_{2,s} \bar{v}_{w,s}, \tag{15}$$

matching the attention sink formulation.

(iii) Top-k Routing via Constructive Hard-Gating. Define a binary gating function $\mathcal{G}_{\text{topk}}(\mathbf{s}_i) \in \{0,1\}^n$ that selects indices with the highest k scores. Setting $M = 1$, $p_1 = 1$, and $\gamma_1 = 1$, the framework reduces to:

$$\mathcal{A}_{\text{MS}}(q_i) = \sum_{j=1}^{n} \frac{[\mathcal{G}_{\text{topk}}(\mathbf{s}_i)]_j \exp(s_{ij})}{\sum_{j'=1}^{n} [\mathcal{G}_{\text{topk}}(\mathbf{s}_i)]_{j'} \exp(s_{ij'})} v_j. \tag{16}$$

By construction, this is identical to dynamic top-k routing attention.

(iv) Block-Sparse Attention via Fixed Pattern Gating. Set uniform compression $p_m = b$ for all m, and define a fixed activation subset $\mathcal{B} \subseteq \{1, \ldots, M\}$. Configure $\gamma_m = \mathbb{I}(m \in \mathcal{B})$. The output becomes:

$$\mathcal{A}_{\text{MS}}(q_i) = \sum_{m \in \mathcal{B}} \sum_{s=0}^{K_b - 1} \frac{\exp(q_i \bar{k}_{b,s}^\top)}{\sum_{m' \in \mathcal{B}} \sum_{s'} \exp(q_i \bar{k}_{b,s'}^\top)} \bar{v}_{b,s}, \tag{17}$$

which exactly matches block-sparse attention with aggregated block representations.

(v) Compression-Based Attention via Single-Scale Activation. Consider the regime where dynamic routing is disabled ($\pi(q_i) \equiv \text{id}$), and only the coarse compression branch is activated via $\gamma_c = 1$ and $\gamma_{m \neq c} = 0$. Fix the compression ratio $p_c = P \gg 1$, partitioning the sequence into $K = \lceil n/P \rceil$ non-overlapping blocks. The attention simplifies to:

$$\begin{aligned} \mathcal{A}_{\text{MS}}(q_i) &= \sum_{s=0}^{K-1} \frac{P \exp(q_i \bar{k}_{P,s}^\top)}{\sum_{s'=0}^{K-1} P \exp(q_i \bar{k}_{P,s'}^\top)} \bar{v}_{P,s} \\ &= \sum_{s=0}^{K-1} \frac{\exp(q_i \bar{k}_{P,s}^\top)}{\sum_{s'=0}^{K-1} \exp(q_i \bar{k}_{P,s'}^\top)} \left(\frac{1}{P} \sum_{t=sP}^{(s+1)P-1} v_t \right) \end{aligned} \tag{18}$$

The scaling factor P cancels in the normalization, leaving a standard Softmax over compressed key-value pairs. This formulation exactly recovers uniform token compression/pooling mechanisms (e.g., Token Merging, Compressive Transformers, and Pooling Attention). When $P \to n$ (extreme compression to a single memory token), it converges to global memory-based attention. The gating mechanism ensures that during training, the model can dynamically suppress finer scales and rely solely on compressed representations when task complexity permits, achieving theoretical convergence to compression-based paradigms.

Since all five paradigms emerge either as asymptotic limits of the optimal routing policy, constructive gating configurations, or single-scale compression limits, the proposition is established. □ □

Proposition E.1 (Convergence of Optimal Sparse Attention)**.** *Let $\mathcal{A}_{full}$ denote the full attention operator and $\mathcal{A}_{MS}^{(M)}$ the multi-scale approximation with M scales. Under the assumption that token correlations follow a power-law decay $|q_i k_j^\top| \sim j^{-\beta}$ for $\beta > 0$, there exists a scale configuration $\{p_m\}_{m=1}^M$ such that:*

$$\|\mathcal{A}_{full}(q_i) - \mathcal{A}_{MS}^{(M)}(q_i)\|_2 \leq \varepsilon(M), \tag{19}$$

where $\varepsilon(M) \to 0$ as $M \to \infty$. Moreover, for any fixed $\varepsilon > 0$, there exists $M^* = \mathcal{O}(\log(1/\varepsilon))$ scales sufficient to achieve ε-approximation.

Proof Sketch. The proof relies on the correlation filtering property established in Section **??**. Since Softmax exponentially suppresses low-correlation tokens, the attention mass concentrates on a small subset of highly relevant tokens. By allocating fine scales ($p_m \approx 1$) to high-correlation regions and progressively coarser scales to low-correlation regions, we can bound the approximation error.

Let $\mathcal{I}_h$ denote the set of high-correlation indices with $|q_i k_j^\top| \geq \tau$ for threshold τ. The error decomposes as:

$$\|\mathcal{A}_{\text{full}} - \mathcal{A}_{\text{MS}}\| \leq \underbrace{\sum_{j \in \mathcal{I}_h} |p_j - \tilde{p}_j|}_{\text{high-correlation error}} + \underbrace{\sum_{j \notin \mathcal{I}_h} p_j}_{\text{tail error}}, \tag{20}$$

where p_j are the true attention weights and $\tilde{p}_j$ the approximated weights.

For the tail error, the power-law decay ensures $\sum_{j \notin \mathcal{I}_h} p_j \leq C\tau^{-\beta}$. For the high-correlation error, using $p_m = 1$ for $j \in \mathcal{I}_h$ eliminates compression error. By choosing $\tau \sim M^{-1/\beta}$ and allocating M scales appropriately, we achieve $\varepsilon(M) \sim M^{-\gamma}$ for some $\gamma > 0$. The logarithmic dependence follows from the exponential suppression property of Softmax. □ □

F Multi-scale Attention Error analysis

Proof. We analyze the error for a single scale partition. The full attention contribution from block $\mathcal{B}_l$ is:

$$\mathcal{A}_{\text{full}}^{(l)} = \frac{\sum_{j \in \mathcal{B}_l} e^{d_{ij}} v_j}{Z_{\text{full}}}, \quad Z_{\text{full}} = \sum_{j=1}^{n} e^{d_{ij}}. \tag{21}$$

The multi-scale approximation compresses this block into:

$$\mathcal{A}_{\text{MS}}^{(l)} = \frac{p_l e^{\bar{d}_{il}} \bar{v}_l}{Z_{\text{MS}}}, \quad \bar{d}_{il} = \frac{1}{p_l} \sum_{j \in \mathcal{B}_l} d_{ij}. \tag{22}$$

The error $\varepsilon_l = \mathcal{A}_{\text{full}}^{(l)} - \mathcal{A}_{\text{MS}}^{(l)}$ shares a common denominator $Z_{\text{full}} Z_{\text{MS}}$, with numerator:

$$N_l = \left(\sum_{j \in \mathcal{B}_l} e^{d_{ij}} v_j \right) Z_{\text{MS}} - \left(p_l e^{\bar{d}_{il}} \bar{v}_l \right) Z_{\text{full}}. \tag{23}$$

Applying the Taylor expansion of $e^{d_{ij}}$ around the block mean $\bar{d}_{il}$ yields:

$$e^{d_{ij}} = e^{\bar{d}_{il}} \left(1 + \delta_j + \tfrac{1}{2}\delta_j^2 + \mathcal{O}(\delta_j^3)\right), \quad \delta_j = d_{ij} - \bar{d}_{il}. \tag{24}$$

Substituting this into N_l and grouping terms by order of δ_j:

$$\begin{aligned} N_l = e^{\bar{d}_{il}} \Bigg[& \underbrace{\left(\sum_{j\in\mathcal{B}_l} v_j\right) Z_{\text{MS}} - p_l \bar{v}_l Z_{\text{full}}}_{\text{Zeroth-order}} \\ & + \underbrace{\left(\sum_{j\in\mathcal{B}_l} \delta_j v_j\right) Z_{\text{MS}}}_{\text{First-order}} \\ & + \underbrace{\tfrac{1}{2}\left(\sum_{j\in\mathcal{B}_l} \delta_j^2 v_j\right) Z_{\text{MS}} + \mathcal{O}(\delta_j^3)}_{\text{Second-order \& higher}} \Bigg]. \end{aligned} \tag{25}$$

By definition of the compressed mean, $\sum_{j\in\mathcal{B}_l} v_j = p_l \bar{v}_l$. For fine-grained partitioning where $Z_{\text{MS}} \approx Z_{\text{full}}$, the zeroth-order term cancels exactly. For the first-order term, we utilize the identity $\sum_{j\in\mathcal{B}_l} \delta_j = 0$ to rewrite:

$$\begin{aligned} \sum_{j\in\mathcal{B}_l} \delta_j v_j &= \sum_{j\in\mathcal{B}_l} \delta_j (v_j - \bar{v}_l) + \bar{v}_l \underbrace{\sum_{j\in\mathcal{B}_l} \delta_j}_{=0} \\ &= \sum_{j\in\mathcal{B}_l} \delta_j (v_j - \bar{v}_l). \end{aligned} \tag{26}$$

This reveals that the *pure first-order score error vanishes identically* due to the mean-centering property $\sum \delta_j = 0$. Consequently, the approximation error is strictly dominated by the second-order residual $\mathcal{R}_l^{(2)} = \mathcal{O}\left(\sum_{j\in\mathcal{B}_l} \delta_j^2\right)$. Furthermore, **when the multi-scale partition function exactly matches the full partition function ($Z_{\text{MS}} = Z_{\text{full}}$), the normalization-induced discrepancy vanishes identically**, leaving only the intrinsic second-order compression residual.

Bounding the coupled first-order term and noting that $|\delta_j| \le \frac{1}{p_l} \sum_{k\in\mathcal{B}_l} |d_{ik} - d_{ij}|$, we obtain:

$$\begin{aligned} \|N_l\|_2 \le e^{\bar{d}_{il}} Z_{\text{MS}} \Bigg[& \left(\sum_{j\in\mathcal{B}_l} |\delta_j|\right) \left(\sum_{j\in\mathcal{B}_l} \|v_j - \bar{v}_l\|_2\right) \\ & + \mathcal{R}_l^{(2)} \Bigg]. \end{aligned} \tag{27}$$

Dividing by $Z_{\text{full}} Z_{\text{MS}}$ and setting $C_l = e^{\bar{d}_{il}} / Z_{\text{full}}$ yields the stated bound. Since $\mathcal{R}_l^{(2)} = \mathcal{O}(1/p_l^2)$ and the score variance within blocks is inherently small for relevant tokens, the overall error is numerically negligible. □ □

G Multi-scale Discussion

Furthermore, based on experiments and Transformer formulations, we observe that this interpolation method achieves a linear hyperplane fitting of the underlying learned manifold data distribution, interpolating relevant data points to positions with closely represented distances in the inner product space. In this process, the role of the feedforward neural network (FFN) is evident. It acts as a spatial transformer, preserving the data distribution on the continuously fitted transformed manifold. Therefore, in this context, the FFN plays a crucial role in storing information.

In contrast, the Attention mechanism operates flexibly with data points. By selecting relevant tokens, Attention utilizes the relevant points to interpolate new data points and expected outputs with similar features in this space. This approach also provides a good explanation for the process of context learning. By using only the relevant new input data points, based on the established manifold distribution and modeled relationships, Attention selects relevant tokens in the designed sequence to interpolate words and sentences related to the expected context. This is why designing prompts with more similar points offers a more accessible way to interpolate and regress the expected context. With more related points, the interpolation can be more precise in regressing the desired tokens. Furthermore, within the model, it is possible to randomly generate KV tokens that are close to the selected tokens, thus using more similar elements for interpolation to improve performance.

It can be further inferred that the establishment of more precise distributions and relationships requires a larger amount of training data. Consequently, in this study, experiments with the CIFAR dataset revealed that, compared to using patch=2, employing coarser compressed tokens with patch=4 for interpolation is more conducive to small datasets. Additionally, based on the relationships established through this distribution process, dataset augmentation can be achieved. For labeled data, one approach to expanding the dataset is by employing cross-interpolation using two images

with similar semantics or labels.

These experiments have demonstrated the effectiveness of the interpolation method and presented a range of possibilities for enhancing flexibility and performance. These insights contribute not only to a deeper understanding of the operational mechanisms of the model but also provide strong inspiration for future research and optimization efforts.

H Polynomial Regression

Proposition H.1 (Polynomial Interpretation of the Output Convex Combination)**.** *The outermost convex combination in attention, $o_i = \sum_j A_{ij} v_j$ with $\sum_j A_{ij} = 1$, can be interpreted as a linear interpolation over the value vectors. This operation corresponds to fitting a linear hyperplane to the data manifold that approximates the target function. By extending the outputs and values to higher orders, e.g., considering $o_i^{(2)} = \sum_j A_{ij} v_j^2$ (element-wise square) or more generally $o_i^{(p)} = \sum_j A_{ij} v_j^{\otimes p}$ (tensor product), the model obtains the ability to represent piecewise polynomial functions of degree p. In particular, using a combination $av^3 + bv^2 + v$ can enhance expressive power by increasing the order of interactions among token features.*

However, such higher-order extensions introduce significant memory overhead and are prone to overfitting; thus they are typically applied only after training (e.g., during inference or fine-tuning) rather than during the main training phase.

Proof. The standard attention output $o_i = \sum_j A_{ij} v_j$ is a convex combination because $A_{ij} \geq 0$ and $\sum_j A_{ij} = 1$. This is a linear interpolation of the vectors $\{v_j\}$, and thus o_i lies in the convex hull of the v_j's. Any smooth function can be approximated locally by a linear function; the attention mechanism effectively learns such a local linear approximation over the sequence.

If we extend the formulation to include quadratic terms, i.e., we consider $o_i^{(2)} = \sum_j A_{ij}(v_j \otimes v_j)$ (flattened), then the output belongs to the convex hull of the quadratic features. More generally, for polynomial degree p, $o_i^{(p)} = \sum_j A_{ij}\, \Pi^{(p)}(v_j)$ where $\Pi^{(p)}$ denotes all monomials up to degree p. This allows the model to represent piecewise polynomial functions of degree p on the data manifold.

The specific combination $av^3 + bv^2 + v$ corresponds to a third-degree polynomial in v with coefficients $a, b, 1$. This can capture more complex interactions among token dimensions. However, the number of terms grows as $O(d^p)$ for d-dimensional vectors, leading to prohibitive memory consumption for $p > 2$ in high dimensions. Moreover, the increased capacity often leads to overfitting when training data is limited. Therefore, such higher-order extensions are rarely used during training; they are instead reserved for post-training enhancement where additional computation can be tolerated. □

I Piecewise Polynomial Theory

Below is a brief explanation of the piecewise polynomial theory.

Firstly, based on the definition of real tasks and category theory or set theory, it can be understood that all tasks can be completed by modeling as a function. The essence of neural network learning is to continuously fit this function using existing input and output through gradient iteration or other machine learning algorithms. So, what is an effective way to fit a function? Polynomial, bump function, or other base functions. Here, I adopt the theory of piecewise polynomials to analyze the function represented by neural networks. In fact, any function can be approximated by piecewise polynomials. For example, for any function, it can be divided into many segments of continuous functions based on its breakpoints, and each segment of continuous function can be approximated by taking a finite number of terms of the polynomial after Taylor expansion.

Next, a simple analysis of the function represented by deep neural networks. Because deep neural networks are computed recursively in a layer-by-layer manner, they can be obtained by using the recursive method of $Y = f(X) = f(x_1, x_2, ...) = f_n(...f_2(f_1(x_1, x_2, ...)))$. First, consider a single layer represented by f_i, where the MLP, CONV, Attention, and other operators are used to establish the manifold required for the task of fitting the data distribution. The activation function is used to implement piecewise-like functions (activation is akin to creating a breakpoint in the domain by discarding certain values).

Furthermore, with each layer of recursion, there are more breakpoints, and the frequency of breakpoints corresponds to the information frequency, i.e., deep layers represent high-frequency information, while shallow layers represent low-frequency information. If there is an ascending operation in

each layer, then the deep layer will contain high-frequency information of high power of x^p (such as deep layers in images represent specific objects, while shallow layers represent low-frequency features such as edges and corners). Combined with the theory of piecewise polynomials, the multi-order fitting capability is definitely not lower than the first-order linear fitting.

According to the above analysis, it can explain or guide many things. Additionally, there may be errors in the analysis, for which we apologize in advance.

The necessity of piecewise, which can enhance the fitting ability, expand the function space that can be fitted, and alleviate overfitting.

The necessity of shortcuts, deep layers need to add low-order polynomials or low-frequency information from shallow layers to build a larger function space.

The selection of (depth, width, dim), it can be seen that they need to be relatively balanced to build a larger function space, fit more functions, but specific tasks may have biases and require an unbalanced setting.

Transformers use higher-order Attention operations, which approximate third-order polynomials and have stronger fitting capabilities than the first-order operations of CONV. However, there are differences in the capabilities of different third-order operations, so it is necessary to design necessary high-order interactions between different x_{ij} based on the task inputs. Moreover, higher-order operations are prone to overfitting and require more data.

Furthermore, it can explain emergence, where models of different depths represent the same object, such as x^{p_1} and x^{p_2} being equal to c; then, the granularity of different layers is c^{1/p_1} and c^{1/p_2}. Combined with knowledge from physics, a large number of simple combinations at a more microscopic level can explain very complex macroscopic problems. Therefore, the simple addition and multiplication of elements in deeper networks may emerge capabilities that are incomprehensible at the macroscopic level.

Of course, the above analysis is too simple to explain complex networks in reality, and there may also be errors. Due to space constraints, we will conduct more complex and deeper modeling and experiments in future work.

J Geometric Interpretation: Manifold Interpolation and Context Learning

Beyond computational efficiency, the multi-scale compression mechanism admits a compelling geometric interpretation that sheds light on the fundamental operating principles of Transformer models.

Proposition J.1 (Manifold Interpolation Interpretation)**.** *Let $\mathcal{M} \subset \mathbb{R}^d$ denote the learned manifold of token representations. The multi-scale attention mechanism performs adaptive interpolation on $\mathcal{M}$ by:*

(i) Selecting relevant anchor points $\{k_j\}$ based on query-key correlations,

(ii) Constructing compressed representations $\{\bar{k}_{p,s}\}$ as convex combinations of nearby points on $\mathcal{M}$,

(iii) Interpolating value vectors $\{\bar{v}_{p,s}\}$ to approximate the target representation at the query location.

This interpolation occurs in the reproducing kernel Hilbert space (RKHS) induced by the exponential kernel $K(q, k) = \exp(q^\top k)$.

Proof Sketch. The compressed representation $\bar{k}_{p,s} = \frac{1}{p}\sum_{j\in\mathcal{B}} k_j$ is a convex combination of tokens in block $\mathcal{B}$, placing it within the convex hull of $\mathcal{M}$. The attention output:

$$\mathcal{A}(q) = \sum_{m,s} \alpha_{m,s}\bar{v}_{p_m,s} \tag{28}$$

is a weighted average of value vectors, where weights $\alpha_{m,s} \propto \exp(q^\top \bar{k}_{p_m,s})$ depend on kernel evaluations. This is precisely the Nyström approximation of kernel ridge regression on $\mathcal{M}$, performing interpolation in the RKHS. □ □

Remark J.1 (Role of Feedforward Networks)**.** *The feedforward network (FFN) acts as a spatial transformer that warps the manifold $\mathcal{M}$ to align with task-specific geometry. While attention performs interpolation in the input space, the FFN learns the nonlinear mapping that preserves semantic relationships on the transformed manifold. This explains why FFN parameters constitute the majority of model capacity—they encode the manifold structure itself.*

Remark J.2 (Context Learning as Manifold Regression)**.** *In-context learning can be interpreted as few-shot manifold regression: given a prompt*

with k demonstration examples, the model identifies the relevant submanifold $\mathcal{M}_{task} \subset \mathcal{M}$ and interpolates to produce the expected output. This explains why:

(i) More similar demonstrations improve performance (denser sampling of $\mathcal{M}_{task}$),

(ii) Prompt engineering is effective (strategic placement of anchor points),

(iii) Models generalize to unseen tasks (smoothness of $\mathcal{M}$ enables interpolation to new regions).

Proposition J.2 (Data Efficiency and Scale Selection)**.** *For small datasets, coarser compression scales ($p \geq 4$) provide stronger regularization by enforcing smoother interpolation on $\mathcal{M}$, reducing overfitting. Conversely, large datasets benefit from finer scales ($p \in \{1, 2\}$) that capture high-frequency variations in the manifold structure.*

Empirical Justification. Experiments on CIFAR-10 demonstrate that patch size $p = 4$ outperforms $p = 2$ for limited training data, while the trend reverses with data augmentation. This aligns with the bias-variance tradeoff: coarse scales introduce bias (smoother interpolation) but reduce variance (fewer parameters), beneficial for small datasets. □ □

Remark J.3 (Dataset Augmentation via Cross-Interpolation)**.** *The manifold interpretation suggests a novel data augmentation strategy: for two semantically similar samples (x_1, y_1) and (x_2, y_2), generate synthetic samples by interpolating their token representations:*

$$\begin{aligned} \tilde{k} &= \lambda \bar{k}(x_1) + (1-\lambda)\bar{k}(x_2), \\ \tilde{y} &= \lambda y_1 + (1-\lambda) y_2, \end{aligned} \tag{29}$$

where $\lambda \in [0, 1]$. This effectively densifies the sampling of $\mathcal{M}$, improving generalization.

K approximate interpolation

K.1 Discussion 1

The theoretical formulae are as follows:

$$f(x, y) = \sum_{i=0}^{h} \sum_{j=0}^{w} W_{ij} \cdot f(x+i, y+j)$$

$$o(x, y) = \sum_{i=0}^{h} \sum_{j=0}^{w} g(d_{ij}) \cdot v(x+i, y+j)$$

In the above equations, $h = 2$ corresponds to bilinear interpolation, and $h = w = 3$ corresponds to cubic spline interpolation. When interpolating a point using the entire image, comparing these interpolation formulas with the Attention formula reveals that their fundamental principles are similar. Both methods involve interpolating with "points in close proximity" by combining vectors with similar features. Consequently, the interpolated results emphasize these shared features, making them more beneficial for task completion.

However, according to the definition of interpolation, it involves fitting a curve through known points. In practice, Attention does not require the values at the original positions to be known, and the resolution remains unchanged, maintaining the size of the entire image. The positions used in the above formulas are intrinsic two-dimensional position vectors of the sequence. However, in models, Attention often incorporates positional encodings, whether relative, absolute, or from CNN, into feature vectors as positional information. Therefore, the actual interpolation positions should align with the encoded vectors used above, representing positions in feature space rather than the true physical sequence of the image.

In feature space, it is akin to forming a linear hyperplane by convexly combining the vertices of the v matrix to fit a manifold. This approach aligns more closely with the definition of regression fitting. Thus, this paper considers it as an approximate "interpolation" in the direction of the sequence entity, essentially a regression fitting in feature space.

The approximate definition of "interpolation": The definition formula for interpolation, as shown in the equation $f(x, y) = \sum_{i=0}^{h} \sum_{j=0}^{w} W_{ij} \cdot f(x+i, y+j)$, depicts coefficients that increase as the distance determined by a certain metric decreases. Ultimately, this achieves the combination of vectors with similar features in close proximity, highlighting relevant characteristics. Note: It is not mandatory for the curve to pass through known points.

K.2 Discussion 2

Self-Attention is an extension of the interpolation formula on Riemannian manifolds given by $Exp_{(P,\lambda)}(T) = P^{1/2} \exp(\lambda P^{-1/2} T P^{-1/2}) P^{1/2}$, where T is the tangent vector matrix, a symmetric matrix decomposable into ww^T, and P is the metric matrix corresponding to a point on the man-

ifold. Extending this to QKV, ww^T is generalized to $w_Q w_K^T$, and $P^{1/2}$ and $P^{-1/2}$ are the X matrices in Attention. Thus, the interpolation formula becomes $Exp_{(P,\lambda)}(T) = \exp(Xw_Q w_K^T X)X^{-1}$, completing the interpolation and transforming it to the V space.