

Dynamic Positional Attention Modulation for Parameter-Efficient Fine-Tuning of Large Language Models

Dayan Pan

School of Computer Science and
Engineering,
MOE Engineering Research Center of
Advanced Computer Application
Technology,
Beihang University
Beijing, China
dayan@buaa.edu.cn

Jingyuan Wang*

School of Computer Science and
Engineering,
School of Economics and
Management,
Beihang University
Beijing, China
jywang@buaa.edu.cn

Xie Yu

School of Computer Science and
Engineering,
MOE Engineering Research Center of
Advanced Computer Application
Technology,
Beihang University
Beijing, China
yuxie_scse@buaa.edu.cn

Abstract

Parameter-efficient fine-tuning (PEFT) has become a standard approach for adapting large language models to downstream tasks. However, most existing PEFT methods rely on uniform and static adaptations, without accounting for the structured heterogeneity of attention across dimensions, heads, layers, and input tokens. In practice, attention representations exhibit non-uniform behavior, and positional encoding mechanisms such as rotary positional embeddings (RoPE) induce dimension-dependent positional structure, making uniform adaptation suboptimal. In this work, we propose DyPAM (Dynamic Positional Attention Modulation), a PEFT method that adapts how positional information contributes to attention by operating directly on the query and key representations. DyPAM combines input-conditioned, dimension-wise modulation with head-wise and layer-wise structural modulation, performing fine-grained adaptation of positional attention aligned with the RoPE-induced structure without modifying the pretrained backbone. Extensive experiments on mathematical and commonsense reasoning benchmarks across multiple backbone models demonstrate that DyPAM consistently outperforms existing strong PEFT baselines. The code is available for reproducibility¹.

CCS Concepts

• Information systems → Data mining; • Computing methodologies → Artificial intelligence.

Keywords

Large Language Models, Parameter-Efficient Fine-Tuning, Positional Attention, Attention Modulation

ACM Reference Format:

Dayan Pan, Jingyuan Wang, and Xie Yu. 2026. Dynamic Positional Attention Modulation for Parameter-Efficient Fine-Tuning of Large Language Models. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery*

*Corresponding authors.

¹<https://github.com/Beihang-BIGSCity/DyPAM>



This work is licensed under a Creative Commons Attribution 4.0 International License. *KDD '26, Jeju Island, Republic of Korea*

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2259-2/2026/08

<https://doi.org/10.1145/3770855.3817911>

and Data Mining V.2 (KDD '26), August 09–13, 2026, Jeju Island, Republic of Korea. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3770855.3817911>

Resource Availability:

The source code of this paper has been made publicly available at <https://doi.org/10.5281/zenodo.20507186>.

1 Introduction

Large language models (LLMs) have been widely adopted across a broad range of real-world applications, including question answering, code generation, and mathematical reasoning [5, 31, 53, 54]. Despite their strong general-purpose capabilities, different application scenarios and downstream tasks often impose substantially different requirements on model behavior. Effectively adapting an LLM to diverse task-specific requirements is essential in practical deployment [24, 25, 56, 61].

Full-parameter fine-tuning for a specific task is effective but computationally and storage intensive, which limits its practicality in many settings. Parameter-efficient fine-tuning (PEFT) addresses this issue by updating only a small subset of parameters while keeping the backbone model frozen, achieving a favorable balance between efficiency and performance [14, 26, 32, 35]. Representative PEFT methods include LoRA[16], which reparameterizes weight updates into a low-rank subspace, IA³ [28], which modulates intermediate activations via lightweight scaling vectors, BOFT [29], which applies orthogonality-constrained multiplicative transformations with butterfly factorization, and Bone [21], which introduces block-wise affine updates with shared parameters.

Despite their strong empirical performance, most existing PEFT methods apply additional parameters in a largely uniform manner across layers, attention heads, and feature dimensions (*i.e.*, individual dimensions of the hidden representations), and therefore lack mechanisms to account for the structured and heterogeneous roles of different model components in a fine-grained manner.

Figure 1 visualizes attention activation patterns of Llama-3.2-3B, focusing on the query (Q) representations used in self-attention. Similar heterogeneous patterns are also observed for key (K) representations, while such patterns are not significant for value (V) representations. Activations vary substantially across query dimensions, attention heads, and layers. As shown in Figure 1(a), attention exhibits distinct activation patterns across dimensions at different

layers, indicating that different dimensions contribute differently to attention. Figure 1(b) further shows clear head-wise variation within the same layer, indicating that different attention heads exhibit distinct activation patterns. Figures 1(c) and (d) show that activation patterns also depend on the input tokens. Tokens with different semantic roles induce systematically different activation distributions, both when aggregated across dimensions and when examined at the level of individual dimensions for a fixed head and layer. Together, these visualizations demonstrate that attention in pretrained LLMs operates in a structured and heterogeneous manner across dimensions, heads, layers, and input tokens.

Such heterogeneous activation patterns are consistent with the fact that different components of LLMs play distinct functional roles [58]. Recent studies have shown that Transformer modules are not functionally uniform, with different layers and components specializing in different aspects [11]. For example, feed-forward networks have been shown to primarily store factual information [33], while attention mechanisms are more closely associated with contextual interaction [7]. Within the attention module, architectural design choices further introduce structured differences in how information is processed. In particular, positional encoding mechanisms in Transformer-based LLMs introduce structured differences in internal representations, resulting in non-uniform usage of attention dimensions across layers and heads [2, 20]. Beyond structural differences across model components, attention behavior also varies with the input, where different tokens can induce different activation patterns depending on their semantic roles and contextual requirements [18, 51]. Taken together, these findings suggest that LLMs exhibit intrinsic functional heterogeneity across components and inputs, and effective adaptation methods should account for such fine-grained structural differences.

These observations suggest three requirements for structure-aware adaptation. First, adaptation should distinguish attention dimensions, since different dimensions exhibit different positional and functional behaviors. Second, it should account for head-wise and layer-wise differences, because the same dimension can play different roles across attention heads and network depths. Third, adaptation should remain input-conditioned, as different tokens and contexts can induce different attention activation patterns. This provides a direct way to adapt dependency modeling while keeping the original attention architecture unchanged.

Motivated by these observations, we propose **DyPAM** (Dynamic Positional Attention Modulation), a PEFT method that adapts how positional information contributes to attention. DyPAM addresses attention heterogeneity from two complementary perspectives. It combines input-conditioned, dimension-wise modulation with head-wise and layer-wise structural modulation to adapt attention behavior. In contrast to prior PEFT methods that apply static adaptation shared across inputs, DyPAM models heterogeneity jointly at the token, dimension, head, and layer levels, and aligns the modulation with the RoPE-induced dimension pairing. Extensive experiments across multiple models and tasks demonstrate the effectiveness of this design.

We summarize the main contributions of this work as follows:

- We propose **DyPAM**, the first parameter-efficient fine-tuning method that explicitly adapts large language models by modulating positional attention in a fine-grained and structured manner.

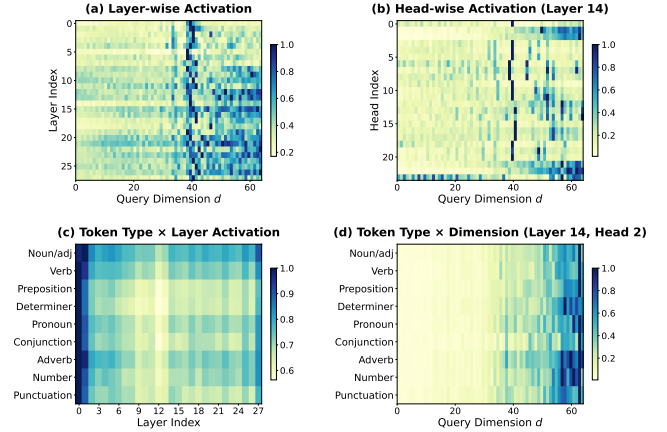


Figure 1: Activation heterogeneity in a pretrained Llama-3.2-3B model. The x-axis in (a), (b), and (d) indexes query dimensions of attention mechanism. Activation patterns vary across layers (a), heads (b), and input token types (c, d), indicating that attention operates heterogeneously across dimensions, heads, layers, and tokens.

- DyPAM introduces *input-conditioned, dimension-wise modulation*, enabling different attention dimensions to be dynamically adjusted according to the input context.
- DyPAM further incorporates *head-wise and layer-wise structural modulation*, allowing different attention heads and network layers to maintain distinct positional preferences.
- Extensive experiments across multiple backbone models and downstream tasks demonstrate the effectiveness and robustness of DyPAM compared to existing PEFT methods.

2 Preliminaries

We briefly review the attention mechanism, Rotary Position Embedding (RoPE), and parameter-efficient fine-tuning (PEFT), which provide the necessary background for describing our method.

2.1 Attention in Large Language Models

Large language models are typically built upon the Transformer architecture [48], where self-attention is the core mechanism for modeling interactions among tokens. Given an input sequence, each layer ℓ of the model produces a sequence of hidden states $\mathbf{H}^{(\ell)} = [\mathbf{h}_1^{(\ell)}, \mathbf{h}_2^{(\ell)}, \dots, \mathbf{h}_T^{(\ell)}]$, where $\mathbf{h}_t^{(\ell)} \in \mathbb{R}^d$ denotes the hidden representation of the t -th token. In each attention layer, the hidden states are linearly projected into query, key, and value representations,

$$\mathbf{Q}^{(\ell)} = \mathbf{H}^{(\ell)} \mathbf{W}_Q^{(\ell)}, \quad \mathbf{K}^{(\ell)} = \mathbf{H}^{(\ell)} \mathbf{W}_K^{(\ell)}, \quad \mathbf{V}^{(\ell)} = \mathbf{H}^{(\ell)} \mathbf{W}_V^{(\ell)}, \quad (1)$$

where $\mathbf{W}_Q^{(\ell)}, \mathbf{W}_K^{(\ell)}, \mathbf{W}_V^{(\ell)} \in \mathbb{R}^{d \times d}$ are learned projection matrices.

The projected representations are then reshaped into H attention heads. For each head $h \in \{1, \dots, H\}$, we denote the per-head matrices as $\mathbf{Q}^{(\ell, h)}, \mathbf{K}^{(\ell, h)}, \mathbf{V}^{(\ell, h)} \in \mathbb{R}^{T \times d_{\text{head}}}$, where $d_{\text{head}} = d/H$. For a given token position t , we denote by $\mathbf{q}_t^{(\ell, h)} \in \mathbb{R}^{d_{\text{head}}}$ and $\mathbf{k}_t^{(\ell, h)} \in \mathbb{R}^{d_{\text{head}}}$ the query and key vectors corresponding to the t -th

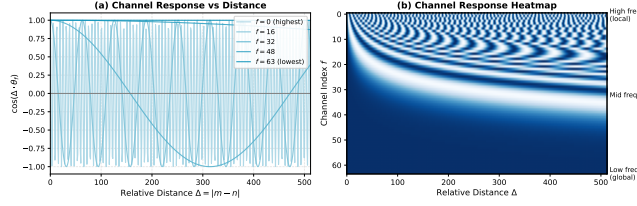


Figure 2: Position-dependent responses across attention dimensions induced by RoPE. (a) Different dimensions respond differently to relative positional distances. (b) Heatmap of all dimensions, showing non-uniform positional sensitivity.

token. We further denote by $q_{t,j}^{(\ell,h)}$ the j -th feature dimension of $\mathbf{q}_t^{(\ell,h)}$. Self-attention is computed independently for each head. For head h at layer ℓ , the attention output is given by

$$\text{Attn}(\mathbf{Q}^{(\ell,h)}, \mathbf{K}^{(\ell,h)}, \mathbf{V}^{(\ell,h)}) = \text{softmax}\left(\frac{\mathbf{Q}^{(\ell,h)} \mathbf{K}^{(\ell,h)\top}}{\sqrt{d_{\text{head}}}}\right) \mathbf{V}^{(\ell,h)}. \quad (2)$$

Notably, the attention operation itself is permutation-invariant and does not encode positional order [38]. As a result, positional information must be explicitly incorporated into the attention computation. Modern LLMs achieve this by applying position-dependent transformations to the query and key representations.

2.2 Rotary Position Embedding

Rotary Position Embedding (RoPE) [45] incorporates positional information into attention by applying position-dependent transformations to the query and key representations. For each attention head, RoPE views the per-head vector as two halves of equal size: a “real” part and an “imaginary” part, each of dimension $d_{\text{head}}/2$. RoPE then applies a 2D rotation to each index i by jointly rotating the paired components $(z_i^{\text{real}}, z_i^{\text{imag}})$. As a consequence, the two halves share the same rotation at each index, and thus exhibit closely related positional behavior across corresponding dimensions.

The rotations vary across dimensions, resulting in different positional behaviors. Figure 2 visualizes this effect, showing that different attention dimensions respond differently to relative positional distances in a non-uniform and multi-scale manner. As a result, different attention dimensions play different roles in encoding positional information, which motivates treating them differently when adapting positional attention.

2.3 Parameter-Efficient Fine-Tuning

Parameter-efficient fine-tuning (PEFT) adapts LLMs by introducing lightweight transformations, while keeping the pretrained backbone fixed. We describe PEFT by applying a constrained transformation to an intermediate representation $\mathbf{z}$ in the model. Formally, the adapted representation can be written as

$$\mathbf{z}' = \mathcal{T}(\mathbf{z}; \theta), \quad (3)$$

where $\mathcal{T}(\cdot; \theta)$ denotes a parameter-efficient transformation with a small number of trainable parameters θ . Many existing PEFT methods adopt an additive update,

$$\mathbf{z}' = \mathbf{z} + \Delta\mathbf{z}, \quad \Delta\mathbf{z} = \mathcal{A}(\mathbf{z}; \theta), \quad (4)$$

where the adaptation module $\mathcal{A}(\cdot)$ is typically applied in a static and largely uniform manner across model components.

While effective, this formulation typically applies the same adaptation mechanism uniformly across layers, attention heads, feature dimensions, and input tokens. However, attention behavior in modern LLMs is highly heterogeneous, particularly in how positional information is encoded. This motivates PEFT approaches that perform fine-grained and structure-aware adaptation.

In this work, we follow the PEFT paradigm and apply multiplicative modulation to attention representations:

$$\mathbf{z}' = \mathbf{s}(\mathbf{x}) \odot \mathbf{z}, \quad \mathbf{s}(\mathbf{x}) = \mathcal{M}(\mathbf{x}; \theta), \quad (5)$$

where $\mathbf{s}(\mathbf{x})$ is an input-conditioned modulation signal and $\odot$ denotes element-wise multiplication. In DyPAM, this modulation is applied to the query and key representations in attention and indexed by the internal structure of attention, including layers, heads, tokens, and feature dimensions. Both formulations follow the same parameter-efficient adaptation paradigm. The key difference is that DyPAM performs input-conditioned, structure-aware modulation, enabling fine-grained adaptation of positional attention.

3 Method

In this section, we introduce DyPAM, a PEFT method for LLMs that explicitly modulates positional information in attention. We first present an overview of the DyPAM framework and its core design principles, and then describe each component in detail.

3.1 Framework Overview

In RoPE-based LLMs, attention exhibits heterogeneous behavior across feature dimensions, layers, heads, and input tokens. However, most existing PEFT methods rely on uniform and static adaptation mechanisms, without accounting for such heterogeneity.

To address this limitation, we propose **Dynamic Positional Attention Modulation (DyPAM)**, a parameter-efficient fine-tuning method that adapts how positional information contributes to attention. DyPAM operates directly on the query and key representations and jointly models input-conditioned, dimension-wise modulation together with head-wise and layer-wise structural modulation, enabling fine-grained and structured adaptation of positional attention. As illustrated in Figure 3, DyPAM introduces modulation into attention representations in a manner aligned with the internal structure of attention and conditioned on the input. The two modulation sources play different roles. The input-conditioned component captures token-dependent variation by generating modulation features from hidden states. The structural bias terms capture persistent preferences associated with heads and layers. Their combination allows DyPAM to adapt to each input while maintaining stable structural differences across the attention hierarchy.

In the following, we present DyPAM by detailing how the modulation is constructed and how it is applied to attention representations, and summarize the overall procedure in Algorithm 1.

3.2 Query–Key Representations and Modulation Features

DyPAM operates on the query and key representations used in self-attention. At each Transformer layer ℓ , these representations are

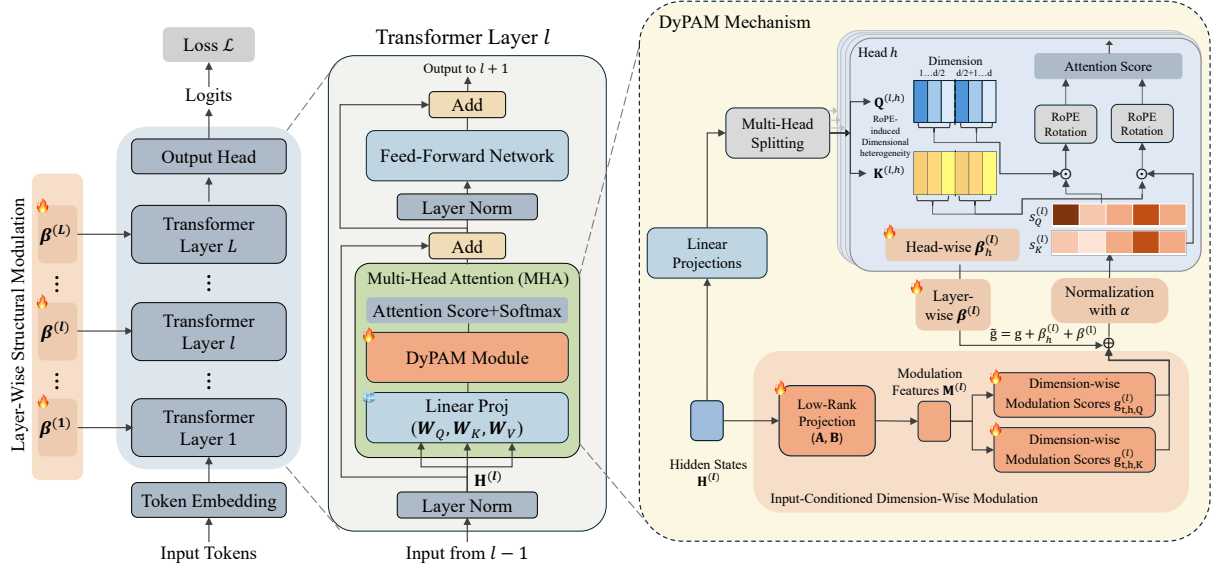


Figure 3: The architecture of DyPAM framework. DyPAM applies input-conditioned, dimension-wise modulation together with head-wise and layer-wise structural biases to the query and key representations before RoPE, enabling fine-grained adaptation of positional attention within the PEFT paradigm.

derived from the token-level hidden states $H^{(l)} \in \mathbb{R}^{B \times T \times d}$, where B is the batch size, T is the sequence length, and d is the model dimension. Following the standard attention formulation described in Section 2.1, the hidden states are linearly projected to obtain the query and key matrices $Q^{(l)}$ and $K^{(l)}$, which are subsequently reshaped into per-head representations $Q^{(l,h)}$ and $K^{(l,h)}$ as defined in Eq. (1).

To enable input-conditioned adaptation of attention behavior, DyPAM derives modulation features directly from the same hidden states $H^{(l)}$. Since the hidden states encode token-specific contextual information, the resulting modulation features are token-dependent and differ across inputs, providing the basis for input-conditioned modulation. Concretely, we apply a lightweight low-rank projection to the hidden states, yielding modulation features:

$$M^{(l)} = H^{(l)} A^{(l)} B^{(l)}, \quad M^{(l)} \in \mathbb{R}^{B \times T \times (H \cdot d_e)}, \quad (6)$$

where $A^{(l)} \in \mathbb{R}^{d \times r}$ and $B^{(l)} \in \mathbb{R}^{r \times (H \cdot d_e)}$ are learnable matrices with rank $r \ll d$, H is the number of attention heads, and d_e denotes the feature dimension per head.

The projected features are reshaped into H head-specific components, yielding modulation features $m_{t,h}^{(l)} \in \mathbb{R}^{d_e}$ for each token position t and attention head h . These features encode contextual information associated with each token and attention head, capturing how the current input is represented at different heads within the layer. They serve as an intermediate representation that bridges the token-level hidden states and the dimension-wise modulation subsequently applied to the query and key representations.

3.3 Input-Conditioned Dimension-Wise Modulation

Given the modulation features constructed from the hidden states, DyPAM maps them to dimension-wise modulation values that are aligned with the query and key representations in attention. This mapping determines how the contribution of each attention dimension is modulated in an input-conditioned manner, allowing the model to dynamically adjust the influence of each dimension. By conditioning on both the token and its context, DyPAM enables fine-grained control over how positional information is utilized across different attention dimensions.

For each layer l , DyPAM introduces learnable dimension embedding matrices that project modulation features to the attention dimension space. Concretely, for the query and key representations, we use separate embedding matrices

$$E_Q^{(l)} \in \mathbb{R}^{\frac{d_{\text{head}}}{2} \times d_e}, \quad E_K^{(l)} \in \mathbb{R}^{\frac{d_{\text{head}}}{2} \times d_e}, \quad (7)$$

where each row corresponds to one pair of attention dimensions. This design reflects the structure induced by RoPE, where each pair of dimensions shares the same positional rotation and therefore exhibits similar positional behavior. By assigning a single modulation value to each dimension pair, DyPAM reduces parameter overhead while respecting the RoPE-induced structure. This pairwise sharing also avoids assigning unrelated modulation values to two coordinates that jointly represent the same rotated subspace. In this way, DyPAM changes the strength of each RoPE-aligned channel without breaking the internal pairing used by the positional transformation. This also supports grouped-query attention (GQA) [1], where multiple attention heads share key and value projections. In such cases, the key-side modulation is shared across

heads that use the same key representation, while the query-side modulation remains head-specific.

Given the modulation feature $\mathbf{m}_{t,h}^{(\ell)} \in \mathbb{R}^{d_e}$ for token position t and attention head h , the dimension-wise modulation scores for queries and keys are computed as

$$\mathbf{g}_{t,h,Q}^{(\ell)} = \mathbf{E}_Q^{(\ell)} \mathbf{m}_{t,h}^{(\ell)}, \quad \mathbf{g}_{t,h,K}^{(\ell)} = \mathbf{E}_K^{(\ell)} \mathbf{m}_{t,h}^{(\ell)}, \quad (8)$$

where $\mathbf{g}_{t,h,Q}^{(\ell)}, \mathbf{g}_{t,h,K}^{(\ell)} \in \mathbb{R}^{\frac{d_{\text{head}}}{2}}$ denote the modulation scores for query and key dimension pairs, respectively. Each entry corresponds to one RoPE dimension pair, and we use i to index dimension pair i .

At this stage, the modulation scores provide input-conditioned, dimension-wise adjustments for the query and key representations.

Discussion. Input-conditioned dimension-wise modulation enables DyPAM to adapt the contribution of individual attention dimensions based on the input context. By aligning modulation with RoPE-induced dimension pairing, DyPAM selectively adjusts how positional information influences attention, while maintaining parameter efficiency. This mechanism provides fine-grained control over positional attention that is sensitive to both token-level context and the structured organization of attention dimensions.

3.4 Head-Wise and Layer-Wise Structural Modulation

While input-conditioned dimension-wise modulation captures token-dependent variation, attention behavior also exhibits differences across attention heads and network layers. To model such structure-level heterogeneity, DyPAM introduces head-wise and layer-wise structural modulation that is independent of the input.

For each layer ℓ , DyPAM maintains layer-wise bias vectors

$$\boldsymbol{\beta}_Q^{(\ell)}, \boldsymbol{\beta}_K^{(\ell)} \in \mathbb{R}^{\frac{d_{\text{head}}}{2}}, \quad (9)$$

which capture layer-specific preferences over attention dimension pairs for the query and key representations. In addition, for each attention head h at layer ℓ , DyPAM introduces head-wise bias vectors

$$\boldsymbol{\beta}_{h,Q}^{(\ell)}, \boldsymbol{\beta}_{h,K}^{(\ell)} \in \mathbb{R}^{\frac{d_{\text{head}}}{2}}, \quad (10)$$

allowing different heads within the same layer to maintain distinct structural biases. Here the subscripts Q and K indicate that the query and key representations use separate bias vectors of the same shape. These bias terms are added to the dimension-wise modulation scores. For queries and keys, the structurally augmented modulation scores are given by

$$\tilde{\mathbf{g}}_{t,h,Q}^{(\ell)} = \mathbf{g}_{t,h,Q}^{(\ell)} + \boldsymbol{\beta}_{h,Q}^{(\ell)} + \boldsymbol{\beta}_Q^{(\ell)}, \quad (11)$$

$$\tilde{\mathbf{g}}_{t,h,K}^{(\ell)} = \mathbf{g}_{t,h,K}^{(\ell)} + \boldsymbol{\beta}_{h,K}^{(\ell)} + \boldsymbol{\beta}_K^{(\ell)}, \quad (12)$$

where $\mathbf{g}_{t,h,Q}^{(\ell)}$ and $\mathbf{g}_{t,h,K}^{(\ell)}$ are the input-conditioned dimension-wise scores from Section 3.3. The bias terms are shared across token positions and encode structural preferences that persist across inputs.

At this stage, the modulation scores integrate input-conditioned, dimension-wise adjustments with head-wise and layer-wise structural biases, capturing both token-dependent variation and persistent structural preferences in attention. These scores are then transformed into bounded modulation factors and applied to the query and key representations.

Algorithm 1 DyPAM: Dynamic Positional Attention Modulation

Input: Input sequence $\mathbf{x} = (x_1, \dots, x_T)$, pretrained RoPE-based LLM, DyPAM parameters

Output: Model output distribution and training loss $\mathcal{L}$

```

1: Obtain token embeddings from  $\mathbf{x}$ 
2: for each Transformer layer  $\ell = 1, \dots, L$  do
3:   Compute hidden states  $\mathbf{H}^{(\ell)}$ 
4:   Project hidden states to query and key representations
5:    $\mathbf{Q}^{(\ell)}, \mathbf{K}^{(\ell)}$  according to Eq. (1)
6:   Reshape  $\mathbf{Q}^{(\ell)}, \mathbf{K}^{(\ell)}$  into per-head representations
7:    $\mathbf{Q}^{(\ell,h)}, \mathbf{K}^{(\ell,h)}$ 
8:   Construct modulation features from hidden states
9:    $\mathbf{m}_{t,h}^{(\ell)}$  according to Eq. (6)
10:  Compute input-conditioned, dimension-wise modulation
    scores
11:   $\mathbf{g}_{t,h,Q}^{(\ell)}, \mathbf{g}_{t,h,K}^{(\ell)}$  according to Eq. (8)
12:  Add head-wise and layer-wise structural biases
13:   $\tilde{\mathbf{g}}_{t,h,Q}^{(\ell)}, \tilde{\mathbf{g}}_{t,h,K}^{(\ell)}$  according to Eq. (11) and (12)
14:  Normalize modulation scores to obtain modulation factors
15:   $s_{t,h,i}^{(\ell)}$  according to Eq. (13)
16:  Apply modulation to query and key representations
17:   $\hat{\mathbf{Q}}^{(\ell,h)}, \hat{\mathbf{K}}^{(\ell,h)}$  according to Eq. (14)
18:  Apply RoPE to modulated query and key representations
19:  Compute attention outputs using modulated queries and
    keys
20: end for
21: Compute model outputs and training loss  $\mathcal{L}$  using Eq. (15)

```

3.5 Applying Modulation in Attention

The combined modulation scores obtained from the previous steps encode both input-conditioned and structural adjustments over attention dimensions. We next apply a normalization step that maps these scores to bounded modulation factors, ensuring stable and controlled adaptation.

Here, $\tilde{g}_{t,h,i}^{(\ell)}$ denotes the modulation score for dimension pair i in the combined score vector for the query or key representation at layer ℓ , head h , and token position t . For brevity, we omit the explicit Q/K subscript when the formulation applies identically to both. For each layer ℓ , token position t , attention head h , and dimension pair i , the normalized modulation factor is computed as

$$s_{t,h,i}^{(\ell)} = 1 + \alpha \cdot (\sigma(\tilde{g}_{t,h,i}^{(\ell)}) - 0.5), \quad (13)$$

where $\sigma(\cdot)$ is the sigmoid function and α controls the modulation strength. This normalization maps the modulation factors to a bounded interval $[1 - \frac{\alpha}{2}, 1 + \frac{\alpha}{2}]$, centering them around the original scale and preventing deviation from the pretrained representations.

The modulation factors are applied to the query and key representations before positional encoding. Let $\mathbf{q}_{t,i}^{(\ell,h)} \in \mathbb{R}^2$ and $\mathbf{k}_{t,i}^{(\ell,h)} \in \mathbb{R}^2$ denote the paired dimensions of the query and key vectors corresponding to dimension pair i . Both dimensions within each pair are modulated using the same factor:

$$\hat{\mathbf{q}}_{t,i}^{(\ell,h)} = s_{t,h,i}^{(\ell)} \cdot \mathbf{q}_{t,i}^{(\ell,h)}, \quad \hat{\mathbf{k}}_{t,i}^{(\ell,h)} = s_{t,h,i}^{(\ell)} \cdot \mathbf{k}_{t,i}^{(\ell,h)}. \quad (14)$$

This operation corresponds to the multiplicative PEFT formulation introduced in Eq. (5), where the pretrained representation $\mathbf{z}$

corresponds to the query or key vectors, and the modulation signal $\mathbf{s}(\mathbf{x})$ is indexed by layer, head, token position, and dimension pair. Following the convention above, $s_{t,h,i}^{(\ell)}$ is computed independently for the query and key sides from their respective scores $\tilde{\mathbf{g}}_{t,h,Q}^{(\ell)}$ and $\tilde{\mathbf{g}}_{t,h,K}^{(\ell)}$, so the query and key representations are modulated by their own factors. As the two coordinates within each pair share the same factor, the modulation signal $\mathbf{s}(\mathbf{x})$ takes a common value over each pair and thus spans the full d_{head} dimensions of the query and key vectors. The modulated query and key representations are then passed through the RoPE mechanism and used in the standard attention computation. By applying modulation prior to RoPE, DyPAM aligns adaptation with the RoPE-induced positional structure.

More broadly, DyPAM addresses attention heterogeneity by jointly modeling input-conditioned, dimension-wise modulation and head-wise, layer-wise structural modulation. This design enables attention dimensions to adapt to input context, while allowing different heads and layers to maintain distinct positional preferences. Rather than introducing uniform parameter updates, DyPAM performs targeted modulation aligned with how positional information is encoded and utilized in attention. As a result, DyPAM provides a structured and fine-grained mechanism for adapting positional attention within the PEFT paradigm.

3.6 Training Details

DyPAM is trained end-to-end using the standard cross-entropy loss for language modeling. Given an input sequence $\mathbf{x} = (x_1, \dots, x_T)$ and the corresponding target sequence $\mathbf{y} = (y_1, \dots, y_T)$, the training loss is defined as

$$\mathcal{L} = - \sum_{t=1}^T \log p(y_t | x_{\leq t}), \quad (15)$$

where $p(y_t | x_{\leq t})$ denotes the model output distribution at position t . The model is parameterized by the pretrained backbone together with the DyPAM parameters. Algorithm 1 summarizes the overall forward computation and training procedure.

4 Experiments

This section evaluates DyPAM across tasks, models, and experimental settings. Our experimental design is organized around a set of research questions that assess overall performance, scalability across model sizes, component contributions, and learned positional modulation behavior.

Specifically, we investigate the following research questions:

- **RQ1:** Does DyPAM outperform existing PEFT methods with comparable numbers of trainable parameters?
- **RQ2:** Is DyPAM effective across different backbone model sizes?
- **RQ3:** Which components of DyPAM contribute most to its performance?
- **RQ4:** How sensitive is DyPAM to its hyperparameters?
- **RQ5:** What positional modulation patterns does DyPAM learn?

We first introduce the experimental setup, followed by a systematic evaluation of DyPAM with respect to each research question.

4.1 Experimental Setup

Datasets. We evaluate DyPAM on mathematical and commonsense reasoning tasks with distinct reasoning patterns and positional sensitivity, and train on two datasets drawn from multiple existing benchmarks [17], emphasizing multi-step arithmetic and general commonsense reasoning, respectively. Mathematical reasoning performance is evaluated on GSM8K [9], AQuA [27], MAWPS [23], AddSub [15], MultiArith [42], SingleEq [22], and SVAMP [37], while commonsense reasoning is evaluated on BoolQ [6], PIQA [4], Social IQA [44], ARC-Easy and ARC-Challenge [8], OpenBookQA [34], HellaSwag [57], and WinoGrande [43]. Across benchmarks, inputs are natural-language questions and outputs are either final numeric answers or discrete choices, and we report accuracy as the evaluation metric with further details provided in the Appendix.

Backbone Models. Experiments are conducted on three widely used RoPE-based LLM families with different architectures and design choices, namely **LLaMA 3.2**, **Qwen3**, and **Gemma 3** [12, 46, 47], to evaluate the robustness and generality of DyPAM.

Baseline Methods. We compare DyPAM with a diverse set of PEFT methods that adopt different adaptation strategies. Low-rank adaptation methods include LoRA [16] and AdaLoRA [58], which parameterize weight updates in a low-dimensional subspace, with AdaLoRA further adjusting rank allocation across layers according to their relative importance. We also consider structured weight reparameterization approaches, including OFT [39] and Bone [21], where OFT constrains updates to orthogonal transformations and Bone employs block-wise affine parameterization to capture structured correlations within weight matrices. In addition, we include lightweight modulation-based methods such as IA³ [28] and LNTuning [60], which adapt the model by rescaling internal activations or normalization parameters with minimal trainable parameters. We further compare against FourierFT [10], which performs adaptation in the frequency domain by learning a compact set of spectral coefficients for weight updates. Finally, we include SHiRA [3], which applies sparse high-rank adapters to update only a small subset of backbone weights, and RoSA [36], which performs RoPE-aware selective adaptation over attention dimensions and layers.

Implementation Details. All experiments are conducted using DeepSpeed [41] with bfloat16 precision on NVIDIA RTX 4090 GPUs. For DyPAM, we use a modulation embedding dimension $d_e = 64$, a low-rank projection rank $r = 128$, and a modulation strength $\alpha = 0.3$. For baseline methods, except for extremely low-parameter approaches, we match the number of trainable parameters to a comparable scale. Additional implementation details are provided in the appendix, and all code and data are released to ease reproducibility².

4.2 Overall Performance (RQ1)

Across both mathematical and commonsense reasoning benchmarks, DyPAM consistently outperforms existing PEFT baselines under comparable parameters, providing a clear positive answer to **RQ1**. The improvements are observed across all evaluated backbone models and task categories, indicating that the effectiveness of DyPAM generalizes across different models and datasets.

²<https://github.com/Beihang-BIGSCity/DyPAM>

Table 1: Comparison of DyPAM with PEFT baselines on mathematical reasoning benchmarks across three backbone models. Micro-avg and macro-avg denote micro- and macro-averaged performance. Best results are highlighted in bold, and second-best results are underlined. * indicates statistically significant improvements over the best baseline (two-sided t-test, $p < 0.05$).

Backbone LLM	Method	Param(%)	MultiArith	GSM8K	AddSub	AQuA	SingleEq	SVAMP	MAWPS	micro-avg(%)↑	macro-avg(%)↑
LLaMA 3.2 3B	LoRA	1.12	71.50	33.21	78.48	22.44	81.50	54.10	76.47	54.96	59.67
	AdaLoRA	2.22	75.67	36.32	80.51	22.83	87.80	55.60	78.57	57.90	62.47
	OFT	0.73	87.17	40.18	<u>85.82</u>	<u>24.02</u>	86.42	61.50	84.03	62.75	<u>67.02</u>
	Bone	1.14	<u>87.50</u>	39.73	85.57	23.62	86.61	63.70	81.93	<u>63.03</u>	66.95
	IA ³	0.02	58.33	27.37	68.61	20.47	72.83	47.90	58.82	46.89	50.62
	LN-Tuning	0.01	58.00	26.38	66.58	21.26	74.80	44.90	60.08	46.01	50.29
	FourierFT	0.73	78.67	33.21	82.03	20.47	85.43	54.30	77.31	56.72	61.63
	SHiRA	1.12	82.50	38.82	84.81	<u>24.02</u>	<u>87.99</u>	56.90	81.93	60.59	65.28
	RoSA	0.54	84.33	37.91	82.78	22.83	87.01	52.50	78.99	59.02	63.77
	DyPAM (ours)	0.92	88.50	<u>39.88</u>	86.33	25.20	88.78	<u>63.00</u>	84.03	63.58*	67.96*
Qwen3 8B	LoRA	0.79	97.67	74.91	89.87	35.83	90.55	84.70	89.08	82.04	80.37
	AdaLoRA	1.57	95.17	73.01	90.63	37.01	92.32	84.80	91.60	81.62	80.65
	OFT	0.51	95.67	73.46	90.38	33.07	94.09	84.90	91.60	81.80	80.45
	Bone	0.81	<u>98.00</u>	72.25	<u>91.65</u>	33.46	93.90	83.80	90.34	81.55	80.49
	IA ³	0.02	92.50	72.18	84.81	35.04	86.61	80.90	86.55	78.49	76.94
	LN-Tuning	0.00	91.67	68.69	85.32	<u>39.76</u>	87.40	78.00	85.71	77.01	76.65
	FourierFT	0.37	94.50	70.05	87.34	31.50	86.81	82.70	81.09	78.28	76.28
	SHiRA	0.79	94.83	<u>75.36</u>	90.13	37.01	93.90	85.70	90.34	<u>82.57</u>	81.04
	RoSA	0.36	97.83	74.07	90.38	35.43	<u>94.49</u>	84.80	<u>92.02</u>	82.48	81.29
	DyPAM (ours)	0.61	99.17	76.72	91.90	40.94	95.28	<u>85.50</u>	92.86	84.24*	83.20*
Gemma 3 4B	LoRA	1.33	86.00	51.25	72.41	25.98	75.59	62.20	75.21	63.26	64.09
	AdaLoRA	2.62	82.67	51.86	66.33	31.50	73.82	62.30	73.95	62.49	63.20
	OFT	0.75	85.83	<u>54.28</u>	72.91	<u>32.28</u>	75.59	63.80	<u>76.47</u>	<u>65.02</u>	<u>65.88</u>
	Bone	1.41	<u>86.17</u>	45.87	71.39	30.31	72.64	55.10	73.11	59.69	62.08
	IA ³	0.03	42.67	38.89	40.51	27.17	40.75	37.20	37.39	38.62	37.80
	LN-Tuning	0.01	32.67	30.63	45.06	23.62	56.69	40.80	37.82	37.64	38.18
	FourierFT	1.10	60.83	31.24	65.32	28.35	66.73	46.30	65.97	47.89	52.10
	SHiRA	1.33	72.67	42.08	<u>73.16</u>	31.50	76.57	61.30	75.63	58.92	61.84
	RoSA	0.40	34.50	38.51	66.84	31.10	63.19	43.70	62.18	45.53	48.58
	DyPAM (ours)	0.62	86.33	55.19	73.42	32.68	<u>76.18</u>	<u>62.70</u>	76.89	65.28*	66.20*

Table 2: Macro-averaged accuracy on mathematical reasoning benchmarks across Qwen3 model scales. The comparison includes representative strong PEFT baselines from the main experiments.

Baseline	Qwen 3 0.6B	Qwen 3 1.7B	Qwen 3 4B	Qwen 3 8B
LoRA	64.06	66.64	75.60	80.37
OFT	<u>65.96</u>	<u>67.81</u>	75.54	80.45
SHiRA	63.95	64.65	73.33	81.04
RoSA	63.99	67.38	<u>77.92</u>	<u>81.29</u>
DyPAM (ours)	66.13	69.24	78.24	83.20

Specifically, on mathematical reasoning benchmarks as shown in Table 1, DyPAM demonstrates strong and stable gains across heterogeneous tasks that require multi-step computation, and numerical reasoning. The improvements are reflected in both micro- and macro-averaged metrics, suggesting that DyPAM enhances overall reasoning capability rather than favoring a small subset of benchmarks. In contrast, existing PEFT baselines exhibit more uneven behavior. Low-rank methods such as LoRA and AdaLoRA tend to yield limited gains, particularly on more challenging datasets. Frequency-domain approaches such as FourierFT show moderate performance but lack robustness across datasets. Methods with extremely small parameter budgets, including IA³ and LN-Tuning, generally underperform due to their limited adaptation capacity. Structured or orthogonal methods such as OFT and Bone often perform well on specific tasks but struggle to maintain consistent improvements

across the full benchmark suite, whereas sparse adaptation methods like SHiRA, which selectively activate a subset of parameters during fine-tuning, achieve strong performance on multiple tasks, highlighting the potential of structured parameter updates. RoSA also benefits from structured adaptation by introducing rotational subspace updates, leading to competitive results on several benchmarks. However, RoSA applies a static RoPE-aware selection that is shared across all inputs, whereas DyPAM modulates query and key dimension pairs in an input-conditioned, token-dependent manner coordinated with head-wise and layer-wise structure. This fine-grained and input-conditioned modulation, rather than the use of positional structure alone, accounts for the consistent margin of DyPAM over RoSA across backbones. Compared with these baselines, DyPAM achieves more balanced improvements across tasks and models, indicating superior robustness and generalization.

A similar pattern is observed on commonsense reasoning tasks as shown in Table 3. DyPAM yields balanced improvements across diverse datasets spanning factual verification and social commonsense. The consistent gains in macro-averaged performance indicate improved robustness at the task level, while the improvements in micro-averaged accuracy reflect stronger overall performance.

Overall, these results show that DyPAM provides a more reliable and general-purpose adaptation mechanism than prior PEFT approaches. By delivering consistent gains across tasks, domains, and backbone models, DyPAM effectively addresses **RQ1** and demonstrates clear advantages over existing PEFT baselines.

Table 3: Comparison of DyPAM with PEFT baselines on commonsense reasoning benchmarks across three backbone models. Micro-avg and macro-avg denote micro- and macro-averaged performance. Best results are highlighted in bold, and second-best results are underlined. * indicates statistically significant improvements over the best baseline (two-sided t-test, $p < 0.05$).

Backbone LLM	Method	Param(%)	BoolQ	PIQA	SocialIQA	ARC-C	ARC-E	OpenBookQA	HellaSwag	WinoGrande	Micro Avg	Macro Avg
LLaMA 3.2 3B	LoRA	1.12	63.61	<u>79.71</u>	66.94	69.45	84.05	67.00	73.94	55.56	71.94	70.03
	AdaLoRA	2.22	63.52	78.94	67.09	68.94	<u>85.14</u>	70.20	78.11	56.35	73.95	71.04
	OFT	0.73	<u>65.63</u>	79.54	<u>70.37</u>	<u>70.39</u>	<u>85.06</u>	71.80	83.15	66.38	<u>77.52</u>	<u>74.04</u>
	Bone	1.14	64.56	75.68	69.34	64.42	79.76	70.20	75.92	<u>65.75</u>	<u>72.77</u>	70.70
	IA ³	0.02	62.32	77.09	59.67	57.94	77.10	57.40	50.48	52.25	58.66	61.78
	LN Tuning	0.01	62.51	76.99	59.52	59.81	76.52	59.00	52.02	52.17	59.42	62.32
	FourierFT	0.73	62.14	79.49	61.98	61.86	80.93	62.40	73.21	49.09	69.75	66.39
	SHiRA	1.12	65.23	79.65	69.14	71.16	84.97	71.20	<u>83.18</u>	65.67	77.35	73.78
	RoSA	0.54	64.53	79.65	69.86	69.28	84.43	70.80	83.12	63.54	77.00	73.15
	DyPAM (ours)	0.92	65.93	79.76	70.88	<u>70.39</u>	85.19	71.80	83.71	65.35	77.83*	74.13*
Qwen3 8B	LoRA	0.79	70.49	86.34	77.18	90.19	96.51	87.60	89.50	72.85	85.19	83.83
	AdaLoRA	1.57	70.73	86.51	76.71	<u>90.36</u>	96.55	87.20	88.92	72.38	84.91	83.67
	OFT	0.51	69.97	86.83	76.56	89.93	<u>96.97</u>	88.00	89.17	76.48	85.20	84.24
	Bone	0.81	69.02	85.31	75.64	88.91	95.58	87.60	89.30	<u>76.56</u>	84.71	83.49
	IA ³	0.02	69.51	86.34	76.71	90.27	96.09	84.40	85.12	66.77	82.59	81.90
	LN Tuning	0.00	69.33	86.40	75.95	90.27	96.00	83.00	83.86	65.43	81.82	81.28
	FourierFT	0.37	69.54	84.49	73.13	85.92	95.29	77.80	80.48	62.27	79.34	78.62
	SHiRA	0.79	<u>70.83</u>	<u>87.05</u>	77.33	<u>90.36</u>	<u>96.97</u>	<u>88.20</u>	89.56	75.77	<u>85.57</u>	<u>84.51</u>
	RoSA	0.36	68.96	86.94	75.33	89.85	<u>96.38</u>	<u>88.20</u>	89.43	76.16	84.99	83.91
	DyPAM (ours)	0.61	70.89	87.11	77.33	90.53	97.05	88.80	<u>89.53</u>	76.80	85.66*	84.75*
Gemma 3 4B	LoRA	1.33	65.72	79.71	69.40	74.49	87.08	71.00	74.53	55.01	73.37	72.12
	AdaLoRA	2.62	<u>66.09</u>	79.49	68.73	76.54	89.02	74.00	73.20	58.09	73.30	73.14
	OFT	0.75	65.69	81.99	74.51	<u>76.71</u>	88.47	78.00	<u>83.86</u>	<u>65.27</u>	<u>79.17</u>	<u>76.81</u>
	Bone	1.41	64.68	75.35	71.24	70.39	82.83	75.80	78.33	64.48	74.70	72.89
	IA ³	0.02	62.17	71.49	57.32	57.51	73.19	55.20	44.89	57.85	55.30	59.95
	LN Tuning	0.00	62.60	66.70	49.85	49.91	63.59	45.20	47.29	60.46	53.90	55.70
	FourierFT	0.37	63.94	75.57	67.14	67.32	76.05	57.80	71.81	59.35	69.76	67.37
	SHiRA	0.79	65.57	<u>82.25</u>	<u>74.53</u>	76.19	89.71	<u>78.20</u>	83.19	64.48	78.94	76.77
	RoSA	0.40	63.70	<u>79.54</u>	<u>67.40</u>	72.27	86.66	69.40	48.53	47.51	60.62	66.88
	DyPAM (ours)	0.62	66.21	82.59	74.82	77.13	<u>89.23</u>	79.20	84.09	65.35	79.56*	77.33*

4.3 Scalability Analysis (RQ2)

Table 2 evaluates the scalability of DyPAM across Qwen3 models from 0.6B to 8B using macro-averaged accuracy on mathematical reasoning benchmarks. As model size increases, DyPAM consistently achieves better performance than the strongest PEFT baselines at each scale. Moreover, the performance difference between DyPAM and baselines becomes larger as the backbone grows, indicating that DyPAM continues to benefit from increased capacity.

Overall, these results demonstrate that DyPAM maintains strong scalability across model sizes and remains effective when applied to both small and large backbone models, directly addressing RQ2.

4.4 Ablation and Sensitivity Analysis (RQ3, 4)

We further analyze DyPAM through ablation studies and parameter sensitivity experiments, aiming to understand the contribution of individual components and the robustness of key hyperparameters.

The ablation results show that each core component of DyPAM plays a complementary role in overall performance. Removing any single component consistently leads to performance degradation, indicating that the gains of DyPAM arise from their joint design rather than isolated architectural choices. In addition, the sensitivity analysis on the modulation strength α demonstrates the effectiveness of this design choice. Proper modulation significantly improves performance compared to weak or overly strong modulation. Overall, these analyses confirm that the performance improvements of DyPAM are structurally grounded and robust.

4.5 Analysis of Modulation Patterns (RQ5)

We analyze the learned modulation from two perspectives: the structural bias terms and the effective modulation ranges.

Figure 5 visualizes the learned layer-wise query bias over attention dimensions on LLaMA-3.2-3B under commonsense and mathematical reasoning settings. Rather than exhibiting uniform shifts, the bias values vary across both layers and dimensions, indicating that DyPAM learns heterogeneous, dimension-specific adjustments. This structured non-uniformity suggests that different attention dimensions develop distinct positional preferences at different depths, aligning with the intuition that positional information is utilized differently across layers. In addition, faint horizontal band patterns can be observed, hinting that certain layers may exhibit consistent preferences over specific subsets of attention dimensions.

Figure 6 summarizes the effective modulation range of query and key representations under the same two settings. The scale factors remain centered around 1, while the range varies across layers and training data. This restrained behavior preserves the pretrained attention structure while allowing flexibility.

Overall, these visualizations demonstrate that DyPAM learns structured and stable positional modulation patterns. By combining dimension-wise modulation with restrained head-wise and layer-wise scaling, DyPAM adapts positional attention in a targeted manner, shaping how token dependencies are formed and thereby helping explain its consistent performance gains across models and tasks.

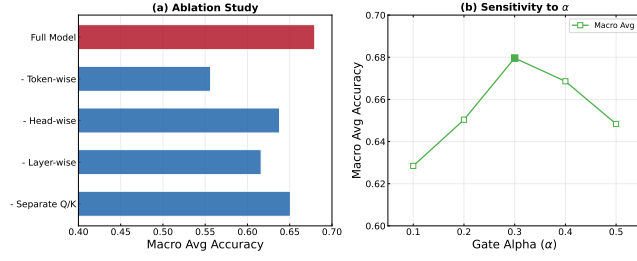


Figure 4: Ablation and hyperparameter sensitivity of DyPAM. The ablation removes individual modulation components, while the sensitivity study varies the modulation strength α .

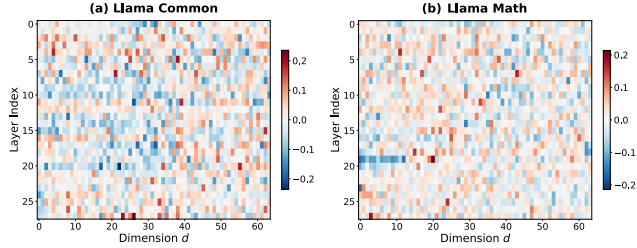


Figure 5: Learned layer-wise query bias over attention dimensions on LLaMA-3.2-3B under commonsense and mathematical reasoning settings. The heatmaps show that the structural bias varies across layers and dimensions rather than following a uniform pattern.

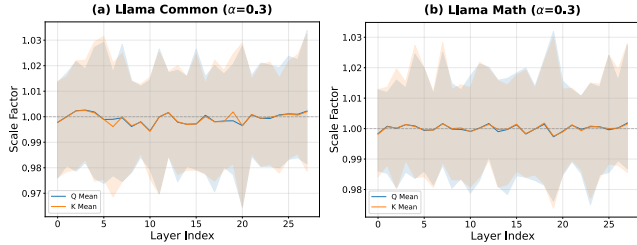


Figure 6: Layer-wise modulation range of query and key representations on LLaMA-3.2-3B under commonsense and mathematical reasoning settings. The mean scale remains close to 1, while the effective range changes across layers and training data.

5 Related Work

5.1 Parameter-Efficient Fine-Tuning

Recent advancements in PEFT focus on reducing the number of trainable parameters while maintaining performance. Low-rank approaches such as LoRA [16] and AdaLoRA [58] adapt pre-trained weights through additive low-rank updates, while scaling-based methods including IA³ [28] and LNTuning [60] employ lightweight gating to modulate activations under strict parameter constraints. Structured PEFT approaches, such as OFT [39] and Bone [21], further constrain the update space by enforcing geometric properties, whereas spectral-domain methods like FourierFT [10] perform

adaptation via frequency-based transformations of model representations. In addition, RoSA [36] is motivated by the structure of RoPE and proposes RoPE-aware selective adaptation by emphasizing low-frequency attention components. Compared with RoSA, which applies static RoPE-aware selection, DyPAM generates input-conditioned modulation over query and key dimension pairs. In contrast to these approaches, our method explicitly models fine-grained, input-conditioned modulation within attention, enabling structure-aware adaptation.

5.2 Structured Modeling in Transformer

Transformer attention has been recognized for its structural heterogeneity, with different attention heads specializing in various semantic or syntactic aspects of the input [40]. Recent works have analyzed head-level specialization and layer-wise differences, demonstrating that attention heads should not be treated uniformly [49, 59]. Beyond language modeling, attention- and frequency-aware designs have also been widely adopted in spatiotemporal modeling [19, 30] and financial prediction [50, 52], further underscoring the value of structure-aware mechanisms. Positional encodings, particularly in RoPE and its variants, are not just external biases but integral to the attention computation, providing a way to model relative positional relationships more effectively [13]. Newer methods, such as ComRoPE [55], introduce dynamic and input-conditioned positional encodings, enhancing the flexibility of position representations in transformers. RoSA [36] is also motivated by the structure of RoPE and explores selective adaptation of attention components guided by frequency characteristics. While these studies highlight the structured and positional nature of attention, they largely rely on static or coarse-grained mechanisms, motivating the need for fine-grained and input-conditioned modulation within attention.

6 Conclusion

In this work, we introduce DyPAM, a PEFT method that adapts LLMs by modulating positional attention in a fine-grained and structured manner. Motivated by heterogeneous attention behavior across dimensions, heads, layers, and input tokens, DyPAM operates directly on the query and key representations and aligns adaptation with the RoPE-induced positional structure. It jointly models input-conditioned, dimension-wise and head-wise, layer-wise structural modulation, providing a principled approach to adapting positional attention within the PEFT paradigm. By reshaping how positional information influences attention, DyPAM adjusts the way dependencies are formed across tokens, which is consistent with its gains on tasks that rely on multi-step and relational reasoning. Extensive experiments across mathematical and commonsense reasoning benchmarks demonstrate that DyPAM consistently outperforms strong PEFT baselines. Future work will further optimize the runtime implementation and extend DyPAM across diverse architectural mechanisms.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 725B2004, 72242101, 72222022), and the Science and Technology Development Fund Macau SAR (0052/2023/RIA1).

References

- [1] Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245* (2023).
- [2] Federico Barbero, Alex Vitvitskiy, Christos Perivolaropoulos, Razvan Pascanu, and Petar Veličković. 2024. Round and round we go! what makes rotary positional encodings useful? *arXiv preprint arXiv:2410.06205* (2024).
- [3] Kartikeya Bhardwaj, Nilesch Prasad Pandey, Sweta Priyadarshi, Viswanath Ganapathy, Shreya Kadambi, Rafael Esteves, Shubhankar Borse, Paul Whatmough, Risheek Garrepalli, Mart Van Baalen, Harris Teague, and Markus Nagel. 2024. Sparse high rank adapters. In *Proceedings of the 38th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) (NIPS '24). Curran Associates Inc., Red Hook, NY, USA, Article 438, 31 pages.
- [4] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 7432–7439.
- [5] Jiawei Cheng, Jingyuan Wang, Yichuan Zhang, Jiahao Ji, Yuanshao Zhu, Zhibo Zhang, and Xiangyu Zhao. 2025. Poi-enhancer: An llm-based semantic enhancement framework for poi representation learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 39. 11509–11517.
- [6] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044* (2019).
- [7] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341* (2019).
- [8] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457* (2018).
- [9] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168* (2021).
- [10] Ziqi Gao, Qichao Wang, Aochuan Chen, Zijiang Liu, Bingzhe Wu, Liang Chen, and Jia Li. 2024. Parameter-efficient fine-tuning with discrete fourier transform. *arXiv preprint arXiv:2405.03003* (2024).
- [11] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer Feed-Forward Layers Are Key-Value Memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 5484–5495.
- [12] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).
- [13] Zihan Gu, Han Zhang, Ruoyu Chen, Yue Hu, and Hua Zhang. 2025. Unpacking Positional Encoding in Transformers: A Spectral Analysis of Content-Position Coupling. *arXiv preprint arXiv:2505.13027* (2025).
- [14] Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608* (2024).
- [15] Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 523–533.
- [16] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).
- [17] Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933* (2023).
- [18] Jiahao Ji, Jingyuan Wang, Yu Mou, Cheng Long, and Junjie Wu. 2026. How to Break It Down for Building It Up? Theory-Guided Graph Decomposition Learning for Spatiotemporal Traffic Prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2026).
- [19] Jiawei Jiang, Chengkai Han, Wayne Xin Zhao, and Jingyuan Wang. 2023. Pdifformer: Propagation delay-aware dynamic long-range transformer for traffic flow prediction. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37. 4365–4373.
- [20] Mingyu Jin, Kai Mei, Wujiang Xu, Mingjie Sun, Ruixiang Tang, Mengnan Du, Zirui Liu, and Yongfeng Zhang. 2025. Massive Values in Self-Attention Modules are the Key to Contextual Knowledge Understanding. *arXiv preprint arXiv:2502.01563* (2025).
- [21] Jiale Kang and Qingyu Yin. 2024. Balancing LoRA Performance and Efficiency with Simple Shard Sharing. *arXiv preprint arXiv:2409.15371* (2024).
- [22] Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics* 3 (2015), 585–597.
- [23] Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies*. 1152–1157.
- [24] Junyi Li, Tianyi Tang, Zheng Gong, Lixin Yang, Zhuohao Yu, Zhipeng Chen, Jingyuan Wang, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Eliteplm: an empirical study on general language ability evaluation of pretrained language models. In *Proceedings of the 2022 conference of the North American chapter of the Association for Computational Linguistics: human language technologies*. 3519–3539.
- [25] Junyi Li, Tianyi Tang, Wayne Xin Zhao, Jingyuan Wang, Jian-Yun Nie, and Ji-Rong Wen. 2023. The web can be your oyster for improving language models. In *Findings of the Association for Computational Linguistics: ACL 2023*. 728–746.
- [26] Zimeng Li, Li Hongjun, Jingyuan Wang, and Ke Tang. 2025. Approximation to Smooth Functions by Low-Rank Swish Networks. In *Forty-second International Conference on Machine Learning*.
- [27] Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146* (2017).
- [28] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems* 35 (2022), 1950–1965.
- [29] Weiyang Liu, Zeju Qiu, Yao Feng, Yuliang Xiu, Yuxuan Xue, Longhui Yu, Haiwen Feng, Zhen Liu, Juyeon Heo, Songyou Peng, et al. 2023. Parameter-efficient orthogonal finetuning via butterfly factorization. *arXiv preprint arXiv:2311.06243* (2023).
- [30] Jingtian Ma, Jingyuan Wang, et al. 2026. Hierarchical Frequency-Decomposition Graph Neural Networks for Road Network Representation Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 40. 15510–15518.
- [31] Jingtian Ma, Jingyuan Wang, Wayne Xin Zhao, Guoping Liu, and Xiang Wen. 2025. Spatio-Temporal Data Enhanced Vision-Language Model for Traffic Scene Understanding. *IEEE Transactions on Intelligent Transportation Systems* (2025).
- [32] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods. <https://github.com/huggingface/peft>.
- [33] Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229* (2022).
- [34] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789* (2018).
- [35] Dayan Pan, Zhaoyang Fu, Jingyuan Wang, Xiao Han, Yue Zhu, and Xiangyu Zhao. 2025. Contextual Attention Modulation: Towards Efficient Multi-Task Adaptation in Large Language Models. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*. 2273–2283.
- [36] Dayan Pan, Jingyuan Wang, Yilong Zhou, Jiawei Cheng, Pengyue Jia, and Xiangyu Zhao. 2026. RoSA: Enhancing Parameter-Efficient Fine-Tuning via RoPE-aware Selective Adaptation in Large Language Models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 40. 15600–15608.
- [37] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191* (2021).
- [38] Ofir Press, Noah A Smith, and Mike Lewis. 2021. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409* (2021).
- [39] Zeju Qiu, Weiyang Liu, Haiwen Feng, Yuxuan Xue, Yao Feng, Zhen Liu, Dan Zhang, Adrian Weller, and Bernhard Schölkopf. 2023. Controlling text-to-image diffusion by orthogonal finetuning. *Advances in Neural Information Processing Systems* 36 (2023), 79320–79362.
- [40] Alessandro Raganato, Yves Scherrer, and Jörg Tiedemann. 2020. Fixed encoder self-attention patterns in transformer-based machine translation. *arXiv preprint arXiv:2002.10260* (2020).
- [41] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. DeepSpeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 3505–3506.
- [42] Subhro Roy and Dan Roth. 2016. Solving general arithmetic word problems. *arXiv preprint arXiv:1608.01413* (2016).
- [43] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 8732–8740.
- [44] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Socialliqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728* (2019).
- [45] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing* 568 (2024), 127063.
- [46] Gemma Team. 2025. Gemma 3. (2025). <https://go.gle/Gemma3Report>

- [47] Qwen Team. 2025. Qwen3 Technical Report. arXiv:2505.09388 [cs.CL]. <https://arxiv.org/abs/2505.09388>
- [48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [49] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418* (2019).
- [50] Jingyuan Wang, Ke Tang, Kai Feng, Xin Lin, Weifeng Lv, Kun Chen, and Fei Wang. 2021. Impact of temperature and relative humidity on the transmission of COVID-19: a modelling study in China and the United States. *BMJ open* 11, 2 (2021), e043863.
- [51] Jingyuan Wang, Chen Yang, Xiaohan Jiang, and Junjie Wu. 2023. WHEN: A Wavelet-DTW hybrid attention network for heterogeneous time series analysis. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*. 2361–2373.
- [52] Jingyuan Wang, Yang Zhang, Ke Tang, Junjie Wu, and Zhang Xiong. 2019. Alphastock: A buying-winners-and-selling-losers investment strategy using interpretable deep reinforcement attention networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1900–1908.
- [53] Xiaolei Wang, Xinyu Tang, Xin Zhao, Jingyuan Wang, and Ji-Rong Wen. 2023. Rethinking the evaluation for conversational recommendation in the era of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 10052–10065.
- [54] Zhongyuan Wu, Jingyuan Wang, Zexuan Cheng, Yilong Zhou, Weizhi Wang, Juhua Pu, Chao Li, and Changqing Ma. 2026. Icad-llm: One-for-all anomaly detection via in-context learning with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 40. 15986–15994.
- [55] Hao Yu, Tangyu Jiang, Shunning Jia, Shannan Yan, Shunning Liu, Haolong Qian, Guanghao Li, Shuting Dong, and Chun Yuan. 2025. ComRoPE: Scalable and Robust Rotary Position Embedding Parameterized by Trainable Commuting Angle Matrices. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 4508–4517.
- [56] Xie Yu, Jingyuan Wang, Yifan Yang, Qian Huang, and Ke Qu. 2025. BIGCity: A universal spatiotemporal model for unified trajectory and traffic state data analysis. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*. IEEE, 4455–4469.
- [57] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830* (2019).
- [58] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512* (2023).
- [59] Xiaofeng Zhang, Yikang Shen, Zeyu Huang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2022. Mixture of attention heads: Selecting attention heads per token. *arXiv preprint arXiv:2210.05144* (2022).
- [60] Bingchen Zhao, Haoqin Tu, Chen Wei, Jieru Mei, and Cihang Xie. 2023. Tuning layernorm in attention: Towards efficient multi-modal llm finetuning. *arXiv preprint arXiv:2312.11420* (2023).
- [61] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023).

A Attention Activation Patterns

A.1 Effect of Positional Encoding Schemes

To verify that the heterogeneous attention behavior motivating DyPAM is tied to the positional encoding mechanism, we compare query activation patterns across models using different positional encodings, including RoPE, ALiBi, and learned positional embeddings. Specifically, we visualize query representations from LLaMA-3.2-3B, BLOOM-560M, and OPT-350M under a unified scheme covering all attention dimensions (Figure 7). RoPE-based models exhibit structured, dimension-dependent activation patterns, whereas models using ALiBi or learned positional embeddings display substantially more homogeneous behavior across dimensions, supporting our focus on RoPE-induced positional structure.

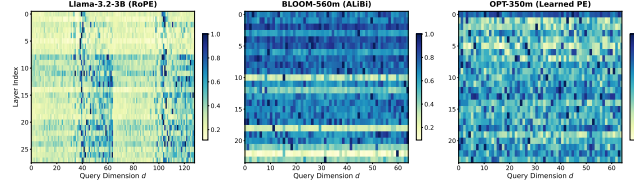


Figure 7: Cross-model comparison of query activation patterns across layers and attention dimensions. RoPE-based LLaMA-3.2-3B shows structured, dimension-dependent patterns, while ALiBi-based BLOOM-560M and OPT-350M with learned embeddings are more homogeneous.

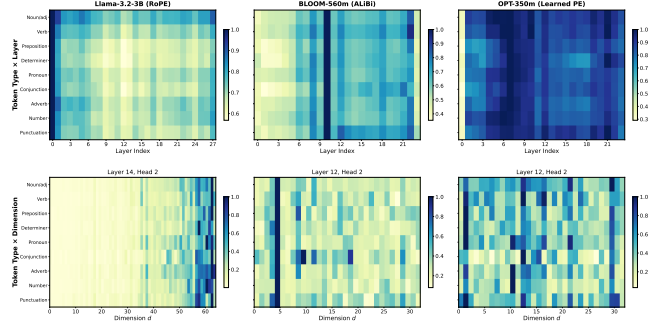


Figure 8: Token-type activation patterns across layers and attention dimensions for LLaMA-3.2-3B, BLOOM-560M, and OPT-350M. Token-dependent variation is most pronounced in the RoPE-based model.

A.2 Token-Dependent Activation Patterns

We further examine token-dependent variation in attention activations. We visualize query activations conditioned on different token types and input contexts, extending the analysis of Fig. 1(c,d) to multiple models (Figure 8). In the RoPE-based model, different tokens induce systematically different activation patterns, while such token-dependent structure is less pronounced under alternative positional encoding schemes. These observations motivate the input-conditioned, token-level component of DyPAM.

B Experimental and Implementation Details

All experiments reported in the main paper use a primary random seed of 2333, with additional runs using seeds 1000, 2000, 3000, and 4000 to assess statistical significance and reproducibility. Training employs mixed-precision (BF16) and DeepSpeed [41] with ZeRO Stage 1. A randomly selected validation set of 300 samples from the training data is used for checkpoint selection, and the checkpoint with the lowest validation loss is used for testing.

The default hyperparameter settings are summarized below:

- Optimizer: AdamW; learning rate 1×10^{-3} with cosine schedule; warmup ratio 0.05; weight decay 0.
- Batch size: 2 (gradient accumulation steps of 2); max sequence length: 2048.
- Modulation feature dimension $d_e = 64$; low-rank projection rank $r = 128$; modulation strength $\alpha = 0.3$.

Table 4: Statistics of the training datasets for commonsense and mathematical reasoning tasks.

Dataset	Samples	Total Tokens	Avg. Tokens/Sample
Math10K	9,919	2,273,016	229.16
Commonsense15K	15,119	1,778,782	117.65

Table 5: Statistics of Mathematical Reasoning Test Datasets.

Dataset	Samples	Answer Type
MultiArith	600	Numeric
GSM8K	1,319	Numeric
AddSub	395	Numeric
AQuA	254	Multiple Choice (A–E)
SingleEq	508	Numeric
SVAMP	1,000	Numeric
MAWPS	238	Numeric

B.1 Software and Environment

The software environment uses PyTorch 2.7.0, DeepSpeed 0.18.4, NumPy 2.2.6, PEFT 0.18.1, Transformers 4.51.0, Tokenizers 0.21.4, and CUDA 12.8. Experiments run on Ubuntu 24.04 LTS with an Intel Xeon Gold 6330 CPU, an NVIDIA GeForce RTX 4090 GPU, and 512GB RAM. For baseline methods, except for extremely low-parameter approaches, we match the number of trainable parameters to a comparable scale; all other hyperparameters follow their default configurations from the PEFT library [32] or the original implementations. Detailed implementation and datasets can be found in our codebase³.

C Dataset Details

We use two instruction-tuning datasets from the LLM-Adapters benchmark suite [17]: **Math10K** for math word problems with step-by-step solutions and **Commonsense15K** for commonsense reasoning questions normalized into a consistent instruction format. Statistics are summarized in Table 4.

We evaluate on well-established commonsense and mathematical reasoning benchmarks; detailed statistics are in Table 5 (Mathematical) and Table 6 (Commonsense).

a) Mathematical Reasoning: MultiArith [42] (multi-step arithmetic), GSM8K [9] (grade-school multi-step reasoning), AddSub [15] (addition/subtraction), AQuA [27] (algebraic problems with rationales), SingleEq [22] (equation-tree parsing), SVAMP [37] (structural variations), and MAWPS [23] (unified word-problem benchmark).

b) Commonsense Reasoning: BoolQ [6] (yes/no QA), PIQA [4] (physical commonsense), SIQA [44] (social commonsense), ARC-Challenge/ARC-Easy [8] (science QA), OBQA [34] (open-book science QA), HellaSwag [57] (adversarial NLI), and WinoGrande [43] (pronoun resolution).

Table 6: Statistics of Commonsense Reasoning Test Datasets.

Dataset	Samples	Answer Format
BoolQ	3,270	true / false
PIQA	1,838	solution1 / solution2
SIQA	1,954	answer1 / answer2 / answer3
ARC-Challenge	1,172	answer1 / answer2 / answer3 / answer4
ARC-Easy	2,376	answer1 / answer2 / answer3 / answer4
OBQA	500	answer1 / answer2 / answer3 / answer4
HellaSwag	10,042	ending1 / ending2 / ending3 / ending4
WinoGrande	1,267	option1 / option2

Table 7: Inference overhead of DyPAM compared with the base model and representative PEFT baselines on LLaMA-3.2-3B. Latency is reported per token. The per-token value is lower at batch size 32 because more tokens are processed in parallel.

Method	BS=1 (ms/tok)	BS=32 (ms/tok)	Mem (GB)	FLOPs overhead
Base	14.49	0.49	6.00	—
LoRA (unmerged)	27.64	1.52	6.06	73.0M (+1.13%)
RoSA	22.40	1.45	6.03	34.9M (+0.54%)
DyPAM (ours)	24.98	1.49	6.06	66.1M (+1.03%)

D Evaluation Protocol and Metrics

All model outputs are generated with greedy decoding (`do_sample=False`, `max_new_tokens=256`) via the `generate()` API in Hugging Face Transformers, using the unified instruction template below.

```
<s>Below is an instruction that describes a task. Write a
response that appropriately completes the request.
### Instruction:
{instruction}

### Response:
```

Predictions are extracted with task-specific regular expressions: for mathematical reasoning, numerical answers (absolute tolerance 10^{-3}) or alphabetic choices (A–E) for AQuA; for commonsense reasoning, exact-match answers (true/false and option labels) against ground-truth labels. All extraction and accuracy computation scripts are provided in our codebase.

E Inference Overhead

We report the inference overhead of DyPAM against the base model and representative PEFT baselines in Table 7, measured on LLaMA-3.2-3B. DyPAM adds 1.03% FLOPs overhead, lower than unmerged LoRA (1.13%), as it avoids low-rank updates to all linear layers and the per-token modulation introduces only a small cost. DyPAM is about 10% faster than unmerged LoRA at batch size 1 and comparable at batch size 32, with negligible memory overhead. Unlike LoRA, DyPAM cannot be merged into the backbone weights due to its input-conditioned design, but the overhead remains controlled and can be further reduced with optimizations such as KV-cache reuse and kernel fusion.

³<https://github.com/Beihang-BIGSCity/DyPAM>