Contextual Attention Modulation: Towards Efficient Multi-Task Adaptation in Large Language Models

Dayan Pan
SCSE, Beihang University
ERC of ACAT, MOE
Beijing, China
City University of Hong Kong
Hong Kong, China
dayan@buaa.edu.cn

Zhaoyang Fu
Huawei Technologies Ltd.
Shenzhen, China
fuzhaoyang2@huawei.com

Jingyuan Wang*
SCSE, Beihang University
Key Lab of DIM, MIIT
SEM, Beihang University
Beijing, China
jywang@buaa.edu.cn

Xiao Han
Zhejiang University of Technology
Hangzhou, China
hahahenha@gmail.com

Yue Zhu*
Huawei Technologies Ltd.
Shenzhen, China
zhuyue9@huawei.com

Xiangyu Zhao*
City University of Hong Kong
Hong Kong, China
xianzhao@cityu.edu.hk

Abstract

Large Language Models (LLMs) possess remarkable generalization capabilities but struggle with multi-task adaptation, particularly in balancing knowledge retention with task-specific specialization. Conventional fine-tuning methods suffer from catastrophic forgetting and substantial resource consumption, while existing parameter-efficient methods perform suboptimally in complex multi-task scenarios. To address this, we propose Contextual Attention Modulation (CAM), a novel mechanism that dynamically modulates the representations of self-attention modules in LLMs. CAM enhances task-specific features while preserving general knowledge, thereby facilitating more effective and efficient adaptation. For effective multi-task adaptation, CAM is integrated into our Hybrid Contextual Attention Modulation (HyCAM) framework, which combines a shared, full-parameter CAM module with multiple specialized, lightweight CAM modules, enhanced by a dynamic routing strategy for adaptive knowledge fusion. Extensive experiments on heterogeneous tasks, including question answering, code generation, and logical reasoning, demonstrate that our approach significantly outperforms existing approaches, achieving an average performance improvement of 3.65%. The implemented code and data are available to ease reproducibility.[1]

CCS Concepts

• **Computing methodologies** → **Natural language processing**.

*Corresponding authors.
[1]https://github.com/Applied-Machine-Learning-Lab/HyCAM

CIKM '25, Seoul, Republic of Korea.

ACM ISBN 979-8-4007-2040-6/2025/11
https://doi.org/10.1145/3746252.3761289

Keywords

Large Language Model, Parameter-efficient fine-tuning , Multi-Task Adaptation

ACM Reference Format:
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 (CIKM '25), November 10–14, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3746252.3761289

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities by their extensive general knowledge and powerful reasoning abilities [1, 51]. More than just a conversation, these models are increasingly proving invaluable as core components in advanced information retrieval [31, 32], critical decision-making systems [4, 57], and spatiotemporal applications [7, 66]. The success has led to increasing demand for adapting such models to specialized domains and, more importantly, to handle multiple diverse tasks simultaneously. This capability is essential for effective deployment in real-world applications [3, 27, 63].

Supervised Fine-Tuning (SFT), a widely adopted adaptation approach, involves further tuning a pre-trained model on task-specific instruction data [61]. However, achieving effective adaptation remains significant challenges. Conventional full parameter fine-tuning, a common SFT implementation that updates all parameters, needs to achieve effective adaptation while preserving foundational capabilities. The training process on a narrow task-specific dataset can significantly change the model's pre-trained weights, leading to catastrophic forgetting [30]. Furthermore, such an approach typically demands substantial computational resources. Such limitations hinder its applicability in many practical scenarios, especially in multi-task settings [13, 58], where models must balance between generalization and specialization. To address these limitations, various Parameter-Efficient Fine-Tuning (PEFT) techniques have been proposed. These approaches adapt pre-trained LLMs to new tasks by updating only a small number of trainable parameters while

leaving the backbone model unchanged, thereby reducing computational cost and overfitting risks [19]. Common PEFT strategies include adapter-based methods [22] that insert lightweight trainable modules, prompt-based methods such as Prefix Tuning [33] that modify input representations, and reparameterization methods like Low-Rank Adaptation (LoRA) [23] and its variants. LoRA, a widely utilized PEFT method, employs low-rank decomposition to weight updates, making it both efficient and effective.

However, these methods face limitations in complex multi-task scenarios due to their limited generalization and representational capacity across diverse tasks and potential interference when adapting to multiple objectives simultaneously [36, 43, 62]. Specifically for low-rank reparameterization approaches like LoRA, the low-rank adaptability may restrict model expressiveness when applied to highly complex tasks, resulting in suboptimal performance [44]. While strategies like incorporating the Mixture-of-Experts (MoE) mechanism, which combines multiple specialized PEFT modules for multi-task adaptation, aim to enhance model capacity for diverse tasks, these MoE-based approaches can introduce additional challenges, including mitigating coupling effects and effectively managing the contributions of different experts [49].

Overall, adapting LLMs to diverse tasks presents two major challenges: (1) preserving rich pretrained general knowledge while specializing for specific tasks, and (2) extending the multi-task capabilities of Parameter-Efficient Methods.

Our approach is motivated by a key observation regarding LLM architectures: different components in the Transformer reveal different roles and activation behaviors. Existing literature suggests that Feed-Forward Network (FFN) layers, constituting the bulk of model parameters, primarily function as key repositories for storing and recalling general knowledge [15]. In contrast, self-attention mechanisms are primarily responsible for processing and integrating contextual information within the input sequence, capturing dependencies between tokens [28]. This functional difference is also reflected in the parameter activation. While FFNs, comprising approximately 90% of model parameters, exhibit high activation sparsity, self-attention mechanisms typically demonstrate denser activation patterns [5, 11, 25]. This denser engagement highlights its critical role in integrating latent general knowledge with contextual information derived from the input.

Given these differences, we argue that focusing on the modulation of self-attention during multi-task adaptation provides a more effective and specialized strategy. The key insight is that large-scale pre-training has equipped LLMs with extensive general knowledge, so effective adaptation should focus on enabling LLMs to better integrate task-specific contextual information. Such an approach can refine how general knowledge is integrated with specific contextual demands of diverse tasks. Importantly, this modulation preserves pre-trained general knowledge, thereby mitigating issues like catastrophic forgetting and task interference.

To this end, we introduce Contextual Attention Modulation (CAM), a novel mechanism designed to dynamically modulate the representations within the self-attention modules of LLMs based on the input context. CAM learns to dynamically modulate self-attention representations to adapt the input context. This context-aware mechanism selectively amplifies task-relevant attentional signals and suppresses irrelevant or interfering ones, thereby enhancing task-specific features while preserving the model's pre-trained general knowledge. Directly modulating the organization of contextual information within attention modules promotes more effective knowledge retention and specialized adaptation, thereby supporting more robust and efficient multi-task learning.

To extend the multi-task capabilities, we embed CAM into our Hybrid Contextual Attention Modulation (HyCAM) framework. HyCAM combines a shared, full-parameter CAM module, which is designed to capture and leverage common knowledge across all tasks, with multiple specialized, lightweight CAM modules. These specialized modules implement the CAM mechanism using PEFT techniques to efficiently capture distinct features, allowing effective multi-task adaptation with minimal additional trainable parameters. A soft-routing strategy, further augmented by a load-balancing constraint, dynamically manages the fusion of knowledge from these shared and specialized CAM components. This design empowers HyCAM to extend multi-task performance by enabling both efficient knowledge sharing and fine-grained specialization.

The main contributions of this paper are summarized as follows:

- We propose Contextual Attention Modulation (CAM), a novel mechanism that learns to dynamically modulate self-attention representations in LLMs based on input context. CAM is designed to enhance task-specific features while preserving pre-trained general knowledge, thereby facilitating more effective knowledge retention and specialized adaptation.
- We introduce the Hybrid Contextual Attention Modulation (HyCAM) framework, which extends multi-task adaptation capabilities by integrating our CAM mechanism in distinct forms. This integration empowers HyCAM to achieve superior multi-task performance by effectively balancing efficient knowledge sharing with fine-grained task specialization.
- We conduct extensive experiments across a range of tasks covering question answering, code generation, logical reasoning, and other domains. Comparative experiments demonstrate that HyCAM significantly outperforms existing state-of-the-art approaches with faster convergence.

2 Preliminaries

This section briefly reviews the fundamental concepts essential for understanding our proposed method. We discuss the relevant components of the Transformer architecture, the basics of task-adaptive fine-tuning, and common PEFT techniques.

2.1 Transformer Architecture

The Transformer architecture [52] serves as the backbone of most LLMs owing to its ability to efficiently process sequences of data through attention mechanisms, making it especially powerful for understanding and generating human language. A Transformer model is typically composed of a stack of identical blocks. Each block primarily contains two core components: the self-attention mechanism and the Feed-Forward Network (FFN). Self-Attention mechanism allows the model to weigh the importance of different tokens in an input sequence and capture contextual relationships by computing attention scores using Query (Q), Key (K), and Value (V) projections, often via scaled dot-product attention:

$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$. Following this, the FFN, typically composed of two linear transformations with a non-linear activation, further processes each token's representation independently to express the complex knowledge of the model.

2.2 Task-Adaptive Fine-Tuning

While LLMs acquire extensive general knowledge and reasoning capabilities, they typically require further adaptation to specialize them for specific tasks and align their behavior with desired objectives, such as following instructions. A common approach for such task-adaptive fine-tuning is Supervised Fine-Tuning (SFT). In SFT, the model learns from examples that provide explicit input-output pairings. These pairings might illustrate a question with its corresponding answer or an instruction followed by the desired model output. The primary goal is to adjust the model's parameters to minimize a task-specific loss function, such as cross-entropy loss for sequence generation or classification tasks.

2.3 Parameter-Efficient Fine-Tuning

Adapting LLMs to specific tasks often involves fine-tuning, but updating all parameters is computationally expensive. PEFT methods enable model adaptation by introducing a small set of new parameters or reparameterizing existing ones while keeping the backbone model weights frozen, significantly reducing computational costs.

A mainstream PEFT category is reparameterization, which introduces a smaller set of trainable parameters that efficiently influence the model's behavior. For instance, a common strategy is to represent the change in a pre-trained weight matrix W_0 during adaptation as a low-rank update, based on the observation that task-specific changes often lie in a subspace of much lower dimensionality than the full parameter space. Thus, instead of learning a large, dense update matrix ΔW, these methods learn a low-rank approximation of it, such as $\Delta W = BA$, where B and A are much smaller matrices [23].

3 Method

We first illustrate an overview of our proposed HyCAM framework. Next, the core CAM mechanism is further detailed. We then provide an in-depth description of the HyCAM framework, including its hybrid components and dynamic knowledge fusion strategies with a soft-routing method and load-balancing constraint, and conclude by specifying the training objective.

3.1 Framework Overview

To address the critical challenge of enabling LLMs to efficiently adapt to diverse tasks while balancing knowledge retention with task-specific specialization, we introduce the Hybrid Contextual Attention Modulation (HyCAM) framework. The core mechanism of HyCAM is Contextual Attention Modulation (CAM), which dynamically learns context-dependent modulation of self-attention representations, selectively amplifying task-relevant signals while suppressing irrelevant or potentially interfering ones to enhance task-specific features and preserve general knowledge. As illustrated in Figure 1, the HyCAM framework employs a novel hybrid architecture that integrates a shared, full-parameter CAM module, designed for capturing common knowledge across tasks, with multiple specialized CAM modules that utilize parameter-efficient techniques for efficient, fine-grained adaptation to distinct task features. The contributions of these diverse CAM modules are managed by a dynamic routing strategy to ensure balanced utilization of the specialized components and adaptive knowledge fusion.

3.2 Contextual Attention Modulation

The CAM mechanism is the core of our HyCAM framework, designed to dynamically modulate self-attention representations at each Transformer block. It learns to dynamically amplify task-relevant attentional signals and suppress irrelevant ones based on the input context, thereby enhancing task-specific features while preserving the model's pre-trained general knowledge, which facilitates more effective and efficient task adaptation.

3.2.1 ***Motivation.*** Our motivation for developing CAM comes from the analysis of distinct roles and activation patterns across different Transformer components, as described in Section 1. While FFN modules account for a large portion of parameters and store a vast amount of an LLM's parameterized knowledge, self-attention modules are crucial for dynamically processing and integrating contextual information. The varying activation patterns of these components highlight the important role of the self-attention modules in integrating latent general knowledge with the specific context derived from an input. With extensive general knowledge from large-scale pre-training of LLMs, the key to effective adaptation lies in enabling them to better integrate this foundational knowledge with task-specific contextual information. Conventional fine-tuning approaches, however, can often overwrite the valuable pre-trained representations by introducing new task-specific knowledge.

This observation motivated us to develop CAM, a mechanism that refines how general knowledge is integrated with specific contextual demands of diverse tasks by modulating self-attention representations. This approach aims to facilitate task-adaptive specialization while preserving valuable pre-trained knowledge.

3.2.2 ***The CAM Mechanism.*** The CAM mechanism is integrated into each Transformer block, operating on the output of the self-attention modules to dynamically modulate its representations based on the input context. This process allows for a fine-grained modulation of contextual information flow. Specifically, the CAM mechanism proceeds as follows:

Input Normalization: Let $h_{in} \in \mathbb{R}^{L\times d}$ be the input hidden state to a Transformer layer, where L denotes the sequence length and d represents the hidden dimension. Consistent with standard Transformer operations, these input hidden states are first normalized using Layer Normalization [2], producing $h_{norm} \in \mathbb{R}^{L\times d}$:

$$h_{norm} = \text{LayerNorm}(h_{in}). \tag{1}$$

The resulting h_{norm} serves as the input for both the conventional self-attention computation and our CAM module.

Modulation Weight Generation: CAM then computes a context-dependent modulation weight tensor, denoted as $\mathbf{A}_{\text{CAM}} \in \mathbb{R}^{L\times d}$. These weights are derived from the normalized hidden state h_{norm} through a linear projection parameterized by a trainable weight

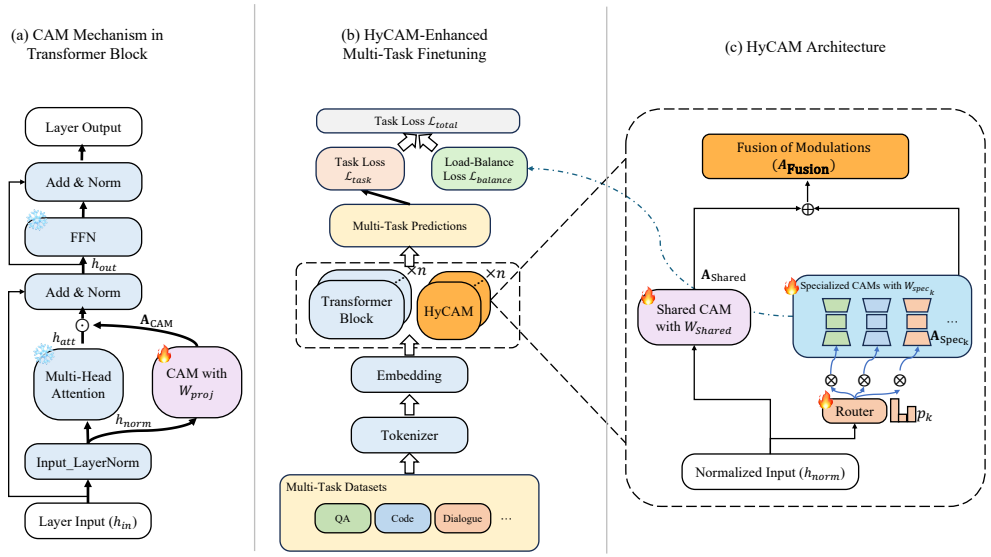

Figure 1: The architecture of the CAM and HyCAM framework. HyCAM applies a hybrid CAM mechanism to the output of the Attention module within each Transformer block, while the backbone LLM remains frozen. Specifically, HyCAM integrates a shared, full-parameter CAM module and multiple lightweight Specialized CAMs for common and task-specific knowledge.

matrix $W_{proj} \in \mathbb{R}^{d\times d}$, followed by a SiLU activation function [10]:

$$\mathbf{A}_{\text{CAM}} = \text{SiLU}(h_{norm}W_{proj}). \tag{2}$$

The matrix W_{proj} is specific to the CAM module and is crucial for learning how to modulate the attention representations based on the input context. To ensure stability during the initial phases of fine-tuning and to allow the model to gradually learn the modulation, W_{proj} is initialized as a zero matrix. This initialization strategy ensures that at the beginning of the fine-tuning, CAM does not alter the pre-trained model's behavior. That is, the model initially maintains its original approach to processing contextual information, which is then gradually modulated as training progresses for a stable adaptation.

Application of Modulation: Concurrently, the standard attention output $h_{att} \in \mathbb{R}^{L\times d}$ is computed using the normalized input h_{norm}:

$$h_{att} = \text{Self-Attention}(h_{norm}). \tag{3}$$

The CAM mechanism then refines this h_{att} by applying the learned modulation weights $\mathbf{A}_{\text{CAM}}$. This is performed via an element-wise Hadamard product ($\odot$). The modulated signal is integrated with the original h_{att} through a residual connection, forming the final output $h_{out} \in \mathbb{R}^{L\times d}$ of the attention mechanism incorporating CAM.

$$h_{out} = h_{att} + h_{att} \odot \mathbf{A}_{\text{CAM}}. \tag{4}$$

3.2.3 ***Advantages.*** By dynamically generating and applying these modulation weights, CAM refines the contextual representation from the self-attention modules to adapt it to specific tasks while preserving the pre-trained general knowledge, thereby mitigating catastrophic forgetting. Thus, CAM facilitates an effective balance between achieving task-specific adaptation and retaining extensive general knowledge. Moreover, by modulating attentional outputs instead of fine-tuning a large number of backbone parameters, CAM achieves computational efficiency.

3.3 The HyCAM Framework

While the CAM mechanism provides a powerful tool for modulating attention representations, adapting LLMs to handle multiple diverse tasks simultaneously presents significant challenges. Conventional full fine-tuning struggles with catastrophic forgetting and resource demands, while existing PEFT methods still face limitations for multi-tasking. Specifically, the limited capacity of representation makes it suboptimal for highly complex tasks, and simple applications of expert-based strategies lead to an imbalance in expert utilization.

To address these multiple challenges and effectively leverage the CAM mechanism for complex multi-task learning scenarios, we introduce the HyCAM framework. The framework is designed to extend the multi-task adaptation capabilities by integrating CAM in hybrid forms, enabling both efficient knowledge sharing and fine-grained task specialization. This is achieved through strategically combining shared, full-parameter CAM module, for efficient knowledge sharing, with multiple specialized, parameter-efficient CAM modules, for fine-grained specialization. The contributions of these components are coordinated by a dynamic routing mechanism with a load-balancing constraint to ensure adaptive knowledge fusion.

3.3.1 ***Hybrid CAM Components.*** The hybrid architecture of the HyCAM framework is designed to leverage both general context understanding and specialized, task-specific adaptation capabilities. This architecture comprises a shared, full-parameter CAM module and multiple lightweight, specialized CAM modules:

Shared CAM Module: The Shared CAM module serves as a global modulator, for capturing and refining common contextual patterns and general knowledge across all tasks. This module is a full-parameter CAM, as detailed in Section 3.2. Its trainable projection matrix, denoted as $W_{Shared} \in \mathbb{R}^{d\times d}$, is shared and updated

across all tasks to produce a modulation weight tensor:

$$\mathbf{A}_{Shared} = \text{SiLU}(h_{norm} W_{Shared}). \tag{5}$$

Specialized CAM Modules: In addition to the shared module, HyCAM incorporates multiple (N_s) lightweight Specialized CAM modules. Specialized CAM modules are designed to learn and apply attention modulations for the distinct features of specific tasks.

Different tasks often require different ways of handling contextual information in the self-attention layer. For example, code generation may need to focus on long-range dependencies, while question answering systems may prioritize specific entities and their relationships in a local context. This design is to enable the model to develop fine-grained adaptations for diverse tasks, thereby mitigating the interference when a single component attempts to learn potentially conflicting objectives from multiple tasks.

The implementation of Specialized CAM modules leverages the PEFT technique for reducing the number of trainable parameters per specialized module, making the framework scalable. Besides, it helps in mitigating overfitting, especially when task-specific data might be limited. Specifically, each Specialized CAM module, indexed by $k \in \{1, ..., N_s\}$, generates its unique modulation weight tensor $\mathbf{A}_{\text{Spec}_k} \in \mathbb{R}^{L \times d}$ as follows:

$$\mathbf{A}_{\text{Spec}_k} = \text{SiLU}(h_{norm} W_{\text{Spec}_k}), \tag{6}$$

where W_{Spec_k} is the trainable projection matrix specific to the k-th specialized module. To achieve parameter efficiency while enhancing representational capacity, we adopt the SLoRA [16] technique for the structure of W_{Spec_k}. Instead of a direct low-rank decomposition like LoRA, typically $W = BA$, SLoRA introduces an intermediate trainable matrix N between B and A. Thus, W_{Spec_k} is parameterized as:

$$W_{Spec_k} = B_k N_k A_k. \tag{7}$$

Here, $A_k \in \mathbb{R}^{r \times d}$ is a matrix that projects the d-dimensional hidden state h_{norm} into a lower-dimensional space of rank r. $N_k \in \mathbb{R}^{r \times r}$ is a trainable intermediate matrix within the low-rank space. $B_k \in \mathbb{R}^{d \times r}$ is a matrix that projects the r-dimensional representation back to the original d-dimensional space. The rank r is significantly smaller than d, ensuring a substantial reduction in trainable parameters compared to a full $d \times d$ matrix.

For initialization, and similar to the zero-initialization of W_{Shared} in the Shared CAM module, we adopt a strategy to ensure training stability. Specifically, the matrices A_k and N_k are initialized using Kaiming Uniform [20]. The matrix B_k is initialized with zeros. This structure allows each Specialized CAM to develop task-specific modulations with very small parameters, thus enhancing the adaptability of the model without sacrificing efficiency.

3.3.2 ***Dynamic Routing.*** To effectively leverage the diverse contributions from the Shared CAM and multiple Specialized CAM modules, HyCAM incorporates a dynamic soft-routing mechanism coupled with a load-balancing constraint. This mechanism adaptively determines the influence of each module based on the input context and promotes load-balance to ensure efficient utilization of all Specialized CAMs.

Routing for Specialized CAMs: The dynamic routing mechanism weights the contributions of the N_s Specialized CAM modules for each input token. This enables HyCAM to adapt its modulation strategy in a fine-grained, context-dependent manner. The routing process is detailed as follows:

For each token representation h_{norm}, derived from h_{in} as described in Section 3.2.2, a lightweight router network first generates $\textbf{logits} \in \mathbb{R}^{N_s}$, produced by a linear layer applied to h_{norm}:

$$\textbf{logits} = h_{norm} W_{router}, \tag{8}$$

where $W_{router} \in \mathbb{R}^{d \times N_s}$ is the trainable weight matrix of the router.

These $\textbf{logits} = [\pi_1, \pi_2, ..., \pi_{N_s}]$ are then transformed into a probability distribution over the specialized modules using the Gumbel-Softmax estimator [24] to obtain differentiable, soft routing probabilities. The Gumbel-Softmax allows for differentiable sampling from a categorical distribution, which facilitates the training process while encouraging exploration, as detailed:

$$p_k = \frac{\exp((\log \pi_k + g_k)/\tau)}{\sum_{j=1}^{N_s} \exp((\log \pi_j + g_j)/\tau)}, \tag{9}$$

where p_k is the resulting soft routing weight for the k-th Specialized CAM module. $g_k \sim \text{Gumbel}(0, 1)$ are *i.i.d.* noise drawn from the Gumbel distribution, adding stochasticity for exploration. τ is a temperature hyperparameter that controls the sharpness of the probability distribution. Lower temperatures make the selection more discrete, while higher temperatures make it softer.

Load Balancing Loss: To prevent routers from over-selecting a few modules, HyCAM adds a load-balancing loss $\mathcal{L}_{balance}$ that encourages more balanced routing across specialized components. For a batch of B tokens, it is computed as:

$$\mathcal{L}_{balance} = \sum_{k=1}^{N_s} \left(\frac{1}{B} \sum_{b=1}^{B} p_{b,k} \right) \cdot \left(\frac{1}{B} \sum_{b=1}^{B} \text{softmax}(\textbf{logits}_b)_k \right), \tag{10}$$

where $p_{b,k}$ is the Gumbel-Softmax output and $\text{softmax}(\textbf{logits}_b)_k$ is the standard softmax output of the router logits.

Fusion of Modulations: Once the routing weights p_k are determined for each token, as described in Equation 9, the final context-dependent modulation tensor, $\mathbf{A}_{Fusion} \in \mathbb{R}^{L \times d}$, is computed by combining the output of the Shared CAM module, $\mathbf{A}_{Shared}$, with the dynamically weighted sum of the modulations from all Specialized CAM modules, $\{\mathbf{A}_{Spec_k}\}_{k=1}^{N_s}$:

$$\mathbf{A}_{Fusion} = \mathbf{A}_{Shared} + \sum_{k=1}^{N_s} p_k \cdot \mathbf{A}_{Spec_k}, \tag{11}$$

Here, p_k denotes the token-specific routing weight of the k-th specialized module, ensuring that the context-based modulation of $\mathbf{A}_{Fusion}$ integrates both general and adaptively selected specialized knowledge. Finally, it is applied to the original self-attention output h_{att}, from Equation 3 in Section 3.2.2, to produce the HyCAM-enhanced output h_{out} using the element-wise Hadamard product and residual connection, as defined in the core CAM mechanism:

$$h_{out} = h_{att} + h_{att} \odot \mathbf{A}_{Fusion}. \tag{12}$$

This entire mechanism, from dynamic routing to the application of the fused modulation, allows HyCAM to dynamically modulate the self-attention process by integrating shared knowledge with specialized insights, thereby enabling the model to effectively balance generalization across diverse tasks with task-specific adaptation.

Table 1: Datasets statistics.

Dataset	Samples	Total Tokens[1]	Avg. Tokens/Sample[1]	Domain	Source
Auto CoT	5,816	943,474	162.22	Arithmetic and other logical reasoning tasks	[67]
iCliniq	7,321	1,826,306	249.46	Conversations between patients and doctors	[34]
Dolly 2.0	15,015	3,061,007	203.86	Closed QA and summarization from Wikipedia	[8]
CodeAlpaca	20,222	2,195,523	109.66	Code generation and optimization	[6]
WebGPT	18,994	13,988,895	736.49	Information retrieval QA	[42]

[1] Calculated by Llama-3 Tokenizer.

Table 2: Experimental results across different backbone LLMs. *indicates the statistically significant improvements (*i.e.,* two-sided t-test with $p < 0.05$) over the best PEFT baseline. Lower PPL↓ is better, where higher BLEU↑ and ROUGE↑ reflect higher quality. The best results are bolded, while the second-best results are underlined.

Backbone LLM	Llama 2 7B			Llama 3 8B			Llama 3.1 8B			Mistral 7B			Qwen 2.5 7B		
Metric	PPL↓	BLEU↑	ROUGE↑	PPL↓	BLEU↑	ROUGE↑	PPL↓	BLEU↑	ROUGE↑	PPL↓	BLEU↑	ROUGE↑	PPL↓	BLEU↑	ROUGE↑
Full Finetune	3.193	0.171	0.231	3.978	0.151	0.203	3.873	0.153	0.205	4.403	0.157	0.192	3.024	0.169	0.225
LoRA	3.222	0.157	0.225	3.556	0.148	0.24	3.537	0.156	0.237	3.418	0.163	0.244	2.840	0.137	0.239
Multi LoRA	3.287	0.121	0.217	3.547	0.157	0.236	3.653	0.134	0.235	3.461	0.141	0.225	3.069	0.136	0.222
RieMoE-LoRA	3.171	0.154	0.232	3.497	0.159	0.242	3.487	0.161	0.238	3.597	0.143	0.24	2.830	0.157	0.227
HyCAM	**3.081***	**0.173***	**0.244***	**3.484***	**0.162***	**0.245***	**3.453***	**0.172***	**0.251***	**3.299***	**0.171***	**0.249***	**2.757***	**0.172***	**0.248***

3.4 Training Details

The HyCAM framework, including the Shared CAM module, the Specialized CAM modules, and the dynamic router, is trained end-to-end. We use a composite objective function that combines a primary task-specific loss with the auxiliary load-balancing loss, described in Section 3.3.2. This approach ensures that the model not only learns to perform the target tasks effectively but also maintains balanced utilization of its specialized components, leading to efficient adaptation across diverse tasks and enhanced overall multi-task performance.

Task-specific Loss: We employ a standard autoregressive training strategy common for LLMs, as introduced in Section 2.2, where the model is trained to predict the next token in a sequence given the input context. Given an input sequence $\mathbf{X} = (x_1, x_2, \ldots, x_m)$ and its corresponding target sequence $\mathbf{Y} = (y_1, y_2, \ldots, y_n)$, the model is trained to predict each token y_t conditioned on the input $\mathbf{X}$ and the previous target tokens $\mathbf{Y}_{<t} = (y_1, y_2, \ldots, y_{t-1})$. The primary objective is to minimize the cross-entropy loss of the target sequences:

$$\mathcal{L}_{\text{task}} = -\sum_{i=1}^{|\mathcal{D}|}\sum_{t=1}^{n_i} \log P(y_{i,t}|\mathbf{X}_i, \mathbf{Y}_{i,<t}; \Theta_{\text{HyCAM}}), \tag{13}$$

where $\mathcal{D}$ represents the batch of training examples, n_i is the length of the i-th target sequence, and $P(y_{i,t}|\mathbf{X}_i, \mathbf{Y}_{i,<t}; \Theta_{HyCAM})$ is the probability of the true token $y_{i,t}$ predicted by the HyCAM framework with its trainable parameters Θ_{HyCAM}.

Overall Training Objective: To ensure diverse utilization of the Specialized CAM modules, we incorporate the auxiliary load-balancing loss $\mathcal{L}_{\text{balance}}$, as defined in Equation 10, into the overall training objective. The final training loss $\mathcal{L}_{\text{total}}$ that HyCAM optimizes is therefore a weighted sum of the task loss and the load-balancing loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \lambda_{\text{balance}} \cdot \mathcal{L}_{\text{balance}}. \tag{14}$$

Here, λ_{balance} is a hyperparameter that controls the contribution of the load-balancing constraint. By optimizing $\mathcal{L}_{\text{total}}$, the HyCAM framework learns to effectively perform the target tasks while maintaining a balanced and efficient use of its specialized components.

4 Experiments

In this section, we present a comprehensive evaluation of our proposed Hybrid Contextual Attention Modulation (HyCAM) framework. To systematically evaluate the performance and characteristics of HyCAM, we conduct experimental analysis around the following key research questions (RQs):

- **RQ1**: How does the HyCAM framework perform overall in multi-task adaptation compared to state-of-the-art baseline methods across various backbone models?
- **RQ2**: How does HyCAM scale when applied to backbone Large Language Models of different sizes?
- **RQ3**: How does HyCAM perform on individual tasks within the multi-task benchmark, and does it demonstrate a balanced adaptation capability across these diverse task types?
- **RQ4**: What are the contributions of the core CAM mechanism and other components of HyCAM to its overall effectiveness, and how sensitive is its performance to key hyperparameters?
- **RQ5**: What qualitative insights do visualizations offer regarding HyCAM's internal working mechanisms?

4.1 Experimental Setup

4.1.1 ***Datasets.*** To better evaluate the effectiveness of HyCAM in complex multi-task scenarios, we construct a comprehensive benchmark dataset comprising five distinct domains: Auto CoT (logical reasoning), iCliniq (medical QA), Dolly 2.0 (general instruction-following), CodeAlpaca (code generation), and WebGPT (information retrieval QA). Detailed information, including statistics, domains, and sources, is presented in Table 1. Further, for robust

Table 3: Results with different sizes of Qwen 2.5-Family.

Backbone	Qwen 2.5 0.5B			Qwen 2.5 1.5B			Qwen 2.5 3B			Qwen 2.5 7B			Qwen 2.5 14B		
Metric	PPL↓	BLEU↑	ROUGE↑	PPL↓	BLEU↑	ROUGE↑	PPL↓	BLEU↑	ROUGE↑	PPL↓	BLEU↑	ROUGE↑	PPL↓	BLEU↑	ROUGE↑
Full Finetune	3.778	0.159	0.219	**3.102**	**0.169**	0.235	2.982	0.161	0.222	3.024	0.169	0.225	2.839	**0.176**	0.214
LoRA	3.764	0.145	0.222	3.344	0.138	0.229	3.106	0.144	0.230	2.840	0.137	0.239	2.889	0.147	0.238
Multi LoRA	3.754	0.144	0.221	3.330	0.148	0.226	3.053	0.157	0.225	3.069	0.136	0.222	2.882	0.152	0.235
RieMoE-LoRA	3.621	0.152	0.232	3.180	0.148	0.230	3.001	0.148	0.238	2.83	0.157	0.227	2.792	0.142	0.238
HyCAM	**3.611**	**0.169**	**0.262**	3.108	0.167	**0.236**	**2.940**	**0.165**	**0.249**	**2.757**	**0.172**	**0.248**	**2.682**	0.160	**0.242**

Table 4: Results with different sizes of Llama 3.2.

Backbone	Llama 3.2 1B			Llama 3.2 3B		
Metric	PPL↓	BLEU↑	ROUGE↑	PPL↓	BLEU↑	ROUGE↑
Full Finetune	**4.221**	**0.164**	0.221	**3.747**	0.159	0.220
LoRA	4.515	0.144	0.227	3.824	0.144	0.234
Multi LoRA	4.533	0.143	0.225	3.876	0.149	0.232
RieMoE-LoRA	4.324	0.161	0.241	3.806	0.154	0.233
HyCAM	4.227	0.163	**0.244**	3.778	**0.167**	**0.243**

evaluation across these datasets, we employ a 7:2:1 split for training, validation, and testing, and conduct five-fold cross-validation.

4.1.2 ***Backbone Models.*** We evaluate HyCAM across three state-of-the-art open-source LLM families to demonstrate its applicability:

LLaMA. We utilize multiple versions from Meta AI's LLaMA series, which is known for strong performance in zero-shot and few-shot scenarios with large amounts of pre-training data. Specifically, our experiments include Llama2-7B, Llama3-8B, Llama3.1-8B, and the smaller Llama3.2-1B/3B models.

Mistral. Introduced by Mistral AI, Mistral features a compact model to achieve comparable performance to the mainstream methods via knowledge distillation. We use Mistral-7B-v0.3 as our base model.

Qwen. Developed by Alibaba, Qwen focuses on efficient inference and robust performance across diverse tasks. We adopt various sizes of Qwen2.5 series, ranging from 0.5B to up to 14B.

4.1.3 ***Baseline Methods.*** We evaluate HyCAM against several representative methods of task adaptation, detailed as follows:

- **Full Fine-Tuning** involves updating all parameters of the backbone LLM for adaptation to the target tasks.
- **LoRA** [23] injects trainable low-rank adaptation matrices, allowing efficient adaptation with reduced trainable parameters.
- **Multi LoRA** [59] applies multiple LoRA adapters in parallel, enabling independent adaptation for different tasks.
- **RieMoE-LoRA** [50] integrates Riemannian gradient rescaling with MoE-LoRA to preserve expressiveness while stabilizing optimization and accelerating convergence.

4.1.4 ***Evaluation Metrics.*** To comprehensively evaluate the performance of methods across diverse tasks and domains, we adopt three commonly used evaluation metrics:
PPL (Perplexity) assesses the fluency and overall language generation quality. Lower PPL scores indicate better performance.
BLEU-4 (Bilingual Evaluation Understudy) [45] measures the n-gram overlap between generated and label texts, particularly relevant for tasks like code generation. Higher BLEU-4 scores are better.
ROUGE-L (Recall-Oriented Understudy for Gisting Evaluation - Longest Common Subsequence) [35] captures content overlap and semantic similarity, especially for longer outputs like summaries. Higher ROUGE-L scores indicate better performance.

4.1.5 ***Implementation Details.*** All our experiments are implemented using PyTorch, and we leverage DeepSpeed for efficient training with BFloat16 mixed-precision. For LoRA-based methods, the LoRA rank (r) is set to 64 and is applied to all linear layers of LLM. For methods involving multiple specialized modules, the number of modules (N_s) is set to 5. The maximum token length is set to 1,200 to accommodate long-text tasks. For HyCAM-specific hyperparameters, the Gumbel-Softmax temperature τ is set to 0.5, and the load-balancing loss coefficient λ_{balance} is set to 0.1. For training, we use the AdamW optimizer with a learning rate of 2e-5, a cosine decay learning rate scheduler, and early stopping based on validation loss to prevent overfitting.

4.2 Overall Performance (RQ1)

To answer **RQ1**, we evaluate the overall multi-task adaptation performance of HyCAM against the baseline methods. The experiment is conducted across different backbone LLMs of comparable scale (7B/8B parameters) with our comprehensive multi-task datasets. The main results, summarized as the average performance across all tasks, are presented in Table 2.

The experimental results demonstrate the superior overall performance of HyCAM. On average, HyCAM achieves a 3.65% relative improvement across all metrics and backbone LLMs compared to the best baseline, with statistically significant ($p < 0.05$, indicated by an asterisk ($*$) in the table).

Compared to Full Fine-Tuning, HyCAM achieves excellent performance while only updating a small fraction of the parameters, underscoring its advantage in computational efficiency. Compared with the single-task PEFT method LoRA, HyCAM shows substantial gains. While LoRA provides a parameter-efficient alternative to full fine-tuning, its capacity can be limited in complex multi-task scenarios. HyCAM, with its CAM mechanism and hybrid architecture design, is better able to handle the diverse demands of multiple tasks simultaneously, thus achieving better adaptation performance.

Furthermore, HyCAM outperforms other multi-task approaches, including Multi LoRA and RieMoE-LoRA. While Multi LoRA allows for parallel task adaptations, it may not facilitate effective knowledge sharing between tasks. RieMoE-LoRA, despite its gradient rescaling mechanism, may struggle with optimal expert utilization.

Table 5: Detailed experimental results across the five datasets.

Backbone LLM	Auto CoT			iCliniq			Dolly 2.0			CodeAlpaca			WebGPT		
Metric	PPL↓	BLEU↑	ROUGE↑	PPL↓	BLEU↑	ROUGE↑	PPL↓	BLEU↑	ROUGE↑	PPL↓	BLEU↑	ROUGE↑	PPL↓	BLEU↑	ROUGE↑
Full Finetune	1.842	0.282	0.287	**7.497**	**0.053**	0.123	6.461	0.088	**0.200**	2.532	0.138	0.195	1.888	**0.182**	**0.341**
LoRA	1.843	0.268	0.291	8.140	0.049	0.124	6.029	0.070	0.181	2.404	0.146	0.202	1.919	0.178	0.331
Multi LoRA	1.952	0.198	0.290	8.846	0.037	0.122	**5.743**	0.101	0.177	**2.312**	0.134	0.189	1.939	0.176	0.337
RieMoE-LoRA	1.813	0.275	**0.298**	8.001	0.051	0.123	5.954	**0.106**	0.183	2.381	0.142	0.207	1.888	0.177	0.336
HyCAM	**1.777**	**0.283**	0.297	7.546	**0.053**	**0.125**	5.893	0.093	0.194	2.359	**0.163**	**0.222**	**1.845**	0.180	0.337

HyCAM's integration of Shared CAM and Specialized CAM, with a dynamic routing strategy, provides a more effective framework for both knowledge sharing and specialized adaptation.

4.3 Scalability Analysis (RQ2)

To address **RQ2** and understand how the effectiveness of HyCAM scales with the size of backbone model, we evaluated its performance across a range of model size within two LLM families: Qwen2.5 (from 0.5B to 14B parameters) and Llama3.2 (1B and 3B parameters), as shown in Table 3, 4. This analysis aims to determine if the advantages are consistent across different model sizes and whether they become more or less significant with increasing size.

The results indicate that HyCAM consistently outperforms other PEFT-based methods across all tested model sizes within both the Qwen and Llama families. While Full Fine-Tuning remains a strong baseline, HyCAM consistently offers a more parameter-efficient solution with competitive, and often superior, performance. Moreover, a key observation is that the relative advantage of HyCAM often becomes more obvious as the model size increases. This suggests that larger models, with their greater capacity and more extensive general knowledge, benefit even more from HyCAM's ability to dynamically modulate attention and integrate shared and specialized knowledge effectively.

4.4 Task-Specific Analysis (RQ3)

To address RQ3, we analyze the performance on individual tasks within our multi-task dataset. We break down the results obtained with the Llama2-7B backbone model, as presented in Table 5. The results indicate that HyCAM generally achieves competitive performance across individual tasks. It is worth noting that performance varies considerably across tasks, due to differences in inherent complexity and task characteristics. In particular, the results on the iClinq and Dolly datasets are significantly lower than those of others. This observation highlights the importance of considering task complexity when designing multi-task learning frameworks for real-world applications.

4.5 Ablation and Hyperparameter Analysis (RQ4)

For **RQ4**, we investigate the contributions of HyCAM components and their sensitivity to key hyperparameters. All studies here are conducted on the Llama2-7B model with PPL metric. First, we performed ablation studies by evaluating several variants of HyCAM:

- **Shared-CAM-Only**: Only the single, shared full-parameter CAM module, removing all specialized CAMs and the routing mechanism, to assess the baseline impact of CAM.

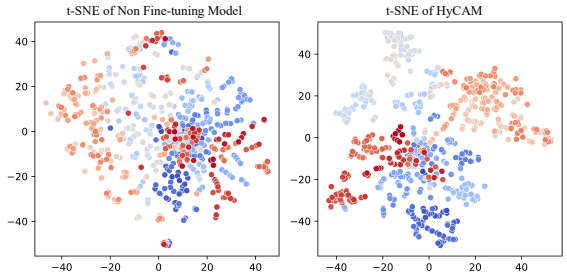

Figure 2: Scatter plots of attention representations. Higher density indicates improved representation capacity of the attention module.

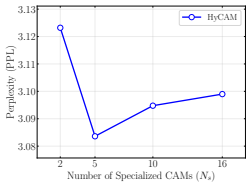

Figure 3: Impact of Number of Specialized CAMs.

Table 6: Ablation study of HyCAM on Llama 2 7B.

Variant	PPL↓
Shared-CAM-Only	3.129
HyCAM-FullSpec	3.102
HyCAM-SpecOnly	3.216
HyCAM-InversePEFT	3.129
HyCAM	**3.081**

- **HyCAM-FullSpec**: Both the shared CAM and all Specialized CAMs are implemented with full parameters, to evaluate the benefit of using PEFT for the specialized components.
- **HyCAM-SpecOnly**: Removes the shared, full-parameter CAM, relying exclusively on the ensemble of PEFT-based Specialized CAMs managed by the dynamic router and load-balancing loss.
- **HyCAM-InversePEFT**: A special configuration where the Shared CAM module is implemented using PEFT, while the Specialized CAMs are full-parameter CAM modules, to validate our architectural design for parameter allocation.

Overall, these experiments demonstrate that each component and design choice in HyCAM contributes positively to its overall performance, as shown in Table 6. We further analyzed HyCAM's sensitivity to the number of Specialized CAM modules (N_s), as illustrated in Figure 3.

4.6 Qualitative Evaluation (RQ5)

To answer **RQ5** and get in-depth insights into the inner mechanisms of HyCAM, we employ several visualization techniques:

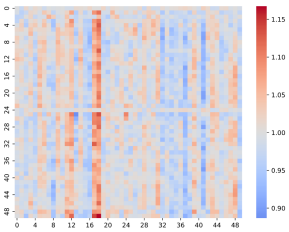

Figure 4: The Weight matrix of HyCAM.

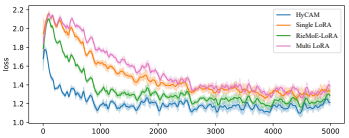

Figure 5: Comparison of the training processes across different methods.

- **Enhanced Representational Coherence:** We utilize t-SNE to visualize the representations of the value matrix (V) within the self-attention module after applying our HyCAM. The vertical axis represents each token, and the horizontal axis indicates the degrees to be enhanced. As shown in Figure 2, CAM leads to more coherent clusters in the learned representations compared to a non-modulated baseline. This suggests an improved capacity to form meaningful representations.
- **Feature Selective Modulation:** To understand how CAM impacts features, we visualize the modulation weight matrix generated by the HyCAM mechanism. As in Figure 4, these weights demonstrate selective amplification of features relevant to the input context, while other features are attenuated. This highlights CAM's ability to perform context-dependent adjustments.
- **Accelerated Convergence:** Figure 5 presents a comparison of training loss curves for HyCAM against baseline methods. These curves illustrate that HyCAM achieves lower loss more rapidly and stably, indicating efficient learning ability.

5 Related Work

5.1 Parameter-Efficient Fine-Tuning

To improve the efficiency of LLMs and make them more practical for real-world applications [38], various methods, such as pruning [54], compression [56], and PEFT methods have been proposed. The primary goal of PEFT methods is to adapt LLM to specific tasks by updating only a small fraction of parameters, thereby preserving pre-trained knowledge and significantly reducing computational and memory cost [12, 19]. Conventional PEFT strategies include additive methods like adapters, selective fine-tuning, and reparameterization techniques such as low-rank adaptation. Adapters, such as AdapterFusion [46], involve inserting lightweight, task-specific modules into the layers of the pre-trained model. Selective fine-tuning approaches, such as BitFit [64], demonstrate that even minimal parameter adjustments can be effective for many tasks. Reparameterization methods, such as Low-rank adaptation (LoRA) [23] and its variants like AdaLoRA, DoRA, and MetaLoRA [39, 55, 65], apply rank decomposition into target layers. LoRA operates on the principle that updates to the weight matrices during adaptation possess a low intrinsic rank. Despite their success in reducing computational demands, existing PEFT methods often perform suboptimally in complex multi-task scenarios. A key challenge is balancing knowledge retention with task-specific specialization across multiple, potentially conflicting, objectives. Many PEFT approaches may exhibit limited generalization and representational capacity across diverse tasks, or suffer from potential interference when adapted to multiple objectives simultaneously.

5.2 Multi-task Adaptation Methods

Multi-task learning (MTL) is fundamental across many real-world domains, including recommendation systems [37, 40], environmental prediction [18, 21, 26], and heterogeneous time-series analysis [9, 53]. Within LLMs, MTL aims to enhance generalization by sharing knowledge across tasks [17, 60, 68], but a key challenge lies in balancing shared knowledge with task-specific specialization [14]. Several strategies have been explored for MTL in LLMs. A common paradigm involves hard parameter sharing, such as T5 [48]. PEFT techniques have been adapted for MTL. For instance, methods like Multi-LoRA [59] and adapter-based methods [47] use small trainable modules into the frozen pre-trained model. Another type of multi-task method is the Mixture-of-Experts (MoE) framework, such as Switch Transformers [11] and GShard [29], which employ a routing mechanism that activates a subset of experts for each input. MoE-LoRA [41] introduces Layer-wise Expert Allocation (MoLA), which assigns experts at different Transformer layers for better adaptability. While powerful, MoE-based approaches often face challenges such as expert load imbalance, which influences training efficiency. Our proposed HyCAM framework offers a novel approach to these challenges by combining CAM with a hybrid strategy for knowledge sharing and specialization, enabling efficient knowledge sharing while preserving task-specific adaptations.

6 Conclusion

In this work, we propose HyCAM, a novel framework designed to enhance multi-task adaptation in Large Language Models by integrating our core CAM mechanism within a hybrid architecture. HyCAM effectively balances generalization and specialization by employing a shared, full-parameter CAM module for broad knowledge retention and multiple specialized, lightweight CAM modules for fine-grained, task-specific feature enhancement. Our dynamic soft-routing strategy, with a load-balancing loss, ensures adaptive knowledge fusion and efficient utilization of these specialized components. Extensive experiments show that HyCAM significantly outperforms existing approaches, demonstrating its ability for efficient adaptation and preserving pre-trained general knowledge.

Acknowledgments

Jingyuan Wang's work was partially supported by the National Natural Science Foundation of China (No. 72171013, 72222022, 72242101), and the Fundamental Research Funds for the Central Universities (JKF-2025017226182). This research was partially supported by Hong Kong Research Grants Council's Research Impact Fund (No.R1015-23), Collaborative Research Fund (No.C1043-24GF), General Research Fund (No.11218325), Institute of Digital Medicine of City University of Hong Kong (No.9229503), National Natural Science Foundation of China (No.62502404), Huawei (Huawei Innovation Research Program), Tencent (CCF-Tencent Open Fund, Tencent Rhino-Bird Focused Research Program), Alibaba (CCF-Alimama Tech Kangaroo Fund No. 2024002), Ant Group (CCF-Ant Research Fund), Kuaishou, Didi (CCF-Didi Gaia Scholars Research Fund), and Bytedance.

7 GenAI Usage Disclosure

This research, including the proposed Contextual Attention Modulation (CAM) and Hybrid Contextual Attention Modulation (HyCAM) framework, the experiment design and the analysis of experimental results, are the original contributions of the authors.

Our work involves efficient multi-task adaptation of LLMs, and we need to use GenAI tools and models during the research process, as follows:

1. Use in experiments: Our study involved evaluating the proposed HyCAM framework for multi-task adaptation in LLMs. To this end, GenAI models, specifically variants from the Llama, Mistral, and Qwen families, are employed as backbone models. These pre-trained LLMs were subjected to fine-tuning using our proposed method and baseline approaches, as described in the experimental sections.

2. Use in Data Origins: Certain datasets utilized in our multi-task benchmark were constructed with the involvement of GenAI. Datasets such as Auto CoT, CodeAlpaca, and WebGPT may incorporate machine generated data, but no additional synthetic data were generated for this study.

The authors declare that they comply with the ACM policy on the usage of GenAI.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[3] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258* (2021).

[4] Erik Brynjolfsson, Danielle Li, and Lindsey Raymond. 2025. Generative AI at work. *The Quarterly Journal of Economics* (2025), qjae044.

[5] Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. 2024. A survey on mixture of experts. *arXiv preprint arXiv:2407.06204* (2024).

[6] Sahil Chaudhary. 2023. Code Alpaca: An Instruction-following LLaMA model for code generation. https://github.com/sahil280114/codealpaca.

[7] 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.

[8] Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. *Free Dolly: Introducing the World's First Truly Open Instruction-Tuned LLM.* https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm

[9] Bowen Du, Xuanxuan Sun, Junchen Ye, Ke Cheng, Jingyuan Wang, and Leilei Sun. 2021. GAN-based anomaly detection for multivariate time series using polluted training set. *IEEE Transactions on Knowledge and Data Engineering* 35, 12 (2021), 12208–12219.

[10] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. 2018. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks* 107 (2018), 3–11.

[11] William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research* 23, 120 (2022), 1–39.

[12] Zichuan Fu, Wentao Song, Yejing Wang, Xian Wu, Yefeng Zheng, Yingying Zhang, Derong Xu, Xuetao Wei, Tong Xu, and Xiangyu Zhao. 2025. Sliding Window Attention Training for Efficient Large Language Models. *arXiv preprint arXiv:2502.18845* (2025).

[13] Zichuan Fu, Xian Wu, Yejing Wang, Wanyu Wang, Shanshan Ye, Hongzhi Yin, Yi Chang, Yefeng Zheng, and Xiangyu Zhao. 2025. Training-free LLM Merging for Multi-task Learning. *arXiv preprint arXiv:2506.12379* (2025).

[14] Chongyang Gao, Kezhen Chen, Jinmeng Rao, Baochen Sun, Ruibo Liu, Daiyi Peng, Yawen Zhang, Xiaoyuan Guo, Jie Yang, and VS Subrahmanian. 2024. Higher layers need more lora experts. *arXiv preprint arXiv:2402.08562* (2024).

[15] 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.

[16] Chenlu Guo, Yuan Wu, and Yi Chang. 2025. NLoRA: Nystr\" om-Initiated Low-Rank Adaptation for Large Language Models. *arXiv preprint arXiv:2502.14482* (2025).

[17] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680* (2024).

[18] Chengkai Han, Jingyuan Wang, Yongyao Wang, Xie Yu, Hao Lin, Chao Li, and Junjie Wu. 2025. Bridging traffic state and trajectory for dynamic road network and trajectory representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 11763–11771.

[19] 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).

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*. 1026–1034.

[21] Kethmi Hirushini Hettige, Jiahao Ji, Shili Xiang, Cheng Long, Gao Cong, and Jingyuan Wang. 2024. Airphynet: Harnessing physics-guided neural networks for air quality prediction. *arXiv preprint arXiv:2402.03784* (2024).

[22] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International conference on machine learning*. PMLR, 2790–2799.

[23] 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).

[24] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).

[25] Sebastian Jaszczur, Aakanksha Chowdhery, Afroz Mohiuddin, Lukasz Kaiser, Wojciech Gajewski, Henryk Michalewski, and Jonni Kanerva. 2021. Sparse is enough in scaling transformers. *Advances in Neural Information Processing Systems* 34 (2021), 9895–9907.

[26] Jiahao Ji, Jingyuan Wang, Yu Mou, and Cheng Long. 2023. Multi-factor spatio-temporal prediction based on graph decomposition learning. *arXiv preprint arXiv:2310.10374* (2023).

[27] Jiahao Ji, Wentao Zhang, Jingyuan Wang, and Chao Huang. 2025. Seeing the unseen: Learning basis confounder representations for robust traffic prediction. (2025).

[28] 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).

[29] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668* (2020).

[30] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691* (2021).

[31] 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 large language models. *arXiv preprint arXiv:2305.10998* (2023).

[32] Xinhang Li, Chong Chen, Xiangyu Zhao, Yong Zhang, and Chunxiao Xing. 2023. E4srec: An elegant effective efficient extensible solution of large language models for sequential recommendation. *arXiv preprint arXiv:2312.02443* (2023).

[33] Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190* (2021).

[34] Yunxiang Li, Zihan Li, Kai Zhang, Ruilong Dan, Steve Jiang, and You Zhang. 2023. ChatDoctor: A Medical Chat Model Fine-Tuned on a Large Language Model Meta-AI (LLaMA) Using Medical Domain Knowledge. *Cureus* 15, 6 (2023).

[35] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.

[36] Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. 2021. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems* 34 (2021), 18878–18890.

[37] Langming Liu, Wanyu Wang, Chi Zhang, Bo Li, Hongzhi Yin, Xuetao Wei, Wenbo Su, Bo Zheng, and Xiangyu Zhao. 2025. Multi-task Offline Reinforcement Learning for Online Advertising in Recommender Systems. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*. 4635–4646.

[38] Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. 2024. When moe meets llms: Parameter efficient fine-tuning for multi-task medical applications. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1104–1114.

[39] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353* (2024).

[40] Ziru Liu, Jiejie Tian, Qingpeng Cai, Xiangyu Zhao, Jingtong Gao, Shuchang Liu, Dayou Chen, Tonghao He, Dong Zheng, Peng Jiang, et al. 2023. Multi-task recommendations with reinforcement learning. In *Proceedings of the ACM web conference 2023*. 1273–1282.

[41] Tongxu Luo, Jiahe Lei, Fangyu Lei, Weihao Liu, Shizhu He, Jun Zhao, and Kang Liu. 2024. Moelora: Contrastive learning guided mixture of experts on parameter-efficient fine-tuning for large language models. *arXiv preprint arXiv:2402.12851* (2024).

[42] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2021. WebGPT: Browser-assisted question-answering with human feedback. In *arXiv*.

[43] Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. 2022. Multi-task learning as a bargaining game. *arXiv preprint arXiv:2202.01017* (2022).

[44] Rui Pan, Xiang Liu, Shizhe Diao, Renjie Pi, Jipeng Zhang, Chi Han, and Tong Zhang. 2024. LISA: Layerwise Importance Sampling for Memory-Efficient Large Language Model Fine-Tuning. *arXiv preprint arXiv:2403.17919* (2024).

[45] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 311–318.

[46] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2020. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247* (2020).

[47] Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. Mad-x: An adapter-based framework for multi-task cross-lingual transfer. *arXiv preprint arXiv:2005.00052* (2020).

[48] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research* 21, 140 (2020), 1–67.

[49] Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. 2022. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. In *International conference on machine learning*. PMLR, 18332–18346.

[50] Mengyang Sun, Yihao Wang, Tao Feng, Dan Zhang, Yifan Zhu, and Jie Tang. 2025. A Stronger Mixture of Low-Rank Experts for Fine-Tuning Foundation Models. *arXiv preprint arXiv:2502.15828* (2025).

[51] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805* (2023).

[52] 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).

[53] 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.

[54] Maolin Wang, Jun Chu, Sicong Xie, Xiaoling Zang, Yao Zhao, Wenliang Zhong, and Xiangyu Zhao. 2025. Put Teacher in Student's Shoes: Cross-Distillation for Ultra-compact Model Compression Framework. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*. 4975–4985.

[55] Maolin Wang, Xiangyu Zhao, Ruocheng Guo, and Junhui Wang. 2025. MetaLoRA: Tensor-Enhanced Adaptive Low-Rank Fine-Tuning. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*. IEEE, 4680–4684.

[56] Maolin Wang, Yao Zhao, Jiajia Liu, Jingdong Chen, Chenyi Zhuang, Jinjie Gu, Ruocheng Guo, and Xiangyu Zhao. 2023. Large multimodal model compression via efficient pruning and distillation at AntGroup. *arXiv preprint arXiv:2312.05795* (2023).

[57] Xiaolei Wang, Xinyu Tang, Wayne Xin Zhao, Jingyuan Wang, and Ji-Rong Wen. 2023. Rethinking the evaluation for conversational recommendation in the era of large language models. *arXiv preprint arXiv:2305.13112* (2023).

[58] Yuhao Wang, Ha Tsz Lam, Yi Wong, Ziru Liu, Xiangyu Zhao, Yichao Wang, Bo Chen, Huifeng Guo, and Ruiming Tang. 2023. Multi-task deep recommender systems: A survey. *arXiv preprint arXiv:2302.03525* (2023).

[59] Yiming Wang, Yu Lin, Xiaodong Zeng, and Guannan Zhang. 2023. Multilora: Democratizing lora for better multi-task learning. *arXiv preprint arXiv:2311.11501* (2023).

[60] Yuhao Wang, Yichao Wang, Zichuan Fu, Xiangyang Li, Wanyu Wang, Yuyang Ye, Xiangyu Zhao, Huifeng Guo, and Ruiming Tang. 2024. Llm4msr: An llm-enhanced paradigm for multi-scenario recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 2472–2481.

[61] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652* (2021).

[62] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems* 33 (2020), 5824–5836.

[63] 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.

[64] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199* (2021).

[65] 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).

[66] Wentao Zhang, Jingyuan Wang, Yifan Yang, et al. 2024. VecCity: A taxonomy-guided library for map entity representation learning. *arXiv preprint arXiv:2411.00874* (2024).

[67] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic Chain of Thought Prompting in Large Language Models. In *The Eleventh International Conference on Learning Representations (ICLR 2023)*.

[68] 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).