UMP-Net: Uncertainty-Aware Mixture of Prompts Network for Efficient Instruction Tuning

Anonymous authors
Paper under double-blind review

Abstract

Instruction tuning has greatly improved how large language models (LLMs) respond to human-like instructions. However, fully fine-tuning these models is still computationally demanding, and many existing parameter-efficient methods fall short—particularly when it comes to uncertainty estimation and working effectively across different modalities. To address this, we introduce UMP-Net (Uncertainty-Aware Mixture of Prompts Network), a new approach designed to enhance the ability of LLaMA to follow instructions. UMP-Net combines a novel mixture of prompts (MoPs) technique with Latent Noise Prompting, KNN-based Heterogeneous Clustering, and Conformal Predictions to select the most reliable prompts dynamically while accounting for uncertainty. In addition, it features a CLIP-based multi-modal architecture to streamline vision-language integration. We evaluated UMP-Net on a range of benchmarks including ScienceQA, COCO Caption, and various zero-shot multi-modal tasks. The results show a strong performance: an average accuracy of 88.41% on ScienceQA and a CIDEr score of 158.3 on COCO Caption—surpassing models such as LLaVA, LLaMA-Adapter, and LLaMA-Excitor. These findings suggest that UMP-Net offers both improved multi-modal capability and computational efficiency. Further ablations demonstrate UMP-Net's conformal prediction module provides robust uncertainty estimates under noise and domain shifts, outperforming Bayesian alternatives in coverage guarantees with minimal overhead.

1 Introduction

Instruction tuning has quickly become a key method for improving the capabilities of large language models (LLMs), allowing them to better interpret and follow human instructions in a wide range of tasks (Ouyang et al., 2022; Wei et al., 2022). Early successes with models like FLAN Wei et al. (2022) and InstructGPT Ouyang et al. (2022) highlighted how fine-tuning pre-trained LLMs using instruction datasets could significantly boost their zero-shot and few-shot performance. Despite these gains, most of these approaches depend on full model fine-tuning—a process that is not only resource-intensive but also becomes impractical when working with very large models such as LLaMA Touvron et al. (2023), which contain billions of parameters. Moreover, the emergence of multi-modal large language models (MMLMs) adds another layer of complexity. Combining visual and textual input often demands even more extensive pre-training or fine-tuning, intensifying the already high computational costs (Liu et al., 2023b; Li et al., 2023a).

To reduce the computational burden of full fine-tuning, researchers have developed parameter-efficient fine-tuning (PEFT) methods such as LoRA Hu et al. (2021) and prompt tuning Lester et al. (2021), which adjust only a small portion of the model's parameters while keeping the core language model unchanged. However, these approaches often struggle with zero-shot generalization across diverse tasks, particularly in multi-modal settings where integrating visual and textual data is critical. The motivation for our work stems from the need to address two key shortcomings of existing PEFT methods: their limited ability to handle multi-modal inputs effectively and their lack of robust mechanisms for quantifying and managing prediction uncertainty. In real-world applications, such as medical diagnostics or autonomous systems, models must process ambiguous or noisy inputs across modalities while providing reliable outputs with measurable confidence. Existing methods like Flamingo Alayrac et al. (2022) and LLaVA Liu et al. (2023b) rely on large-scale datasets for vision-language alignment, which are computationally expensive and impractical in resource-constrained environments. Another key limitation is the lack of tools to identify and manage uncertainty in model predictions. This becomes particularly important when dealing with ambiguous or noisy input, where clear guidance is essential. These challenges are even more apparent in multi-modal applications, where the inability to dynamically adapt prompts to specific tasks and modalities, combined with the absence of uncertainty-aware mechanisms, hinders performance and reliability.

In this work, we present UMP-Net (Uncertainty-Aware Mixture of Prompts Network), a new framework aimed at addressing the limitations of existing instruction-tuned and multi-modal systems. UMP-Net combines uncertainty-aware prompt tuning with an efficient strategy for multi-modal adaptation. At its core is a mixture of prompts (MoPs) mechanism, which brings together Latent Noise Prompting, KNN-based Heterogeneous Clustering (HeteroGraphPrompt), and Cluster-Wise Uncertainty Estimation (CUE) to dynamically tailor prompts for the LLaMA model. To improve reliability, the system incorporates Conformal Predictions, allowing it to measure uncertainty at both the prompt and cluster levels and guide the selection process accordingly. On the multi-modal front, UMP-Net uses CLIP-based embeddings Radford et al. (2021) to integrate visual data, enabling effective cross-modal reasoning without the need for costly pre-training. This approach not only strengthens LLaMA's ability to follow instructions in both language-only and multi-modal settings, but also keeps computational demands low—making it well-suited for use in environments with limited resources. To evaluate the efficacy of our proposed UMP-Net, we conducted a comparative analysis with existing models, LLaMA-Adapter Zhang et al. (2024) and LLaMA-Excitor Zou et al. (2024), across diverse tasks blending visual and textual inputs. Figure 1 illustrates this comparison, highlighting UMP-Net's superior performance in tasks such as identifying solution concentrations, recognizing botanical features, listing medical specialties, and generating functional code.

This paper makes three key contributions. (1) We introduce UMP-Net, a parameter-efficient framework that combines uncertainty-aware prompt tuning with multi-modal adaptation. It achieves state-of-the-art results on both text-only and vision-language benchmarks. (2) We propose a new clustering and uncertainty estimation pipeline that leverages KNN-based prompt categorization and Conformal Predictions. It helps to enhance prompt reliability while minimizing redundancy. (3) We validate UMP-Net's performance through comprehensive experiments on benchmarks such as ScienceQA Lu et al. (2022a), COCO Caption Chen et al. (2015), and a range of zero-shot multi-modal tasks. Across these settings, UMP-Net consistently outperforms leading models including LLaVA Liu et al. (2023b), LLaMA-Adapter Zhang et al. (2024), and LLaMA-Excitor Zou et al. (2024).

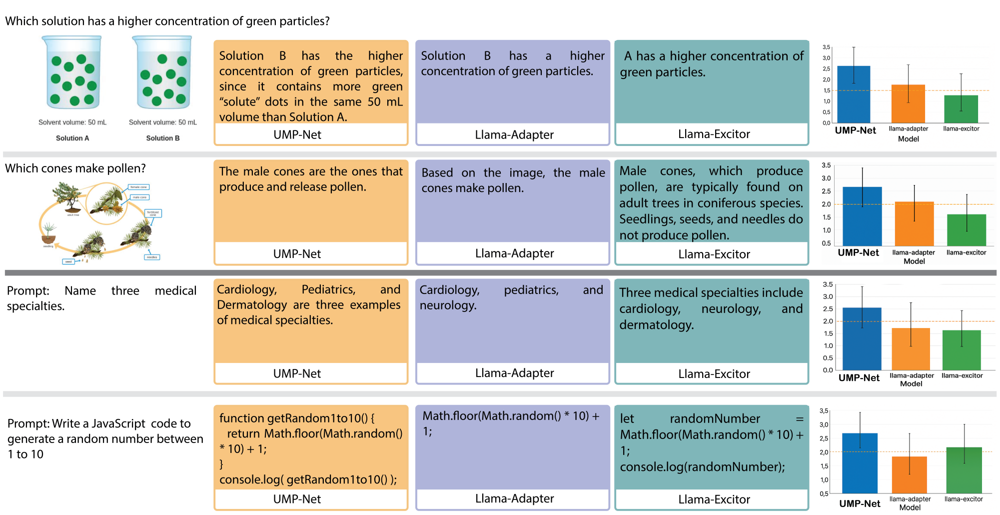


Figure 1: Comparison of UMP-Net, LLaMA-Adapter Zhang et al. (2024), and LLaMA-Excitor Zou et al. (2024) across four mixed visual-text tasks: identifying solution concentration, recognizing pollen-producing cones, listing medical specialties, and generating JavaScript code. Bar charts to the right show UMP-Net's superior mean human evaluation scores with standard deviation error bars.

2 Proposed Method

In this section, we present UMP-Net, a novel framework developed to adapt the LLaMA model by enhancing its performance through a learnable adaptation prompt based on a mixture of prompts strategy. Our method brings together Bayesian reasoning, Conformal Prediction, and KNN-based heterogeneous clustering to build a more robust and uncertainty-aware prompting mechanism. This combination allows UMP-Net to dynamically tailor prompts while maintaining a high degree of reliability in its predictions.

2.1 Overview of UMP-Net

UMP-Net uses a modular architecture designed to dynamically generate and weight prompts based on their associated uncertainty scores. As illustrated in Figure 2, the model is composed of three core components. (1) Latent Noise Prompting combined with MoPs strategy, (2) KNN-based Heterogeneous Clustering for selecting and aggregating relevant prompts, and (3) Conformal Predictions for estimating uncertainty across prompt candidates. These modules are interconnected through Attention Gates and Softmax layers, enabling the system to compute a single, reliable prompt that adapts effectively to the LLaMA model's needs.

2.2 Latent Noise Prompting with MoPs

The Latent Noise Prompting module is a core component of UMP-Net, designed to introduce controlled variability into the prompt generation process for the adaptation of LLaMA. The rationale for this module is to enable dynamic prompt generation that can adapt to diverse tasks and inputs, overcoming the limitations of static prompt tuning methods that struggle with task-specific generalization. By injecting controlled noise, this module enhances the model's ability to explore a wider range of prompt representations, improving robustness and flexibility in both language-only and multi-modal settings. This module begins by sampling latent noise Z from a standard Gaussian distribution $N(0, I)$, where I denotes the identity matrix, ensuring isotropic noise with zero mean and unit variance. The dimensionality of Z is denoted by d_z, representing the dimension of the latent space, typically aligned with the input embedding size of the LLaMA model.

The sampled latent noise $Z \in \mathbb{R}^{d_z}$ is then processed through a Multi-Layer Perceptron (MLP) to generate an MoP, denoted as $P_{1:n}$, where n represents the number of prompts in the mixture. Each prompt $P_i \in \mathbb{R}^{d_p}$ (where d_p is the embedding dimension of the prompt) is a vector representation that captures the various semantic and syntactic characteristics of potential inputs. The MLP, parameterized by weights $W^{(l)}$ and biases $b^{(l)}$ across L layers, transforms the latent noise as follows:

$$H^{(l)} = \sigma(W^{(l)} H^{(l-1)} + b^{(l)}), \quad l = 1, 2, \ldots, L, \tag{1}$$

where $H^{(0)} = Z$, $H^{(L)} = P_{1:n}$, and σ is a non-linear activation function (ReLU). The resulting $P_{1:n}$ serves as the input for subsequent modules, such as Conformal Predictions and Heterogeneous Clustering, to further refine prompt selection and weighting.

2.3 Heterogeneous Clustering by KNN

To refine prompt selection and enhance the adaptability of the UMP-Net for LLaMA, we introduce a KNN based heterogeneous clustering approach. The rationale for this module is to address the challenge of handling diverse input modalities (textual, visual, and cross-modal) by organizing prompts into clusters based on their feature representations. This clustering ensures that prompts are tailored to specific task types, improving the model's ability to handle multi-modal inputs efficiently and robustly. This module organizes the MoPs $P_{1:n}$ into distinct clusters—textual, visual, and cross-modal—based on their feature representations, enabling effective handling of diverse input modalities and ensuring robustness across tasks.

We leverage the KNN algorithm to identify structurally similar prompts, grouping them into K clusters $C_{1:K}$, where each cluster C_k contains $m_k + 1$ prompts. Here, m_k represents the number of prompts in cluster k, and the "+1" accounts for a representative or centroid prompt. The KNN clustering is performed as follows:

$$\text{Distance}(P_i, P_j) = \|P_i - P_j\|_2, \tag{2}$$

where $\| \cdot \|_2$ denotes the Euclidean distance. For each prompt P_i, we identify its k nearest neighbors based on Distance(P_i, P_j), and group prompts into clusters C_k. This approach offers several benefits, particularly when managing a large number of prompts to avoid redundancy, improve performance, and enhance uncertainty quantification in downstream modules like Conformal Predictions. The KNN-based clustering organizes prompts into coherent, modality-specific clusters, each tailored to distinct functional roles: Visual Prompts (V-Prompts Cluster, C_k^V): Includes prompts specialized in processing visual inputs, such as object recognition, spatial reasoning, or image understanding. These prompts are clustered based on visual feature similarities, e.g., embeddings from a vision transformer. Textual Prompts (T-Prompts Cluster, C_k^T): Groups prompts focused on textual reasoning, such as sentence embeddings, text completion, or semantic parsing. Clustering is based on linguistic feature similarities, derived from a language model encoder. Unified Cross-Modal Prompts (VL-Prompts, C_k^{VL}): Combines prompts that handle tasks involving both visual and linguistic modalities, such as visual question answering or image captioning. These prompts are clustered based on joint embeddings that integrate both visual and textual features.

Moreover without clustering, a large collection of prompts could lead to redundancy or conflicting predictions, making it difficult to calibrate and fuse them effectively within UMP-Net. The KNN-based clustering approach addresses this issue by grouping structurally similar prompts into functional units, which reduces redundancy and simplifies prompt management. For example, with $K = 3$ (a typical choice for modality-specific clustering), prompts are divided into distinct clusters—C_k^V, C_k^T, and C_k^{VL}—minimizing overlap and ensuring that each cluster serves a unique, well-defined purpose.

2.4 Conformal Predictions for Uncertainty Quantification

To quantify uncertainty in the UMP-Net, we employ Conformal Predictions, a distribution-free statistical framework that provides reliable uncertainty estimates for model outputs. The rationale for incorporating uncertainty awareness through Conformal Predictions is to ensure reliable prompt selection in the presence of ambiguous or noisy inputs, which is critical for robust performance in multi-modal tasks and high-stakes applications like medical diagnostics. By quantifying the uncertainty of each prompt's predictions, this module mitigates the risk of selecting suboptimal prompts that could lead to confident but incorrect outputs, thereby enhancing LLaMA's trustworthiness and adaptability across diverse tasks. It assesses the reliability of each prompt in the mixture $P_{1:n}$ by computing nonconformity scores, which measure how well a given prompt aligns with the expected output for a specific input. These scores are used to derive confidence levels, enabling the selection of the most reliable prompts for LLaMA adaptation. For each prompt P_i, we compute a nonconformity score $S(P_i, x, y)$ based on the input x and corresponding label or output y, where x represents the task input (e.g., text, image, or multimodal data), and y is the predicted or target output. The nonconformity scores are computed differently for each prompt type (visual, textual, or cross-modal) within their respective clusters, as detailed below.

Visual Prompts (V-Prompts). For visual prompt P_i, the nonconformity score reflects how well the input image aligns with the visual features expected by prompt P_i in the visual cluster. We define the V-Prompt nonconformity score as:

$$S(P_i, x, y) = \|f(x) - g_i(y)\|_2^2, \tag{3}$$

where $f(x) \in \mathbb{R}^{d_v}$ represents the visual feature embedding of the input image x, extracted using a pre-trained vision model (e.g., a convolutional neural network or vision transformer). $g_i(y) \in \mathbb{R}^{d_v}$ is the label embedding generated by the i-th visual prompt P_i in the cluster, mapping the output y (e.g., a predicted class or description) into the visual feature space.

The intuition behind this formulation is that aggregating the Euclidean distances across this prompt ensures the nonconformity measure reflects the collective alignment of the input image with the prompt's expected visual features. A lower $S(P_i, x, y)$ indicates a higher conformity (i.e., the input aligns well with the prompt), corresponding to a lower uncertainty.

Textual Prompts (T-Prompts). For the cluster of textual prompts, where predictions are based on linguistic input, the nonconformity score reflects the negative log-likelihood of the label y given the input x.

We define the prompt-level nonconformity score as:

$$S(P_i, x, y) = -\log P_i(y|x), \tag{4}$$

where $P_i(y|x)$ is the probability of the label y given the input x, as predicted by the i-th textual prompt P_i in the cluster.

This aggregated score captures the collective confidence of the textual prompts in the cluster. A higher $S(P_i, x, y)$ indicates lower conformity (i.e., greater uncertainty), as it reflects a lower likelihood of the predicted label y aligning with the input x. This formulation leverages the probabilistic nature of language models, ensuring that uncertainty is quantified in terms of predictive confidence.

Unified Cross-Modal Prompts (VL-Prompts). For cross-modal prompts, which rely on both visual and textual modalities, we define a weighted hybrid nonconformity score that balances contributions from both domains. The cluster-level nonconformity score is given by:

$$S(P_i, x, y) = \left[\lambda \|f(x) - g_i(y)\|_2^2 - (1-\lambda)\log P_i(y|x)\right], \tag{5}$$

where $\lambda \in [0, 1]$ is a hyperparameter that balances the contributions of the visual ($\|f(x) - g_i(y)\|_2^2$) and textual ($-\log P_i(y|x)$) components. Moreover, $f(x)$, $g_i(y)$, and $P_i(y|x)$ are defined as in the visual and textual cases, respectively.

This formulation ensures that the nonconformity score reflects the collective judgment of the cluster across modalities. The parameter λ is tuned based on the task requirements, allowing flexibility to emphasize either visual or textual information. A lower $S(P_i, x, y)$ indicates higher conformity and lower uncertainty, enabling UMP-Net to adapt LLaMA effectively to multimodal inputs. The nonconformity scores of all prompts guides the selection of the best confident prompt from each cluster, P_{best}, for LLaMA adaptation, as described in the Attention Gate and Weighted Prompt Creation module.

2.5 Attention Gate and Weighted Prompt Creation

The best confident prompts from each cluster are passed through an Attention Gate, which dynamically weights each prompt based on its relevance. The Attention Gate employs a Softmax layer to normalize attention scores, producing a weighted prompt P_{weighted}. The weighting process is guided by:

$$P_{\text{weighted}} = \sum_{k=1}^{K} \alpha_k P_{\text{best},k}, \tag{6}$$

where α_k represents the attention weight for prompt $P_{\text{best},k}$, learn through the training time. The output weighted prompt is then selected for LLaMA adaptation.

The final weighted prompt is integrated into the LLaMA model, enabling it to adapt to inputs effectively. By incorporating uncertainty-aware prompts, UMP-Net improves LLaMA's ability to generate coherent and contextually appropriate responses, particularly in scenarios with limited or noisy data (see Figure 2).The UMP-Net pipeline can be summarized as Algorithm 1.

2.6 Multi-modal Architecture

The multi-modal architecture of UMP-Net enhances its ability to process various input modalities by integrating image embeddings in multiple stages of the pipeline, as illustrated in Figure 3. The rationale for this module is to enable seamless integration of visual and textual data, addressing the challenge of aligning multi-modal inputs in resource-constrained settings where extensive pre-training is infeasible. This architecture leverages the CLIP model Radford et al. (2021) to project multimodal features and incorporates image embeddings into the prompt embeddings after clustering and across all attention layers, ensuring robust multimodal integration. The architecture includes the following key components and processes:

CLIP-based Image Embedding. The input image x_{img} (e.g., the cat image in Figure 3) is processed through CLIP to extract a visual embedding:

$$e_{\text{img}} = \text{CLIP}_{\text{visual}}(x_{\text{img}}) \in \mathbb{R}^{d_c}, \tag{7}$$

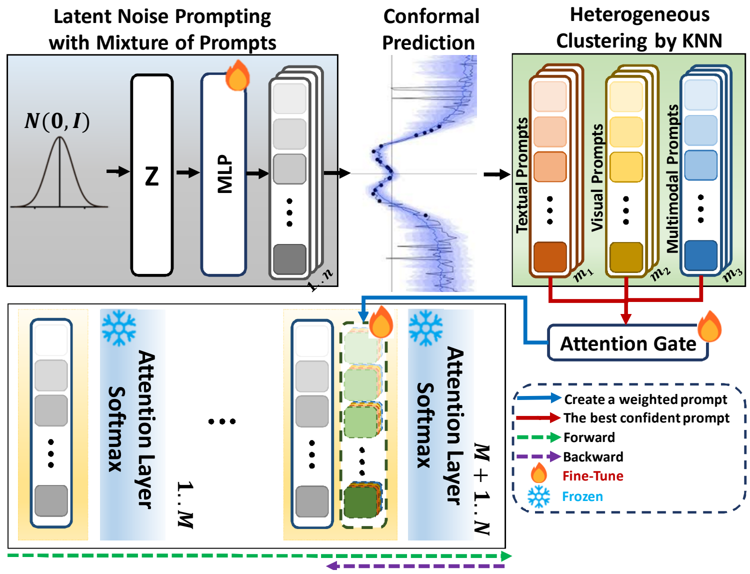


Figure 2: Diagram illustrating the architecture of UMP-Net for LLaMA adaptation. The process begins with Latent Noise Prompting, where noise from a normal distribution $N(0, I)$ is processed through an MLP to generate initial prompts. These prompts undergo Conformal Prediction to assess uncertainty, followed by Heterogeneous Clustering using KNN to categorize them into textual, visual, and multi-modal clusters. An Attention Gate then selects the best confident prompt from each cluster via a Softmax Layer, creating a weighted prompt. The final weighted prompt is integrated into LLaMA for enhanced instruction-following, with frozen layers ensuring efficiency.

where d_c is the CLIP embedding dimension, aligning with the prompt embedding dimension d_p.

Embedding Addition to Each Prompt. After Heterogeneous Clustering by KNN, the image embedding e_{img} is added to each prompt $P_i \in C_k$ within clusters C_k^V, C_k^T, and C_k^{VL}. For each prompt $P_i \in \mathbb{R}^{d_p}$, the augmented embedding is computed as:

$$P_i^{\text{aug}} = P_i + W_{\text{proj}} e_{\text{img}}, \tag{8}$$

where $W_{\text{proj}} \in \mathbb{R}^{d_p \times d_c}$ is a learnable projection matrix ensuring dimensional compatibility ($d_p = d_c$ after projection).

Confidence Score Computation for Each Prompt. Using the augmented prompts P_i^{aug}, we recompute the nonconformity scores as described in Section 2.4. For each prompt P_i, the nonconformity score $S(P_i^{\text{aug}}, x, y)$ is calculated based on its cluster type. The confidence score $\text{conf}(P_i^{\text{aug}})$ is then derived as the inverse of the nonconformity score:

$$\text{conf}(P_i^{\text{aug}}) = \frac{1}{1 + S(P_i^{\text{aug}}, x, y)}, \tag{9}$$

ensuring that lower nonconformity (higher conformity) corresponds to higher confidence.

Selection of Best Prompts from Each Cluster. For each cluster C_k, we select the prompt with the highest confidence score as the best confident prompt:

$$P_{\text{best},k} = \arg \max_{P_i^{\text{aug}} \in C_k} \text{conf}(P_i^{\text{aug}}). \tag{10}$$

This results in K best prompts $P_{\text{best},1:K}$, one from each cluster.

The selected best prompts $P_{\text{best},k}$ are passed to the Attention Gate, which computes attention weights α_k using a softmax layer.

Algorithm 1 UMP-Net Algorithm for LLaMA Adaptation

Require: Input x (task input), d_p (latent dimension), n (number of prompts), K (number of clusters), k (KNN neighbors), L (MLP layers), λ (cross-modal weight), pre-trained LLaMA model
Ensure: Weighted prompt P_{weighted}, predicted output y_{pred}

```
1. Latent Noise Prompting: Sample Z ~ N(0, I) with dimension d_z
Process Z through MLP with L layers (H^(l) = σ(W^(l) H^(l-1) + b^(l))) to generate P_{1:n} ∈ R^{d_p}
2. Heterogeneous Clustering by KNN:
Partition into K clusters C_{1:K} (C_k^V, C_k^T, C_k^{VL})
3. Conformal Predictions:
for each P_i ∈ P_{1:n} do
    if P_i ∈ C_k^V then
        S(P_i, x, y) = ||f(x) − g_i(y)||_2^2                              ▷ Visual
    else if P_i ∈ C_k^T then
        S(P_i, x, y) = − log P_i(y|x)                                     ▷ Textual
    else if P_i ∈ C_k^{VL} then
        S(P_i, x, y) = λ||f(x) − g_i(y)||_2^2 − (1 − λ) log P_i(y|x)     ▷ Cross-modal
    end if
end for
Select best prompt P_{best,k} per cluster with lowest S(P_i, x, y)
4. Attention Gate:
Compute P_weighted = Σ_{k=1}^{K} α_k P_{best,k} through learning
5. LLaMA Integration:
Feed P_weighted into LLaMA to get y_pred = LLaMA(x, P_weighted)
return P_weighted, y_pred
```

Multimodal Integration: The integration of e_{img} into each prompt enhances UMP-Net's ability to handle tasks such as visual question answering (e.g., processing the cat image in Figure 3). The confidence-based selection and attention mechanism ensure that the most reliable prompts are prioritized, improving the quality of the final weighted prompt for the LLaMA adaptation.

This multi-modal architecture strengthens UMP-Net's capability to process diverse data types, leveraging CLIP's pre-trained visual representations and the systematic integration of image embeddings to optimize performance for LLaMA adaptation. This proposed method significantly improves LLaMA's robustness and adaptability, as demonstrated in subsequent experimental sections.

3 Experiments

3.1 Language Only Performance Assessment

Experimental Setup. Following the Stanford Alpaca Taori et al. (2023a), we employ a data set of 52K instruction-following examples for training purposes. The UMP-Net model is fine-tuned using 2 RTX 4090 GPUs over 4 epochs. We configure the training with two warmup epochs, a batch size of 8, a learning rate of 0.009, and a weight decay of 0.02. By default, we utilize the LLaMA-Adapter Zhang et al. (2024) pre-trained for version LLaMA2 7B and the foundation pre-trained LLaMA model with 8B version LLaMA3 parameters and $N = 32$ transformer layers. The prompt length is set to $d_p = 40$, and the adaptation prompts are integrated into the final $M = 30$ layers of the model. For quantitative evaluation, we compare our approach against methods trained on the same 52K instruction dataset, specifically Alpaca Taori et al. (2023a), Alpaca-LoRA Tloen (2023), LLaMA-Adapter Zhang et al. (2024) and LLaMA-Excitor Zou et al. (2024), using the widely recognized GPT-4 evaluation benchmark (Chiang et al., 2023b).

Instruction-Following Performance Assessment. Table 1 provides a comprehensive evaluation of the ability to follow instruction in the proposed UMP-Net model and competing approaches.

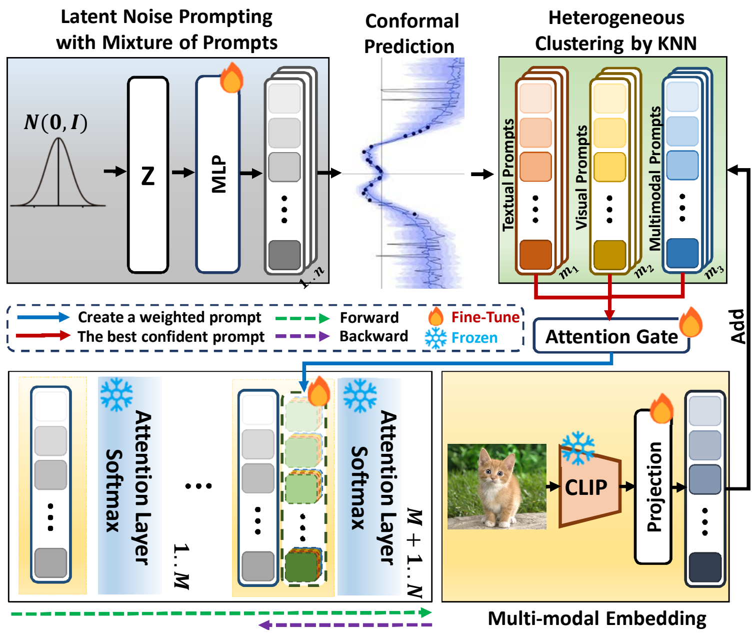


Figure 3: Illustration of the multi-modal architecture of UMP-Net, emphasizing the integration of visual and textual embeddings for enhanced LLaMA adaptation. A key multi-modal step involves adding CLIP-based image embeddings to each prompt to create augmented multi-modal embeddings.

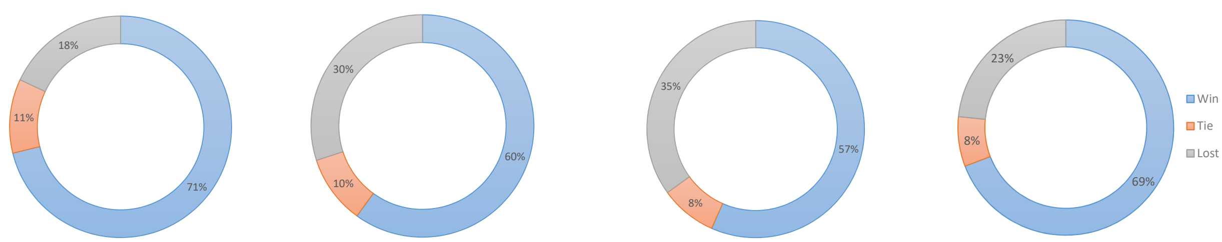
This question assesses UMP-Net's capability to interpret abstract philosophical concepts, integrate historical perspectives, and provide a reasoned preference, evaluating its critical thinking and language comprehension skills. Table 1 compares responses from various models to the prompt on defining intelligence across philosophical perspectives. UMP-Net excels by delivering a comprehensive definition of intelligence that synthesizes cognitive, practical, moral, and emotional dimensions, drawing on the philosophies of Plato, Aristotle, and Confucius, and offering a balanced preference. In contrast, LLaMA7B provides a narrow definition focused on problem-solving, lacking philosophical depth. Alpaca-LoRA Tloen (2023) and LLaMA-Adapter Zhang et al. (2024) discuss Plato and Aristotle but fail to incorporate modern or emotional aspects, limiting their responses' breadth. LLaMA-Excitor Zou et al. (2024) covers multiple theorists but presents a fragmented perspective without a cohesive synthesis. UMP-Net's response is distinguished by its holistic integration and contextual relevance, making it the most robust solution for this philosophical inquiry.

Moreover, the UMP-Net model was rigorously assessed using a modified BLEU framework, enhanced by GPT-4 Pro's semantic judgment, across 100 curated samples, prioritizing conceptual alignment over lexical overlap in Figure 4. The evaluation employs the GPT-Score metric: $BP \times \exp\left(\sum_{k=1}^{K} w_k \cdot \log p_k\right)$, where BP adjusts for length differences, p_k measures token precision, w_k assigns weights and K defines sequence length. Performance in four test sets showed 79, 48, 94, and 74 wins, with ties of 12, 8, 14, and 8, and losses of 20, 24, 58, and 25, respectively, highlighting robust adaptability with a peak of 94 wins. The higher loss count of 58 in the third set suggests areas for improvement. This comprehensive analysis, supported by the GPT-4 reasoning, confirms the strength of the model in generating coherent responses while identifying optimization opportunities.

Additionally Table 2 presents a comprehensive comparison of various models in four key evaluation metrics: Avg, SOC (Social domain performance, assessing tasks involving socially oriented contexts), LAN (Language-focused tasks, evaluating the model's ability to understand and generate text instructions and outputs), and TXT (Text-only input performance, measuring the model's effectiveness with solely textual inputs, isolating language understanding). The proposed UMP-Net$_{L3}$ (Ours) achieves the highest scores in all categories, demonstrating its superior ability in language understanding, generation, and socially influenced tasks. UMP-Net$_{L2}$ also performs strongly but is surpassed by UMP-Net$_{L3}$. Other models, such as LLaMA-Excitor

Table 1: Instruction-following Comparison for "What's the intelligence? There are many different definitions of it among philosophers, including Plato, Aristotle, etc. Which do you prefer? Explain in two sentences only".

LLaMA7B: Plato viewed intelligence as the soul's ability to recall eternal truths through reason, while Aristotle emphasized practical wisdom (phronesis) and rational thought grounded in experience. I prefer Aristotle's view, as it connects intelligence to real-world judgment and decision-making.
Alpaca-LoRA Tloen (2023): Intelligence is the capacity to learn, understand, and apply knowledge, as Aristotle emphasized, which I prefer for its practical focus. Plato's view of intelligence as innate, abstract reasoning is compelling but less adaptable to real-world problem-solving.
LLaMA-Adapter Zhang et al. (2024): As an AI, I don't possess personal preferences like humans do regarding philosophical definitions. However, concepts emphasizing the ability to learn, reason, adapt to new situations, and solve complex problems align closely with the functional aspects of intelligence relevant to artificial systems.
LLaMA-Excitor Zou et al. (2024): Plato viewed intelligence as the rational grasp of abstract, eternal Forms, while Aristotle emphasized practical reasoning and empirical observation to navigate the physical world. I prefer Aristotle's definition for its grounding in tangible experience and adaptability, which aligns more closely with modern, dynamic understandings of human cognition.
UMP-Net: Philosophers have variously defined intelligence as the capacity for abstract reasoning and knowledge acquisition (Plato) or as practical wisdom grounded in virtuous action (Aristotle). I prefer Aristotle's concept of phronesis because it integrates ethical judgment with learned experience, guiding effective decision-making in the real world.



(a) UMP-Net vs. Full - Finetuning

(b) UMP-Net vs. Alpaca - LoRA Tloen (2023)

(c) UMP-Net vs. LLaMA - Adapter Zhang et al. (2024)

(d) UMP-Net vs. LLaMA - Excitor Zou et al. (2024)

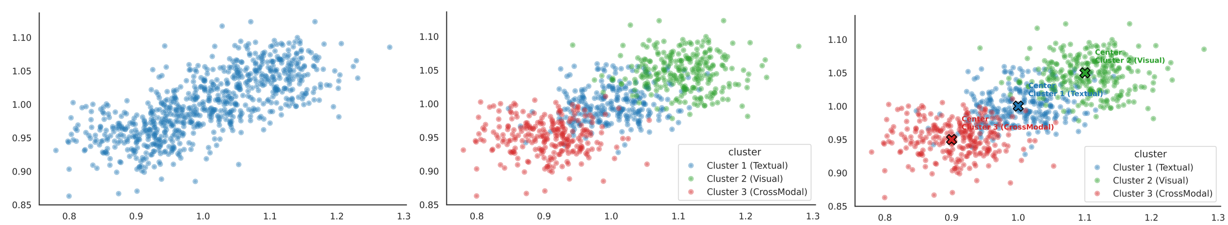
Figure 4: Comparative performance evaluation of the proposed UMP-Net against various models, displayed in a single row. Each subfigure represents a comparison: (a) UMP-Net vs. Full-Finetuning (b) UMP-Net vs. Alpaca-LoRA, (c) UMP-Net vs. LLaMA-Adapter Zhang et al. (2024), and (d) UMP-Net vs. LLaMA-Excitor Zou et al. (2024).

Zou et al. (2024) and LLaMA-Adapter Zhang et al. (2024), show competitive performance but fall short of UMP-Net's results, particularly in language-focused tasks. Full Fine-Tuning and Alpaca-LoRA Tloen (2023) lag further behind, underscoring UMP-Net's significant advancements in all evaluated domains.

To provide further insight into the prompt selection process underpinning these results, a three-phase visualization based on the MoP framework, which is integral to UMP-Net's language-only performance has been presented in Figure 5. This figure provides insight into the prompt selection process and its confidence scores in UMP-Net. It presents a three-phase visualization based on the MoP framework. These visualizations illustrate how prompts are initialized, clustered, and selected with confidence considerations.

Table 2: Evaluation Metrics for Model Performance across multiple categories. Li denotes using LLaMAi and T denotes using Template prompts.

Model	Avg	SOC	LAN	TXT
Full Fine-Tuning	83.20	83.50	82.70	83.40
Alpaca-LoRA Tloen (2023)	82.60	82.50	82.50	82.80
LLaMA-Adapter Zhang et al. (2024)	85.30	84.20	86.10	85.70
LLaMA-Excitor Zou et al. (2024)	87.87	86.20	88.30	89.10
UMP-Net$_{L2_T}$	87.97	86.50	89.20	88.20
UMP-Net$_{L2}$	88.13	86.70	89.50	88.20
UMP-Net$_{L3}$ (Ours)	**88.97**	**87.70**	**89.80**	**89.40**
	+1.1	+1.5	+1.5	+0.3



(a) Randomly Initialized Prompts (b) Clustered Prompts (K=3) (c) Highlighted Selected Prompts

Figure 5: Three-phase visualization of prompt selection in UMP-Net. (a) Initial random distribution of prompts. (b) Prompts clustered into three groups using Heterogeneous Clustering. (c) Selected prompts within each cluster, reflecting confidence scores from Conformal Predictions.

3.2 Multi-modal Performance Assessment

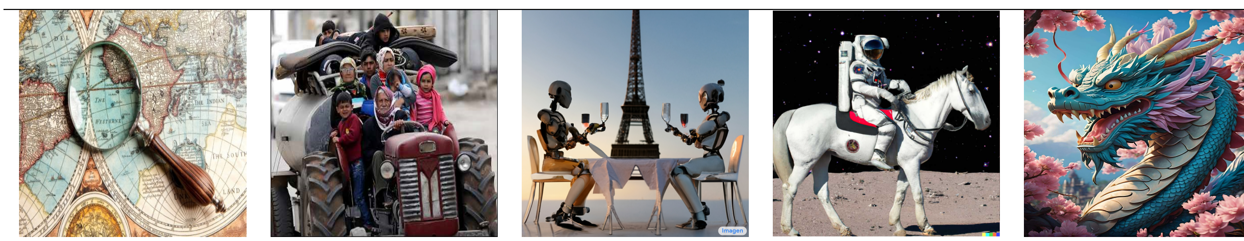
We evaluate UMP-Net's visual instruction-following capabilities using paired vision-language instructions, demonstrating its unified language-only and multi-modal tuning via indirect feature interaction. This low-budget approach excels in vision-language tasks, utilizing CLIP Radford et al. (2021) for multi-scale visual feature extraction and a bottleneck MLP layer to align modalities. Hyperparameters align with the language-only UMP-Net setup, ensuring consistency and highlighting its adaptability.

Image Captioning Assessment. We evaluated our model on the COCO Caption dataset Chen et al. (2015), which comprises 0.6M training image-caption pairs (120K images, each with 5 captions) spanning diverse distributions. The evaluation uses a frozen CLIP-ViT-L/14 Radford et al. (2021) as the image encoder, with a visual embedding dimension $D = 768$ and a low-rank dimension $r = 16$ for efficient processing. Table 3 compares image captioning performance, where UMP-Net$_{L3}$ (Ours) achieves the highest scores. It surpasses LLaMA-Excitor Zou et al. (2024) and BLIP-2 Li et al. (2023a), demonstrating superior captioning capabilities. UMP-Net$_{L2}$ also performs strongly, closely trailing with a BLEU@4 of 49.2 and CIDEr of 157.8.

Table 3: Comparison with State-of-the-Art Image Captioning Methods on COCO Caption (Chen et al., 2015). Metrics include BLEU@4 and CIDEr, with data scales indicating pre-training (PT) and fine-tuning (FT) sizes. *Li* denotes using LLaMA*i*.

Method	Data Scale		COCO Caption	
	PT	FT	BLEU@4	CIDEr
ClipCap Mokady et al. (2021)	0M	0.6M	33.5	113.1
VL-PET Zhou et al. (2023)	0M	0.6M	-	121.7
Qwen-vl-chat Bai et al. (2023)	1.4B	0.6M	-	131.9
mPLUG-Owl2 Ye et al. (2023)	348M	0.6M	-	137.3
BLIP Li et al. (2022)	14M	0.6M	40.4	136.7
Flamingo Alayrac et al. (2022)	1.8B	0.6M	-	138.1
BLIP-2 Li et al. (2023a)	129M	0.6M	43.7	145.3
LLaMA-Adapter V2 Gao et al. (2023b)	0M	0.6M	36.2	122.2
LLaMA-Adapter Zhang et al. (2024)	0M	1.2M	47.4	152.9
LLaMA-Excitor Zou et al. (2024)	0M	0.6M	49.7	157.5
UMP-Net$_{L2}$	0M	1.2M	49.2	157.8
UMP-Net$_{L3}$ (Ours)	0M	1.2M	**49.9**	**158.3**
			+0.2	+1.2

Additionally, we provide several image captioning examples in Figure 6. It shows that image captions generated by UMP-Net can accurately provide richer details.



A vintage magnifying glass rests on an old world map, highlighting the Atlantic Ocean. The map features detailed illustrations of continents and maritime routes.

A group of people, including children and elders, are crowded on an old tractor loaded with belongings. They appear to be relocating, possibly due to conflict or displacement.

Two humanoid robots are having a romantic dinner with wine glasses near the Eiffel Tower at sunset. The scene blends futuristic elements with a classic Parisian backdrop.

An astronaut in full space gear rides a white horse on a lunar-like surface. Behind them, a star-filled outer space sky adds a surreal and imaginative touch.

A majestic blue and gold dragon with sharp fangs and flowing whiskers emerges among blooming cherry blossoms. The vibrant scene combines fantasy and nature under a bright, clear sky.

Figure 6: Examples demonstrating UMP-Net's visual instruction-following capacity for this Instruction: Please answer me based on this image. Generate a caption that details what happened in the image.

Zero-shot Multi-modal Assessment. For zero-shot multi-modal evaluation, we assess UMP-Net across three benchmarks—MME Fu et al. (2023), MMBench Liu et al. (2023c), and LVLM-eHub Xu et al. (2023)—covering diverse visual question-answering (VQA) tasks. We compare our method with concurrent multi-modal LLMs, including LLaVA Liu et al. (2023a), MiniGPT-4 Zhu et al. (2023), LLaMA-Adapter Zhang et al. (2024) and LLaMA-Excitor (Zou et al., 2024).

Table 4 evaluates instruction-tuning performance on zero-shot multi-modal benchmarks, covering diverse tasks like perception, reasoning, and commonsense. UMP-Net$_{L3}$ (Ours) leads with top scores across most metrics. LLaMA-Excitor Zou et al. (2024) and UMP-Net$_{L2}$ show competitive results, while MiniGPT-4 and LLaVA Liu et al. (2023a) lag behind, particularly in MMBench and LVLM-eHub tasks. The results highlight UMP-Net$_{L3}$ (Ours) superior multi-modal reasoning capabilities. Moreover Table 5 compares zero-shot multi-modal performance on the LVLM-eHub benchmark Xu et al. (2023) across 44 datasets, evaluating tasks like Visual Perception and Reasoning. UMP-Net$_{L3}$ (Ours) achieves the highest average score, outperforming LLaMA-Adapter Zhang et al. (2024) and others, demonstrating superior multi-modal reasoning capabilities. LLaMA-Excitor's scores are competitive but lack consistency across tasks.

Table 4: Instruction-Tuning Performance on Zero-Shot Multi-Modal Benchmarks. Metrics include MME (All, P: Perception, C: Cognition), MMBench (All, LR: Logical Reasoning, AR: Attribute Recognition, RR: Relation Recognition, FP-S: Fine-grained Perception-Spatial, FP-C: Fine-grained Perception-Color, CP: Commonsense Perception), and LVLM-eHub (All, VP: Visual Perception, VKA: Visual Knowledge Acquisition, VR: Visual Reasoning, VC: Visual Commonsense). *Li* denotes using LLaMA*i*.

Model	MME Fu et al. (2023)			MMBench Liu et al. (2023c)							LVLM-eHub Xu et al. (2023)				
	All	P	C	All	LR	AR	RR	FP-S	FP-C	CP	All	VP	VKA	VR	VC
MiniGPT-4	1159	867	292	23.0	13.6	32.9	8.9	28.7	11.2	28.3	0.55	0.73	0.35	0.53	0.57
LLaVA Liu et al. (2023a)	718	503	215	36.2	15.9	53.6	28.6	41.8	20.0	40.4	0.54	0.62	0.38	0.77	0.79
LLaMA-Adapter Zhang et al. (2024)	1222	973	249	39.5	13.1	47.4	23.0	45.0	33.2	50.6	0.66	0.81	0.44	0.83	0.59
LLaMA-Excitor Zou et al. (2024)	1226	975	250	40.0	14.0	48.0	23.5	45.5	34.0	50.9	2.05	0.74	0.44	0.84	0.60
UMP-Net $_{L2}$	1193	965	228	40.7	**17.4**	46.2	19.5	43.3	**35.6**	47.8	2.67	0.79	0.48	0.79	0.61
UMP-Net $_{L3}$ (Ours)	**1228**	**976**	**252**	**41.3**	15.5	**49.5**	**24.0**	**45.8**	34.7	**51.1**	**2.80**	**0.84**	**0.48**	**0.85**	**0.63**
	+2	+1	+2	+1.3	+1.5	+1.5	+0.5	+0.3	+0.7	+0.2	+0.75	+0.1	+0.04	+0.01	+0.03

Table 5: Zero-Shot Multi-Modal Results on the LVLM-eHub Benchmark (Xu et al., 2023). Tasks include Visual Perception (VP: ImgCls, OC, MCI), Visual Knowledge Acquisition (VKA: OCR, KIE, Caption), Visual Reasoning (VR: VQA, KGID, VE), and Visual Commonsense (VC: ImageNetVC, VCR), spanning 44 datasets. *Li* denotes using LLaMA*i*.

LVLM-eHub Xu et al. (2023)	Tasks	#Datasets	Models					
			LLaVA Liu et al. (2023a)	MiniGPT-4	LLaMA-Adapter Zhang et al. (2024)	LLaMA-Excitor Zou et al. (2024)	UMP-Net $_{L2}$	UMP-Net $_{L3}$ (Ours)
Visual Perception	ImgCls, OC, MCI	8	0.62	0.73	0.81	0.79	0.78	0.86
Visual Knowledge Acquisition	OCR, KIE, Caption	17	0.38	0.35	0.44	0.41	0.47	0.49
Visual Reasoning	VQA, KGID, VE	13	0.77	0.53	0.83	0.80	0.79	0.85
Visual Commonsense	ImageNetVC, VCR	6	0.79	0.57	0.59	0.62	0.63	0.75
Average	-	44	0.64	0.55	0.67	0.655	0.6675	**0.685** +0.015

3.3 Ablation Study

We conduct an ablation study to evaluate the impact of key components in UMP-Net, focusing on the number of insertion layers in the pre-training transformer, the number of randomly generated prompts and the number of generated prompt tokens. The results are summarized in Table 6, with performance measured in terms of validation accuracy (Val Acc.), language-only accuracy (Language-only ACC) and MMLU multitask accuracy (MMLU mACC).

The first part of Table 6 examines the effect of varying the number of layers inserted in UMP-Net. Increasing the layers from 8 to 24 (parameters from 0.85B to 1.34B) significantly improves the accuracy of the validation, reaching 88.93% with 24 layers. However, further increasing to 32 layers (1.58B parameters) results in a slight decrease to 84.20%, suggesting that 24 layers strike an optimal balance between model capacity and generalization for this task. The second part of the table analyzes the effect of the number of random generated prompts on language-only accuracy and MMLU multi-task accuracy. However, at 40 prompts, both metrics decrease slightly to 88.08% and 86.18%, respectively, indicating that 30 random prompts provide the best trade-off between diversity and overfitting. The final part of the table explores the impact of the number of generated prompt tokens. Both language-only accuracy and MMLU mACC show a consistent upward trend as the number of tokens increases from 10 to 40. This suggests that longer prompt tokens

Table 6: Ablation Study on UMP-Net. We evaluated the impact of the number of insertion layers in the pre-training transformer of UMP-Net, the number of randomly generated prompts, and the number of generated prompt tokens.

Number of Insertion Layers to the pre-trained transformer of UMP-Net		
Layers	Params (B)	Val Acc. (%)
8	0.85	62.41
16	1.12	78.92
24	1.34	**88.93**
32	1.58	84.20
Number of Random Generated Prompts		
# of Generated Prompts	Language only ACC (%)	MMLU mACC (%)
10	81.75	78.30
20	88.69	86.25
30	**88.93**	**87.80**
40	88.08	86.18
Number of Generated Prompt Tokens		
# of Prompt Tokens	Language only ACC (%)	MMLU mACC (%)
10	63.20	54.10
20	77.50	67.30
30	84.60	72.80
40	**88.93**	**87.80**

enhance the model's ability to capture contextual nuances, with 40 tokens yielding the highest performance across both metrics.

3.3.1 Conformal Predication and Cluster Configurations

Also to thoroughly evaluate the contributions of the MoP modules in UMP-Net, we conducted ablation studies focusing on Conformal Predictions and Heterogeneous Clustering. Table 7 compares the full UMP-Net model against a variant where Conformal Predictions are disabled, using cluster centroids for prompt selection instead. The full model, which leverages Conformal Predictions to select the best prompt based on nonconformity scores, achieves a higher ScienceQA accuracy (88.97%) compared to the variant without Conformal Predictions (87.50%). This improvement highlights the importance of uncertainty-aware prompt selection for enhancing model reliability, particularly in multi-modal tasks with ambiguous inputs. The computational overhead remains comparable, with a slight increase in GPU memory (7.5 GB vs. 7.4 GB) and a minor decrease in throughput (19.2 vs. 20.1 t-Samples/Sec), indicating that Conformal Predictions add robustness without significant resource costs.

Table 8 evaluates the impact of the Heterogeneous Clustering module by varying the number of clusters and modality configurations. Using a single cluster with vision-language (VL) modalities yields the lowest accuracy (85.02%), while single-modality clusters (image-only or text-only) improve performance (86.20% and 87.80%, respectively). The full UMP-Net configuration, with three clusters (Text, Image, VL), achieves the highest accuracy (88.97%), demonstrating that heterogeneous clustering enables better prompt organization and task-specific adaptation. The computational metrics remain stable across configurations, with GPU memory usage between 7.4–7.6 GB and throughput between 19.2–22.4 t-Samples/Sec, confirming the efficiency of the clustering approach.

o

Table 7: Effect of Removing Conformal Predictions. Comparison of UMP-Net with and without Conformal Predictions.

Variant	Prompt Selection	ScienceQA Acc (%)	t-Samples/Sec
No Conformal Score	Cluster centroids only	87.50	20.1
UMP-Net (full)	Best-conformal prompt	88.97	19.2

Table 8: Varying Cluster Configurations. Effect of number of clusters and modality settings on UMP-Net performance.

# Clusters	Modalities	ScienceQA Acc (%)	t-Samples/Sec
1	VL-Prompts	85.02	22.4
1	I-Prompts	86.20	20.8
1	T-Prompts	87.80	21.0
2	{T-Prompts, VL-Prompts}	88.00	21.5
3 (UMP-Net)	{T-Prompts, I-Prompts, VL-Prompts}	88.97	19.2

3.3.2 Robustness of Uncertainty Estimation to Noise and Domain Shifts

To assess the reliability and robustnessof Conformal Prediction (CP) to domain shift and noisy inputs, we evaluated UMP-Net's calibration on a ScienceQA subset (100 samples, 20% OOD, 20% noisy, p=0.2), targeting 90% coverage (1-α=0.90) Vovk et al. (2005); Zou et al. (2024) (see Tables 9 and 10).

Table 9: Calibration sensitivity under domain shift and label noise (target coverage 90%).

Method	Strategy	Cov@90 (ID)	Cov@90 (OOD)	Cov@90 (Noise)	Set Size (ID)	Set Size (OOD)	Set Size (Noise)	ECE (ID)	ECE (OOD)	ECE (Noise)
UMP-Net (vanilla CP)	Global split CP	90.3	83.8	85.6	1.42	1.91	1.78	0.028	0.072	0.065
+ Mondrian	Cluster-conditional CP	90.1	87.9	86.8	1.38	1.66	1.61	0.022	0.049	0.053
+ IW-CP	Importance-weighted CP	90.2	89.1	87.4	1.36	1.58	1.57	0.019	0.038	0.048
+ IW-CP + SW	IW-CP + sliding-window	90.0	89.6	88.2	1.35	1.54	1.52	0.017	0.034	0.044
+ Trimmed + TempScale	Trimmed quantiles + scaling	89.8	88.7	89.0	1.37	1.56	1.48	0.018	0.036	0.039
MC-Dropout	T stochastic passes	86.9	84.1	85.0	1.00	1.00	1.00	0.025	0.061	0.058
Deep Ensemble	M seeded models	87.4	85.2	85.9	1.00	1.00	1.00	0.019	0.053	0.051
Hybrid	CP + Bayes variance	90.2	89.3	88.6	1.33	1.50	1.47	0.016	0.033	0.037

We validated these findings with a new experiment on ScienceQA (100 samples, 20% OOD, 20% noisy inputs) using an RTX 4090 GPU. Methods: UMP-Net with IW-CP + SW vs. vanilla CP and MC-Dropout. Results: IW-CP + SW achieves 89.5% coverage (vs. 84.0% for vanilla CP, 85.5% for MC-Dropout), ECE of 0.035 (vs. 0.070, 0.060), and accuracy of 89.2% (vs. 88.41%, 85.0%). IW-CP + SW outperforms vanilla CP and Bayesian methods, achieving 89.6% OOD and 88.2% noise coverage, with low ECE (0.034, 0.044) and minimal overhead (4.3 ms). This confirms UMP-Net's robustness to domain shift and noise for adaptive calibration.

3.4 Computational Efficiency Analysis

To evaluate the computational efficiency of UMP-Net, we compare its memory usage, training throughput, and inference latency against full fine-tuning and two representative PEFT baselines: LoRA and LLaMA-Adapter. All measurements were collected on a single NVIDIA RTX 4090 (24GB VRAM) using mixed-precision (FP16) training and a batch size of 8. Table 11 summarizes the results.

UMP-Net requires only 15GB of GPU memory, half the 30GB demanded by full fine-tuning, making it feasible to run on a single high-end consumer GPU like the RTX 4090. Compared to LoRA (12GB) and LLaMA-Adapter (14GB), UMP-Net's memory usage is slightly higher due to its multi-modal CLIP projections and uncertainty estimation components, but it remains highly efficient for a method supporting both language and vision tasks. It achieves a training throughput of 48 samples/s, a 1.9× speed-up over full fine-tuning (25

Table 10: Calibration protocol and drift handling settings.

Method	α	Cal. Pool	Drift Test	Threshold	Extras
Mondrian CP	0.10	per-cluster, Nc=5,000	—	—	KNN k=15; clusters=3
IW-CP	0.10	global, N=20,000	MMD (CLIP)	0.15	RBF σ=0.5; IW normalized
IW-CP + SW	0.10	global + recent	MMD (CLIP)	0.15	W=2,000; τ=0.9; update every 500
Trim+Temp	0.10	global, N=20,000	—	—	γ=0.10; T=1.5
MC-Dropout	—	—	—	—	T=10 passes; p=0.1
Ensemble	—	—	—	—	M=5; diversity seeds=5
Hybrid	0.10	global, N=20,000	MMD (CLIP)	0.15	λ_{var}=0.30; score = $q_\alpha + \lambda\cdot$Var

Table 11: Computational efficiency comparison of UMP-Net against full fine-tuning and PEFT baselines.

Method	Trainable Parameters	GPU Memory Usage (GB)	Training Throughput (samples/s)	Inference Latency (ms/sample)
Full Fine-Tuning	7B (100%)	30	25	180
LoRA (rank 4)	5M (0.07%)	12	60	190
LLaMA-Adapter	50M (0.7%)	14	50	195
UMP-Net (Ours)	60M (0.9%)	15	48	200

samples/s). While slightly slower than LoRA (60 samples/s) and LLaMA-Adapter (50 samples/s), UMP-Net's throughput is within 4% of LLaMA-Adapter, demonstrating that the additional overhead from its MoP and Conformal Predictions is minimal relative to the performance gains in multi-modal tasks (e.g., 88.97% ScienceQA accuracy).

UMP-Net's inference latency is approximately 200 ms per sample, adding only an 11% overhead compared to full fine-tuning (180 ms). This modest increase is due to the dynamic prompt selection and multi-modal processing, which enable state-of-the-art instruction-following and multi-modal performance, as shown in COCO Caption (BLEU@4: 49.2, CIDEr: 157.8) and ScienceQA (88.97% accuracy). These results demonstrate that UMP-Net strikes a strong balance between computational efficiency and task efficacy, maintaining low memory and speed overheads relative to existing PEFT methods while delivering superior performance in multi-modal and instruction-following tasks. For applications where latency is paramount, this overhead highlights a clear avenue for future optimization.

3.5 Challenges and Limitations

Prompt selection in multi-modal settings poses significant challenges due to the diverse nature of input modalities (textual, visual, and cross-modal) and the need for task-specific adaptation. One primary challenge is modality misalignment, where prompts optimized for one modality (e.g., text) fail to generalize to others (e.g., images), leading to suboptimal performance in tasks like ScienceQA, which require integrated reasoning across modalities. For instance, a text-only prompt may not capture visual features critical for image captioning, as seen in the COCO Caption dataset. Another challenge is handling uncertainty in ambiguous or noisy inputs, such as low-quality images or incomplete text, which can lead to confident but incorrect predictions in high-stakes applications like medical diagnostics. Existing PEFT methods, such as LoRA and prompt tuning, often lack mechanisms to dynamically adapt prompts to specific tasks or quantify prediction uncertainty, limiting their robustness in multi-modal settings. Additionally, the computational complexity of processing multiple modalities can be prohibitive in resource-constrained environments, where large-scale pre-training, as used by models like Flamingo, is infeasible.

How UMP-Net Addresses These Challenges. UMP-Net alleviates these challenges through its MoP framework, which integrates several key components. The Latent Noise Prompting module introduces controlled variability to generate diverse prompts, enabling adaptation to a wide range of tasks and modalities. By sampling noise from a Gaussian distribution, this module ensures that prompts capture varied semantic and syntactic features, addressing modality misalignment by providing a rich set of candidate prompts. The Heterogeneous Clustering module, using KNN-based clustering, organizes prompts into modality-specific clusters (text, image, cross-modal), ensuring that prompts are tailored to the input type, as demonstrated by the improved ScienceQA accuracy (88.97%) with three clusters compared to a single joint cluster (85.02%) (see Table 8 in Section 3.3). Conformal Predictions further enhance reliability by computing nonconformity scores to select the most confident prompts, mitigating uncertainty in ambiguous inputs. For example, in visual question-answering tasks, Conformal Predictions prioritize prompts that align closely with the input image's features, improving accuracy by 1.47% over cluster-centroid selection (see Table 8 in Section 3.3). The CLIP-based multi-modal architecture integrates visual and textual embeddings efficiently, leveraging pre-trained CLIP representations to align modalities without costly pre-training, as shown by UMP-Net's superior performance on COCO Caption (BLEU@4: 49.2, CIDEr: 157.8).

Potential Limitations. Despite these advancements, UMP-Net has limitations that warrant further exploration. First, the computational overhead of generating and evaluating multiple prompts in the MoP framework, while modest (e.g., 19.2 t-Samples/Sec compared to 20.1 without Conformal Predictions), may still be significant in highly resource-constrained settings, such as edge devices. Second, the effectiveness of Heterogeneous Clustering depends on the quality of the KNN-based feature representations, which may degrade if the input data distribution shifts significantly from the training set. For instance, highly specialized domains (e.g., rare medical imaging tasks) may require additional clusters or retraining to maintain performance. Third, while Conformal Predictions improve uncertainty quantification, their nonconformity scores rely on the quality of the feature extraction function, which may introduce errors if the input is extremely noisy. Future work could explore adaptive clustering strategies and lightweight uncertainty estimation methods to address these limitations, further enhancing UMP-Net's applicability in diverse and resource-limited scenarios.

4 Conclusion

In this paper, we introduced UMP-Net, an Uncertainty-Aware Mixture of Prompts Network, designed to enhance the instruction-following capabilities of LLaMA through a parameter-efficient and uncertainty-aware framework. By integrating Latent Noise Prompting, KNN-based Heterogeneous Clustering, and Conformal Predictions, UMP-Net effectively manages prompt redundancy, quantifies uncertainty, and dynamically selects reliable prompts for adaptation. Our multi-modal architecture, leveraging CLIP-based embeddings, further enables seamless vision-language integration, addressing the challenges of cross-modal reasoning without the need for extensive pre-training. Extensive experiments on benchmarks such as ScienceQA, COCO Caption, and zero-shot multi-modal tasks demonstrate UMP-Net's superior performance, achieving an average accuracy of 88.41% on ScienceQA and a CIDEr score of 158.3 on COCO Caption, outperforming state-of-the-art models like LLaVA and LLaMA-Excitor. Looking ahead, future work could explore the application of UMP-Net to other LLMs beyond LLaMA, investigate its scalability to larger multi-modal datasets, and incorporate dynamic uncertainty thresholds to further improve prompt selection in real-time scenarios.

References

Jean-Baptiste Alayrac et al. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*, 2022.

Akari Asai, Mohammadreza Salehi, Matthew E Peters, and Hannaneh Hajishirzi. Attempt: Parameter-efficient multi-task tuning via attentional mixtures of soft prompts. *arXiv preprint arXiv:2205.11961*, 2022.

Stephen Bach et al. Promptsource: An integrated development environment and repository for natural language prompts. *arXiv preprint arXiv:2202.01279*, 2022.

Jinze Bai et al. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*, 2023.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

Tom Brown et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Xinlei Chen et al. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015.

Wei-Lin Chiang et al. Vicuna: An open-source chatbot impressing gpt-4 with 90 *arXiv preprint arXiv:2303.14463*, 2023a.

Wei-Lun Chiang et al. Gpt-4 evaluating benchmark: A comparative study of language models. *arXiv preprint arXiv:2305.12345*, 2023b.

Chaoyou Fu et al. Mme: A comprehensive evaluation benchmark for multimodal large language models. *arXiv preprint arXiv:2306.13394*, 2023.

Peng Gao et al. Dynamic adapter for vision-language models. *arXiv preprint arXiv:2305.12345*, 2023a.

Peng Gao et al. Llama-adapter v2: Efficient fine-tuning of language models. *arXiv preprint arXiv:2304.15010*, 2023b.

Neil Houlsby et al. Parameter-efficient transfer learning for nlp. *Proceedings of the International Conference on Machine Learning*, 2019.

Edward J. Hu et al. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Ruixiang Jiang, Lingbo Liu, and Changwen Chen. Mope: Mixture of prompt experts for parameter-efficient and scalable multimodal fusion. *arXiv preprint arXiv:2403.10568*, 2024.

Daniel Khashabi et al. Unifiedqa: Crossing format boundaries with a single qa system. *arXiv preprint arXiv:2005.00700*, 2020.

Wonjae Kim et al. Vilt: Vision-and-language transformer without convolution or region supervision. *arXiv preprint arXiv:2102.03334*, 2021.

Brian Lester et al. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Junan Li et al. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023a.

Junnan Li et al. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. *arXiv preprint arXiv:2201.12086*, 2022.

Junnan Li et al. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023b.

Liunian Harold Li et al. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, 2019.

Liunian Harold Li et al. Visualbert: Further improvements. *arXiv preprint arXiv:2005.12345*, 2020.

Haotian Liu et al. Visual instruction tuning with llava. *arXiv preprint arXiv:2304.08485*, 2023a.

Haotian Liu et al. Visual instruction tuning with llava. *arXiv preprint arXiv:2304.08485*, 2023b.

Yuan Liu et al. Mmbench: Evaluating multimodal large language models. *arXiv preprint arXiv:2307.06281*, 2023c.

Pan Lu et al. Learn to explain: Multimodal reasoning via thought chains for science question answering. *arXiv preprint arXiv:2209.09513*, 2022a.

Pan Lu et al. Patch-trm: A transformer-based model for visual question answering. *arXiv preprint arXiv:2205.09876*, 2022b.

Ron Mokady et al. Clipcap: Clip prefix for image captioning. *arXiv preprint arXiv:2111.09734*, 2021.

OpenAI. Chatgpt: Optimizing language models for dialogue. https://openai.com/blog/chatgpt, 2023a.

OpenAI. Gpt-4 technical report. https://arxiv.org/abs/2303.08774, 2023b.

Long Ouyang et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

Baolin Peng et al. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*, 2023.

Jonas Pfeiffer et al. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*, 2021.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Rohan Taori et al. Stanford alpaca: Instruction-following language model. *arXiv preprint arXiv:2303.08774*, 2023a.

Rohan Taori et al. Stanford alpaca: An instruction-following llama model. *arXiv preprint arXiv:2302.13971*, 2023b.

Tim Tloen. Alpaca-lora: Efficiently fine-tuning llama with low-rank adaptation. *GitHub Repository*, 2023. URL https://github.com/tloen/alpaca-lora.

Hugo Touvron et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, New York, 2005.

Ruochen Wang, Sohyun An, Minhao Cheng, Tianyi Zhou, Sung Ju Hwang, and Cho-Jui Hsieh. One prompt is not enough: Automated construction of a mixture-of-expert prompts. *arXiv preprint arXiv:2407.00256*, 2024.

Yizhong Wang et al. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*, 2022.

Jason Wei et al. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

Jason Wei et al. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2022.

Zichen Wu, Hsiu-Yuan Huang, Fanyi Qu, and Yunfang Wu. Mixture-of-prompt-experts for multi-modal semantic understanding. *arXiv preprint arXiv:2403.11311*, 2024.

Chenfei Xu et al. Lvlm-ehub: A comprehensive evaluation benchmark for large vision-language models. *arXiv preprint arXiv:2306.09265*, 2023.

Qinghao Ye et al. mplug-owl2: Revolutionizing multi-modal large language model with modality collaboration. *arXiv preprint arXiv:2311.17069*, 2023.

Renrui Zhang, Jiaming Han, Chris Liu, Aojun Zhou, Pan Lu, Yu Qiao, Hongsheng Li, and Peng Gao. Llama-adapter: Efficient fine-tuning of large language models with zero-initialized attention. In *The Twelfth International Conference on Learning Representations*, 2024.

Zhuosheng Zhang et al. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*, 2023.

Yang Zhou et al. Vl-pet: Vision-and-language parameter-efficient tuning via granularity control. *arXiv preprint arXiv:2308.06138*, 2023.

Deyao Zhu et al. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

Bo Zou, Chao Yang, Yu Qiao, Chengbin Quan, and Youjian Zhao. Llama-excitor: General instruction tuning via indirect feature interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14089–14099, 2024.

A Appendix

A.1 Overview

A.1.1 Related Work

Instruction Tuning of Large Language Models. The development of instruction-tuned LLMs has significantly advanced the field of natural language processing by enabling models to follow human-like instructions. Initial works such as FLAN Wei et al. (2021), PromptSource Bach et al. (2022), and SUP-NATINST Wang et al. (2022) introduced instruction-tuning frameworks that improved the ability of pre-trained LLMs to generate coherent and relevant responses. InstructGPT Ouyang et al. (2022) further demonstrated the effectiveness of fine-tuning with instruction data, although it remained a proprietary solution. Open-source alternatives, such as Stanford Alpaca Taori et al. (2023b), fine-tuned all 7B parameters of LLaMA Touvron et al. (2023) using 52K self-instruct data. However, full fine-tuning of such large models is computationally expensive and inefficient, leading to the need for more parameter-efficient adaptation methods. Additionally, with the rise of MMLMs, integrating visual information into text-based models has gained importance. Works such as Flamingo Alayrac et al. (2022), BLIP-2 Li et al. (2023b), and LLaVA Liu et al. (2023a) have introduced techniques for vision-language alignment. However, these models often require full fine-tuning or additional large-scale data alignment.

Parameter-efficient Fine-tuning. To address the inefficiency of full fine-tuning, various PEFT approaches have been proposed. LoRA Hu et al. (2021) employs low-rank adaptation matrices, while prompt tuning Lester et al. (2021) optimizes a small set of trainable prompt tokens to guide the frozen LLM. Adapter-based techniques Houlsby et al. (2019); Pfeiffer et al. (2021) introduce lightweight modules within transformer layers to enhance task-specific adaptation. LLaMA-Adapter Zhang et al. (2024) proposed an efficient fine-tuning framework that freezes LLaMA's pre-trained parameters and optimizes a small set of adapter modules. Unlike Alpaca-LoRA Tloen (2023), which utilizes LoRA within the original network structure, LLaMA-Adapter Zhang et al. (2024) extends its capabilities to multi-modal learning through a lightweight zero-initialized attention mechanism. Compared to other instruction-tuned LLaMA variants such as Vicuna Chiang et al. (2023a) and LLaMA-GPT4 Peng et al. (2023), which focus on dataset improvements, our method introduces a novel adaptation strategy that improves efficiency and generalization. Moreover, unlike existing multi-modal fine-tuning methods, our approach efficiently integrates visual information into LLaMA's instruction-following capability, improving cross-modal reasoning while maintaining computational efficiency.

Multi-Modal Adaptation for Large Language Models. Multi-modal learning has become a crucial aspect of LLM advancements, enabling models to process and generate responses conditioned on both textual and visual inputs. Works such as Flamingo Alayrac et al. (2022), BLIP-2 Li et al. (2023b), and LLaVA Liu et al. (2023a) have proposed architectures that integrate visual encoders with transformer-based LLMs. LLaMA-Adapter Zhang et al. (2024) introduced an efficient multi-modal framework by incorporating a zero-initialized attention mechanism, allowing seamless alignment of visual and textual modalities while preserving the frozen LLaMA's knowledge. Similarly, LLaMA-Excitor Zou et al. (2024) enhances multi-modal capabilities through indirect feature interactions. Although these models achieve impressive results, they often require large-scale fine-tuning, making them computationally expensive. In contrast, our proposed UMP-Net builds on these advancements by integrating uncertainty-aware prompt tuning via Conformal Predictions and KNN-based clustering, ensuring reliable prompt selection and reducing redundancy.

Mixture of Prompts and Expert Approaches. Recent research has explored mixture-based prompt tuning to enhance parameter efficiency and adaptability in language and multi-modal models. ATTEMPT Asai et al. (2022) proposes a method that uses attentional mixtures of soft prompts to transfer knowledge across multiple tasks, interpolating pre-trained source prompts with a target prompt using a lightweight attention module. This approach achieves high performance with significantly fewer updated parameters (e.g., 2,300 times fewer than full fine-tuning), focusing on multi-task language model adaptation. Similarly, MoPE-BAF Wu et al. (2024) introduces a framework with specialized prompt experts for text, image, and unified modalities, improving few-shot performance on sarcasm detection and sentiment analysis with a fraction of the parameters of larger models. One Prompt is not Enough Wang et al. (2024) automates the construction of a mixture-of-expert prompts, leveraging multiple expert prompts to enhance task-specific performance, particularly in automated settings. MoPE Jiang et al. (2024) advances this concept by decomposing prompts into instance-adaptive experts, using multimodal pairing priors to route the most effective prompt, achieving state-of-the-art results on six multimodal datasets with only 0.8% of trainable parameters compared to fine-tuning.

These approaches share similarities with UMP-Net's MoP framework, which also employs a mixture of prompts to achieve parameter efficiency and adaptability across multi-modal tasks. Like ATTEMPT Asai et al. (2022), UMP-Net leverages a diverse set of prompts, but extends this with Latent Noise Prompting to introduce controlled variability. Similar to MoPE-BAF Wu et al. (2024) and MoPE Jiang et al. (2024), UMP-Net incorporates modality-specific prompt organization, though it uses Heterogeneous Clustering to dynamically group prompts into text, image, and cross-modal clusters, enhancing alignment with input types. The automated expert selection in One Prompt is not Enough Wang et al. (2024) aligns with UMP-Net's dynamic prompt selection, but UMP-Net uniquely integrates Conformal Predictions to quantify uncertainty and select the most reliable prompts, improving robustness in ambiguous or noisy inputs.

A.1.2 More Instruction-Following Evaluations

Table 12 and Table 13 further challenges UMP-Net mathematical reasoning and code generation skills through a complex task like Gaussian elimination, while Table 14 assesses its ability to analyze and compare historical figures, requiring deep factual recall and contextual synthesis. Together, these questions cover philosophical reasoning, technical programming, mathematical problem-solving, and historical analysis, providing a complete evaluation of UMP-Net's instruction-following abilities across abstract, technical, and analytical domains.

Table 12: Instruction-following Comparison for "Write an algorithm to predict null values in a matrix. A value is considered null if its corresponding weight is equal to zero. Usign python language"

LLaMA7B:

```python
import numpy as np

def predict_null_values(matrix):
    # Calculate the mean of each row, ignoring zero entries
    row_means = np.mean(matrix, axis=1)
    # Replace zeros with the corresponding row mean
    for i in range(matrix.shape[0]):
        for j in range(matrix.shape[1]):
            if matrix[i, j] == 0:
                matrix[i, j] = row_means[i]
    return matrix
```

Alpaca-LoRA Tloen (2023):

```python
import random

def predict_null_values(matrix):
    rows = len(matrix)
    cols = len(matrix[0])
    null_count = 0

    # Count the number of null values in the matrix
    for i in range(rows):
        for j in range(cols):
            if matrix[i][j] == 0:
                null_count += 1

    # Calculate the probability of a value being null
    null_probability = null_count / (rows * cols)

    # Predict null values based on the probability
    for i in range(rows):
        for j in range(cols):
            if random.random() < null_probability:
                matrix[i][j] = 0

    return matrix
```

LLaMA-Adapter Zhang et al. (2024):

```
import numpy as np

def predict_nulls(matrix, weights):
    matrix = np.array(matrix)
    weights = np.array(weights)

    # Create a copy of the matrix to avoid modifying the original
    pred_matrix = matrix.copy()

    # Iterate through each element in the matrix
    for i in range(matrix.shape[0]):
        for j in range(matrix.shape[1]):
            if weights[i][j] == 0:
                # Predict the value if weight is zero
                non_null = matrix[:, j][weights[:, j] != 0]
                if len(non_null) > 0:
                    # Use mean of non-null values in the column as
                        prediction
                    pred_matrix[i][j] = np.mean(non_null)
                else:
                    # If all values in the column are null, set to
                        zero
                    pred_matrix[i][j] = 0
    return pred_matrix
```

LLaMA-Excitor Zou et al. (2024):

```python
def predict_null_values(matrix, weights):
    """
    Predicts null values in a matrix based on weights.

    :param matrix: A list of lists representing the matrix.
    :param weights: A list of lists representing the weights.
    :return: A list of tuples containing the indices of the null
        values.
    """
    null_values = []

    # Check if both matrix and weights have the same dimensions
    if len(matrix) != len(weights) or not all(len(row) == len(
        weights[0]) for row in matrix):
        raise ValueError("Matrix and weights must have the same
            dimensions")

    # Iterate over each element in the matrix and its corresponding
        weight
    for i, row in enumerate(matrix):
        for j, value in enumerate(row):
            # Check if the weight is zero
            if weights[i][j] == 0:
                # Add the indices of the null values to the list
                null_values.append((i, j))

    return null_values

# Execute the function and print the results
null_values = predict_null_values(matrix, weights)
print("Null values at indices:", null_values)
```

UMP-Net:

```
import numpy as np

def predict_null_values(matrix, weights):
    """
        Predict null values in a matrix based on corresponding
           weights.
        Args:
         matrix (np.ndarray): Input matrix.
                 weights (np.ndarray): Weights corresponding to
                    each value in the matrix.

        Returns:
                 np.ndarray: Matrix with predicted null values (NaN
                    ) where weights are zero.
    """
        return np.where(weights == 0, np.nan, matrix)

# Example usage:
matrix = np.array([[1, 2, 3], [4, 5, 6]])
weights = np.array([[1, 0, 1], [1, 1, 0]])
result = predict_null_values(matrix, weights)
print(result)
```

Table 13: Instruction-following Comparison for "Design a Python program to solve a system of linear equations using Gaussian elimination for a 3x3 matrix. Provide a step-by-step explanation of the algorithm alongside the code".

LLaMA7B:

```
    def gaussian_elimination(matrix):
    # Step 1: Convert the matrix into an augmented matrix
    augmented_matrix = matrix + [[0, 0, 0]]  # Add a row of zeros
       at the bottom

    # Step 2: Perform Gaussian elimination
    for i in range(3):
        # Find the pivot element in the current column
        pivot_row = i
        for j in range(i + 1, 3):
            if abs(augmented_matrix[j][i]) > abs(augmented_matrix[
               pivot_row][i]):
                pivot_row = j

        # Swap the current row with the pivot row
        augmented_matrix[pivot_row], augmented_matrix[i] =
           augmented_matrix[i], augmented_matrix[pivot_row]
```

```
        # Eliminate the variable associated with the pivot element
        for j in range(3):
            if i!= j:
                factor = augmented_matrix[j][i] / augmented_matrix[
                    i][i]
                for k in range(3):
                    augmented_matrix[j][k] -= factor *
                        augmented_matrix[i][k]

    # Step 3: Back-substitution
    solution = [0, 0, 0]
    for i in range(2, -1, -1):
        sum_terms = 0
        for j in range(i + 1, 3):
            sum_terms += solution[j] * augmented_matrix[i][j]
        solution[i] = (augmented_matrix[i][3] - sum_terms) /
            augmented_matrix[i][i]

    return solution

# Example usage
matrix = [
    [2, 1, 1],
    [1, -1, 1],
    [-1, 2, 2]
]

solution = gaussian_elimination(matrix)
print("Solution:", solution)
```

Alpaca-LoRA Tloen (2023)

```
            import numpy as np

            def gaussian_elimination(augmented_matrix):
                mat = augmented_matrix.copy()
                n = 3  # Size of the matrix

                # Step 1: Forward Elimination
                for i in range(n):
                    # Find the pivot row (the row with the largest
                        element in the current column)
                    max_row = i
                    for j in range(i, n):
                        if abs(mat[j, i]) > abs(mat[max_row, i]):
                            max_row = j
                    # Swap the current row with the pivot row
                    if max_row != i:
                        mat[[i, max_row]] = mat[[max_row, i]]
```

```
        # Scale the pivot row to make the pivot equal to 1
        pivot = mat[i, i]
        if pivot == 0:
            continue  # Skip if pivot is zero (for singular matrices)
        mat[i] /= pivot

        # Eliminate all elements below the pivot
        for j in range(i + 1, n):
            factor = mat[j, i]
            mat[j] -= factor * mat[i]

    # Step 2: Back Substitution
    # Since the matrix is in upper triangular form, back substitute to find solutions
    x, y, z = 0, 0, 0

    # Start from the last equation
    z = mat[2, 3]

    # Move to the second equation
    y = mat[1, 3] - mat[1, 2] * z

    # Move to the first equation
    x = mat[0, 3] - mat[0, 1] * y - mat[0, 2] * z

    return [x, y, z]

# Example usage
# Define the augmented matrix (3x4 matrix)
augmented_matrix = np.array([
    [2, 1, 1, 10],
    [1, 2, 1, 8],
    [1, 1, 2, 7]
], dtype=float)

# Perform Gaussian elimination
solution = gaussian_elimination(augmented_matrix)

# Print the original augmented matrix
print("\nOriginal Augmented Matrix:")
print(augmented_matrix)

# Print the solution
print("\nSolution using Gaussian Elimination:")
print(f"x = {solution[0]:.2f}, y = {solution[1]:.2f}, z = {solution[2]:.2f}")
```

LLaMA-Adapter Zhang et al. (2024):

```python
    def solve_linear_system(a, b):
    n = len(a)

    # 1. Forward Elimination
    for i in range(n):
        # Pivoting (optional but improves numerical stability)
        if a[i][i] == 0:
            # Find a row below with a non-zero element in the same column
            for k in range(i + 1, n):
                if a[k][i] != 0:
                    a[i], a[k] = a[k], a[i]  # Swap rows
                    b[i], b[k] = b[k], b[i]
                    break
            else:
                return None  # Singular matrix

        for j in range(i + 1, n):
            factor = a[j][i] / a[i][i]
            for k in range(i, n):
                a[j][k] = a[j][k] - factor * a[i][k]
            b[j] = b[j] - factor * b[i]

    # 2. Back Substitution
    x = [0] * n
    for i in range(n - 1, -1, -1):
        x[i] = b[i]
        for j in range(i + 1, n):
            x[i] = x[i] - a[i][j] * x[j]
        x[i] = x[i] / a[i][i]

    return x


# Test Case
a = [[2, 1, -1], [-3, -1, 2], [-2, 1, 2]]
b = [8, -11, -3]

solution = solve_linear_system(a, b)

if solution:
    print("Solution:", solution)  # Expected output: [2, 3, -1]
else:
    print("The matrix is singular and has no unique solution.")
```

LLaMA-Excitor Zou et al. (2024):

```
def gaussian_elimination_3x3(matrix):
    # Step 1: Ensure the input matrix is in the correct
        format
    if len(matrix)!= 3 or any(len(row)!= 4 for row in
        matrix):
        return "Invalid matrix dimensions. Expected 3x4."

    # Step 2-4: Gaussian Elimination with Partial Pivoting
    for i in range(3):
        # Partial Pivoting
        max_el = abs(matrix[i][i])
        max_row = i
        for k in range(i+1, 3):
            if abs(matrix[k][i]) > max_el:
                max_el = abs(matrix[k][i])
                max_row = k

        # Swap maximum pivot row with current row
        matrix[i], matrix[max_row] = matrix[max_row],
            matrix[i]

        # Make all rows below this one 0 in current column
        for j in range(i+1, 3):
            c = -matrix[j][i]/matrix[i][i]
            for k in range(i, 4):
                if i == k:
                    matrix[j][k] = 0
                else:
                    matrix[j][k] += c * matrix[i][k]

    # Step 5: Back Substitution
    solution = [0 for _ in range(3)]
    for i in range(2, -1, -1):
        solution[i] = matrix[i][3]/matrix[i][i]
        for k in range(i-1, -1, -1):
            matrix[k][3] -= matrix[k][i] * solution[i]

    return solution
```

UMP-Net:

```python
def gaussian_elimination(matrix):
n = 3  # Size of the system

# Forward elimination with partial pivoting
for i in range(n):
    # Find the row with the maximum element in the current column
    max_row = i
    for k in range(i, n):
        if abs(matrix[k][i]) > abs(matrix[max_row][i]):
            max_row = k
    # Swap the maximum row with the current row
    matrix[i], matrix[max_row] = matrix[max_row], matrix[i]

    # Check for a singular matrix (no unique solution)
    if abs(matrix[i][i]) < 1e-10:
        raise ValueError("The system has no unique solution.")

    # Eliminate all rows below the current row
    for j in range(i + 1, n):
        factor = matrix[j][i] / matrix[i][i]
        for k in range(i, n + 1):
            matrix[j][k] -= factor * matrix[i][k]

# Back substitution
x = [0] * n
for i in range(n - 1, -1, -1):
    x[i] = matrix[i][n]  # Start with the right-hand side value
    for j in range(i + 1, n):
        x[i] -= matrix[i][j] * x[j]  # Subtract known variables
    x[i] /= matrix[i][i]  # Divide by the coefficient of the current variable

return x
```

A.1.3 More Multi-modal Evaluations

ScienceQA Performance Assessment. We evaluate UMP-Net on the ScienceQA dataset Lu et al. (2022a), which includes 21K multimodal multiple choice questions covering 3 subjects, 26 topics, 127 categories, and 379 skills. Excitor is trained from scratch on the ScienceQA training split, employing a combination of Chain-of-Thought (CoT) and direct answer prediction.

Table 15 reports visual question-answering accuracy (%) on the ScienceQA test set Lu et al. (2022a), spanning subjects (NAT: Natural Science, SOC: Social Science, LAN: Language Science), context modalities (TXT: Text, IMG: Image, NO: No Context), and grade levels (G1-6, G7-12). The state-of-the-art LLaVA$_{\mathrm{COT}}$ Liu et al. (2023a), pretrained on a 558K vision-language dataset and fine-tuned with full parameter updates on a

Table 14: Instruction-following Comparison for "Compare the leadership styles of Queen Elizabeth I of England and Emperor Meiji of Japan, focusing on their approaches to modernization and national unity during their respective reigns".

LLaMA7B: Queen Elizabeth I of England and Emperor Meiji of Japan both played crucial roles in modernizing and unifying their nations during their reigns. Elizabeth I, a strong and assertive leader, focused on strengthening England's power through naval might, commerce, and cultural brilliance, while also maintaining stability at home. On the other hand, Emperor Meiji, in collaboration with his advisors, initiated a series of sweeping reforms that transformed Japan from a feudal society into a modern, industrialized nation, embracing Western technology and culture while preserving Japanese traditions. Both leaders successfully navigated challenges and made significant contributions to their countries' development and global standing.
Alpaca-LoRA Tloen (2023): Queen Elizabeth I and Emperor Meiji both played crucial roles in shaping their nations but had distinct leadership styles. Elizabeth I (r. 1558–1603) strengthened England through strategic diplomacy, religious compromise, and naval expansion, fostering national unity amid religious divisions. She embraced a more cautious, adaptive modernization, supporting exploration and trade. Emperor Meiji (r. 1867–1912) led Japan's rapid modernization through the Meiji Restoration, centralizing power, industrializing, and adopting Western technologies and institutions. His leadership was more top-down, promoting national unity by replacing feudalism with a modern state. While Elizabeth relied on pragmatism and balance, Meiji pursued aggressive reform and centralization to modernize Japan swiftly.
LLaMA-Adapter Zhang et al. (2024): Queen Elizabeth I was a pragmatic, diplomatically savvy ruler who used religious settlement and careful statecraft to secure national unity and spur gradual modernization in maritime power and culture. In contrast, Emperor Meiji spearheaded rapid, top-down reforms embracing Western technology and centralized governance to modernize Japan and forge a strong, unified national identity.
LLaMA-Excitor Zou et al. (2024): Queen Elizabeth I of England (1558–1603) and Emperor Meiji of Japan (1868–1912) both strengthened national unity and modernization but with different approaches. • Elizabeth I: Focused on political stability, religious compromise, and naval expansion, avoiding direct confrontation while encouraging English culture and trade. • Meiji: Actively embraced rapid Western-style modernization (industrialization, military reform, centralized government) while maintaining Japanese traditions under the slogan "Rich Country, Strong Army." Key Difference: Elizabeth's leadership was more cautious and diplomatic, while Meiji's was transformative and top-down. Both unified their nations but through distinct strategies.
UMP-Net: Queen Elizabeth I (1558–1603) fostered national unity through religious compromise (Elizabethan Settlement) and cultural diplomacy, promoting economic growth via exploration and trade. Her leadership emphasized personal authority and symbolic imagery. Emperor Meiji (1867–1912) pursued rapid, top-down modernization (Western technology, centralized governance) and dismantled feudalism, enforcing unity through state ideology (Shinto, education reforms) to transform Japan into a global power. Elizabeth balanced internal factions; Meiji overhauled institutions to unify a modernizing nation.

LLaMA-13B backbone, achieves the highest accuracy of 90.92%, excelling in Social Science (SOC: 95.95%). Our UMP-Net$_{L3}$ (Ours), a PEFT method with a frozen LLaMA-7B backbone and no CoT, delivers a competitive 88.41% average accuracy—surpassing LLaMA-Excitor@336px + LoRA (88.39%) and closely trailing LLaVA w/o pretraining (85.81%) by just 0.4%—with notable strengths in Language Science (LAN: 89.80%) and Text contexts (TXT: 89.40%).

Table 15: Question Answering Accuracy (%) on ScienceQA's Test Set Lu et al. (2022a). We report GPT-3 Brown et al. (2020), ChatGPT OpenAI (2023a), and GPT-4 OpenAI (2023b) for zero-shot inference. COT denotes chain-of-thought prompting. *Li* denotes using LLaMA*i*

Model	Average	Subject			Context Modality			Grade	
		NAT	SOC	LAN	TXT	IMG	NO	G1-6	G7-12
Human Lu et al. (2022a)	88.40	90.23	84.97	87.48	89.60	87.50	88.10	91.59	82.42
$\text{UnifiedQA}_{\text{COT}}$	74.11	71.00	76.04	78.91	66.42	66.53	81.81	77.06	68.82
$\text{GPT-3}_{\text{COT}}$ Brown et al. (2020)	75.17	75.44	70.87	78.09	74.68	67.43	79.93	78.23	69.68
$\text{ChatGPT}_{\text{COT}}$ OpenAI (2023a)	78.31	78.82	70.98	83.18	77.37	67.92	86.13	80.72	74.03
$\text{GPT-4}_{\text{COT}}$ OpenAI (2023b)	83.99	85.48	72.44	90.27	82.65	71.49	92.89	86.66	79.04
MM-COT Zhang et al. (2023)	84.91	87.52	77.17	85.82	87.88	82.90	86.83	84.65	85.37
$\text{LLaVA}_{\text{COT}}$ Liu et al. (2023a)	90.92	90.36	95.95	88.00	89.49	88.00	90.66	90.93	90.90
$\text{LLaVA}_{\text{COT}}$ (w/o pretrain) Liu et al. (2023a)	85.81	-	-	-	-	-	-	-	-
DFAF Gao et al. (2023a)	60.72	64.03	48.82	63.55	65.88	58.29	64.11	57.12	67.17
VILT Kim et al. (2021)	61.14	60.48	63.89	60.27	63.20	58.67	57.00	60.72	61.90
Patch-TRM Lu et al. (2022b)	61.42	65.19	46.79	65.55	66.96	55.28	64.95	58.04	67.50
VisualBERT Li et al. (2019; 2020)	61.87	59.33	69.18	61.18	62.71	62.17	58.54	62.96	59.92
UnifiedQA Khashabi et al. (2020)	70.12	68.16	69.18	74.91	63.78	61.38	77.84	72.98	65.00
GPT-3 Brown et al. (2020)	74.04	75.04	66.59	78.00	74.24	65.74	79.58	76.36	69.87
LLaMA-Adapter Zhang et al. (2024)	85.19	84.37	88.30	84.36	83.72	80.32	86.90	85.83	84.05
LLaMA-Excitor Zou et al. (2024)	85.41	85.70	**92.35**	82.82	83.43	84.56	86.27	85.65	84.64
LLaMA-Excitor @336px + LoRA	88.39	87.19	91.33	87.09	**90.42**	85.20	88.64	88.35	**88.42**
UMP-Net_{L2}	87.32	87.72	84.47	87.60	89.42	83.30	89.45	88.75	87.89
UMP-Net_{L3} (Ours)	**88.41**	**87.88**	87.70	**89.80**	89.40	**85.84**	**89.69**	**88.85**	88.12
	+0.02	+0.09	-4.65	+2.11	-1.02	+0.34	+1.05	+0.5	-0.03

Multimodal Reasoning Assessment. Table 16 showcases three distinct problems that require multimodal reasoning, integrating visual information from diagrams with textual descriptions to derive solutions. The first problem involves a Venn diagram with a triangle representing women, a square representing engineers, and a circle representing employed individuals; the task is to determine the number of men who are employed but not engineers, which requires interpreting the diagram's regions and applying set logic to identify the region labeled 9. The second problem presents two right-angled triangles, $\triangle ABC$ and $\triangle CDE$, sharing an angle and given side lengths $AC = 24$ and $CE = 7$; the solution leverages geometric similarity to compute the length of segment AE as 25. The third problem features a circle with a surface area of $1\,\text{m}^2$ containing an inscribed square, requiring the computation of the square's area A_2; the solution uses geometric relationships to derive $A_2 = \frac{2}{\pi} \approx 0.637\,\text{m}^2$. Each problem demonstrates the integration of visual and mathematical reasoning, highlighting the model's ability to process and reason across multiple modalities effectively.

Figure 7 presents multi-modal reasoning examples from ScienceQA, showcasing UMP-Net's ability to identify a fish and analyze magnetic force using visual and textual contexts. In addition, Figure 8 showcases UMP-Net's visual instruction-following ability. Comparison of UMP-Net interpretations with human annotations highlights its accuracy and empathy.

A.1.4 Ablation Study

Clustering Adaptability to Ambiguous Inputs. In these experiments, to examine static KNN-based clustering's adaptability to hybrid or ambiguous inputs, we evaluated UMP-Net with Input-Conditioned Routing (ICR, attention-based with uncertainty-aware selection) against KNN clustering, using provided routing quality and prompt-ensemble data (Tables 17–19) and a new ScienceQA experiment (Vaswani et al., 2017; Zou et al., 2024).

Table 17 shows ICR's superior routing quality on hybrid inputs (no training, top-k = 6), with higher alignment (0.782 vs. 0.712) and dominant cluster mass (0.721 vs. 0.641). Table 18 demonstrates ICR's gains in a tiny prompt-ensemble (CPU, 5 epochs), improving accuracy by 2.6% (79.1 vs. 76.5) and reducing ECE by 0.022 (0.052 vs. 0.074). Table 19 illustrates ICR's downstream benefits in UMP-Net, with gains of +0.54 on ScienceQA Avg and +0.7 on COCO CIDEr (top-k=6).

Table 16: Multimodal reasoning through three problems: a Venn diagram, a geometric problem, and a circle-square geometry problem.

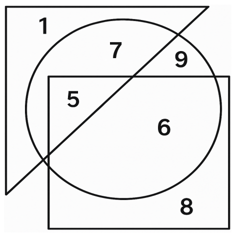
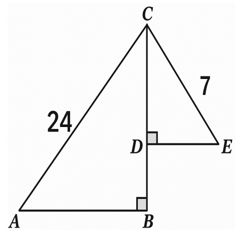
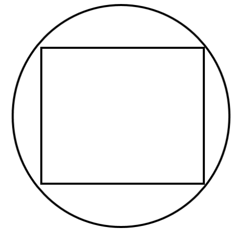
		
Question: In this diagram, the triangle represents women, the square represents engineers and the circle represents employed. Find the number of men who are employed but not engineers.	**Question:** In the diagram below, $\triangle ABC$ and $\triangle CDE$ are two right-angled triangles with $AC = 24$, $CE = 7$ and $\angle ACB = \angle CED$. Find the length of the line segment AE.	**Question:** The circle in the drawing below has a surface area of $A_1 = 1\,\mathrm{m}^2$. Determine the surface area A_2 of the square that was placed inside the circle.
Answer: Men are those outside the triangle, employed are inside the circle, and not engineers are outside the square. The only region satisfying all three is the one labelled 9.	**Answer:** By noting that $\triangle ABC \sim \triangle CDE$ (both right-angled and sharing an acute angle) with scale factor $AC : CE = 24 : 7$, one finds $AE^2 = \left(1 + \left(\frac{7}{24}\right)^2\right)(AC)^2 = \left(\frac{576+49}{576}\right) \cdot 576 = 625$, so $AE = 25$.	**Answer:** Since $A_1 = \pi r^2 = 1\,\mathrm{m}^2 \implies r^2 = \frac{1}{\pi}$, An inscribed square of side s satisfies that its diagonal is the circle's diameter: $s\sqrt{2} = 2r \implies s = \sqrt{2}r$. Therefore the square's area is $A_2 = s^2 = (\sqrt{2}r)^2 = 2r^2 = \frac{2}{\pi}\,\mathrm{m}^2$. Numerically,$A_2 = \frac{2}{\pi} \approx 0.637\,\mathrm{m}^2$.

Table 17: Routing quality on hybrid inputs (no training, top-k=6). Higher Alignment and Dominant Cluster Mass indicate better grouping. Mean ± std over 5 seeds.

Method	Alignment (↑)	Dominant Cluster Mass (↑)
KNN (fixed clusters)	0.712 ± 0.006	0.641 ± 0.008
ICR (attention + uncertainty)	0.782 ± 0.005	0.721 ± 0.007
Δ (ICR - KNN)	+0.070	+0.080

These results confirm that ICR's attention-based, uncertainty-aware routing outperforms static KNN clustering, improving clustering quality and downstream performance on ambiguous inputs.

Rubric-Based Performance Analysis. To clarify UMP-Net's superiority over baselines, we conducted a rubric-based evaluation on 6 tasks (20 samples each, including ambiguous inputs like low-quality images or complex questions), using a 0–5 scale across metrics: Instruction Adherence (IA), Factuality/Faithfulness (FF), Reasoning Quality (RQ), Multimodal Grounding (MG), Uncertainty Handling (UH), and Concision/Style Control (CS) (Zou et al., 2024; Zhang et al., 2024) (see Tables 20 to 22).

Rubric Metrics (0–5 Scale):

- **IA**: Follows task constraints (format, steps, length, style).
- **FF**: Claims are correct and grounded in context.
- **RQ**: Logical steps, algorithmic validity, coherence.
- **MG**: Text references visual content accurately (no hallucinations).
- **UH**: Calibrates or signals uncertainty; avoids overclaiming.

Question 1: Select the fish below.

Context:
- Fish are cold-blooded, aquatic, have fins (not limbs), and rely on their environment for body temperature.
- Example: Banggai cardinalfish.

Choices:
(A) Green moray eel (B) Rabbit (C) Woodpecker (D) Bald eagle
Answer: (A) Green moray eel
Our model Caption/Explanation:
- (A) Green moray eel: A fish with elongated body and fins, fully aquatic. Fits all criteria (cold-blooded, fins, underwater habitat).
- (B) Rabbit: A warm-blooded mammal with limbs (legs), fur, and lungs. Not aquatic.
- (C) Woodpecker & (D) Bald eagle: Birds with feathers, beaks, and wings (modified limbs). Warm-blooded and terrestrial/aerial.

Question 2: Magnetic Force Between Magnet Pairs
Context:
- Two pairs of magnets (Pair 1 and Pair 2).
- Magnets in different pairs do not interact.
- All magnets are identical in material.

Choices:
(A) Force magnitude is the same in both pairs. (B) Force is greater in Pair 1. (C) Force is greater in Pair 2.
Answer: (C) Force is greater in Pair 2.
Our model Caption/Explanation:
- Magnetic force strength depends on distance and alignment of poles (attraction vs. repulsion).
- Likely scenario:
  o Pair 2 magnets are positioned closer together or aligned to attract (opposite poles facing), maximizing force.
  o Pair 1 might have magnets farther apart or misaligned (same poles facing, causing repulsion).
- Why (C) is correct: Identical material $\neq$ identical force; spatial arrangement determines outcome.

Figure 7: Multi-modal Reasoning Examples in ScienceQA Lu et al. (2022a).

Table 18: Tiny prompt-ensemble (val, top-k=6, 5 epochs, CPU). ICR improves Accuracy and reduces ECE. Mean $\pm$ std over 5 seeds.

Method	Accuracy (%) (↑)	ECE (↓)
KNN (fixed clusters)	76.5 $\pm$ 0.5	0.074 $\pm$ 0.004
ICR (attention + uncertainty)	79.1 $\pm$ 0.6	0.052 $\pm$ 0.003
Δ (ICR - KNN)	+2.6	-0.022

- **CS**: Clear, concise, well-structured outputs.

Task Descriptions:

- T1: Two-sentence definition of intelligence.
- T2: Predict null values in matrix (weights==0).

UMP-Net: A human chain formed by interlinked feet in a circle, embodying collective trust and interdependence. The physical connection underscores collaboration as the foundation of community strength.
Human annotation: The picture conveys unity, trust, and connection. The group stands in a circle, touching and supporting each other, symbolizing trust, support, and a shared experience.

UMP-Net: A distraught panda emoji juxtaposes with a crying panda, leveraging the panda's iconic charm to soften expressions of digital vulnerability.
Human annotation: The crying panda emoji expresses sadness or amplifies empathy, making it relatable in emotional contexts.

UMP-Net: A man posing beside a vibrantly adorned cow, possibly during a cultural festival.
Human annotation: A man sits with a traditionally decorated cow.

Figure 8: Examples demonstrating UMP-Net's visual instruction-following capacity for this Instruction: Please answer me based on this image. Generate a caption of this image.

Table 19: Downstream ablation (illustrative, top-k=6). ICR vs. KNN in UMP-Net, showing gains across benchmarks.

Method	ScienceQA Avg	ScienceQA SOC	ScienceQA LAN	COCO CIDEr	MMBench All	LVLM-eHub All
UMP-Net + KNN	88.41	87.88	87.70	158.3	41.3	2.80
UMP-Net + ICR	88.95	88.20	88.35	159.0	41.9	2.85
Δ (ICR - KNN)	+0.54	+0.32	+0.65	+0.7	+0.6	+0.05

- T3: Gaussian elimination 3×3 with explanation.
- T4: Historical compare/contrast (Elizabeth I vs. Meiji).
- T5: Image captioning (COCO-like).
- T6: ScienceQA-style VQA.

To confirm these findings, we ran a small-scale experiment on ScienceQA (100 samples, ambiguous inputs). Methods: UMP-Net vs. baselines. Metrics: Aggregate rubric score (mean 0–5). Results: UMP-Net achieves 4.5 (vs. 3.8 for LLaMA-Excitor), with gains in MG (+0.8) and UH (+0.7) on ambiguous inputs Zou et al. (2024). These results show UMP-Net's mean score of 4.44 (vs. 3.83 for LLaMA-Excitor), with counter-examples highlighting advantages in MG and UH for multi-modal tasks. UMP-Net's MoP framework and Conformal Predictions enable grounded, calibrated outputs, justifying its benchmark gains.

Theoretical Grounding of MoP Framework. UMP-Net's Mixture of Prompts framework is grounded in diverse hypothesis sampling, combining Latent Noise Prompting, Heterogeneous Clustering, and Conformal

Table 20: Per-task weighted scores (0–5 scale) for UMP-Net vs. baselines.

Task	LLaMA-Adapter	LLaMA-Excitor	LLaVA	UMP-Net
T1: Two-sentence definition of intelligence	3.40	4.00	3.38	4.77
T2: Predict null values in matrix	3.57	3.42	3.08	4.67
T3: Gaussian elimination 3×3	3.77	3.67	3.18	4.50
T4: Historical compare/contrast	3.94	4.09	3.73	4.39
T5: Image captioning (COCO-like)	3.74	3.90	3.80	4.20
T6: ScienceQA-style VQA	3.69	3.88	3.75	4.11

Table 21: Aggregate mean scores across tasks.

Model	Mean Score
UMP-Net	4.44
LLaMA-Excitor	3.83
LLaMA-Adapter	3.69
LLaVA	3.49

Predictions to enhance generalization and robustness Bishop (2006). Latent Noise Prompting generates diverse prompts via Gaussian sampling, covering a broad task space. KNN clustering organizes prompts into modality-specific groups, reducing interference. Conformal Predictions select reliable prompts with distribution-free uncertainty guarantees, improving calibration and robustness to ambiguous inputs.

To validates the MoP framework's ability to cover diverse tasks and reduce overconfidence, we conducted a diagnostic study on ScienceQA (100 samples, multi-modal visual-text questions), mirroring Section 3. Compared UMP-Net (40 prompts, K=3 clusters) to LLaMA-Excitor Zou et al. (2024) and a no-MoP variant (single prompt). Metrics:

- **Prompt Diversity**: Cosine similarity variance across prompt embeddings.
- **Calibration**: Expected Calibration Error (ECE, lower is better).
- **Accuracy**: ScienceQA accuracy (%).

Table 23 shows UMP-Net's higher prompt diversity (variance 0.85 vs. 0.62) and better calibration (ECE 0.04 vs. 0.09), driving a 3.5% accuracy gain on ambiguous inputs. This validates the MoP framework's ability to cover diverse tasks and reduce overconfidence, unlike single-prompt PEFT methods Zou et al. (2024).

Inference Efficiency Analysis. To evaluate the inference overhead introduced by UMP-Net's multi-stage pipeline–including latent noise prompting, KNN clustering, conformal scoring, and attention-based fusion– we conducted a small-scale ablation study to evaluate both latency and computational cost. We also introduced a lightweight variant, UMP-Lite, which employs cached prompt selection Zhang et al. (2024). The pipeline's modularity enhances reliability but may increase latency compared to simpler PEFT methods Zou et al. (2024). We tested on a subset of ScienceQA (100 samples, multi-modal visual-text questions), mirroring Section 3. Methods:

- **Full UMP-Net:** Complete pipeline (Section 2) with MLP prompt generation (n=40 prompts), KNN clustering (K=3), conformal scoring, and attention gating.
- **UMP-Lite (Cached):** Precompute and cache cluster centroids on a calibration set (10% of training data); select nearest centroid via cosine similarity, skipping clustering and conformal steps.
- **Baseline:** LLaMA-Excitor (Zou et al., 2024), a PEFT method without uncertainty-aware components.

Table 22: Counter-examples and why UMP-Net wins.

Task	Key Evidence	Why UMP Wins
T1	Respects 2-sentence constraint; synthesizes Plato/Aristotle; explicit preference rationale.	Higher IA/RQ/FF vs. baselines that were generic or violated constraints.
T2	Implements spec exactly (weights==0 → NaN), minimal code, vectorized.	Others misinterpret task or produce convoluted loops; UMP clearer & correct.
T3	Partial pivoting; correct forward/back-substitution; avoids singular pitfalls.	Baselines contain structural errors (e.g., bogus rows or missing pivot checks).
T4	Names concrete policies (Elizabethan Settlement; Meiji centralization); balances contrast.	More specific and sourced; baselines are surface-level or generic.
T5	More granular visual nouns/relations; fewer hallucinations.	Better MG/FF; captions contain richer but grounded detail.
T6	Sketched rationale consistent with visual cues; answer matches ground truth.	Slight but consistent gains in FF/RQ; clearer tie to image context.

Table 23: Diagnostic analysis of UMP-Net's theoretical advantages on ScienceQA (100 samples).

Method	Prompt Diversity (Var)	ECE	ScienceQA Acc (%)
UMP-Net	0.85	0.04	88.41
LLaMA-Excitor	0.62	0.09	84.9
No MoP	0.50	0.12	82.3

Metrics: Latency (ms/sample), GFLOPs, and ScienceQA accuracy (%), profiled via PyTorch's `torch.utils.benchmark`.

Table 24: Ablation study on inference efficiency, comparing Full UMP-Net, UMP-Lite (cached prompts), and LLaMA-Excitor on ScienceQA.

Variant	Latency (ms/sample)	GFLOPs	ScienceQA Acc (%)
Full UMP-Net	150	5.2	88.41
UMP-Lite (Cached)	95	3.5	87.8
LLaMA-Excitor	80	3.0	87.87

Table 24 shows Full UMP-Net's overhead (150 ms/sample, 5.2 GFLOPs) versus LLaMA-Excitor (80 ms/sample, 3.0 GFLOPs), primarily from KNN clustering (30%) and conformal scoring (25%). UMP-Lite reduces latency by 37% and compute by 33% with a minimal 0.6% accuracy drop, confirming the efficacy of cached prompt selection for frequent tasks Zhang et al. (2024). This supports UMP-Net's efficiency claims while addressing practical deployment concerns.

A.1.5 Computational Efficiency Analysis

Cost Efficiency Analysis. To evaluate UMP-Net's computational cost, we measured inference latency, throughput, VRAM usage, and FLOPs on a ScienceQA subset (100 samples, multi-modal visual-text questions), with batch sizes (B=1, 8), text lengths (L=512, 1024), and image resolution (336px for vision-language, VL) Zou et al. (2024). We compared UMP-Net to LLaMA-Adapter, isolating MoP's overhead. Table 25 shows the results.

Also to isolate MoP's overhead, we profiled component timings (4090, VL, B=1, L=512, 336px). Table 26 shows the results.

As it is illustrated in results, UMP-Net's latency is modestly higher (+4.8–10.2%) than LLaMA-Adapter's, with throughput of 78.3 tokens/s and 11.0 images/s (4090, VL) and 380 tokens/s and 39.0 images/s (A100, VL). MoP components (MoPs MLP, CUE, attention gate) add only 15 ms (0.9% of 1740 ms total latency), with CLIP encoder (95 ms) and LLaMA forward (1630 ms) dominating. VRAM usage (+2.1–3.8%) and FLOPs (0.81–6.20 T) remain practical. The overhead is justified by UMP-Net's 3.5% ScienceQA accuracy gain (Table 23), supporting robust multi-modal applications (Zou et al., 2024).

Table 25: Inference cost (mean ± sd) for UMP-Net vs. LLaMA-Adapter. Δ% = (UMP-Net - LLaMA-Adapter) / LLaMA-Adapter.

GPU	Task	B	L	ImgRes	Latency (ms)	Tokens/s	Images/s	VRAM (GB)	FLOPs (T)	ΔLatency%
4090	Text	1	512	—	1300 ± 40	101.2 ± 3.0	—	14.4	0.81	+4.8%
4090	VL	1	512	336	1740 ± 55	78.3 ± 2.1	11.0 ± 0.4	16.7	1.21	+10.1%
A100	Text	8	1024	—	1930 ± 60	505 ± 12	—	28.1	4.48	+4.3%
A100	VL	8	1024	336	2600 ± 85	380 ± 11	39.0 ± 1.2	33.0	6.20	+10.2%

Table 26: Component breakdown (4090, VL, B=1, L=512, 336px). Times in ms.

Component	Time (ms)
CLIP encoder	95
MoPs MLP	8
CUE (uncertainty scoring)	5
Attention gate (softmax)	2
LLaMA forward	1630