# Quantifying and Mitigating Selection Bias in LLMs: A Transferable LoRA Fine-Tuning and Efficient Majority Voting Approach

**Anonymous ACL submission**

## Abstract

Multiple-choice question answering (MCQA) is a widely-used method for evaluating the performance of Large Language Models (LLMs). However, LLMs often exhibit selection bias in MCQA tasks, where their choices are influenced by factors like answer position or option symbols rather than the content. This bias undermines the reliability of MCQA as an evaluation framework. Most existing selection bias metrics require answer labels and measure divergences between prediction and answer distributions, but do not fully capture the consistency of a model's predictions across different orderings of answer choices. Existing selection bias mitigation strategies have notable limitations: majority voting, though effective, is computationally prohibitive; calibration-based methods require validation sets and often fail to generalize across datasets. To address these gaps, we propose three key contributions: (1) a new unsupervised label-free **Permutation Bias Metric (PBM)** that directly quantifies inconsistencies in model predictions across answer permutations, providing a more precise measure of selection bias, (2) an efficient majority voting approach called Batch Question-Context KV caching (BaQCKV), to significantly reduce computational costs while preserving bias mitigation effectiveness, and (3) an unsupervised Low-Rank Adaptation (LoRA)-1 fine-tuning strategy based on our proposed metric and the BaQCKV that mitigates selection bias, providing a computationally efficient alternative that maintains model generalizability. Experiments across multiple MCQA benchmarks demonstrate that our approaches reduce bias, increasing consistency in accuracy while minimizing computational costs.

## 1 Introduction

Selection bias in Large Language Models (LLMs) has been increasingly recognized as a significant challenge, particularly in multiple-choice question (MCQ) answering tasks (Wei et al., 2024a; Zheng et al., 2024; Zong et al., 2023). This bias occurs when models exhibit a preference for certain answer choices based on factors like their position or symbolic representation, rather than the content itself (Wei et al., 2024a). For instance, LLMs may disproportionately favor the last option or option "A" across different questions. Such biases are especially problematic in evaluation settings, where multiple-choice formats are widely used for example, in standardized testing, professional certification exams, and educational assessments. These biases undermine the fairness and reliability of model evaluations, as they can lead to inconsistent answers across equivalent permutations, eroding trust in LLM-based decision systems.

The presence of selection bias in LLMS was highlighted by (Zheng et al., 2024), demonstrating how factors like answer position and symbolic representation can lead to systematic errors in MCQ answering. Effectively addressing selection bias requires a well-defined metric for bias quantification. Several metrics have been proposed to measure the selection bias such as the Choice Kullback-Leibler Divergence (CKLD) (Choi et al., 2024), Standard Deviation of Recalls (RStd) (Zheng et al., 2024), and Relative Standard Deviation (RSD) (Croce et al., 2021; Reif and Schwartz, 2024), which primarily evaluate bias in terms of divergence from ground truth distributions (i.e., CKLD) or variability in class-wise performance (i.e., RStd and RSD). However, they do not adequately capture the bias exhibited by models to option permutations. Also, the Fluctuation Rate proposed by (Wei et al., 2024b) only considers two permutations of the options, which may not capture the full permutation bias. We therefore introduce a **new permutation bias metric (PBM)** that evaluates selection bias in LLM without requiring ground truth distributions while considering all possible option permutations. The primary intuition behind our metric is that *log-*

*ically, an answer's correctness does not change based on its position in a list of options and we therefore want language models to possess this behaviour.*

Addressing the problem of bias requires not just quantifying but also mitigating bias. Prior mitigation strategies like majority voting (Zong et al., 2023) that aggregates predictions across all permutations of answer choices - has been shown to reduce bias. However, its computational cost increases factorially with the number of choices, making it impractical for real-time inference. Thus, a key challenge is to develop an efficient method for bias quantification and bias mitigation that can be integrated into real-world systems. We therefore propose **Batch Question-Context KV caching** (BaQCKV), an efficient implementation of majority voting that reduces computational cost considering all permutations. Additionally, we introduce an unsupervised Low-Rank Adaptation finetuning strategy that optimizes the model on our proposed metric.

Our contributions can be summarized as follows:

- We propose a **novel, unsupervised, and label-free Permutation Bias metric (PBM)** that captures inconsistencies in model predictions across all permutations of answer choices. Unlike prior metrics, it requires no access to ground-truth labels and directly measures permutation sensitivity.

- We introduce **BaQCKV (Batch Question-Context KV caching)**, a computationally efficient variant of majority voting that significantly reduces the overhead associated with evaluating all permutations of multiple-choice questions.

- We develop a LoRA-based fine-tuning strategy that leverages our proposed bias metric as a differentiable objective, enabling parameter-efficient debiasing without the need for labeled data or full model retraining.

Our efficient BaQCKV method achieves token savings of up to **54.4%**, while our unsupervised LoRA-1 fine-tuning reduces the PBM bias by an average of **58%** and improves standard deviation of accuracy by **27%**, outperforming existing approaches. BaQCKV is particularly well-suited for evaluation or deployment scenarios where deterministic and fully permutation-invariant responses are required, as it can achieve 0 bias but with additional compute and latency. In contrast, LoRA-1 fine-tuning offers a lightweight, one-pass inference alternative practical for large-scale LLM deployments or latency-sensitive settings.Together, these contributions can lead to a unified framework for quantifying and mitigating selection bias in LLMs, particularly in the context of multiple-choice question answering

## 2 Related Work

Large Language Models (LLMs) exhibit systematic selection biases in multiple-choice question answering (MCQA), favoring options by position (e.g., last choice) or by identifier (e.g., option "A") rather than semantic content. In this review, we focus on existing approaches for bias quantification and mitigation.

### 2.1 Bias Evaluation in LLMs

Several metrics have been proposed to quantify selection bias in LLMs' predictions. A common approach is to compute some divergence between the predicted answer distribution and the ground truth answer distribution. Choice Kullback–Leibler Divergence (CKLD) (Choi et al., 2024) measures the divergence between the model's answer frequency and the ground-truth answer frequency (e.g., how often the correct answer is A, B, C, etc.). However, CKLD requires access to the labelled answers. Other metrics focus on variability in per-option accuracy or recall. The Relative Standard Deviation (RSD) (Reif and Schwartz, 2024) computes the standard deviation of the model's accuracy for each answer option ID, while the Standard Deviation of Recall (RStd) (Zheng et al., 2024) uses recall instead of accuracy. These capture whether the model performs significantly better on one option (e.g., 'A') than on others, which could indicate the presence of bias. While useful, such label-dependent metrics fail to capture *inconsistency across permutations*: they only consider the original order of options, but not all possible permutations. They do not consider if a model would answer the same question differently when choices are presented in a different option permutation, since they only evaluate against the single correct label in the original ordering.

Wei et al. (2024a) introduced the Fluctuation Rate (FR) as a label-free measure of sensitivity. FR computes the rate at which an LLM's answer changes when the order of options is reversed,

quantifying how option shuffling alone affects its output. This metric revealed substantial instability in model answers due to option order. However, FR considers only two permutations (original vs. reversed), which limits its expressiveness—real-world biases may span more complex permutation patterns. Moreover, FR captures only discrete prediction flips (i.e., changes in the argmax choice) and ignores subtle changes in model confidence. *This limitation also makes FR non-differentiable, which prevents it from being directly used as an optimization objective in gradient-based fine-tuning.*

In summary, current metrics are either label-dependent (CKLD, RSD/RStd) or permutation-limited (FR), offering incomplete views of bias. To address this, we propose a ***permutation-sensitive, label-agnostic metric*** that captures prediction consistency across all answer orderings. Our approach enables broad applicability on unlabeled datasets and introduces a ***differentiable objective for debiasing during fine-tuning*** (section 3.3.2).

### 2.2 Bias Mitigation Strategies

Several research works have explored various strategies to mitigate selection bias in LLMs, ranging from calibration, voting, to finetuning-based approaches.

**Calibration-based** methods aim to adjust the model's output probabilities to compensate for the skewed bias distributions. Most of these methods target the recalibration to improve accuracy. CalibraEval (Li et al., 2024) reweights predictions to reduce positional bias in model evaluation, while label bias calibration (Reif and Schwartz, 2024) improves few-shot accuracy using known label statistics. (Zheng et al., 2024), proposed PriDe, which estimates a prior probability for all options IDs (the symbol that precedes an option content. For example A,B,C, or D) and debiases model predictions by dividing predictions by the priors and normalizing the resulting probabilities. The approach was shown to reduce selection bias as measured by the RStd metric. These approaches usually require an in-domain validation set or known answer statistics to compute the calibration parameters (prior probabilities). Even though calibration can correct distributional bias (making the overall frequency of each option more balanced), it may not ensure consistency across permutations because it uses the same calibration parameters across all permutations. A model could still change its answer when options are permuted, even if its overall option frequencies look unbiased. Prompt-level tactics like Auxiliary Option Injection (AOI) (Choi et al., 2024) offer a lightweight alternative by inserting a dummy option to nudge the model toward balanced outputs. While simple and requiring no model changes, such prompt-level fixes may only partially reduce bias and might not generalize across different prompt formats or tasks.

**Majority voting** across permutations is a simple but effective method to mitigate permutation bias by aggregating predictions over multiple answer orderings, biases tied to option positions can be reduced (Zong et al., 2023). In practice, this means querying the LLM with every possible ordering of the options and taking the majority or consensus answer (Wei et al., 2024b). However, this approach is computationally prohibitive: with $k$ options, there are $k!$ permutations, which grows factorially. Even for a 4-option multiple choice, 24 passes through the model are needed per question, which is infeasible for large-scale or real-time use. Recent works attempt to retain the benefits of majority voting while cutting down its cost. Zhou et al. (2024) introduced a batch prompting and calibration technique that allows the model to consider multiple re-ordered options in a single forward pass, making the process more efficient. Guda et al. (2024) showed that a smaller subset of random permutations can be used to approximate the full majority vote, achieving a more uniform selection without evaluating all orderings. Another line of work is self-consistency: rather than permuting inputs, the model is run multiple times with variations in its reasoning process (e.g., different sampled chains-of-thought) and the most consistent answer is chosen (Wang et al., 2024; Kim et al., 2024). (Chen et al., 2023) showed that this can be extended to open-ended generation problems. Enforcing this kind of self-consistency has been shown to improve answer reliability and reduce biases, in a manner analogous to majority voting by averaging over diverse inference paths(Wang et al., 2024). These voting-based methods share a drawback: they require multiple model evaluations, incurring high computational cost and latency. Our approach tackles this by proposing Batch Question-Context KeyValue caching (BaQCKV) – an optimized majority voting by minimizing repeated computation across permutations to cut cost.

Instead of repeatedly querying a biased model at inference time, another strategy is to **adjust the parameters** that induce bias within the model itself.

Teacher–student distillation (Liusie et al., 2024) transfers behavior from a debiased teacher to a compact student, achieving bias reduction with lower inference cost. Choi et al. (2024) proposed Bias Node Pruning (BNP), which identifies a bias vector in the final decoder layer and prunes parameters in the projection matrix based on their interactions with this vector. The resulting model can then be used in MCQ tasks without additional runtime overhead. While such methods reduce metrics like FR, they may negatively impact the model's performance on other tasks due to irreversible weight pruning or overfitting to specific bias patterns.

In summary, training-time debiasing techniques including knowledge distillation, fine-tuning, and structural pruning aim to internalize bias mitigation and reduce the need for repeated inference-time interventions. However, these often require labeled datasets and can compromise generalization.

**In contrast, our approach introduces a fully differentiable, label-free bias objective-PBM-that enables targeted fine-tuning to reduce permutation sensitivity.** This allows for unsupervised debiasing that generalizes across datasets.

## 3 Methodology

Our approach to addressing selection bias involves bias quantification and mitigation. In this section, we define our bias quantification metric which accounts for all option permutations and also describe the mitigation strategies - BaQCKV and LoRa-1 finetuning.

### 3.1 Permutation Bias Metric (PBM)

PBM is based on the intuition that, logically, a model's confidence in an option should be invariant to the permutation of the options. Also, we argue that this quantification should be label-free because the confidence for each option content across all permutations should be constant regardless of option correctness. Let $Q$ represent a question, and $O = \{o_1, o_2, \ldots, o_n\}$ represent a set of $n$ options for the question. A model processes the sequence $S_\pi = Q \oplus O_\pi$ for a permutation $\pi$ of the options $O$, where $\oplus$ denotes the concatenation operator. Passing a permuation of the options $S_\pi$ through the model assigns probabilities $P(o_{\pi(i)} \mid Q, O_\pi)$ to each option content $o_i$. Similarly,for a different permutation $\pi^{'}$ it assigns $P(o_{\pi^{'}(i)} \mid Q, O_{\pi^{'}})$. We define our selection bias metric mathematically for a model in Equation (1) as the variance of the probabilities for each option content across all permutations. This will capture how much the model's confidence fluctuates due to reordering.

**Definition 1** (Permutation Bias Metric – PBM)**.** Given a question $Q$ and a set of answer options $O = \{o_1, o_2, \ldots, o_n\}$, the selection bias $B(Q, O)$ is defined as:

$$B(Q,O) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{Var}_\pi\left(P(o_{\pi(i)} \mid Q, O_\pi)\right), \quad (1)$$

We refer to this selection bias score as the **Permutation Bias Metric (PBM)**, where:

$$\begin{aligned} &\mathrm{Var}_\pi\left(P(o_{\pi(i)} \mid Q, O_\pi)\right) = \\ &\frac{1}{n!}\sum_{\pi}\Bigg(P(o_{\pi(i)} \mid Q, O_\pi) \\ &-\mathbb{E}_\pi\left[P(o_{\pi(i)} \mid Q, O_\pi)\right]\Bigg)^2, \end{aligned} \quad (2)$$

and the expectation over all permutations $\pi$ is:

$$\mathbb{E}_\pi[P(o_{\pi(i)} \mid Q, O_\pi)] = \frac{1}{n!}\sum_{\pi} P(o_{\pi(i)} \mid Q, O_\pi).$$

By defining the **PBM** to be proportional to the variance of a model's prediction across all permutations, the metric captures the inconsistency in the predictions. A lower variance indicates more stable and consistent predictions across permutations. Therefore, lower values of the **PBM** are preferred. **PBM** is label-free compared to previously proposed metrics like RStd (equation (4)), FR (equation (3)) and CKLD (equation (5)) which capture some form of divergence from the answer distribution, requiring the ground truth answers. The RStd and CKLD do not capture any information about the option permutations because they only use the original permutations of the options.This assumes the original permutation corresponds to the fixed answer order provided in the dataset, which is often arbitrary (i.e, randomly assigned during test construction). Also, while **PBM** goes into the granular confidence level by considerng option probabilities, the FR does not capture granular information of the changes in confidence but only checks if the chosen answer (argmax) is the same in the forward and reverse permutations. By considering only 2 permutations, it does not capture information from other permutations. The flunctuation rate (FR) is expressed as;

$$FR = \frac{1}{N}\sum^{N}\Big(\underset{i}{\text{argmax}}\big(P(O \mid Q, O_{\pi})\big) \neq \underset{i}{\text{argmax}}\big(P(O \mid Q, O_{\pi^r})\big)\Big) \quad (3)$$

where $\pi^r$ is the reverse permutation of $\pi$ and $N$ is the total number of questions.

$$R_{\sigma} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_i - \mu_r)^2} \quad (4)$$

where $n$ represents the number of option classes, $r_i$ is the recall for the $i$-th class, and $\mu_r$ denotes the average recall across all option classes.

$$CKLD = \sum_{i}^{n} p_i \log \frac{p_i}{q_i} \quad (5)$$

where $p_i$ is the ratio of ground truth choice label for option ID $i$ and $q_i$ is the ratio for predictions.

### 3.2 Investigation into the cause of Selection Bias in LLMs

In the decoder-only transformer architecture, which is prevalent in most large language models (LLMs), each token is generated based on causal attention. This causal attention mechanism ensures that predictions are conditioned only on previously generated tokens. To preserve the sequential structure of the input, positional encodings are applied during attention computation. When a question and its permuted-option variant are provided as input, the set of unique tokens remains unchanged. However, the reordering alters the positional encodings assigned to each token. Since positional encodings influence attention scores, this modification can lead to differences in the model's output, even if the semantic token content remains the same (see Appendix A.1 and Appendix A.2).

### 3.3 Bias Mitigation Methodology

In the following sections, we explain our efficient BaQCKV approach and the unsupervised LoRa-1 debiasing. BaQCKV, an efficient majority voting variant that enforces permutation invariance (zero bias) with reduced compute via batched inference, ideal for critical evaluations requiring strict consistency; and LoRA-1, a lightweight adaptation method trained using our unsupervised bias metric to debias models for single-pass inference, suitable for large-scale deployments. BaQCKV trades computational overhead for robustness, while LoRA-1 prioritises scalability with minimal latency This can enable users to balance bias mitigation against operational constraints.

#### 3.3.1 Efficient Majority Voting with BaQCKV

The majority voting is an effective mitigation strategy for selection bias (Zong et al., 2023). It passes all option permutations of a question through the model and chooses the option with the highest average score across the permutations. This scheme enforces permutation invariance (0 bias on our metric) by ensuring that an option has the same confidence across all permutations, making it an ideal selection bias mitigation strategy. Mathematically, majority voting calculates

$$i^* = \underset{i \in O}{\text{argmax}}\, \mathbb{E}_{\pi}\left[P(o_{\pi(i)} \mid Q, O_{\pi})\right], \quad (6)$$

where $O = \{o_1, \ldots, o_n\}$ is the set of answer options, $\pi$ denotes a permutation of the options, and $\mathbb{E}_{\pi}$ represents the average over all $n!$ permutations (as defined in Section 3.1).

In spite of its effectiveness in mitigating bias,the majority voting has not been widely adopted, as the computational complexity of making predictions on all possible permutations is $n!$ for an MCQ with $n$ options. This can be easily reduced by defining a fixed number $k$ and considering only $k$ permutations instead of $n!$, reducing the computational cost (Guda et al., 2024). Thus, $p_i = \frac{1}{k}\sum_{j=1}^{k} p_{ji}$. However, this scheme can be made even more efficient, without a corresponding loss in bias, by employing a KV cache. To do so, we leverage the insight that while an MCQ consists of a question $Q$ (with or without a context), and a set of options, $O$, *the question, $Q$, remains the same across all possible option permutations.*

For $k$ permutations of the options, the original formulation of the majority voting (Guda et al., 2024) requires $k$ passes through the LLM, resulting in an additional overhead of $(k-1) \times |Questions \oplus Context \oplus Options|$ tokens per question. We however, note that the set of $Questions \oplus Context$ tokens remains constant across all $k$ passes for each question in a batch. To eliminate the redundant computation of these tokens across the batch, we are motivated by the KV cache in (Pope et al., 2023) to introduce the BaQCKV, which caches and reuses the KV states of the $Questions \oplus Context$ tokens for a set of $k$ permutations. This cached KV state is pre-pended to the KV states of the $k$ permuted options. The attention mask of the permuted options is then expanded based on the length of the

$Questions \oplus Context$ tokens to ensure that the LLM's attention is correctly computed. We show in

**Algorithm 1** Efficient Majority Inference with BaQCKV

**procedure** BAQCKVINFERENCE($Q_C, O_k, \mathcal{M}$)
  **Input:** $Q_C$ - Question $\oplus$ Context tokens, $O_k$ - $k$ permutations of options, $\mathcal{M}$ - Language Model, **Output:** $\mathcal{Y}_k$ - Model outputs

  **Step 1: Cache Question-Context KV States**
  $\texttt{KV}_{Q_C} \leftarrow \mathcal{M}.\text{encode}(Q_C)$
  **Step 2: Compute KV States for Permuted Options**
  **for** $i = 1$ to $k$ **do**
    $\texttt{KV}_{O_i}, \texttt{mask}_i \leftarrow \mathcal{M}.\text{encode}(O_i)$
  **end for**
  **Step 3: Merge and Adjust KV States**
  **for** $i = 1$ to $k$ **do**
    $\texttt{KV}_i \leftarrow \texttt{KV}_{Q_C} \oplus \texttt{KV}_{O_i}, \quad \texttt{mask}_i \leftarrow \mathbf{1}_{|Q_C|} \oplus \texttt{mask}_i$
  **end for**
  **Step 4: Compute Batch Outputs**
  $\mathcal{Y}_k \leftarrow \{\mathcal{M}.\text{decode}(\texttt{KV}_i, \texttt{mask}_i) \mid i = 1, 2, \ldots, k\}$
  **return** $\mathcal{Y}_k$
**end procedure**

Appendix A.3 that the percentage of tokens by using the BaQCKV is defined by Equation (7), where $C$ is the optional set of context tokens for the Question $Q$.

$$\text{Token savings (\%)} = \frac{(k-1) \times |Q \oplus C|}{k \times |Q \oplus C \oplus O|} \times 100 \tag{7}$$

In Equation (7), the savings are maximized when $|C|$ is large, as in Retrieval-Augmented Generation (RAG), where redundant computation is minimized. Even when $|C| = 0$, savings persist due to the shared $|Q|$ tokens. Larger permutation sizes $k$ further amplify savings by increasing redundancy in $|Q \oplus C|$ across permutations. Thus, BaQCKV is most effective in tasks with substantial shared context, multiple options, and large permutation sizes.

### 3.3.2 Unsupervised Lora-1 Bias Mitigation

We introduce an unsupervised fine-tuning of **PBM**, our permutation-based bias metric to mitigate the selection bias. This is because PBM is fully differentiable unlike the Fluctuation Rate (FR) and the Standard Deviation of Recalls (RStd). In addition, it is also label-free, unlike all the other metrics including the CKLD. We make two adjustments to the metric when using it as a loss for fine-tuning (equation (8)). Firstly, to ensure that there is an adequate flow of information from the gradients we take the variance of the log of the probabilities (Equation (9)). When obtaining the option probabilities from the model, we only consider the logits that correspond to the option IDs instead of logits for the entire vocabulary. Secondly, we observe that a model can learn to minimize the bias by simply predicting a uniform probability for all options IDs across all permutations. In that case, the mean probability for all option IDs would be the same as the uniform probabilities assigned to all option IDs for all permutations. To prevent this, we regularize the bias with the entropy across the option IDs (Equation (10)). This helps in making the model maintain its confidence in the chosen answer while also minimizing the bias across different permutations. $\lambda$ is a hyper-parameter that balances the model's confidence in an answer and minimizing the bias. Computing the loss defined in Equation (8) can be expensive because each example needs to be expanded to the full permutations. To reduce this cost, first, we apply the BaQKCV to compute the loss and only sample a maximum of 24 permutations for questions with more than 4 options.

$$\text{Loss} = B(Q, O)_{log} + \lambda H(Q, O) \tag{8}$$

where

$$B(Q, O)_{\text{log}} = \sum_{\pi} \Big( \log \left( P(o_{\pi(i)} \mid Q, O_{\pi}) \right) - \log \left( \mathbb{E}_{\pi} \left[ P(o_{\pi(i)} \mid Q, O_{\pi}) \right] \right) \Big)^2 \tag{9}$$

$$H(Q, O) = -\sum_{\pi} \sum_{i} (P(o_{\pi(i)} \log(P(o_{\pi(i)})) \tag{10}$$

As LLMs are desired to be used for a wide variety of tasks and not just answering MCQs, we adopt the LoRa fine-tuning to preserve the original performance of the LLM on non-MCQ tasks while avoiding expensive training. The LoRa debiasing weight adapters can be connected when the model is used for MCQ.

## 4 Results

**Datasets and Models :** For our experiments, we employed three small language models of comparable size: Qwen2.5-3B-Instruct (Bai et al., 2023), Phi-2 (Javaheripi et al., 2023), and Llama3.2-3B (Grattafiori et al., 2024). We experiment with these models on four diverse datasets across different domains: TeleQnA (Maatouk et al., 2023), MedMCQA(Pal et al., 2022), QASC(Khot et al., 2020) and ARC Challenge(Clark et al., 2018). ARC Challenge and MedMCQA have 4 options and TeleQnA has 2-5 options, while QASC has 8.

**Experiments:** We conduct experiments to evaluate both the accuracy and bias of various models across all datasets, and to assess the effectiveness of different selection bias mitigation strategies. First, we demonstrate that *majority voting not only reduces or eliminates selection bias (quantified by PBM) but also improves model accuracy*. Furthermore, by introducing BaQCKV, we show that *majority voting can be made significantly more efficient*, yielding substantial savings in both computation time and token usage. In addition, we assess the impact of our proposed **LoRA-1 fine-tuning method**, which consistently reduces all existing bias metrics and exhibits strong transferability across datasets.

We compare our approach against three alternative bias mitigation methods: (1) **GRAY**, a gray-box technique that leverages both forward and backward predictions to reduce bias; (2) **BNP** (Bias Node Pruning), which prunes parameters in the final projection layer that contribute to bias; and (3) **PRIDE**, which normalizes model predictions using prior probabilities estimated from the dataset.

**However, these prior methods do not achieve the consistency of majority voting, a technique well-established for mitigating selection bias** (Zong et al., 2023; Wang et al., 2024). The prompt templates and hyperparameter configurations used in our experiments are documented in Appendix C.

**Our unsupervised bias metric (PBM) correlates with the difficulty of the MCQ:** The results in Table 1 show that all models exhibit varying degrees of bias, correlating with the difficulty of the problem. Across all models, the bias is seen to be highest with the MedMCQA benchmark due to its difficulty (having the lowest accuracy). This confirms that selection bias is present and measurable using our proposed metric (PBM). This also means that PBM may be used to compare the difficulty of different MCQ datasets without having access to the labels. Notably, after applying majority voting (MV) with help of BaQCKV, the PBM value consistently drops to 0.00. Additionally, applying majority voting shows substantial improvements in accuracy, particularly in QASC, where scores increase significantly (e.g., from 0.630 to 0.9329 for Phi-2 and 0.4892 to 0.837 for Llama), validating the effectiveness of our metric in capturing and mitigating bias.

**Efficiency of the BaQCKV:** Beyond bias reduction, the BaQCKV enhances real-world applicability by significantly reducing computational costs of applying majority voting. As shown in Table 1, our efficient BaQCKV approach for the majority voting results in significant time savings of up to 88.6% (Llama3.2-3B) and over 90% across token savings across all models.This efficiency gain is crucial for deploying bias-mitigation strategies at scale during inference, making our approach feasible for real-world applications where computational cost is a limiting factor.

**The unsupervised Lora-1 Bias Mitigation demonstrates the best performance in maintaining consistency:**

The scatter plot in Figure 1a illustrates that, overall, the `lora-1` fine-tuning approach exhibits greater consistency in accuracy, characterized by a smaller standard deviation and lower selection bias **(PBM)** compared to other mitigation strategies. Ideally, an effective model should have its corresponding points converge near the origin of the plot, indicating minimal variability and bias. Moreover, as shown in Table 2, when evaluating the percentage change relative to the undebiased models, `lora-1` fine-tuning achieves the greatest average reduction in both standard deviation of accuracy ($-27\%$) and selection bias ($-58\%$) across all evaluated models and datasets. These results highlight the effectiveness of `lora-1` in mitigating variability and bias simultaneously. It also on average improves the accuracy of the models by 20% even though this is not up to the 41.92% demonstrated by PriDe. However, the PriDe demonstrates the highest bias (1.8) and hardly offers any improvements in standard deviation of accuracy. We show a training graph of the unsupervised finetuning for TeleQnA dataset process in Figure 1b.

**The transferability of the unsupervised Lora-1 Bias Mitigation:** We investigate the transferability of the unsupervised LoRA-1 fine-tuning approach by training a model on a single dataset and evaluating the resulting checkpoint on all other datasets. The average performance across each model/dataset pair is reported in Table 3, with complete results provided in Appendix B.2. As shown in Table 3, the unsupervised fine-tuning generally transfers well: it consistently reduces our permutation bias metric (PBM), fluctuation rate, and the standard deviation of accuracy across option permutations. In many cases, it also yields modest improvements in accuracy. However, similar to the non-transfer setting, this approach does not improve the CKLD metric. Notably, for the Qwen-MEDCQ model, CKLD actually increases signifi-

| Model Name | TeleQnA | | | | MedMCQA | | | | QASC | | | | ARC | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Acc | PBM | TS (seconds) | TokS (%) | Acc | PBM | TS (seconds) | TokS (%) | Acc | PBM | TS (seconds) | TokS (%) | Acc | PBM | TS (seconds) | TokS (%) |
| Qwen2.5-3B | 0.5464 | 0.021 | 1.009 | - | 0.479 | 0.058 | - | - | 0.737 | 0.011 | 0.996 | - | 0.804 | 0.029 | 0.9610 | - |
| Qwen2.5-3B + MV | 0.5710 | 0.000 | 0.0491 | 0.4583 | 0.487 | 0.000 | 0.068 | 0.5328 | 0.947 | 0.000 | 0.1562 | 0.5377 | 0.839 | 0.00 | 0.0680 | 0.3600 |
| Phi-2 | 0.2568 | 0.0303 | 0.912 | - | 0.359 | 0.082 | 0.888 | - | 0.630 | 0.024 | 0.9443 | - | 0.552 | 0.0269 | 0.8463 | - |
| Phi-2 + MV | 0.328 | 0.000 | 0.0985 | 0.3596 | 0.369 | 0.000 | 0.0576 | 0.551 | 0.9329 | 0.000 | 0.063 | 0.0543 | 0.4547 | 0.000 | 0.0579 | 0.366 |
| Llama3.2-3B | 0.4536 | 0.0053 | 0.9856 | - | 0.370 | 0.017 | 0.9549 | - | 0.4892 | 0.005 | 1.065 | - | 0.5179 | 0.0091 | 0.9371 | - |
| Llama3.2-3B + MV | 0.516 | 0.000 | 0.065 | 0.4618 | 0.384 | 0.000 | 0.0619 | 0.533 | 0.837 | 0.000 | 0.0551 | 0.545 | 0.537 | 0.00 | 0.0804 | 0.3639 |

Table 1: Accuracy and bias values for different models across multiple datasets, along with computational efficiency improvements using Majority Voting (MV) with BaQCKV.

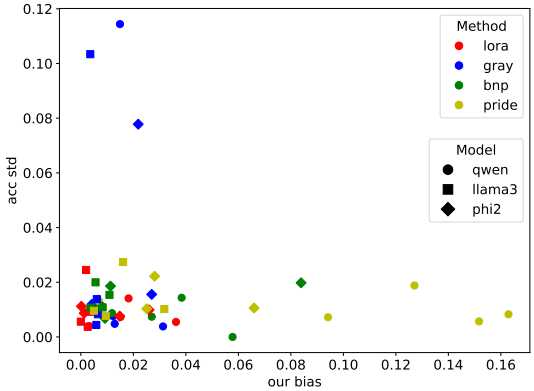
.

| Method | **PBM** ↓ | RStd ↓ | CKLD ↓ | FR ↓ | Acc ↑ | AccStd ↓ |
|---|---|---|---|---|---|---|
| LoRA | **-0.586** | -0.076 | 0.928 | **-0.525** | 0.200 | **-0.276** |
| Gray | -0.364 | -0.045 | 0.677 | – | 0.077 | 0.940 |
| BNP | -0.119 | 0.000 | 0.653 | -0.250 | 0.064 | -0.131 |
| Pride | 1.880 | **-0.240** | **0.432** | -0.137 | **0.419** | -0.040 |

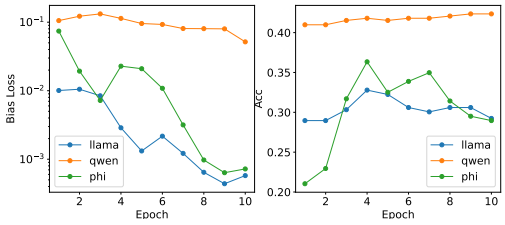
Table 2: Comparison of mitigation methods across bias, stability, and accuracy. We did not compute FR for Gray box, since it is always 0 by design. **LoRA** is our method. **Gray** (Choi et al., 2024), **BNP** (Choi et al., 2024), and **Pride** (Zheng et al., 2024) are prior approaches.

| Model-Dataset$_{train}$ | PBM ↓ | RStd ↓ | CKLD ↓ | FR ↓ | Acc ↑ | AccStd ↓ |
|---|---|---|---|---|---|---|
| QWEN-MEDCQ | -0.497 | -0.411 | 207.74 | -0.150 | 0.028 | -0.279 |
| QWEN-TeleQNA | -0.509 | 0.710 | 4.01 | -0.160 | 0.029 | -0.206 |
| QWEN-ARC | -0.413 | 1.017 | 4.63 | -0.126 | 0.026 | -0.234 |
| QWEN-QASC | -0.319 | 1.168 | 5.06 | -0.006 | -0.000 | -0.153 |
| Phi2-ARC | -0.640 | -0.712 | -0.90 | -0.764 | 0.423 | -0.371 |
| Llama3.2-TeleQNA | -0.687 | -0.425 | -0.60 | -0.760 | 0.282 | -0.479 |
| Llama3.2-ARC | -0.799 | -0.314 | 0.18 | -0.651 | 0.220 | -0.300 |
| Llama3.2-QASC | -0.863 | 0.213 | 0.54 | -0.627 | -0.085 | -0.073 |

Table 3: The average percentage change in bias metrics and accuracy for transferability experiments. For each row, the model-dataset is evaluated on all other datasets excluding the one used for finetuning




(a) Standard deviation of accuracy across permutations vs bias across all models and datasets for different mitigation strategies.




(b) Lora1 finetuning training graph showing bias and accuracy for the TeleQnA dataset.

Figure 1: Visualization of bias-related behaviors across models and strategies.

cantly, accompanied by a more than 200% rise in RStd. This may be due to the model shifting toward greater consistency across permutations, which can lead to more uniform confidence distributions that diverge from dataset-specific label frequencies. Importantly, this behavior aligns with our objective of reducing positional sensitivity.

## 5 Conclusion

In this work, we address a critical yet underexplored challenge in Multiple-Choice Question Answering (MCQA)—selection bias in Large Language Models (LLMs). We introduced a novel unsupervised, label-free bias metric (PBM) that directly quantifies inconsistencies in predictions across permuted answer choices, offering a more faithful measure of selection bias than existing methods. To mitigate this bias without incurring prohibitive computational costs, we proposed BaQCKV, an efficient majority voting strategy, and LoRA-1, a lightweight fine-tuning method grounded in our bias metric. Our experiments across diverse MCQA datasets and models demonstrate that these techniques not only reduce bias and improve accuracy but also significantly cut down inference time and token usage, making them scalable and practical for real-world deployment. Ultimately, our work provides both a theoretical and practical framework for more reliable and efficient MCQA with LLMs, laying the foundation for future advancements in fair and robust AI systems.

## 6 Limitations

This work only focuses on decoder only transformer language models and did not investigate bias in other language models such as encoder-

decoder models. Also, we only investigate MCQs where the model has to choose one option and do not consider other types of MCQs.

## References


Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, and 29 others. 2023. Qwen technical report. *Preprint*, arXiv:2309.16609.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. 2023. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*.

Hyeong Kyu Choi, Weijie Xu, Chi Xue, Stephanie Eckman, and Chandan K. Reddy. 2024. Mitigating selection bias with node pruning and auxiliary options. *Preprint*, arXiv:2409.18857.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457.

Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. 2021. Robustbench: A standardized adversarial robustness benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Blessed Guda, Gabrial Zencha Ashungafac, Lawrence Francis, and Carlee Joe-Wong. 2024. Qmos: Enhancing llms for telecommunication with question masked loss and option shuffling. *Preprint*, arXiv:2409.14175.

Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, and 1 others. 2023. Phi-2: The surprising power of small language models. *Microsoft Research Blog*.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition. *Preprint*, arXiv:1910.11473.

Minjoon Kim and 1 others. 2024. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2404.56789*.

Haitao Li, Junjie Chen, Qingyao Ai, Zhen Chu, Yujie Zhou, Qian Dong, and Yong Liu. 2024. Calibraeval: Calibrating prediction distribution to mitigate selection bias in llms-as-judges. *arXiv preprint arXiv:2410.15393*.

Adian Liusie, Yassir Fathullah, and Mark J. F. Gales. 2024. Teacher-student training for debiasing: General permutation debiasing for large language models. *Preprint*, arXiv:2403.13590.

Ali Maatouk, Fadhel Ayed, Nicola Piovesan, Antonio De Domenico, Merouane Debbah, and Zhi-Quan Luo. 2023. Teleqna: A benchmark dataset to assess large language models telecommunications knowledge. *Preprint*, arXiv:2310.15051.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. Medmcqa : A large-scale multi-subject multi-choice dataset for medical domain question answering. *Preprint*, arXiv:2203.14371.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5:606–624.

Yuval Reif and Roy Schwartz. 2024. Beyond performance: Quantifying and mitigating label bias in llms. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6784–6798.

Xinyi Wang and 1 others. 2024. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2403.45678*.

Sheng-Lun Wei, Cheng-Kuang Wu, Hen-Hsen Huang, and Hsin-Hsi Chen. 2024a. Unveiling selection biases: Exploring order and token sensitivity in large language models. *arXiv preprint arXiv:2406.03009*.

Sheng-Lun Wei, Cheng-Kuang Wu, Hen-Hsen Huang, and Hsin-Hsi Chen. 2024b. Unveiling selection biases: Exploring order and token sensitivity in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5598–5621, Bangkok, Thailand. Association for Computational Linguistics.

Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. 2024. Large language models are not robust multiple choice selectors. In *The Twelfth International Conference on Learning Representations*.

Han Zhou, Xingchen Wan, Lev Proleev, Diana Mincu, Jilin Chen, Katherine Heller, and Subhrajit Roy. 2024. Batch calibration: Rethinking calibration for in-context learning and prompt engineering. *arXiv preprint arXiv:2309.17249*.

Yongshuo Zong, Tingyang Yu, Ruchika Chavhan, Bingchen Zhao, and Timothy Hospedales. 2023. Fool your (vision and) language model with embarrassingly simple permutations. *arXiv preprint arXiv:2310.01651*.

# A Appendix

## A.1 Positional Encoding and Sensitivity of Transformers to Option Permutations

In decoder-only transformers, token generation is conditioned on causal self-attention, where each token attends to prior tokens using both content-based embeddings and positional encodings. When a question $Q$ and its associated options $O = \{o_1, o_2, \ldots, o_n\}$ are presented, the model processes the sequence:

$$S = Q \oplus O.$$

For a permutation $\pi$ of the options, the modified sequence becomes:

$$S_\pi = Q \oplus O_\pi.$$

While the token set remains unchanged, the reordering affects positional encodings, altering attention computations. The self-attention mechanism computes attention scores between tokens at positions $i$ and $j$ as:

$$\text{Attention}_{i,j} = \frac{Q(o_i + p_i) \cdot (K(o_j + p_j))^T}{\sqrt{d}}.$$

For the permuted sequence $S_\pi$, the updated scores are:

$$\text{Attention}^{\pi}_{i,j} = \frac{Q(o_i + p_{\pi(i)}) \cdot (K(o_j + p_{\pi(j)}))^T}{\sqrt{d}}.$$

Since $p_i \neq p_{\pi(i)}$, the attention patterns for $S$ and $S_\pi$ differ, resulting in distinct contextual representations for the same token set.

As attention weights directly influence token representations, these changes propagate through the network, modifying the sequence representation and ultimately affecting the model's output distribution. Let $P(y \mid S)$ and $P(y \mid S_\pi)$ denote the probability distributions over possible answers. Then,

$$P(y \mid S) \neq P(y \mid S_\pi).$$

## A.2 Impact of option permutations on attention scores

The impact of the permutations owing to the positional encoding on the attention scores is illustrated in Figure 2. The figure shows that there are more fluctuations in the attention scores on the option tokens (later parts of the x-axis) compared to the question tokens (earlier tokens on the x-axis).

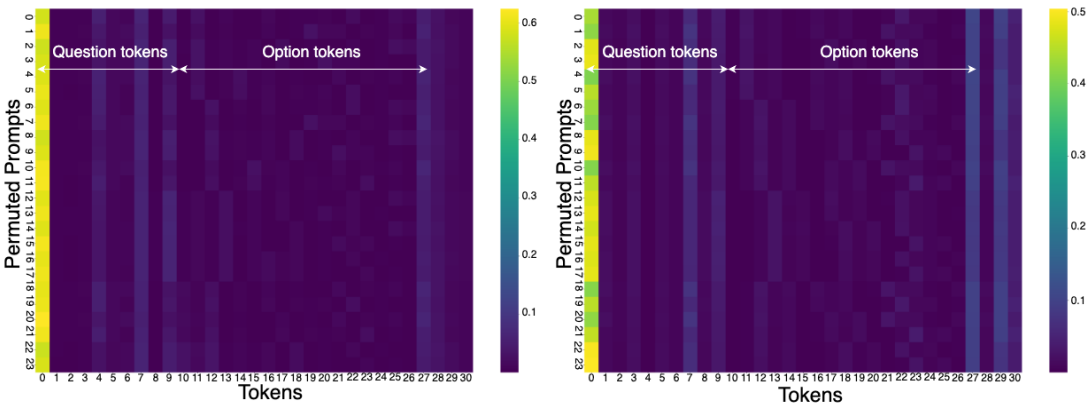



Figure 2: Attention scores for the last token in the last layer of the Llama model across different prompt permutations, shown for two transformer heads.

### A.3 Proof of Token Savings in BaQCK

In the original Majority Voting (MV) framework, each question undergoes $k$ passes through the LLM, processing the full sequence of $Q \oplus C \oplus O$ each time. The total token cost per question is:

$$\text{Cost}_{\text{MV}} = k \times |Q \oplus C \oplus O| \tag{11}$$

In BaQCK, the shared $Q \oplus C$ tokens are processed only once, while the $O$ tokens are processed $k$ times. Thus, the total token cost per question is:

$$\text{Cost}_{\text{MV}} = |Q \oplus C| + k \times |O| \tag{12}$$

The token savings is computed as:

$$\begin{aligned} \text{Savings} &= \text{Cost}_{\text{MV}} - \text{Cost}_{\text{BaQCK}} && (13)\\ &= k \times |Q \oplus C \oplus O| - (|Q \oplus C| + k \times |O|) && (14)\\ &= k \times |Q \oplus C| + k \times |O| - |Q \oplus C| - k \times |O| && (15)\\ &= (k-1) \times |Q \oplus C| && (16) \end{aligned}$$

Expressing this as a percentage of the original cost:

$$\text{Token savings (\%)} = \frac{(k-1) \times |Q \oplus C|}{k \times |Q \oplus C \oplus O|} \times 100 \tag{17}$$

This result shows that BaQCK significantly reduces token computations, particularly when $|C|$ is large (e.g., in Retrieval-Augmented Generation). Even for small or zero-context cases ($|C| = 0$), savings persist due to shared $|Q|$ tokens. Increasing $k$ further amplifies efficiency by reducing redundant recomputation across shuffled options.

## B Experimental Results

### B.1 Bias Mitigation Results on all models and datasets

Table 5: Phi-2 Performance across Datasets (Raw Scores)

| Metric | Bias | RStd | CKLD | Fluct. Rate | Accuracy | Perm Acc Std |
|---|---|---|---|---|---|---|
| *ARC Dataset* | | | | | | |
| Baseline | 0.0269 | 0.2207 | 0.2853 | 0.6334 | 0.5520 | 0.0110 |
| LoRa FineTuning | 0.0214 | 0.2102 | 0.2501 | 0.6225 | 0.5589 | 0.0082 |
| Gray-Box Weighting | 0.0187 | 0.1965 | 0.2394 | – | 0.5661 | 0.0071 |
| BNP | 0.0276 | 0.2251 | 0.2923 | 0.6881 | 0.5498 | 0.0104 |
| Pride | 0.1507 | 0.2769 | 0.2933 | 0.6987 | 0.5512 | 0.0107 |
| *TeleQNA Dataset* | | | | | | |
| Baseline | 0.0303 | 0.3559 | 1.5208 | 0.8114 | 0.2568 | 0.0216 |
| LoRa FineTuning | 0.0255 | 0.3471 | 1.4801 | 0.8041 | 0.2642 | 0.0173 |
| Gray-Box Weighting | 0.0196 | 0.3432 | 1.4273 | – | 0.2697 | 0.0152 |
| BNP | 0.0388 | 0.3674 | 1.5404 | 0.7463 | 0.2553 | 0.0201 |
| Pride | 0.1301 | 0.3865 | 1.4311 | 0.7522 | 0.2374 | 0.0214 |
| *MedMCQ Dataset* | | | | | | |
| Baseline | 0.0512 | 0.3981 | 1.4585 | 0.9369 | 0.3409 | 0.0194 |
| LoRa FineTuning | 0.0418 | 0.3907 | 1.4132 | 0.9286 | 0.3467 | 0.0168 |
| Gray-Box Weighting | 0.0356 | 0.3849 | 1.4017 | 0.0000 | 0.3523 | 0.0150 |
| BNP | 0.0542 | 0.4072 | 1.4682 | 0.8624 | 0.3397 | 0.0175 |
| Pride | 0.1489 | 0.4138 | 1.4425 | 0.8796 | 0.3402 | 0.0189 |
| *QASC Dataset* | | | | | | |
| Baseline | 0.0314 | 0.3232 | 2.9914 | 0.9676 | 0.1577 | 0.0122 |
| LoRa FineTuning | 0.0249 | 0.3014 | 2.7412 | 0.9548 | 0.1643 | 0.0103 |
| Gray-Box Weighting | 0.0221 | 0.2876 | 2.6785 | – | 0.1671 | 0.0094 |
| BNP | 0.0327 | 0.3357 | 3.0141 | 0.9251 | 0.1552 | 0.0109 |
| Pride | 0.1273 | 0.3428 | 2.8345 | 0.9387 | 0.1499 | 0.0118 |

Table 4: QWEN Performance across Datasets (Raw Scores)

| Metric | Bias | RStd | CKLD | Fluct. Rate | Accuracy | Perm Acc Std |
|---|---|---|---|---|---|---|
| *ARC Dataset* | | | | | | |
| Baseline | 0.0268 | 0.0158 | 0.0003 | 0.1997 | 0.8038 | 0.0088 |
| LoRa FineTuning | 0.0151 | 0.0101 | 0.0004 | 0.1869 | 0.808 | 0.0073 |
| Gray-Box Weighting | 0.0129 | 0.0083 | 0.0002 | – | 0.8283 | 0.0048 |
| BNP | 0.0270 | 0.0169 | 0.0004 | 0.2550 | 0.7994 | 0.0074 |
| Pride | 0.1629 | 0.0261 | 0.0007 | 0.2374 | 0.8039 | 0.0083 |
| *TeleQNA Dataset* | | | | | | |
| Baseline | 0.0214 | 0.0564 | 0.0070 | 0.5874 | 0.5464 | 0.0189 |
| LoRa FineTuning | 0.0182 | 0.0879 | 0.0223 | 0.6093 | 0.5437 | 0.0141 |
| Gray-Box Weighting | 0.0149 | 0.0179 | 0.0070 | – | 0.5492 | 0.1144 |
| BNP | 0.0384 | 0.0881 | 0.0145 | 0.2158 | 0.5546 | 0.0143 |
| Pride | 0.1271 | 0.0778 | 0.0040 | 0.3389 | 0.4186 | 0.0188 |
| *MedMCQ Dataset* | | | | | | |
| Baseline | 0.0577 | 0.0179 | 0.0006 | 0.8964 | 0.4805 | 0.0057 |
| LoRa FineTuning | 0.0363 | 0.0771 | 0.0082 | 0.9031 | 0.4798 | 0.0055 |
| Gray-Box Weighting | 0.0313 | 0.0087 | 0.0062 | 0.0000 | 0.4867 | 0.0038 |
| BNP | 0.0578 | 0.0096 | 0.0056 | 0.2421 | 0.4797 | 0.0000 |
| Pride | 0.1517 | 0.0238 | 0.0078 | 0.2476 | 0.4801 | 0.0057 |
| *QASC Dataset* | | | | | | |
| Baseline | 0.0110 | 0.0547 | 0.0117 | 0.2062 | 0.8952 | 0.0068 |
| LoRa FineTuning | 0.0031 | 0.0101 | 0.0003 | 0.1188 | 0.9698 | 0.0040 |
| Gray-Box Weighting | 0.0123 | 0.0324 | 0.0146 | – | 0.8801 | 0.0076 |
| BNP | 0.0119 | 0.0573 | 0.0129 | 0.1619 | 0.8866 | 0.0087 |
| Pride | 0.0942 | 0.0594 | 0.0060 | 0.1690 | 0.8729 | 0.0072 |

Table 6: LLama 3.2 Performance across Datasets (Raw Scores)

| Metric | Bias | RStd | CKLD | Fluct. Rate | Accuracy | Perm Acc Std |
|---|---|---|---|---|---|---|
| *ARC Dataset* | | | | | | |
| Baseline | 0.0091 | 0.1082 | 0.1050 | 0.5171 | 0.5179 | 0.0097 |
| LoRa FineTuning | 0.0034 | 0.1780 | 0.1144 | 0.1373 | 0.5339 | 0.0092 |
| Gray-Box Weighting | 0.0065 | 0.0239 | 0.0999 | 0.0000 | 0.5957 | 0.0083 |
| BNP | 0.0083 | 0.0500 | 0.0230 | 0.3279 | 0.5392 | 0.0108 |
| Pride | 0.0318 | 0.0911 | 0.0125 | 0.4789 | 0.5615 | 0.0102 |
| *TeleQNA Dataset* | | | | | | |
| Baseline | 0.0053 | 0.1010 | 0.0907 | 0.5574 | 0.4536 | 0.0165 |
| LoRa FineTuning | 0.0002 | 0.1998 | 0.3120 | 0.3306 | 0.3830 | 0.0245 |
| Gray-Box Weighting | 0.0036 | 0.0330 | 0.0828 | – | 0.4004 | 0.1034 |
| BNP | 0.0056 | 0.1786 | 0.0614 | 0.1746 | 0.4754 | 0.0199 |
| Pride | 0.0161 | 0.1638 | 0.0483 | 0.5575 | 0.4317 | 0.0274 |
| *MedMCQ Dataset* | | | | | | |
| Baseline | 0.0167 | 0.3182 | 0.5597 | 0.8128 | 0.3696 | 0.0161 |
| LoRa FineTuning | 0.0000 | 0.0052 | 0.0531 | 0.2116 | 0.3213 | 0.0056 |
| Gray-Box Weighting | 0.0059 | 0.0221 | 0.8249 | – | 0.4014 | 0.0044 |
| BNP | 0.0109 | 0.3098 | 0.5049 | 0.7547 | 0.3851 | 0.0154 |
| Pride | 0.0095 | 0.0165 | 0.0112 | 0.8217 | 0.3973 | 0.0077 |
| *QASC Dataset* | | | | | | |
| Baseline | 0.0046 | 0.2021 | 0.4818 | 0.6544 | 0.4892 | 0.0138 |
| LoRa FineTuning | 0.0026 | 0.0130 | 0.0014 | 0.1263 | 0.9514 | 0.0037 |
| Gray-Box Weighting | 0.0061 | 0.2500 | 0.3122 | – | 0.6210 | 0.0138 |
| BNP | 0.0037 | 0.1357 | 0.1357 | 0.3359 | 0.7441 | 0.0107 |
| Pride | 0.0050 | 0.0545 | 0.0017 | 0.4841 | 0.8715 | 0.0096 |

## B.2 Transferability Results over all datasets and models

Table 7: Transferability of Model-Checkpoints Across Datasets (Complete Results)

| Source/Target | Bias | RStd | CKLD | Fluct. Rate | Accuracy | Perm Acc Std |
|---|---|---|---|---|---|---|
| **QWEN/MEDCQ → ARC Dataset** | | | | | | |
| Baseline | 0.0268 | 0.0158 | 0.0003 | 0.1997 | 0.8038 | 0.0088 |
| LoRa FineTune | 0.0101 | 0.0004 | 0.1869 | 0.1869 | 0.8080 | 0.0073 |
| % Change | -43.6% | -36.1% | +333.3% | -6.4% | +0.5% | -17.0% |
| LoRa Transfer | 0.0215 | 0.0116 | 0.0001 | 0.2438 | 0.8034 | 0.0075 |
| **QWEN/MEDCQ → TeleQNA Dataset** | | | | | | |
| Baseline | 0.0214 | 0.0564 | 0.0070 | 0.5874 | 0.5464 | 0.0189 |
| LoRa FineTune | 0.0182 | 0.0879 | 0.0223 | 0.6093 | 0.5437 | 0.0141 |
| % Change | -15.0% | +55.9% | +218.6% | +3.7% | -0.5% | -25.4% |
| LoRa Transfer | 0.0254 | 0.0671 | 0.0096 | 0.2076 | 0.5574 | 0.0116 |
| **QWEN/MEDCQ → QASC Dataset** | | | | | | |
| Baseline | 0.0110 | 0.0547 | 0.0117 | 0.2062 | 0.8952 | 0.0068 |
| LoRa FineTune | 0.0031 | 0.0101 | 0.0003 | 0.1188 | 0.9698 | 0.0040 |
| % Change | -72.2% | -81.5% | -97.4% | -42.4% | +8.3% | -41.2% |
| LoRa Transfer | 0.0097 | 0.0720 | 0.0291 | 0.1847 | 0.8585 | 0.0071 |

Table 8: Transferability of Model-Checkpoints (Continued)

| Source/Target | Bias | RStd | CKLD | Fluct. Rate | Accuracy | Perm Acc Std |
|---|---|---|---|---|---|---|
| **QWEN/TeleQNA → ARC Dataset** | | | | | | |
| Baseline | 0.0268 | 0.0158 | 0.0003 | 0.1997 | 0.8038 | 0.0088 |
| LoRa FineTune | 0.0151 | 0.0101 | 0.0004 | 0.1869 | 0.8080 | 0.0073 |
| LoRa Transfer | 0.0247 | 0.0163 | 0.0006 | 0.2635 | 0.8068 | 0.0069 |
| **QWEN/TeleQNA → MedMCQ Dataset** | | | | | | |
| Baseline | 0.0577 | 0.0179 | 0.0006 | 0.8964 | 0.4805 | 0.0057 |
| LoRa FineTune | 0.0363 | 0.0771 | 0.0082 | 0.9031 | 0.4798 | 0.0055 |
| LoRa Transfer | 0.0484 | 0.0279 | 0.0024 | 0.2639 | 0.4803 | 0.0049 |
| **QWEN/TeleQNA → QASC Dataset** | | | | | | |
| Baseline | 0.0110 | 0.0547 | 0.0117 | 0.2062 | 0.8952 | 0.0068 |
| LoRa FineTune | 0.0031 | 0.0101 | 0.0003 | 0.1188 | 0.9698 | 0.0040 |
| LoRa Transfer | 0.0167 | 0.0913 | 0.0516 | 0.2408 | 0.8337 | 0.0097 |

Table 9: Transferability of Model-Checkpoints (Continued)

| Source/Target | Bias | RStd | CKLD | Fluct. Rate | Accuracy | Perm Acc Std |
|---|---|---|---|---|---|---|
| **QWEN/ARC → TeleQNA Dataset** | | | | | | |
| Baseline | 0.0214 | 0.0564 | 0.0070 | 0.5874 | 0.5464 | 0.0189 |
| LoRa FineTune | 0.0182 | 0.0879 | 0.0223 | 0.6093 | 0.5437 | 0.0141 |
| LoRa Transfer | 0.0025 | 0.1363 | 0.0805 | 0.1694 | 0.5464 | 0.0139 |
| **QWEN/ARC → MedMCQ Dataset** | | | | | | |
| Baseline | 0.0577 | 0.0179 | 0.0006 | 0.8964 | 0.4805 | 0.0057 |
| LoRa FineTune | 0.0363 | 0.0771 | 0.0082 | 0.9031 | 0.4798 | 0.0055 |
| LoRa Transfer | 0.0041 | 0.1532 | 0.1005 | 0.4023 | 0.4604 | 0.0079 |
| **QWEN/ARC → QASC Dataset** | | | | | | |
| Baseline | 0.0110 | 0.0547 | 0.0117 | 0.2062 | 0.8952 | 0.0068 |
| LoRa FineTune | 0.0031 | 0.0101 | 0.0003 | 0.1188 | 0.9698 | 0.0040 |
| LoRa Transfer | 0.0214 | 0.2950 | 1.3080 | 0.8293 | 0.3013 | 0.0134 |

Table 10: Transferability of Model-Checkpoints (Continued)

| Source/Target | Bias | RStd | CKLD | Fluct. Rate | Accuracy | Perm Acc Std |
|---|---|---|---|---|---|---|
| **QWEN/QASC → ARC Dataset** | | | | | | |
| Baseline | 0.0268 | 0.0158 | 0.0003 | 0.1997 | 0.8038 | 0.0088 |
| LoRa FineTune | 0.0151 | 0.0101 | 0.0004 | 0.1869 | 0.8080 | 0.0073 |
| LoRa Transfer | 0.0279 | 0.0183 | 0.0011 | 0.2481 | 0.7923 | 0.0074 |
| **QWEN/QASC → TeleQNA Dataset** | | | | | | |
| Baseline | 0.0214 | 0.0564 | 0.0070 | 0.5874 | 0.5464 | 0.0189 |
| LoRa FineTune | 0.0182 | 0.0879 | 0.0223 | 0.6093 | 0.5437 | 0.0141 |
| LoRa Transfer | 0.0408 | 0.0620 | 0.0223 | 0.2158 | 0.5601 | 0.0140 |
| **QWEN/QASC → MedMCQ Dataset** | | | | | | |
| Baseline | 0.0577 | 0.0179 | 0.0006 | 0.8964 | 0.4805 | 0.0057 |
| LoRa FineTune | 0.0363 | 0.0771 | 0.0082 | 0.9031 | 0.4798 | 0.0055 |
| LoRa Transfer | 0.0573 | 0.1089 | 0.0067 | 0.2175 | 0.4735 | 0.0044 |

Table 11: Transferability of Phi-2 Model-Checkpoints

| Source/Target | Bias | RStd | CKLD | Fluct. Rate | Accuracy | Perm Acc Std |
|---|---|---|---|---|---|---|
| **Phi-2/TeleQnA → ARC Dataset** | | | | | | |
| Baseline | 0.0269 | 0.2207 | 0.2853 | 0.6334 | 0.5520 | 0.0110 |
| LoRa FineTune | 0.0148 | 0.0257 | 0.0021 | 0.2858 | 0.7553 | 0.0076 |
| LoRa Transfer | 0.0217 | 0.0545 | 0.0067 | 0.2576 | 0.6500 | 0.0080 |
| **Phi-2/TeleQnA → MedMCQ Dataset** | | | | | | |
| Baseline | 0.0512 | 0.3981 | 1.4585 | 0.9369 | 0.3409 | 0.0194 |
| LoRa FineTune | 0.0013 | 0.0893 | 0.0527 | 0.3358 | 0.3569 | 0.0087 |
| LoRa Transfer | 0.0256 | 0.0709 | 0.0478 | 0.3799 | 0.3344 | 0.0051 |
| **Phi-2/TeleQnA → QASC Dataset** | | | | | | |
| Baseline | 0.0314 | 0.3232 | 2.9914 | 0.9676 | 0.1577 | 0.0122 |
| LoRa FineTune | 0.0002 | 0.2373 | 1.0080 | 0.1003 | 0.3002 | 0.0112 |
| LoRa Transfer | 0.0073 | 0.0605 | 0.0067 | 0.0950 | 0.8542 | 0.0008 |

Table 12: Transferability of Phi-2 Model-QASC

| Source/Target | Bias | RStd | CKLD | Fluct. Rate | Accuracy | Perm Acc Std |
|---|---|---|---|---|---|---|
| **Phi-2/Qasc → ARC Dataset** | | | | | | |
| Baseline | 0.0269 | 0.2207 | 0.2853 | 0.6334 | 0.5520 | 0.0110 |
| LoRa FineTune | 0.0148 | 0.0257 | 0.0021 | 0.2858 | 0.7553 | 0.0076 |
| LoRa Transfer | 0.0129 | 0.2855 | 0.5250 | 0.7511 | 0.4489 | 0.0091 |
| **Phi-2/Qasc → TeleQNA Dataset** | | | | | | |
| Baseline | 0.0303 | 0.3559 | 1.5208 | 0.8114 | 0.2568 | 0.0216 |
| LoRa FineTune | 0.0260 | 0.0271 | 0.0087 | 0.0267 | 0.3530 | 0.0099 |
| LoRa Transfer | 0.0494 | 0.4000 | 2.9890 | 1.0000 | 0.2100 | 0.0233 |
| **Phi-2/Qasc → MedMCQ Dataset** | | | | | | |
| Baseline | 0.0512 | 0.3981 | 1.4585 | 0.9369 | 0.3409 | 0.0194 |
| LoRa FineTune | 0.0013 | 0.0893 | 0.0527 | 0.3358 | 0.3569 | 0.0087 |
| LoRa Transfer | 0.0300 | 0.4259 | 2.4919 | 0.9931 | 0.3201 | 0.0199 |

Table 13: Transferability of Phi-2 Model-MedMCQA)

| Source/Target | Bias | RStd | CKLD | Fluct. Rate | Accuracy | Perm Acc Std |
|---|---|---|---|---|---|---|
| **Phi-2/MedMCQ → ARC Dataset** | | | | | | |
| Baseline | 0.0269 | 0.2207 | 0.2853 | 0.6334 | 0.5520 | 0.0110 |
| LoRa FineTune | 0.0148 | 0.0257 | 0.0021 | 0.2858 | 0.7553 | 0.0076 |
| LoRa Transfer | 0.0149 | 0.0257 | 0.0021 | 0.2858 | 0.7553 | 0.0076 |
| **Phi-2/MedMCQ → TeleQNA Dataset** | | | | | | |
| Baseline | 0.0303 | 0.3559 | 1.5208 | 0.8114 | 0.2568 | 0.0216 |
| LoRa FineTune | 0.0260 | 0.0271 | 0.0087 | 0.0267 | 0.3530 | 0.0099 |
| LoRa Transfer | 0.0260 | 0.0960 | 0.0115 | 0.1939 | 0.4125 | 0.0170 |
| **Phi-2/MedMCQ → QASC Dataset** | | | | | | |
| Baseline | 0.0314 | 0.3232 | 2.9914 | 0.9676 | 0.1577 | 0.0122 |
| LoRa FineTune | 0.0002 | 0.2373 | 1.0080 | 0.1003 | 0.3002 | 0.0112 |
| LoRa Transfer | 0.0004 | 0.0291 | 0.0017 | 0.1134 | 0.9265 | 0.0051 |

Table 14: Transferability of Phi-2/ARC Model-ARC

| Source/Target | Bias | RStd | CKLD | Fluct. Rate | Accuracy | Perm Acc Std |
|---|---|---|---|---|---|---|
| **Phi-2/ARC → ARC Dataset** | | | | | | |
| Baseline | 0.0269 | 0.2207 | 0.2853 | 0.6334 | 0.5520 | 0.0110 |
| LoRa FineTune | 0.0148 | 0.0257 | 0.0021 | 0.2858 | 0.7553 | 0.0076 |
| LoRa Transfer | 0.0217 | 0.0545 | 0.0067 | 0.2575 | 0.6498 | 0.0076 |
| **Phi-2/ARC → TeleQNA Dataset** | | | | | | |
| Baseline | 0.0303 | 0.3559 | 1.5208 | 0.8114 | 0.2568 | 0.0216 |
| LoRa FineTune | 0.0260 | 0.0271 | 0.0087 | 0.0267 | 0.3530 | 0.0099 |
| LoRa Transfer | 0.0261 | 0.0964 | 0.0115 | 0.1939 | 0.4126 | 0.0099 |
| **Phi-2/ARC → MedMCQ Dataset** | | | | | | |
| Baseline | 0.0512 | 0.3981 | 1.4585 | 0.9369 | 0.3409 | 0.0194 |
| LoRa FineTune | 0.0013 | 0.0893 | 0.0527 | 0.3358 | 0.3569 | 0.0087 |
| LoRa Transfer | 0.0269 | 0.0893 | 0.0214 | 0.2737 | 0.3749 | 0.0045 |
| **Phi-2/ARC → QASC Dataset** | | | | | | |
| Baseline | 0.0314 | 0.3232 | 2.9914 | 0.9676 | 0.1577 | 0.0122 |
| LoRa FineTune | 0.0002 | 0.2373 | 1.0080 | 0.1003 | 0.3002 | 0.0112 |
| LoRa Transfer | 0.0042 | 0.0291 | 0.0017 | 0.1133 | 0.9265 | 0.0059 |

Table 15: Transferability of Llama 3.2 Model-ARC

| Source/Target | Bias | RStd | CKLD | Fluct. Rate | Accuracy | Perm Acc Std |
|---|---|---|---|---|---|---|
| **Llama 3.2/ARC → TeleQNA Dataset** | | | | | | |
| Baseline | 0.0053 | 0.1010 | 0.0907 | 0.5574 | 0.4536 | 0.0165 |
| LoRa FineTune | 0.0002 | 0.1998 | 0.3120 | 0.3306 | 0.3830 | 0.0245 |
| LoRa Transfer | 0.0007 | 0.2344 | 0.5903 | 0.6393 | 0.4235 | 0.0209 |
| **Llama 3.2/ARC → MedMCQ Dataset** | | | | | | |
| Baseline | 0.0167 | 0.3182 | 0.5597 | 0.8128 | 0.3696 | 0.0161 |
| LoRa FineTune | 0.0000 | 0.0052 | 0.0531 | 0.2116 | 0.3213 | 0.0056 |
| LoRa Transfer | 0.0012 | 0.3943 | 1.2546 | 0.9132 | 0.3541 | 0.0189 |
| **Llama 3.2/ARC → QASC Dataset** | | | | | | |
| Baseline | 0.0046 | 0.2021 | 0.4818 | 0.6544 | 0.4892 | 0.0138 |
| LoRa FineTune | 0.0026 | 0.0130 | 0.0014 | 0.1263 | 0.9514 | 0.0037 |
| LoRa Transfer | 0.0020 | 0.2963 | 1.7180 | 0.8920 | 0.2397 | 0.0119 |

Table 16: Transferability of Llama 3.2 Model-TeleQnA

| Source/Target | Bias | RStd | CKLD | Fluct. Rate | Accuracy | Perm Acc Std |
|---|---|---|---|---|---|---|
| **Llama 3.2/TeleQNA → ARC Dataset** | | | | | | |
| Baseline | 0.0091 | 0.1082 | 0.1050 | 0.5171 | 0.5179 | 0.0097 |
| LoRa FineTune | 0.0034 | 0.1780 | 0.1144 | 0.1373 | 0.5339 | 0.0092 |
| LoRa Transfer | 0.0010 | 0.1828 | 0.2423 | 0.5416 | 0.4343 | 0.0136 |
| **Llama 3.2/TeleQNA → MedMCQ Dataset** | | | | | | |
| Baseline | 0.0167 | 0.3182 | 0.5597 | 0.8128 | 0.3696 | 0.0161 |
| LoRa FineTune | 0.0000 | 0.0052 | 0.0531 | 0.2116 | 0.3213 | 0.0056 |
| LoRa Transfer | 0.0004 | 0.3840 | 1.6399 | 0.9013 | 0.3376 | 0.0056 |
| **Llama 3.2/TeleQNA → QASC Dataset** | | | | | | |
| Baseline | 0.0046 | 0.2021 | 0.4818 | 0.6544 | 0.4892 | 0.0138 |
| LoRa FineTune | 0.0026 | 0.0130 | 0.0014 | 0.1263 | 0.9514 | 0.0037 |
| LoRa Transfer | 0.0012 | 0.3290 | 3.5746 | 0.9946 | 0.1369 | 0.0124 |

Table 17: Transferability of Llama 3.2 Model-QASC

| Source/Target | Bias | RStd | CKLD | Fluct. Rate | Accuracy | Perm Acc Std |
|---|---|---|---|---|---|---|
| **Llama 3.2/QASC → ARC Dataset** | | | | | | |
| Baseline | 0.0091 | 0.1082 | 0.1050 | 0.5171 | 0.5179 | 0.0097 |
| LoRa FineTune | 0.0034 | 0.1780 | 0.1144 | 0.1373 | 0.5339 | 0.0092 |
| LoRa Transfer | 0.0003 | 0.1195 | 0.1184 | 0.4515 | 0.4944 | 0.0094 |
| **Llama 3.2/QASC → TeleQNA Dataset** | | | | | | |
| Baseline | 0.0053 | 0.1010 | 0.0907 | 0.5574 | 0.4536 | 0.0165 |
| LoRa FineTune | 0.0002 | 0.1998 | 0.3120 | 0.3306 | 0.3830 | 0.0245 |
| LoRa Transfer | 0.0007 | 0.2344 | 0.5903 | 0.6393 | 0.4235 | 0.0175 |
| **Llama 3.2/QASC → MedMCQ Dataset** | | | | | | |
| Baseline | 0.0167 | 0.3182 | 0.5597 | 0.8128 | 0.3696 | 0.0161 |
| LoRa FineTune | 0.0000 | 0.0052 | 0.0531 | 0.2116 | 0.3213 | 0.0056 |
| LoRa Transfer | 0.0012 | 0.3943 | 1.2545 | 0.9132 | 0.3541 | 0.0018 |

Table 18: Transferability of Llama 3.2 Model-MedMCQ

| Source/Target | Bias | RStd | CKLD | Fluct. Rate | Accuracy | Perm Acc Std |
|---|---|---|---|---|---|---|
| **Llama 3.2/MedMCQ → ARC Dataset** | | | | | | |
| Baseline | 0.0091 | 0.1082 | 0.1050 | 0.5171 | 0.5179 | 0.0097 |
| LoRa FineTune | 0.0034 | 0.1780 | 0.1144 | 0.1373 | 0.5339 | 0.0092 |
| LoRa Transfer | 0.0011 | 0.0914 | 0.0645 | 0.4129 | 0.4893 | 0.0133 |
| **Llama 3.2/MedMCQ → TeleQNA Dataset** | | | | | | |
| Baseline | 0.0053 | 0.1010 | 0.0907 | 0.5574 | 0.4536 | 0.0165 |
| LoRa FineTune | 0.0002 | 0.1998 | 0.3120 | 0.3306 | 0.3830 | 0.0245 |
| LoRa Transfer | 0.0003 | 0.1730 | 0.2920 | 0.4508 | 0.4153 | 0.0202 |
| **Llama 3.2/MedMCQ → QASC Dataset** | | | | | | |
| Baseline | 0.0046 | 0.2021 | 0.4818 | 0.6544 | 0.4892 | 0.0138 |
| LoRa FineTune | 0.0026 | 0.0130 | 0.0014 | 0.1263 | 0.9514 | 0.0037 |
| LoRa Transfer | 0.0009 | 0.2999 | 1.8713 | 0.9039 | 0.2441 | 0.0154 |

# C Prompt Templates and Hyperparameter Configurations

## C.1 Prompt Templates Used in All Experiments

This appendix provides the exact prompt templates used during evaluation of transferability experiments across datasets and models.

### MedMCQA Dataset (All Models)

```
Instruct = Youre a Medical Question Answering Expert, answer the following question.
Please generate only answer choice (1, 2, 3 or 4)
{question}
```

```
{options}
Output: option
```

**TeleQnA Dataset (Qwen & Llama)**

```
Instruct: Answer the following question using the context provided.
Your answer must start with the correct option letter (1, 2, 3, 4, or 5):
{question}
{options}
Output: option
```

**TeleQnA or ARC Dataset (Phi-2)**

```
Instruct: Answer the following question.
Your answer must start with the correct option letter (1, 2, 3, 4, or 5) followed by the
text of the answer:
{question}
{options}
Output: option
```

**ARC Dataset (All Models Except Phi)**

```
Instruct: {question}
{options}
Output: option
```

**QASC Dataset (Qwen, Phi-2)**

```
Instruct: Answer the following question using the context provided, reason over it.
Please generate only answer choice (1, 2, 3, 4, 5, 6, 7 or 8) without any explanations

{question}
context: {context}

{options}
{question}
Output: option
```

**QASC Dataset (Llama)**

```
Instruct: Answer the following question using the context provided, reason over it.
Please generate only answer choice (1, 2, 3, 4, 5, 6, 7 or 8) without any explanations

{question}
context: {context}

{options}
Output:
```

### C.2 Hyper-Parameter Configurations

**Unsupervised Lora Finetuning:** During each iteration we randomly sample 64 samples every epoch from the train split of the dataset and use it for training. For the Lora adapters, we target the attention QKV and Output projection weights, with a dropout of 0.05 and Lora alpha of 16. We used an AdamW optimizer with learning rate of $1e^{-4}$ and weight decay of $0.001$ for all experiments. In computing the loss, we use an $\alpha$ of 0.1 to weight the entropy.

### C.3 Model Size And Budget

We evaluated our methods on a set of compact yet capable LLMs, including Qwen2.5–3B, Phi-2, and LLaMA3.2–3B, each with approximately 3 billion parameters. All experiments were conducted on NVIDIA L40 GPUs (48 GB VRAM). Using AWS g5.12xlarge instances (approximate L40 equivalent) priced at $3.06/hour on-demand, the cost per fine-tuning run is $6–$9, and BaQCKV inference costs are $1–$2 per dataset. This keeps the total cost for running all experiments within a practical research budget, demonstrating that our methods are efficient and deployable even on mid-sized mod **Bias Metric Computation**: To compute the Our Bias metric, we do not use the raw probabilities over the vocab size. We rather use the logits for only the option IDs to compute the probabilities. Also, we limit the number of permutations to a maximum of 24 ($4!$). This is due to our GPU memory memory budget and our considerations that most MCQs would have around 4 options.