Embracing Positional Bias in Multiple-Choice Question Answering via Permutation Equivariant Neural Networks

Chengyu Jiao*, Siyin Huang*, Yu Zhang†

Southern University of Science and Technology

Abstract

Several studies have demonstrated that large language models (LLMs) exhibit positional bias when answering multiple-choice questions (MCQs). Previous methods have identified such bias to be detrimental, leading to the development of techniques to mitigate it. However, we observe that certain permutations of options can actually improve the performance. Therefore, instead of eliminating such bias, we propose an EMbracing the Bias EquivaRiantly (EMBER) network. Specifically, the EMBER network, which outputs a permutation of options in MCQs, is optimized towards the beneficial permutations to which the LLM is biased. Additionally, to solve the positional bias among different permutations of options, the EMBER network is designed to grant the equivariance to the permutation to the LLMs. Theoretically and empirically, we show that the proposed EMBER network can effectively utilize the positional bias and demonstrate state-of-the-art performance over various baselines.

1 Introduction

Large language models (LLMs) (Brown et al. 2020; Touvron et al. 2023a,b) have shown a significant breakthrough in Natural Language Processing (NLP), and have achieved exceptional performance across various language tasks. Though LLMs have been demonstrated remarkable potential on a wide range of applications, recent researches (Zong et al. 2024; Zheng et al. 2024; Pezeshkpour and Hruschka 2024; Li et al. 2024b) have shown the presence of implicit positional biases when they are asked to select the correct option from several ones in multiple-choice questions (MCQs) (Hendrycks et al. 2021; Li et al. 2024a; Yüksel et al. 2024). That is, the probability of selecting a particular option can vary based on its position, as illustrated in Figure 1a.[1]

The positional bias issue of LLMs in MCQs has attracted considerable attention as it may degrade the performance and reliability of LLMs. To address this issue, calibration-based methods (Zheng et al. 2024; Li et al. 2024b; Li and Gao 2024) adjust the output distribution of LLMs during inference, ensuring consistent predictions across different orders of options. In contrast, structural adjustment strategies (Zhou et al. 2024; Xue et al. 2024; Wiegreffe et al. 2024) investigate the internal mechanism and mitigate the bias by suppressing bias components in model architecture.

Although previous approaches focus on mitigating this bias, we think that positional bias is not inherently detrimental. To see that, we adopt the *any wrong* setting from permutation attacks (Zong et al. 2024), where a question is considered to be incorrectly answered if at least one incorrect prediction occurs across all the permutations of options. Moreover, we propose a new setting, *any right*, to evaluate the upper bound of the LLM performance, where a question is considered to be correctly answered if the LLM provides a correct response in at least one permutation of its options. As illustrated in Figure 1b, we observe a substantial performance gap between the original setting and *any wrong* setting, indicating that "bad" permutations can significantly degrade the performance. Conversely, LLMs under the *any right* setting outperforms those under the original setting, suggesting that "good" permutations can unlock a greater potential in LLMs, which remains largely unexplored.

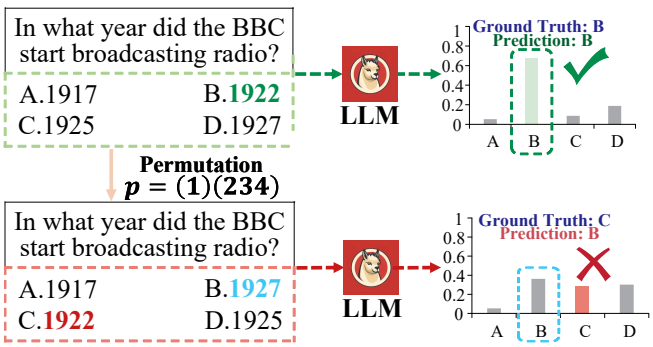
Building upon the observations above, we consider the positional bias as a double-edged sword. "Good" permutations can help LLM reach better performance, while "bad" permutations can drastically degrade the performance. Thus, our objective is to **embrace** such positional bias to learn "good" permutations, while incorporating the **equivariance** to the permutation of options (Gens and Domingos 2014; Bronstein et al. 2017) to avoid the performance degradation caused by "bad" permutations.

To achieve that, in this paper, we propose a novel network, EMBER, to EMbrace the Bias Equivariantly. Specifically, by taking the embeddings of the question and options as the input, the EMBER network adopts a novel feature expansion block that transforms features for all the n options into $n!$ permutation features, representing all possible permutations of options. Then after going through multiple attention blocks and pooling operations, the EMBER network identifies the favorite permutation corresponding to the highest feature values as the output. To alleviate the collapse of the permutation features, we introduce a diversity loss to train the EMBER network.

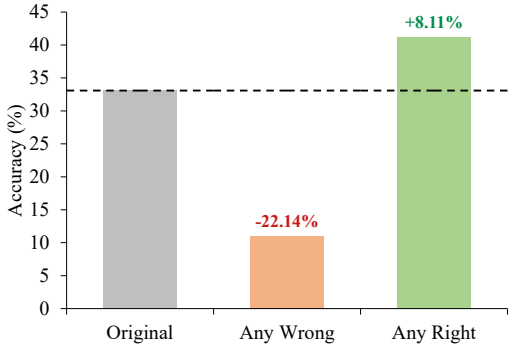
*These authors contributed equally.

†Corresponding author.

[1] In this study, unless otherwise noted, the term "option" refers solely to the content of the option, excluding its ID. Our analysis focuses on permuting only the option content, while the option IDs (e.g., A, B, etc.) remain unchanged.



(a) An example of positional bias.



(b) LLama-2-7B performance on MMLU under different settings.

Figure 1: (Left) When the permutation $p = (1)(234)$ is applied to the original four options, the output probabilities of Llama 3.1 for the permuted questions differ substantially from those of the original. (Right) Llama-2-7B accuracies in the original setting and the *any wrong/right* settings.

Based on the design, the EMBER network satisfies two crucial properties. Firstly, applying the permutation generated by the EMBER network to options results in an invariant order of options, reflecting a fact that the learned order of options is unique for a given question and a set of options, regardless of the initial order of options. Secondly, the combination of EMBER and LLM could output equivariant probabilities, which is essential for addressing the positional bias issue.

Our main contributions are summarized as follows.

1. To the best of our knowledge, this work is the first to analyze the positional bias problem from the perspective of the permutation equivariance.
2. The proposed EMBER network provides the LLM with equivariance to the permutation of options while utilizing "good" permutations to enhance the performance of the LLM and the robustness to the permutation attack.
3. Theoretically, we prove that the permutation equivariance is an effective solution to the positional bias issue.
4. Empirically, EMBER shows notable improvements on MCQ answering task with minimal increase in parameters. Additionally, the proposed EMBER method are shown to be robust to the permutation attack.

2 Related Works

2.1 Equivariance in Deep Learning

Equivariance (Gens and Domingos 2014; Bronstein et al. 2017) in machine learning refers to the ability of a model to incorporate and preserve certain symmetries. Equivariant neural networks are initially prove to be successful in image processing, which naturally involves translations, rotations, and reflections. In particular, Group Convolutional Neural Networks (G-CNNs) (Cohen and Welling 2016) along with subsequent extensions (Cohen and Welling 2017; Cohen et al. 2018; Weiler and Cesa 2019; Gerken et al. 2022), achieve equivariance to these symmetries and demonstrate significant improvements over traditional convolutional neural networks (CNNs). Beyond rotation and reflection, permutation equivariance also plays a significant role, in tasks like weight space learning (Unterthiner et al. 2020; Zhou et al. 2023) and code processing (Pei et al. 2024), showing enhanced performance compared to non-equivariant models.

In principle, incorporating equivariance into pretrained foundation models is considered beneficial. However, due to the computational costs of retraining from scratch, directly modifying the structure of foundation models is not a realistic approach. Currently, two main approaches exist, both focusing on rotation and reflection equivariance for image processing. The first approach involves equi-tuning (Basu et al. 2023), which fine-tunes pretrained models to accommodate group equivariance without extensive retraining. Alternatively, learning canonicalization functions (Kaba et al. 2023; Mondal et al. 2023; Shumaylov et al. 2024) provides a decoupled framework in which image inputs are rotated to the model's preferred angles, referred as the canonical positions, before feeding them into neural networks.

In the context of natural language processing (NLP), few studies focus on equivariance because identifying symmetries in linguistic structures can be challenging. As a concrete problem, multiple-choice questions (MCQs) exhibit a straightforward symmetry: the options can be rearranged in any order without changing the meaning of the whole MCQ. We anticipate that LLMs should be able to determine the correct answer regardless of the order in which the options are presented. However, as shown in Figure 1b, current LLMs struggle with this due to the inherent positional bias.

2.2 Mitigating Positional Bias

Multiple-choice questions (MCQs) have extensively been used as benchmarks to evaluate the capabilities of large language models (LLMs) across various domains, with notable datasets such as the Massive Multitask Language Understanding (MMLU) (Hendrycks et al. 2021) and AI2 Reasoning Challenge (ARC) (Clark et al. 2018) . However, recent research (Zong et al. 2024; Zheng et al. 2024; Pezeshkpour and Hruschka 2024) indicates that LLMs exhibit biases related to the order of answer choices, leading to substantial performance degradation when these options are rearranged.

To address the concern, recent studies (Zheng et al. 2024;

Li et al. 2024b; Li and Gao 2024) have introduced diverse debiasing techniques. As one of the most effective method, PriDe (Zheng et al. 2024) employs permutation-based methods to estimate and remove prior biases, offering an efficient inference-time debiasing strategy. Concurrently, recent investigations (Zhou et al. 2024; Xue et al. 2024; Wiegreffe et al. 2024) have examined the internal mechanisms within LLMs that contribute to positional biases. One such approach, UniBias (Zhou et al. 2024), identifies and suppresses biased components in feedforward neural networks and attention heads, thereby reducing the model's sensitivity to input perturbations. Different from existing works, we aim to incorporate the permutation equivariance into the LLM via the proposed EMBER network.

3 Preliminary

3.1 Permutation Groups

A general *group* G can be thought as a collection of certain symmetries. In this paper, we focus on the permutation group of n elements, denoted by S_n.To define S_n, we first introduce the concept of cycle notation. A cycle c is represented as $c = (a_1\ a_2\ \dots\ a_k)$, which maps each element to the next element in a circular fashion. Specifically, for i such that $1 \leq i \leq k-1$, we have $c(a_i) = a_{i+1}$, and for the last element, $c(a_k) = a_1$.

The permutation group S_n consists of $n!$ elements. Each permutation $p \in S_n$ represents a unique way of permuting n elements, and can be uniquely represented in disjoint cycle notations. Given a integer i, $p(i)$ is applying the circular notation containing i on i. The identity element is $e = (1)(2)\dots(n)$ is the identity element, mapping every i back to itself. The multiplication $\cdot$ of permutation can be understood of function composition. We often omit $\cdot$ and directly write p_1p_2 for $p_1 \cdot p_2$. The inverse of p is denoted by p^{-1}, represent the unique element satisfying $p \cdot p^{-1} = p^{-1} \cdot p = e$. Every element of S_n has an inverse.

In this paper, given a MCQ composed of a question Q and n possible options $O = \{o_i\}_{i=1}^n$, S_n acts on O by permuting index of n options. That is, for any $p \in S_n$, $pO = \{o_{p(i)}\}_{i=1}^n$.

To help the understanding, we show two examples of the permutation acting on O. The permutation $(13)(24)$ performs the swapping of option 1 with option 3, and option 2 with option 4. The second example is $p = (1)(234)$, which keeps option 1 unchanged while moving option 2 to the third position, option 3 to the fourth position, and option 4 to the second position.

3.2 Positional Bias in MCQs

Based on previous studies (Mondal et al. 2023), current vanilla LLMs suffer from the positional bias, where they have inconsistent performance when the permutation is applied directly on the options. In Figure 1a, Llama 3.1-8b is able to correctly answer the original question, but fails to recognize the correct answer after a simple permutation. This is a general case (Zong et al. 2024), where most LLMs suffer performance decrease under the any wrong setting. This positional bias can be formally defined as follows:

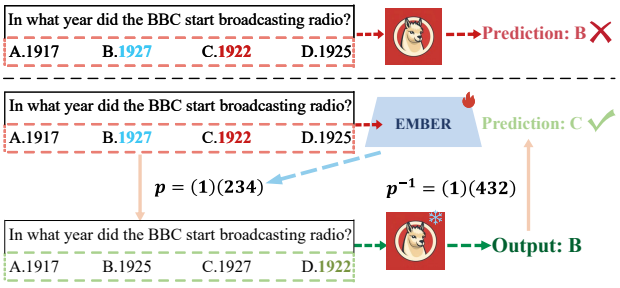


Figure 2: An overview of the proposed EMBER framework as shown in the bottom figure. The top figure shows how the LLM handles MCQs.

Definition 1. A neural network f for answering MCQs has positional bias if its outputs are inconsistent when a permutation acts on the options O. That is, for any $p \in S_n$, we have

$$f(Q, pO) \neq pf(Q, O). \tag{1}$$

3.3 Permutation Equivariance and Invariance

Given the previously defined permutation on options, we can also establish the definition of permutation equivariance. A neural network f is **permutation equivariant** if it commutes with the permutation action. The formal definition of the permutation equivariance is as follows.

Definition 2. Let f be a neural network that takes a question Q and n possible options O as inputs. Then, f is *permutation equivariant* if for any $p \in S_n$, we have

$$f(Q, pO) = pf(Q, O). \tag{2}$$

For a special case that $f(Q, pO) = f(Q, O)$, we say that f is invariant.

4 Methodology

It is trivial to see that the definition of equivariance in Eq. (2) would exactly solve the positional bias issue shown in Eq. (1). Therefore, it is critical to incorporate permutation equivariance into LLMs. The following theorem shows that current representative debiasing methods fail to do so.[2]

Theorem 3. *Assume f is a non-equivariant MCQ answering model. Let ρ be the calculated bias prior in Calibration (Zhao et al. 2021) or PriDe (Zheng et al. 2024). Then, affine transformations in Calibration or dividing it in PriDe cannot make f permutation equivariant.*

Moreover, as discussed in Sec. 2.1, it is unrealistic to modify the architecture of existing LLMs to be permutation equivariant and then re-train them from scratch, since the cost incurred is too high. In this section, we demonstrate how our proposed EMBER network embraces the positional bias equivariantly.

[2]Proofs of all the theorems are put in Appendix.

4.1 The Framework

Given an LLM M taking a question Q and n different options $O = \{o_i\}_{i=1}^n$ as the inputs, O can be permuted in $n!$ different ways, each of which corresponds to a permutation $p \in S_n$. Since there are only finitely many such permutations, there has to be one permutation p^* such that the output $M(Q, p^*O)$ has the minimum loss with the ground truth. We refer to this permutation p^* as the "optimal" permutation, and the resulted p^*O can be viewed as the model's "favorite" order for this question, containing the implicit information related to model's positional bias.

To embrace such positional bias in LLM, the proposed EMBER network E takes (Q, O) as the input, and outputs a permutation $p \in S_n$. This design can be interpreted as leveraging information from both the question and the options to learn the optimal permutation p^* for the options. We denote the output permutation by $p_O = E(Q, O)$, and details of E is shown in Sec. 4.3.

After that, we permute the options according to the generated permutation, resulting in a permuted input $(Q, p_O O)$. This permuted input, along with the rest of prompt template and option IDs (e.g., A, B, etc.) is then fed into the LLM M. We collect the output corresponding to all option IDs, denoted by $M(Q, p_O O)$. Since the results corresponds to the permuted $p_O O$, we naturally permute back by using the inverse of p_O to obtain the final probability for the original options O. The final probability can be formulated as

$$\text{Prob} = p_O^{-1} M(Q, p_O O), \text{ where } p_O = E(Q, O). \quad (3)$$

where p_O^{-1} denotes the inverse of p_O. As abbreviation, we define $M \circ E(Q, O)$ as

$$M \circ E(Q, O) = E(Q, O)^{-1} M(Q, E(Q, O)O), \quad (4)$$

allowing us to rewrite Eq. (3) as

$$\text{Prob} = M \circ E(Q, O). \quad (5)$$

To satisfy the permutation equivariance, E should satisfy some properties, which are summarized in the following proposition.

Proposition 4. *To achieve the permutation equivariance, for any $p \in S_n$, E needs to satisfy the following two properties:*

1. *$M \circ E$ needs to be permutation equivariant. That is, $M \circ E(Q, pO) = p(M \circ E(Q, O))$.*
2. *Among all the possible permutations S_n on O, the favorite permutation of LLM should be the same. Therefore, we expect $E(Q, O)O$ to be invariant to permutations. That is, $E(Q, O)O = E(Q, pO)pO$.*

4.2 Difference Function and Permutation Features

Before introducing the EMBER network, we explains two important concepts in this section: the difference function and permutation features.

We denote features of n options by $\{A_i\}_{i=1}^n$. A **difference function**, denoted by D, inputs two different A_i, A_j and outputs a feature in the same shape. As an abbreviation, we write $D(i, j) = D(A_i, A_j)$ and refer to it as the difference between i and j. A simple example of a difference function is simply $D(i, j) = A_i - A_j$. If $D(a, b) > D(l, k)$ for all $l \neq k \leq n$, we say a has the largest difference with b.

For clarity, all option features A_i are represented as a scalar. An **elementary approach** to permute n options is to **simply reorder options based on the difference between options features**. If a has the largest difference with b, we place option a at the first place and option b last. If there are two or more options left, we repeat the reordering process by finding $D(c, d)$ as the largest difference among the remaining A_i, and putting option c to the furthermost left, option d to the furthermost right on the remaining positions. If only one option is left, it is placed in the middle. We terminate the process if no options are left.

Instead of comparing difference for $\lfloor n/2 \rfloor$ times[3], the following "decayed sum" enables us to compute all differences in a single operation called the decayed sum.

Definition 5. Let $\{a_i\}_{i=1}^{\lfloor n/2 \rfloor}$ and $\{b_i\}_{i=1}^{\lfloor n/2 \rfloor}$ be two sets of option index with an empty intersection between them. The **decayed sum** of $\{a_i\}_{i=1}^{\lfloor n/2 \rfloor}$ and $\{b_i\}_{i=1}^{\lfloor n/2 \rfloor}$ is expressed as

$$\sum_{i=1}^{\lfloor n/2 \rfloor} \lambda^i D(a_i, b_i) \quad (6)$$

for any fixed decay factor $0 < \lambda < 1$.

Proposition 6. *a_1 has the largest difference with b_1, a_2 has the second largest difference with b_2, and so on, if and only if $\{a_i\}_{i=1}^{\lfloor n/2 \rfloor}$ and $\{b_i\}_{i=1}^{\lfloor n/2 \rfloor}$ have the largest decayed sum compared to that of any possible index sets $\{c_i\}_{i=1}^{\lfloor n/2 \rfloor}, \{d_i\}_{i=1}^{\lfloor n/2 \rfloor}$ that have no intersection.*

When Prop. 6 happens, based on the elementary approach of reordering options based on the difference, we put a_i to the i-th position, and b_i to the $n + 1 - i$-th position. This corresponds to the permutation p where $p(a_i) = i$ and $p(b_i) = n + 1 - i$. Equivalently, $a_i = p^{-1}(i)$ and $b_i = p^{-1}(n + 1 - i)$. Therefore, the expression of the decayed sum in the Eq. (6) can be rewritten as

$$B_p = \sum_{1 \leq i \leq \frac{n}{2}} \lambda^i D\left(p^{-1}(i), p^{-1}(n + 1 - i)\right). \quad (7)$$

By Prop. 6, the elementary approach can be viewed as exhausting through all $\{B_p\}_{p \in S}$ to find the largest B_q. The corresponding permutation q is exactly the one we desire.

As we are choosing the corresponding p based on values in B_p, we name such B_p as the **permutation feature**. Since we only use all option features $\{A_i\}_{i=1}^n$ and the fixed difference function, $\{B_p\}_{p \in S_n}$ can be viewed an **feature expansion** from n option features to $n!$ permutation features. Also, Eq. (7) can be easily extended to higher dimensional inputs.

As an elementary approach, directly searching largest B_p often yields unsatisfactory results. However, such expansion from n option features $\{A_i\}_{i=1}^n$ to $n!$ permutation features $\{B_p\}_{p \in S_n}$ will play a crucial role in our EMBER network, as demonstrated in the following section.

[3]The notation $\lfloor \cdot \rfloor$ stands for the floor operation.

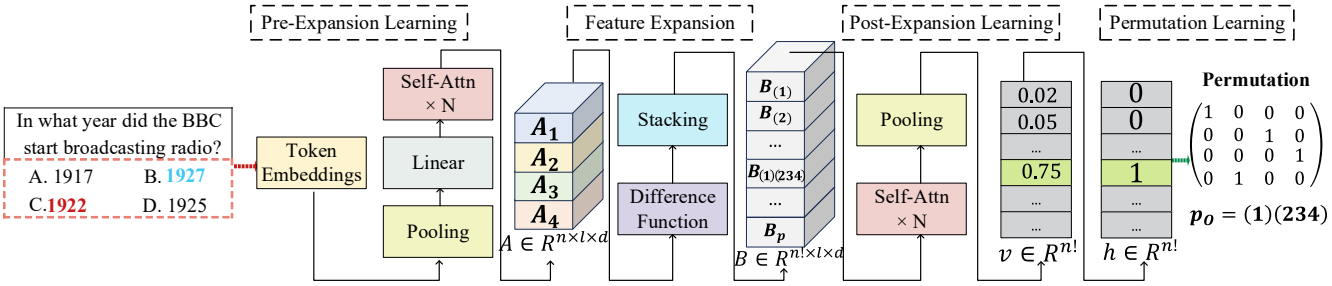


Figure 3: An overview of the EMBER network. $\{A_i\}_{i=1}^n$ and $\{B_p\}_{p\in S_n}$ are the option features and permutation features, respectively, in Sec. 4.3. The exact expression of B_p is shown in Eq. (7).

4.3 The EMBER Network

The EMBER network is divided into four main components, including pre-expansion learning, feature expansion, post-expansion learning, and permutation matching.

First, we embed the input option $O = \{o_i\}_{i=1}^n \in \mathbb{R}^{n\times l}$ using the frozen embedding layer of LLM in the pre-expansion learning module, where l denotes the length of tokens in each option after padding. Note that there is no positional encoding in the EMBER networks, which causes the LLM to lose equivariance (Vaswani et al. 2017). Next, we pass the resulting embeddings through a pre-expansion layer consisting of a pooling layer, a linear layer, and two self-attention blocks. The outputs of this pre-expansion process, $A = \{A_i\}_{i=1}^n \in \mathbb{R}^{n\times l\times d}$, represent the learned features of the options, where d is the attention dimension.

As discussed in Sec. 4.2, we can expand option features $\{A_i\}_{i=1}^n$ to permutation features $\{B_p\}_{p\in S_n}$ through feature expansion with a fixed difference function. Our implementation of feature expansion module utilize the cross attention mechanism CA[4] as the difference function and perform the feature expansion using Eq. (7). We compute $B = \bigcup_{p\in S_n}\{B_p\}$ where the union symbol represents stacking features.

After obtaining B, we add four more attention blocks to process these permutation features as the post-expansion learning. Then, we reduce the dimension by averaging the sequence length dimension, max pooling the attention dimension and summing over the attention dimension, to get final output $v \in \mathbb{R}^{n!}$. We convert this vector to a one-hot vector h and this corresponds to the final permutation p_O. This finishes the EMBER network for the options O, demonstrated in Figure 3.

For a question Q with length l_Q, the pre-expansion process is the same, getting features $A_Q \in \mathbb{R}^{l_Q\times d}$. During the expansion phase, we compute the self-attention of the questions using the same attention module, thereby obtaining the permutation representation $B_Q \in \mathbb{R}^{l_Q\times d}$. We simply perform average pooling from l_Q to l, create $n!$ copies, and add to the permutation features B. This allows us to incorporates context of question when learning the best permutation, finishing the design of the EMBER network.

[4]CA (A_i, A_j) denotes the standard cross attention, using the query from the first input, keys and values from the second input.

Theorem 7. *The proposed EMBER network satisfies all the properties in Prop. 4.*

4.4 The Objective Function

As discussed in Sec. 4.1, the output probability is denoted by $M \circ E(Q, O)$. We compute the cross entropy loss between the probability and ground truth y by $L_{\text{CE}}(M \circ E(Q, O), y)$.

The final output of the EMBER network's post-expansion learning is a length-$n!$ vector v. We locate the largest value in v to match the final p. However, if there are multiple occurrences of the largest value in v, the correspondence between v to the chosen ρ is not unique anymore, leading to the loss of equivariance.

To mitigate this issue, we propose a divergence loss as

$$L_{div}(E) = \max(\varepsilon_1 - L^2(E), 0) + \max(\varepsilon_2 - s(E), 0), \quad (8)$$

where $\|\cdot\|_2$ denotes the ℓ_2 norm of a vector, $L^2(E) = \frac{1}{nld}\sum_{i=1}^{n-1}\|A_{i+1} - A_i\|_2^2$, and $s(E)$ computes the standard deviation of the output feature v after post-expansion learning in Sec. 4. Each term in the divergence loss is to keep $L^2(E)$ and $s(E)$ over thresholds ε_1 and ε_2 accordingly. Both thresholds are chosen to be relatively small, as our goal is to prevent repeating largest value in v, and minimal level of divergence should be enough to encourage such goal. Therefore, our final loss for training is defined as

$$L(E, Q, O) = L_{\text{CE}} + \gamma L_{div}. \quad (9)$$

where γ is a weighting hyperparameter. To optimize problem (9), we can use stochastic gradient descent or its variants, where for the gradient calculation of h in the permutation matching component of the EMBER network, we rewrite h as $h - v.\text{detach}() + v$, where $v.\text{detach}()$ standing for v without its gradient information.

5 Experiment

5.1 Experimental Setup

We evaluate EMBER on tasks: (i) MMLU (Hendrycks et al. 2021), which is a general benchmark that covers 57 subjects from 4 domains; and (ii) ARC-Challenge (Clark et al. 2018), which is a reasoning benchmark with different grade-school level questions. As the MMLU does not have a default split, we group the subjects by domains and randomly split each domain into the training (70%) and testing (30%) sets. For

Model	Method	MMLU	ARC-Challenge
Llama-3.1-8B	Base	52.2	61.2
	UniBias	53.4(+1.2)	62.2(+1.0)
	PriDe	53.9(+1.7)	62.5(+1.3)
	EMBER	**57.4(+5.2)**	**64.6(+3.4)**
Llama -3.2-3B	Base	28.7	40.2
	UniBias	31.6(+2.9)	41.9(+1.7)
	PriDe	32.1(+3.4)	42.3(+2.1)
	EMBER	**34.8(+6.1)**	**46.5(+6.3)**
Llama -2-7B	Base	33.1	43.3
	UniBias	36.7(+3.6)	45.9(+2.6)
	PriDe	37.2(+4.1)	46.4(+3.1)
	EMBER	**39.5(+6.4)**	**47.8(+4.5)**
Falcon -7B	Base	24.8	34.8
	UniBias	26.8(+2.0)	36.7(+1.9)
	PriDe	27.4(+2.6)	37.4(+2.6)
	EMBER	**33.2(+8.5)**	**41.5(+6.7)**
Vicuna -7B-v1.5	Base	45.5	54.4
	UniBias	46.6(+1.2)	55.0(+0.6)
	PriDe	47.0(+1.6)	55.3(+0.9)
	EMBER	**50.1(+4.6)**	**57.4(+3.0)**
Qwen -2.5-7B	Base	54.3	63.1
	UniBias	55.3(+1.0)	63.8(+0.7)
	PriDe	55.5(+1.2)	64.1(+1.0)
	EMBER	**56.9(+2.6)**	**64.8(+1.7)**

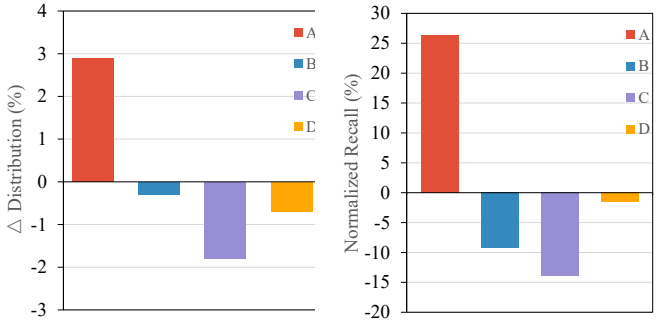
Table 1: Test accuracy (%) comparison of the baseline LLM, UniBias, PriDe, and EMBER. The accuracy improvements over the baseline are in parentheses, with the best in **bold**.

the ARC-Challenge dataset, we use the default training and testing sets. We choose six baseline LLMs, three in the Llama family (Touvron et al. 2023b), Falcon (Almazrouei et al. 2023), Vicuna (Zheng et al. 2023) and Qwen (Yang et al. 2024). The training details of the EMBER network are in Appendix. Additional experiments, including hyperparameter sensitivity, transferability, and performance with various numbers of options, are also included in Appendix.

To the best of our knowledge, there currently is no work addressing the issue of positional bias using **equivariant networks**. Therefore, other than the baseline LLMs, EMBER is primarily compared to PriDe (Zhao et al. 2021) and UniBias (Zhou et al. 2024). PriDe is the state-of-the-arts methods in debiasing. Despite being training-free, PriDe also require large-scale data. We follow Zhao et al. (2021) in implementing PriDe, use the training set to compute the debiasing term, and debias the output probability of LLMs on the testing set. For UniBias, we take the support set from the training set and test the masked model on the testing set.

5.2 Performance Results

Table 1 shows the testing accuracy on MMLU and ARC-Challenge tasks for different models. For MMLU, experiments are conducted on each of the four categories, and we report the average performance. The extended table are present in the appendix. As can be seen, EMBER consistently achieves the best accuracy across models and tasks.



(a) Distribution Difference (b) Normalized Recall

Figure 4: (a) A visualization illustrating how EMBER network influences the distribution of the dataset. (b) Recall rate after shifting all correct answers to each option ID, normalized by subtracting the original accuracy.

Compared with baseline LLMs, EMBER brings more than 5.2% accuracy improvement on average with only 2.36M (approximately 0.03% of 7B) additional parameters, indicating the efficiency of EMBER. Furthermore, EMBER surpasses PriDe by a significant margin. For example, EMBER outperforms PriDe on average by a large margin of 6% on average for Llama 3.2 and 2. This empirically shows that optimizing EMBER to embrace positional bias is beneficial.

5.3 Impact of EMBER on Data Distribution

In this section, we investigate how the EMBER network alters the distribution of the original dataset and whether these changes are analogous to the positional bias. We choose the base model to be Llama-2-7B, and we focus on the STEM testing set. We first collect the original distribution of the correct answer's option IDs. Then, we apply EMBER on each MCQ and collect the distribution of where the correct answer is moved to. We compute the difference between these two distributions, and a visualization is provided in Figure 4a. Following Zheng et al. (2024), we reflect the LLM's positional bias for each option ID using the normalized recall. Specifically, we move all correct answers to a specific option ID and compute the recall rate, normalized by subtracting the original accuracy, visualized in Figure 4.

As illustrated, EMBER shifts correct answers toward options where LLMs exhibit stronger performance, suggesting that EMBER effectively embraces the positional bias.

5.4 Robustness Against Permutation Attack

As discussed in Sec. 2.2, an attacker can permute a multiple-choice question (MCQ) to a certain position, intentionally leading a LLM to provide incorrect answers. This tactic is referred to as a permutation attack (Zong et al. 2024). The setting used to measure vulnerability to this attack is termed "any wrong", marking a question as incorrect if any permuted version of the question is answered incorrectly. The larger the difference between the "any wrong" setting and the original accuracy, the more vulnerable the model is to the permutation attack. Since permutation solves the positional bias, here we verify that EMBER network can suc-

Base LLM	Method	STEM	Social Sciences	Humanities	Others	ARC-Challenge
Llama-3.1-8B	Basic	28.6(-15.1)	41.2(-15.6)	35.5(-17.8)	37.3(-17.6)	42.8(-21.4)
	UniBias	37.4(-7.2)	51.0(-6.9)	47.6(-6.8)	49.4(-7.1)	53.9(-8.3)
	PriDe	40.2(-4.8)	53.1(-5.2)	49.8(-5.4)	51.5(-5.5)	56.0(-6.5)
	EMBER	**48.3(-0.1)**	**61.8(0.0)**	**58.5(0.0)**	**60.5(-0.2)**	**64.6(0.0)**
Llama-3.2-3B	Basic	9.9(-15.7)	12.3(-17.6)	12.0(-17.4)	10.7(-19.3)	20.1(-20.1)
	UniBias	18.7(-9.2)	23.6(-9.4)	22.5(-10.2)	23.1(-9.7)	31.9(-10.0)
	PriDe	22.2(-6.8)	25.3(-8.1)	24.7(-7.7)	26.1(-7.4)	34.0(-8.3)
	EMBER	**31.7(0.0)**	**36.1(0.0)**	**35.6(-0.1)**	**35.5(0.0)**	**46.5(0.0)**
Llama-2-7B	Basic	6.4(-23.0)	11.7(-23.1)	7.9(-25.0)	8.7(-26.5)	10.4(-32.9)
	UniBias	21.5(-11.7)	28.1(-10.2)	26.0(-10.6)	26.5(-12.1)	31.3(-14.6)
	PriDe	27.3(-6.3)	33.8(-5.2)	30.9(-6.1)	34.4(-4.9)	38.2(-8.2)
	EMBER	**36.0(-0.2)**	**41.4(-0.1)**	**39.3(-0.1)**	**40.8(-0.1)**	**47.7(-0.1)**
Falcon-7B	Basic	4.2(-19.6)	4.9(-19.8)	5.3(-20.3)	4.3(-20.6)	14.2(-20.6)
	UniBias	15.0(-10.4)	16.4(-10.4)	16.2(-11.6)	16.3(-10.7)	24.2(-12.5)
	PriDe	18.2(-8.3)	19.2(-8.5)	19.9(-7.9)	19.0(-8.5)	28.6(-8.8)
	EMBER	**33.7(0.0)**	**30.0(-0.1)**	**35.3(0.0)**	**33.6(-0.1)**	**41.4(-0.1)**
Vicuna-7B-v1.5	Basic	20.5(-22.6)	27.5(-19.2)	26.3(-19.5)	24.9(-21.3)	34.0(-20.4)
	UniBias	29.1(-14.9)	34.5(-13.2)	34(-13.3)	33.6(-13.8)	39.4(-15.6)
	PriDe	35.2(-9.1)	39.0(-9.7)	38.3(-9.4)	37.5(-9.8)	47.9(-7.4)
	EMBER	**47.5(-0.1)**	**50.6(0.0)**	**50.3(0.0)**	**51.6(-0.2)**	**57.4(0.0)**
Qwen-2.5-7B	Basic	33.9(-13.5)	40.6(-18.3)	37.4(-14.6)	39.1(-16.0)	42.8(-20.3)
	UniBias	45.2(-7.2)	51.5(-7.9)	44.8(-8.1)	48.7(-7.6)	54.0(-9.8)
	PriDe	47.4(-5.0)	55.6(-4.2)	47.0(-6.1)	49.7(-6.8)	57.1(-7.0)
	EMBER	**54.4(0.0)**	**60.9(-0.1)**	**54.8(0.0)**	**57.1(-0.1)**	**64.8(0.0)**

Table 2: Comparison of the wrong accuracy (%) of all models, with differences from regular accuracy in parentheses.

	CA	Div	Accuracy	Any Wrong
Base Model			33.09	8.68
w/o CA		✓	36.69	36.60
w/o Div	✓		39.14	12.92
EMBER	✓	✓	**39.49**	**39.38**

Table 3: Test accuracy (%) and any wrong accuracy (%) of different variants of EMBER.

Decay Factor	0.9	0.7	0.5	0.3	0.1
Δ Accuracy (%)	6.42	6.39	6.40	6.39	6.36

Table 4: Test accuracy (%) of EMBER with different decay factors, normalized by subtracting the accuracy of the baseline model and averaged over all categories of MMLU.

cessfully defend the permutation attack. We also included the PriDe and UniBias for comparison. As shown in Table 2, baseline LLMs completely fail against permutation attacks. While the PriDe and UniBias show greater robustness, they still do not match the original accuracy. In contrast, EMBER demonstrates strong robustness to permutation attacks, remaining largely unaffected by any wrong settings, due to the equivariant property of EMBER in Prop. 4.

5.5 Ablation Study

We conducted experiments to analyze the effects of cross-attention and divergence loss. First, we simply use subtraction as the distance function instead of cross attention, refer to as EMBER w/o CA. Second, we trained a model without the divergence loss, referred to as EMBER w/o Div.

Both models were trained and tested on four domains of the MMLU dataset, with average accuracy reported as a percentage. As shown in Table 3, cross attention outperforms subtraction, and EMBER w/o Div exhibits a significant drop in the "any wrong" metric. We conclude that cross attention enhances model performance, while divergence loss is essential for maintaining equivariance in EMBER.

5.6 Sensitivity of Decay Factor

In Eq. (6), the decay factor λ is introduced. We trained EMBER on Llama-2-7B with various decay factors. As shown in Table 4, it has a small effect on the performances.

6 Conclusion

Researches have shown that options order can impact model outputs, indicating positional bias. We analyze such bias from an equivariance perspective, and present a novel strategy, leveraging the prefered permutations of LLMs through an equivariant network. Theoretically, we demonstrate that permutation equivariance addresses positional bias. Empirically, we show that EMBER improves performance across various LLMs with minimal additional parameters.

Acknowledgments

This work was supported by National Natural Science Foundation of China (NSFC) under grant number 62250710682 and 62136005.

References

Almazrouei, E.; Alobeidli, H.; Alshamsi, A.; Cappelli, A.; Cojocaru, R.; Debbah, M.; Goffinet, É.; Hesslow, D.; Launay, J.; Malartic, Q.; et al. 2023. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*.

Basu, S.; Sattigeri, P.; Ramamurthy, K. N.; Chenthamarakshan, V.; Varshney, K. R.; Varshney, L. R.; and Das, P. 2023. Equi-Tuning: Group Equivariant Fine-Tuning of Pretrained Models. In *AAAI*.

Bronstein, M. M.; Bruna, J.; LeCun, Y.; Szlam, A.; and Vandergheynst, P. 2017. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Process. Mag.*, 34: 18–42.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. In *NeurIPS*.

Clark, P.; Cowhey, I.; Etzioni, O.; Khot, T.; Sabharwal, A.; Schoenick, C.; and Tafjord, O. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Cohen, T. S.; Geiger, M.; Köhler, J.; and Welling, M. 2018. Spherical CNNs. *CoRR*, abs/1801.10130.

Cohen, T. S.; and Welling, M. 2016. Group Equivariant Convolutional Networks. In *ICML*.

Cohen, T. S.; and Welling, M. 2017. Steerable CNNs. In *ICLR*.

Gens, R.; and Domingos, P. M. 2014. Deep Symmetry Networks. In *NIPS*.

Gerken, J. E.; Carlsson, O.; Linander, H.; Ohlsson, F.; Petersson, C.; and Persson, D. 2022. Equivariance versus Augmentation for Spherical Images. *CoRR*, abs/2202.03990.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2021. Measuring Massive Multitask Language Understanding. In *ICLR*.

Kaba, S.; Mondal, A. K.; Zhang, Y.; Bengio, Y.; and Ravanbakhsh, S. 2023. Equivariance with Learned Canonicalization Functions. In *ICML*.

Li, H.; Zhang, Y.; Koto, F.; Yang, Y.; Zhao, H.; Gong, Y.; Duan, N.; and Baldwin, T. 2024a. CMMLU: Measuring massive multitask language understanding in Chinese. In *ACL*.

Li, J.; Hu, R.; Huang, K.; Zhuang, Y.; Liu, Q.; Zhu, M.; Shi, X.; and Lin, W. 2024b. PertEval: Unveiling Real Knowledge Capacity of LLMs with Knowledge-Invariant Perturbations. *CoRR*, abs/2405.19740.

Li, R.; and Gao, Y. 2024. Anchored Answers: Unravelling Positional Bias in GPT-2's Multiple-Choice Questions. *CoRR*, abs/2405.03205.

Mondal, A. K.; Panigrahi, S. S.; Kaba, O.; Mudumba, S.; and Ravanbakhsh, S. 2023. Equivariant Adaptation of Large Pretrained Models. In *NeurIPS*.

Pei, K.; Li, W.; Jin, Q.; Liu, S.; Geng, S.; Cavallaro, L.; Yang, J.; and Jana, S. 2024. Exploiting Code Symmetries for Learning Program Semantics. In *ICML*.

Pezeshkpour, P.; and Hruschka, E. 2024. Large Language Models Sensitivity to The Order of Options in Multiple-Choice Questions. In *NAACL-HLT*.

Shumaylov, Z.; Zaika, P.; Rowbottom, J.; Sherry, F.; Weber, M.; and Schönlieb, C. 2024. Lie Algebra Canonicalization: Equivariant Neural Operators under arbitrary Lie Groups. *CoRR*, abs/2410.02698.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; Rodriguez, A.; Joulin, A.; Grave, E.; and Lample, G. 2023a. LLaMA: Open and Efficient Foundation Language Models. *CoRR*, abs/2302.13971.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; Bikel, D.; Blecher, L.; Canton-Ferrer, C.; Chen, M.; Cucurull, G.; Esiobu, D.; Fernandes, J.; Fu, J.; Fu, W.; Fuller, B.; Gao, C.; Goswami, V.; Goyal, N.; Hartshorn, A.; Hosseini, S.; Hou, R.; Inan, H.; Kardas, M.; Kerkez, V.; Khabsa, M.; Kloumann, I.; Korenev, A.; Koura, P. S.; Lachaux, M.; Lavril, T.; Lee, J.; Liskovich, D.; Lu, Y.; Mao, Y.; Martinet, X.; Mihaylov, T.; Mishra, P.; Molybog, I.; Nie, Y.; Poulton, A.; Reizenstein, J.; Rungta, R.; Saladi, K.; Schelten, A.; Silva, R.; Smith, E. M.; Subramanian, R.; Tan, X. E.; Tang, B.; Taylor, R.; Williams, A.; Kuan, J. X.; Xu, P.; Yan, Z.; Zarov, I.; Zhang, Y.; Fan, A.; Kambadur, M.; Narang, S.; Rodriguez, A.; Stojnic, R.; Edunov, S.; and Scialom, T. 2023b. Llama 2: Open Foundation and Fine-Tuned Chat Models. *CoRR*, abs/2307.09288.

Unterthiner, T.; Keysers, D.; Gelly, S.; Bousquet, O.; and Tolstikhin, I. O. 2020. Predicting Neural Network Accuracy from Weights. *CoRR*, abs/2002.11448.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *NIPS*, 5998–6008.

Weiler, M.; and Cesa, G. 2019. General E(2)-Equivariant Steerable CNNs. In *NeurIPS*.

Wiegreffe, S.; Tafjord, O.; Belinkov, Y.; Hajishirzi, H.; and Sabharwal, A. 2024. Answer, Assemble, Ace: Understanding How Transformers Answer Multiple Choice Questions. *CoRR*, abs/2407.15018.

Xue, M.; Hu, Z.; Liu, L.; Liao, K.; Li, S.; Han, H.; Zhao, M.; and Yin, C. 2024. Strengthened Symbol Binding Makes Large Language Models Reliable Multiple-Choice Selectors. In *ACL*, 4331–4344.

Yang, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Li, C.; Liu, D.; Huang, F.; Wei, H.; et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Yüksel, A.; Köksal, A.; Senel, L. K.; Korhonen, A.; and Schütze, H. 2024. TurkishMMLU: Measuring Massive Multitask Language Understanding in Turkish. In *EMNLP*.

Zhao, Z.; Wallace, E.; Feng, S.; Klein, D.; and Singh, S. 2021. Calibrate Before Use: Improving Few-shot Performance of Language Models. In *ICML*.

Zheng, C.; Zhou, H.; Meng, F.; Zhou, J.; and Huang, M. 2024. Large Language Models Are Not Robust Multiple Choice Selectors. In *ICLR*.

Zheng, L.; Chiang, W.-L.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E.; et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36: 46595–46623.

Zhou, A.; Yang, K.; Burns, K.; Cardace, A.; Jiang, Y.; Sokota, S.; Kolter, J. Z.; and Finn, C. 2023. Permutation Equivariant Neural Functionals. In *NeurIPS*.

Zhou, H.; Feng, Z.; Zhu, Z.; Qian, J.; and Mao, K. 2024. UniBias: Unveiling and Mitigating LLM Bias through Internal Attention and FFN Manipulation. *CoRR*, abs/2405.20612.

Zong, Y.; Yu, T.; Chavhan, R.; Zhao, B.; and Hospedales, T. M. 2024. Fool Your (Vision and) Language Model with Embarrassingly Simple Permutations. In *ICML*.