Breaking the Ceiling of the LLM Community by Treating Token Generation as a Classification for Ensembling

Anonymous ACL submission

Abstract

Ensembling multiple models has always been an effective approach to push the limits of existing performance and is widely used in classification tasks by simply averaging the classification probability vectors from multiple classifiers to achieve better accuracy. However, in the thriving open-source Large Language Model (LLM) community, ensembling methods are rare and typically limited to ensembling the full-text outputs of LLMs, such as selecting the best output using a ranker, which leads to underutilization of token-level probability information. In this paper, we treat the **G**eneration of each token by LLMs **a**s a **C**lassification (**GaC**) for ensembling. This approach fully exploits the probability information at each generation step and better prevents LLMs from producing early incorrect tokens that lead to snowballing errors. In experiments, we ensemble state-of-the-art LLMs on several benchmarks, including exams, mathematics and reasoning, and observe that our method breaks the existing community performance ceiling. Furthermore, we observed that most of the tokens in the answer are simple and do not affect the correctness of the final answer. Therefore, we also experimented with ensembling only key tokens, and the results showed better performance with lower latency across benchmarks.

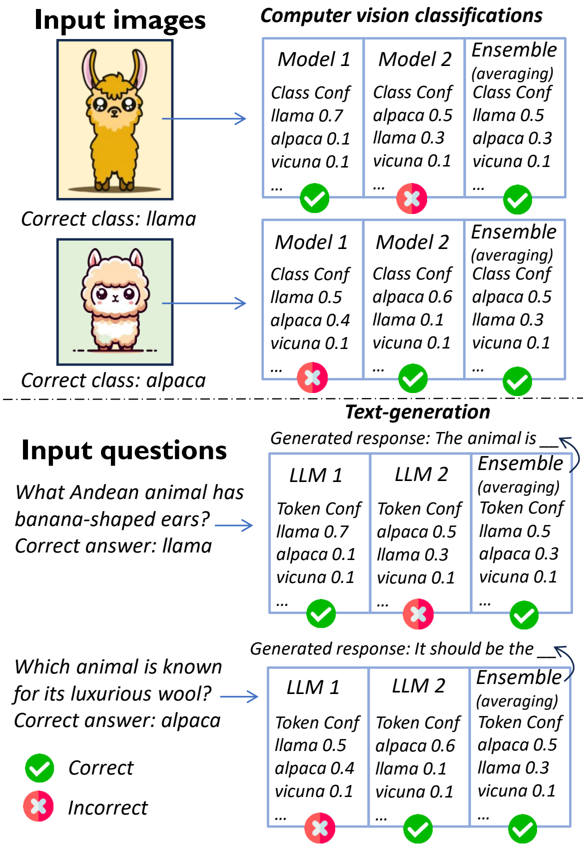

Figure 1: Motivation of GaC. The upper part shows CV classification ensemble, while the lower part illustrates ensemble at one text generation step.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in a wide range of natural language processing tasks (Achiam et al., 2023; Touvron et al., 2023). Over time, new and more powerful LLMs are continually being released, pushing the boundaries of the LLM community (Meta, 2024; Alibaba, 2024). Due to the diversity of data sources, architectures and training methods, different LLMs have strengths and weaknesses in different tasks and contexts (Jiang et al., 2023). In addition to investing significant resources in training a superior LLM, ensembling multiple existing models is another effective way to break through the community performance ceiling (Huang et al., 2016), especially given the current trend in the open source LLM community to contribute only model weights rather than training data and procedures (AllenAI, 2024).

Taking computer vision (CV) classification as an example, it is common to ensemble the output probability vectors of multiple models (e.g. by averaging) to achieve superior results (Krizhevsky et al., 2017). This approach remains effective even with

CV Models on ImageNet	*Acc* [%]	*ECE*
EfficientNet-B1	78.55	0.072
RepGhostNet	78.81	0.053
PVTv2-B1	78.71	0.119
Ensembled CV Models (Averaged)		
EfficientNet-B1 + PVTv2-B1	80.20↑**1.49**	-
EfficientNet-B1 + RepGhostNet	80.06↑**1.25**	-
EfficientNet-B1 + RepGhostNet + PVTv2-B1	80.62↑**1.81**	-
LLMs on MMLU	*Acc* [%]	*ECE*
Llama-3-70b-Instruct	79.68	0.095
Qwen1.5-72b-chat	77.79	0.089
Yi-34B-Chat	72.75	0.090

Table 1: Performance of various CV models on ImageNet and LLMs on MMLU. ↑ indicates improvement over a single model.

recent CV models. As shown in Tab.1, we selected several common CV models (Chen et al., 2022; Tan and Le, 2019; Wang et al., 2022) for ensembling and observed better accuracy on ImageNet (Deng et al., 2009) compared to using a single model. Similarly, the popular decoder-only LLM architecture generates text by producing tokens one by one, with each generation step resulting in a probability vector of the length of the vocabulary. Inspired by CV, we propose to treat each generation step as a classification task, and by ensembling multiple models, we can achieve higher accuracy, as shown in Fig.1. There is already work that simplifies problems into binary tasks, exploiting the collective wisdom of LLMs and achieving better results (Schoenegger et al., 2024), demonstrating the feasibility of this approach.

Another advantage is that early errors in LLMs often snowball into later errors (Zhang et al., 2023). Ensembling during generation helps prevent the generation of inaccurate tokens at each step, thereby reducing misleading cues for subsequent token generation. In this paper, we conducted experiments at several points in time between November 2023 and June 2024, ensembling available state-of-the-art (SOTA) LLMs up to each of these points. We found that this approach **significantly** outperformed any single model available at those times on five popular benchmarks involving subject examination, mathematics, reasoning, and knowledge-based QA.

In addition, we found that for text generation it seemed unnecessary to ensemble every token. For example, for the question "*What Andean animal has banana-shaped ears?*" shown in Fig.1, the most critical part is for the LLM to generate the key token "*llama*". The initial part of the answer "*It should be _*" or "*The animal is _*" do not significantly affect the correctness of the final answer. Ideally, the step that produces the token "*llama*" is the one we want to ensemble.

Studies in CV classification have also shown that most samples are "simple" and can be correctly classified by most models (Wang et al., 2017), including cost-efficient ones, making the use of expensive models wasteful. To address this, CV classification used cascade inference (Jazbec et al., 2024; Enomoro and Eda, 2021), where a gate model passes a sample to a more powerful model only if its confidence falls below a threshold, thereby improving efficiency. Obviously, it is very important for cascading that the confidence of the gate model accurately reflects the accuracy. To ensure that LLMs are also suitable as gate models, we measured the Expected Calibration Error (ECE) (Guo et al., 2017) of CV models and LLMs on ImageNet and MMLU (Hendrycks et al., 2020), as shown in Tab.1. ECE is a metric that reflects the difference between a model's confidence and its accuracy. We found that the ECE of CV models and LLMs were close. Therefore, in this paper, we also applied the cascade inference to LLMs by ensembling only the "key" tokens to speed up generation. Our experiments showed that this approach consistently achieved better performance with lower latency across different benchmarks.

2 Analysis and Prior Work

In this chapter, we review previous LLM ensemble studies and their features, as well as the problems our approach addresses. Previous studies can be categorized as follows:

Output-level ensemble methods select multiple candidate models and use their complete outputs for ensembling. Jiang et al. (2023) trained an additional ranking model (*PairRanker*) to score each candidate output and select the best one. Lu et al. (2023) and Shnitzer et al. (2023) trained a router to select the most appropriate candidate model given a question. However, these methods are limited to the existing candidate outputs and become ineffective if all the outputs are incorrect. Other studies have trained a fusion model to blend the outputs (Jiang et al., 2023; Wang et al., 2023b), overcoming the limitation of selecting only existing candidate outputs and often achieving superior results. However, the generalization of the fusion model is a major challenge, and they cannot fully exploit the probability information from each generation step.

Weight-level ensemble methods merge the weights of multiple models and are primarily used in multi-task learning (Yadav et al., 2024). The expectation is that the merged model will inherit capabilities across multiple tasks. However, a limitation is that the architectures of the models to be merged must be homologous, which limits the use of the capabilities of the LLM community. And it is rare to observe that the merged model outperforms the original models (Yu et al., 2023).

Training-level ensemble like FuseLLM (Wan et al., 2024) uses the output probability vectors of multiple models during training to ensemble as labels, rather than one-hot labels. In effect, this is a specific form of distillation that allows the model being trained to gain more information from the probability outputs of the ensembled (teacher) models. However, distillation is mainly used to improve small models, making it difficult to further improve the SOTA LLMs.

Our work can overcome the above limitations by ensembling at each generation step, allowing the output not to be confined to the original candidate output space and homologous architectures, while fully exploiting the probability information at each step. Our experiments in Sec.4 will also show that the ensemble consistently outperforms any single model, even SOTA LLMs. The main challenge, however, is that different LLMs typically have inconsistent vocabularies, leading to different dimensions in the probability vectors produced by different models. The most intuitive solution is to take the union of the vocabularies of the ensembled LLMs, denoted V^U, which includes all tokens from the participating models. Then, at each generation step, the output is first mapped to this union space $\mathbb{R}^{|V^U|}$ before ensembling.

A potential problem with this approach is that different models may tokenize the same word differently, leading to conflicts. However, most mainstream LLMs use BPE or BBPE (Sennrich et al., 2015; Wang et al., 2020) to train tokenizers on sampled corpora, which tend to have similar sources (e.g. CommonCrawl) and distributions. This results in consistent tokenization for common words. For example, both Qwen1.5 and Llama3 (Bai et al., 2023; Meta, 2024) tokenize the word " *alphabetically*" into ["*Ġalphabet*", "*ically*"]. If both models

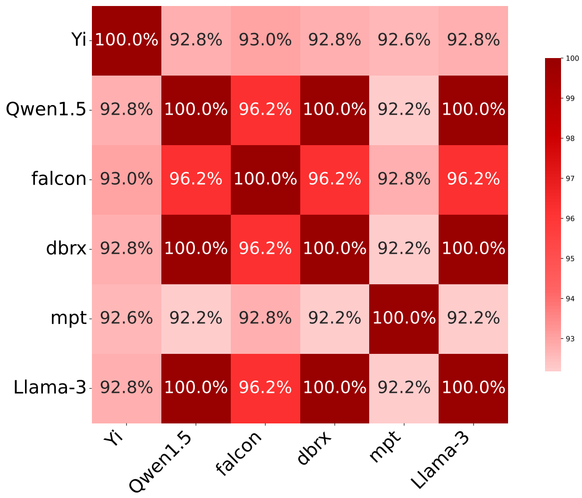

Figure 2: The rate of identical tokenization for Oxford 5000 common words between different LLMs.

intend to output this word, they will assign a higher probability to "*Ġalphabet*" first. We selected several popular LLMs (Young et al., 2024; Databricks, 2024; Almazrouei et al., 2023) and tokenized 5,000 commonly used English words (Oxford, 2018), and then calculated the proportion of identical tokenization results between each pair of LLMs, as shown in Fig.2. The proportion is above 90% for all pairs, indicating that such conflicts can be ignored in most cases.

3 Proposed Method

In this section, we will first introduce the overall ensemble process of our GAC framework, and then explain the details in the following subsections.

3.1 Overall Process of GAC

When generating text, LLMs output a probability vector of the same dimension as their vocabulary. Given n LLMs to be ensembled, we first take the union of their vocabularies and create a mapping matrix that can project the probability vectors to the union dimensions (Sec.3.2). At each generation step, all LLMs produce outputs that are mapped to the union vocabulary dimensions and ensembled to sample the next token. The tokenizer of each LLM then converts the sampled token into token IDs for the next step (Sec.3.3). As mentioned in Sec.1, not all tokens have the necessity for ensembling, so we also try to ensemble only certain key tokens (Sec.3.4).

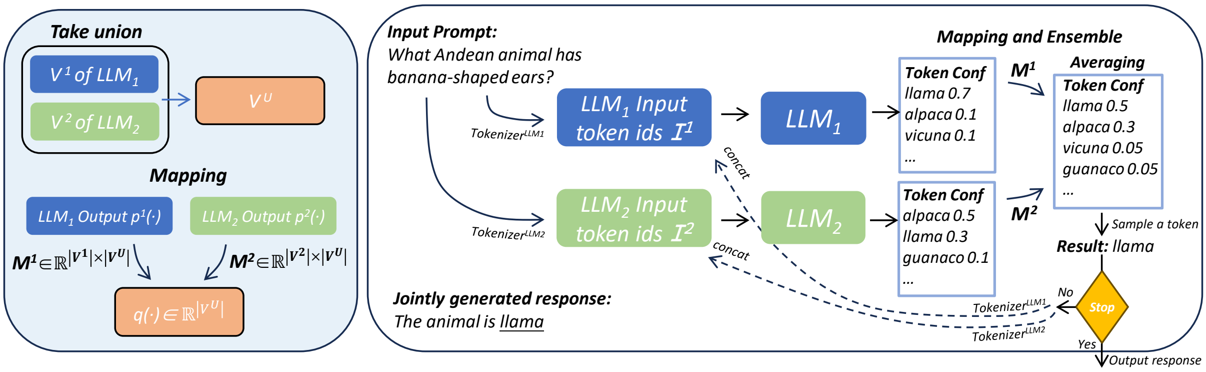

Figure 3: Overview of GAC. The left side shows the creation of the mapping matrix, and the right side shows the ensembling during text generation with two LLMs.

3.2 Creating the Union Mapping

Given $\{LLM_1, LLM_2, \ldots, LLM_n\}$ to ensemble, with their respective vocabularies $\{V^1, V^2, \ldots, V^n\}$, we first take the union of the vocabularies:

$$V^U = \bigcup_{i=1}^{n} V^i. \tag{1}$$

During this process, we record the positions of tokens from V^i in V^U and create corresponding mapping matrices $\mathbf{M}^i \in \mathbb{R}^{|V^i| \times |V^U|}$.

3.3 GAC Ensembling

At the start of text generation, we convert the input *prompt* into token ID sequences for each LLM. We denote the tokenizer of LLM_i as $\mathcal{T}^i : text \to (\tau_1, \tau_2, \ldots, \tau_m)$, which converts the input text into a sequence of token IDs. We calculate:

$$\mathcal{I}^i = \mathcal{T}^i(prompt) \quad \text{for } i = 1, \ldots, n \tag{2}$$

where $\mathcal{I}^i$ is the input token ID sequence for LLM_i.

For each generation step, we input $\mathcal{I}^i$ into LLM_i to obtain $p^i(\cdot \,|\, \mathcal{I}^i) \in \mathbb{R}^{|V^i|}$, which represents the probability vector for the next token. These vectors are then mapped to the union vocabulary dimensions and averaged:

$$q(\cdot) = \frac{1}{n} \sum_{i=1}^{n} p^i(\cdot \,|\, \mathcal{I}^i) \cdot \mathbf{M}^i, \tag{3}$$

where $q(\cdot)$ is the ensemble probability vector. In Sec.4.3, we experimented with different ensemble weights and decided to use the average. We then sample a token $x \sim q(\cdot)$ as the result of this step. Finally, the sampled token is converted back into token IDs for each LLM and appended to $\mathcal{I}^i$:

$$\mathcal{I}^i \leftarrow \mathcal{I}^i \frown \mathcal{T}^i(x) \quad \text{for } i = 1, \ldots, n \tag{4}$$

We repeat (3) and (4) until the stopping criteria are met, such as outputting an end-of-sentence token or reaching the maximum length, as shown in Fig.3. In our implementation, different LLMs run in parallel on different GPUs, so the duration of each step is equal to the time taken by the slowest LLM. Since we have not modified a complete forward pass, our approach is compatible with techniques such as vLLM, DeepSpeed, quantization, and hardware optimizations (Kwon et al., 2023; Rasley et al., 2020).

3.4 Ensembling Key Tokens with Threshold

As mentioned in the last part of Sec.1, most tokens do not significantly affect the correctness of the response. From Tab.1, we can see that LLMs and CV models have similar ECE levels, suggesting that the confidence scores of LLMs may reflect accuracy to some extent. Therefore, we also experiment with ensembling only the tokens with confidence below a threshold t. We choose a model as the gate, denoted LLM_g, and use its maximum probability at each step as the confidence score. During the ensemble, we replace the original (3) with:

$$q(\cdot) = \begin{cases} \frac{1}{n} \sum_i p^i(\cdot \,|\, \mathcal{I}^i) \cdot \mathbf{M}^i & \textit{if } \max(p^g(\cdot \,|\, \mathcal{I}^g)) \le t \\ p^g(\cdot \,|\, \mathcal{I}^g) \cdot \mathbf{M}^g & \textit{otherwise}. \end{cases} \tag{5}$$

Note that apart from LLM_g, the other LLMs are not computed at every step, so their KV caches become stale. While there has been research using partial KV caches (Barad et al., 2023), for simplicity our work disables the KV caches of all LLMs except LLM_g. This is an area for improvement and is listed in our future work.

4 Experiments

4.1 Overview

In this section, we first present the experimental setup, including the benchmarks and hardware used (Sec.4.2). We then test the effects of different ensemble weights for GAC (Sec.4.3) and compare it with other methods (Sec.4.4). We also select SOTA LLMs available at different times for ensembling to explore the performance ceiling at each time period (Sec.4.5). Finally, we experiment with thresholded ensembling to explore variations in latency and performance (Sec.4.6).

4.2 Experimental Settings

Benchmarks. GAC is not limited to specific tasks, so we tested it as broadly as possible. We selected a total of five benchmarks. For general capabilities, we chose MMLU (Hendrycks et al., 2020). For maths, we utilized GSM8K (Cobbe et al., 2021). For reasoning, we employed BBH (Suzgun et al., 2023). For knowledge capabilities, we included TriviaQA (Joshi et al., 2017) and NaturalQuestions (NQ) (Kwiatkowski et al., 2019). Note that all scores, including those for individual models, were computed locally under the same environment to ensure fairness[1], using lm-evaluation-harness[2] v0.4.1 (Gao et al., 2023).

Hardware and Latency. Each LLM was loaded on 1(n) A100 GPU(s) according to its memory requirements[3], using naive model parallelism (HuggingFace, 2024) without optimization for inference. During ensembling, different LLMs were loaded on separate GPU(s) and executed in parallel, managed and communicated via Ray (Moritz et al., 2018). We also recorded the latency (ms/token). Each model performs a "dry run" after being loaded onto the GPU, generating 1024 tokens to warm up CUDA before experimentation, following the practice of Mehta et al. (2024).

4.3 Different Ensemble Weights

Before proceeding with further experiments, we tested different ensemble weights for GAC. We used a simple averaging of the probabilities from each model in Eq.3. We now replace Eq.3 by $\frac{1}{\sum_i w^i} \sum_i w^i p^i(\cdot \,|\, \mathcal{I}^i)\mathbf{M}^i$, where w^i is the ensemble weight for LLM_i. We set w^i separately to each LLM's score on MMLU, the MMLU score minus the ECE and 1 (i.e. averaging). We selected two sets of LLMs (Wang et al., 2023a; Kim et al., 2023; Young et al., 2024; Bai et al., 2023; NousResearch, 2024a) listed in Tab.2 with different sizes for GAC ensemble. The results on MMLU are shown in Fig.4. We observed no significant differences between the different weights, so for simplicity we decided to use averaging.

[1]Please see Appendix.A for more benchmarks details.
[2]https://github.com/EleutherAI/lm-evaluation-harness
[3]We listed each model and its hardware in Appendix.B.

Model	*Acc* [%]	*ECE*
OpenChat-3.5-0106	64.53	0.0833
Qwen1.5-14B-Chat	67.20	0.1312
SOLAR-10.7B-Instruct-v1.0	64.48	0.2884
Yi-34B-Chat	72.75	0.0903
Qwen1.5-32B-Chat	75.12	0.1003
Nous-Hermes-2-Mixtral-8x7B-DPO	72.65	0.0789

Table 2: Two sets of LLMs of different sizes on MMLU. Smaller models on top, larger models on bottom.

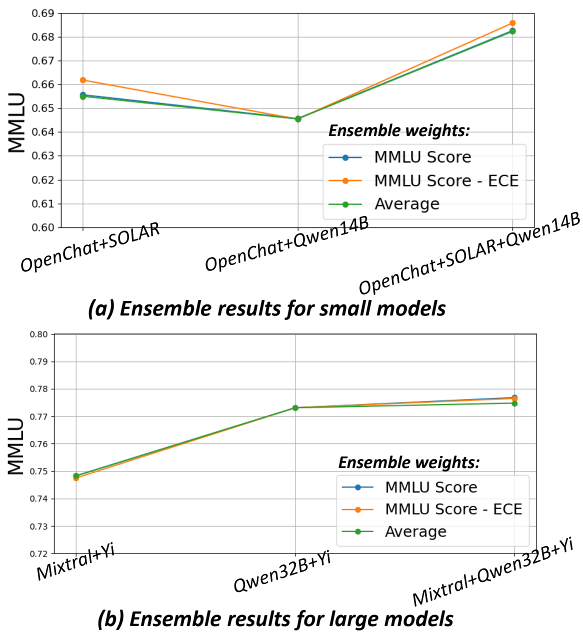

Figure 4: Results of GAC ensemble with different weights for models from Tab.2. Smaller models ensembles on top, larger ones on bottom. The x-axis shows names participating in the ensemble (abbreviated).

4.4 Comparison with Other Methods

Baselines. We compared GAC with existing methods. First, we considered LLM Blender (Jiang et al., 2023), which employs PairRanker to rank the outputs of candidate LLMs and GenFuser to fuse these outputs. However, we found that GenFuser refused to answer a significant proportion of the questions in our chosen benchmarks. We therefore only used PairRanker to ensure fairness. We also

Id	Models	MMLU	GSM8K	BBH	TriviaQA	NQ	Avg.	Latency
1	openchat_3.5	63.87	68.46	47.96	68.12	29.75	55.63	28.01*ms/token*
2	Nous-Hermes-2-SOLAR-10.7B	64.88	72.86	49.92	71.33	32.21	58.24	50.89*ms/token*
	Ensemble Results for the Above Two Models (openchat and SOLAR)							
3	LLM Blender (PairRanker)	64.23	74.07	50.18	70.55	32.55	58.32	57.86*ms/token*
4	OAssistRM	64.87	72.93	49.02	70.06	31.64	57.70	55.62*ms/token*
5	UltraRM	64.94	75.51	50.65	71.23	32.03	58.87	74.51*ms/token*
6	GaC (***ours***)	66.51	74.30	51.19	72.50	33.82	59.66	51.32*ms/token*
7	FuseLLM	63.94	65.50	46.32	64.57	29.06	53.88	28.31*ms/token*
8	$\text{GaC}^{t=0.5}_{7.68\%}$ (***ours***)	65.14	73.18	50.32	69.68	31.75	58.01	31.34*ms/token*
9	Mixtral-8x7B-Instruct-v0.1	70.89	66.82	49.84	76.54	34.35	59.69	96.64*ms/token*
10	Yi-34B-Chat	72.75	68.76	50.88	70.01	29.81	58.44	67.96*ms/token*
	Ensemble Results for the Above Two Models (Mixtral and Yi)							
11	LLM Blender (PairRanker)	72.69	69.59	51.70	72.37	32.24	59.72	105.21*ms/token*
12	OAssistRM	73.34	70.15	51.91	72.79	30.69	59.78	99.75*ms/token*
13	UltraRM	69.49	71.09	52.27	73.82	32.36	59.81	114.57*ms/token*
14	GaC (***ours***)	74.83	71.21	52.64	75.60	33.52	61.56	98.13*ms/token*

Table 3: Results of comparison with other methods. Upper and lower halves represent different ensemble combinations. Blue indicates the best result for each ensemble. For id 8, GaC bottom right shows ensembled token proportion, top right shows threshold.

included other rankers, such as OAssistRM[4] (Köpf et al., 2024) and UltraRM (Cui et al., 2023), which we ran in parallel on the GPUs hosting the ensemble LLMs to ensure low latency. These rankers scored the outputs and selected the best answers. Furthermore, we included the FuseLLM (Wan et al., 2024) (OpenChat-3.5-7B-Solar[5]), which uses probability information from multiple models during training for distillation.

Models for Ensemble. We chose two sets of models of different sizes. The smaller models included openchat-3.5 (Wang et al., 2023a) and Nous-Hermes-2-SOLAR-10.7B (NousResearch, 2024b) (teacher models for OpenChat-3.5-7B-Solar distillation). Larger models included Mixtral-8x7B-Instruct-v0.1 and Yi-34B-Chat (Jiang et al., 2024; Young et al., 2024).

Experimental Results. We ensemble the above two sets of LLMs using both our and baseline methods, and present the results in Tab.3. Our method showed superior performance with the lowest latency for both combinations (row ids 6 and 14). For row id 8, we used openchat-3.5 as the gate model with a threshold of 0.5 (Eq.5), resulting in only 7.68% of tokens being ensembled. This slightly increased the latency from 28.01 to 31.34 ms/token, but achieved a performance close to that of SOLAR-10.7B (average score of 58.01 vs. 58.24), whose latency is 50.89 ms/token, further demonstrating the effectiveness of our method.

[4] https://huggingface.co/OpenAssistant/reward-model
[5] https://huggingface.co/FuseAI/OpenChat-3.5-7B-Solar

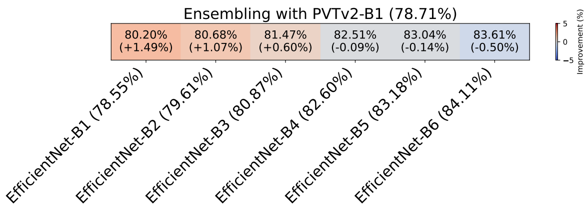

Figure 5: Ensemble results of CV models with different accuracy gaps on ImageNet. Models' accuracies are next to their names. Each cell shows the ensemble accuracy, with the improvement over the best single model in parentheses.

4.5 Breaking the Ceiling

In this experiment, we aimed to break the performance ceiling of the open source LLM community at different times. We chose the SOTA LLMs released between November 2023 and June 2024, as listed in the upper part of Tab.4, excluding models with more than 100 billion parameters due to hardware limitations. We then ensemble the SOTA LLMs available at different times, as shown in row ids 6-10, and observe an improvement of **3.13%** to **4.47%** over the best single model at each time. An exception is 2024/04/18, when Llama-3-70B-Instruct was released and significantly improved performance over the previous SOTA LLMs (aver-

Id	Models	MMLU	GSM8K	BBH	TriviaQA	NQ	Avg.	Date	Latency
1	Yi-34B-Chat	72.75	68.76	50.88	70.01	29.81	58.44	2023/11/08	67.96*ms/token*
2	Mixtral-8x7B-Instruct-v0.1	70.89	66.82	49.84	76.54	34.35	59.69	2023/12/11	96.64*ms/token*
3	Qwen1.5-72B-Chat	77.79	83.33	48.94	65.69	27.02	60.55	2024/02/04	102.11*ms/token*
4	Llama-3-70B-Instruct	79.68	90.00	57.13	79.12	35.57	68.30	2024/04/18	150.32*ms/token*
5	Qwen2-72B-Instruct	82.30	89.70	62.57	73.58	33.11	68.25	2024/06/07	113.91*ms/token*
					Ensemble the Above Models with GAC				
6	Yi + Mixtral	74.83	71.21	52.64	75.60	33.52	61.56↑**3.13%**	~2023/12/11	98.13*ms/token*
7	Qwen1.5-72B + Yi	79.83	77.27	52.05	70.88	33.80	62.77↑**3.65%**	~2024/02/04	103.69*ms/token*
8	Qwen1.5-72B + Mixtral	79.55	75.76	54.19	75.71	31.09	63.26↑**4.47%**	~2024/02/04	112.83*ms/token*
9	Llama-3 + Qwen1.5-72B	81.49	87.06	56.73	78.60	36.01	67.98↓**0.47%**	~2024/04/18	153.96*ms/token*
10	Qwen2-72B + Llama-3	83.54	90.91	63.99	79.29	37.65	71.08↑**4.06%**	~2024/06/07	151.56*ms/token*
					Task-specific Top-2 Model Ensemble with GAC				
11	Top-2 (~2024/04/18)	81.49	87.96	58.64	80.84	37.95	69.38↑**1.58%**	~2024/04/18	-
12	Top-2 (~2024/06/07)	83.54	90.91	63.99	80.84	37.95	71.45↑**4.61%**	~2024/06/07	-

Table 4: Ensemble of available SOTA LLMs from different periods. The top part lists the individual models, while the bottom part shows the ensemble results (model names abbreviated). ↑ indicates the percentage improvement over the individual models.

age score increased from 60.55 of Qwen1.5-72B-Chat to 68.30), resulting in a drop in performance after ensemble due to the large gap.

However, with the release of Qwen2-72B-Instruct, which showed comparable performance to Llama-3-70B-Instruct, the ensemble again led to significant improvements (row id 10). In rows 11 and 12, we ensemble the top two best-performing models for each benchmark at the two most recent times, including the challenging time of 2024/04/18, and observe performance gains with this task-specific top-two ensemble even on 04/18 (row id 11). Finally, row id 12 shows the **best** results available for the open source community on 2024/06/07. By pushing the boundaries of the community, we can narrow the gap with proprietary models and promote the democratization of LLMs.

Since ensembling models with large performance differences could lead to performance degradation (row id 9 in Tab.4), we also tested this hypothesis with CV models. We ensemble PVTv2-B1 (Wang et al., 2022) with different sizes of EfficientNet (Tan and Le, 2019) on ImageNet (Deng et al., 2009) by averaging their outputs, as shown in Fig.5. We observed that as the accuracy gap between the two models increased, the ensemble gains decreased and eventually became negative. This suggests that it is advisable to ensemble models with similar levels of performance.

4.6 Ensemble with Threshold

In this experiment, we used the thresholded ensemble (Sec.3.4) to explore variations in latency and performance. We selected models of different sizes (Abdin et al., 2024; Meta, 2024; Bai et al., 2023), listed in the upper part of Tab.5, pairing a smaller model with a larger model and using the smaller model as the gate model for the ensemble. We aimed to match the performance (average score) of Qwen1.5-72B-chat and Qwen1.5-32B-chat with our ensemble, but with lower latency, and the results are shown in row ids 6-9. Interestingly, even when combining the two Qwen models themselves with a threshold of 0.5 (row id 7), where 6.31% of the tokens were ensembled, we observed slightly higher performance than Qwen1.5-72B-chat (average score increased from 60.55 to 60.96) and lower latency (102.11 to 77.86 ms/token). We believe this is a promising new way to speed up inference.

We also observed lower latency in row ids 6, 8 and 9 of Tab.5 with comparable performance to Qwen1.5-72B-chat or Qwen1.5-32B-chat. In addition, similar trends were observed in the MT-Bench (Zheng et al., 2023) using LLM as judge (GPT-4-0613) in Tab.6, with scores calculated using FastChat[6] in our local environment. For the combinations of Llama-3-8B-Instruct or Phi-3-mini-4k-instruct with Llama-3-70B-Instruct, we directly adopted the output of the larger model if the probability of the smaller model was below the threshold. This is based on our observations in Fig.5, where ensembling models with large performance gaps resulted in reduced performance.

[6] https://github.com/lm-sys/FastChat

Id	Models	Threshold	MMLU	GSM8K	BBH	TriviaQA	NQ	Avg.	Latency
1	Llama-3-70B-Instruct	-	79.68	90.00	57.13	79.12	35.57	68.30	150.32*ms/token*
2	Llama-3-8B-Instruct	-	65.08	76.26	44.72	67.67	26.48	56.04	34.30*ms/token*
3	Phi-3-mini-4k-instruct	-	67.11	79.00	47.26	55.78	17.98	53.43	31.82*ms/token*
4	Qwen1.5-72B-chat	-	77.79	83.33	48.94	65.69	27.02	60.55	102.11*ms/token*
5	Qwen1.5-32B-Chat	-	75.12	75.97	53.89	62.57	22.96	58.10	59.01*ms/token*
	*Ensemble with threshold to match Qwen1.5-**72B**-Chat performance (avg. 60.55)*								
6	Llama-3-8B + Llama-3-70B$_{9.30\%}$	0.5	69.51	82.86	47.65	74.35	33.74	61.62	68.97*ms/token*
7	Qwen1.5-32B + Qwen1.5-72B$_{6.31\%}$	0.5	75.53	81.82	55.87	63.91	27.69	60.96	77.86*ms/token*
	*Ensemble with threshold to match Qwen1.5-**32B**-Chat performance (avg. 58.10)*								
8	Llama-3-8B + Llama-3-70B$_{6.98\%}$	0.45	68.06	81.79	46.66	73.43	33.74	60.74	58.99*ms/token*
9	Phi-3 + Llama-3-70B$_{7.59\%}$	0.5	68.46	78.57	50.34	69.08	29.51	59.19	51.61*ms/token*

Table 5: Thresholded ensemble results. The top lists individual models, while the bottom shows ensemble combinations (model names abbreviated). The percentage in the bottom right of the combination names represents the proportion of tokens ensembled.

Models	Threshold	MT-Bench	Latency
Llama-3-8b-Instruct	-	8.03	33.75*ms/token*
Llama-3-70b-Instruct	-	8.80	128.50*ms/token*
Qwen1.5-72b-chat	-	8.33	87.39*ms/token*
GAC (Llama-3-8b + Llama-3-70b)$_{10.10\%}$	0.55	8.34	69.28*ms/token*
Qwen1.5-32b-chat	-	8.12	58.49*ms/token*
GAC (Llama-3-8b + Llama-3-70b)$_{7.24\%}$	0.5	8.16	57.28*ms/token*

Table 6: Thresholded ensemble on MT-Bench. GAC shows the ensemble combinations (model names abbreviated), with the proportion of tokens ensembled shown at the bottom right.

5 Conclusion

In this paper, we present a token-level ensembling framework called GAC, which fully exploits the probability information at each generation step. In our experiments, we have surpassed the performance ceiling of open-source SOTA LLMs available at different time periods (Sec.4.5), further narrowing the gap between open-source and proprietary models. This progress promotes the democratization of LLMs and provides new motivations for future research, enabling better exploitation of collective intelligence. In addition, we experimented with ensembling just a few tokens and found that this approach can achieve better performance with lower latency (Sec.4.6), opening up new avenues for accelerating inference.

Contemporaneous Works

We have noticed several contemporaneous works related to our research, all of which aim to address the vocabulary discrepancy between different models. Xu et al. (2024) proposed EVA, which trains a projection matrix between each pair of LLMs, using the overlapping tokens from their vocabularies as a bridge. DEEPEN (Huang et al., 2024) converts the output probabilities to a relative representation using anchor tokens before ensembling, and then inverts back to the original model's vocabulary space using gradient descent, which requires an additional 7% to 29% of time per generation step. In contrast, our method requires no additional training and only a single matrix multiplication and tokenization for each model during ensembling, with minimal time cost.

Limitation

Like other ensemble methods, the approach proposed in this paper requires more computational resources. Although different models can be run in parallel on separate GPUs, so that latency only depends on the slowest model, the overall computational load is additive, raising the threshold for use.

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

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

Alibaba. 2024. Qwen-2: Blog post. Accessed: 2024-06-07.

AllenAI. 2024. Hello olmo: A truly open llm. Accessed: 2024-06-07.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. 2023. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Haim Barad, Ekaterina Aidova, and Yury Gorbachev. 2023. Leveraging speculative sampling and kv-cache optimizations together for generative ai using openvino. *arXiv preprint arXiv:2311.04951*.

Chengpeng Chen, Zichao Guo, Haien Zeng, Pengfei Xiong, and Jian Dong. 2022. Repghost: a hardware-efficient ghost module via re-parameterization. *arXiv preprint arXiv:2211.06088*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2023. Ultrafeedback: Boosting language models with high-quality feedback. *arXiv preprint arXiv:2310.01377*.

Databricks. 2024. Introducing dbrx: A new state-of-the-art open llm. Accessed: 2024-06-07.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Shohei Enomoro and Takeharu Eda. 2021. Learning to cascade: Confidence calibration for improving the accuracy and computational cost of cascade inference systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 8, pages 7331–7339.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. 2016. Snapshot ensembles: Train 1, get m for free. In *International Conference on Learning Representations*.

Yichong Huang, Xiaocheng Feng, Baohang Li, Yang Xiang, Hui Wang, Bing Qin, and Ting Liu. 2024. Enabling ensemble learning for heterogeneous large language models with deep parallel collaboration. *arXiv preprint arXiv:2404.12715*.

HuggingFace. 2024. Transformers documentation: Model parallelism. https://huggingface.co/docs/transformers/v4.15.0/parallelism. Accessed: 2024-06-07.

Metod Jazbec, James Allingham, Dan Zhang, and Eric Nalisnick. 2024. Towards anytime classification in early-exit architectures by enforcing conditional monotonicity. *Advances in Neural Information Processing Systems*, 36.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14165–14178.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Dahyun Kim, Chanjun Park, Sanghoon Kim, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, et al. 2023. Solar 10.7 b: Scaling large language models with simple yet effective depth up-scaling. *arXiv preprint arXiv:2312.15166*.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. 2024. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems*, 36.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2017. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2023. Routing to the expert: Efficient reward-guided ensemble of large language models. *arXiv preprint arXiv:2311.08692*.

Sachin Mehta, Mohammad Hossein Sekhavat, Qingqing Cao, Maxwell Horton, Yanzi Jin, Chenfan Sun, Iman Mirzadeh, Mahyar Najibi, Dmitry Belenko, Peter Zatloukal, et al. 2024. Openelm: An efficient language model family with open-source training and inference framework. *arXiv preprint arXiv:2404.14619*.

Meta. 2024. Meta ai blog: Meta llama 3. Accessed: 2024-06-07.

Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. 2018. Ray: A distributed framework for emerging {AI} applications. In *13th USENIX symposium on operating systems design and implementation (OSDI 18)*, pages 561–577.

NousResearch. 2024a. Nous-hermes-2-mixtral-8x7b-dpo. `https://huggingface.co/NousResearch/Nous-Hermes-2-Mixtral-8x7B-DPO`. Accessed: 2024-06-07.

NousResearch. 2024b. Nous-hermes-2-solar-10.7b. `https://huggingface.co/NousResearch/Nous-Hermes-2-SOLAR-10.7B`. Accessed: 2024-06-07.

Oxford. 2018. The oxford 5000. Accessed: 2024-06-07.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Philipp Schoenegger, Indre Tuminauskaite, Peter S Park, and Philip E Tetlock. 2024. Wisdom of the silicon crowd: Llm ensemble prediction capabilities match human crowd accuracy. *arXiv preprint arXiv:2402.19379*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Tal Shnitzer, Anthony Ou, Mirian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. 2023. Large language model routing with benchmark datasets. In *Annual Conference on Neural Information Processing Systems*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051.

Mingxing Tan and Quoc Le. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. 2024. Knowledge fusion of large language models. *arXiv preprint arXiv:2401.10491*.

Changhan Wang, Kyunghyun Cho, and Jiatao Gu. 2020. Neural machine translation with byte-level subwords. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 9154–9160.

Guan Wang, Sijie Cheng, Xianyuan Zhan, Xiangang Li, Sen Song, and Yang Liu. 2023a. Openchat: Advancing open-source language models with mixed-quality data. In *The Twelfth International Conference on Learning Representations*.

Hongyi Wang, Felipe Maia Polo, Yuekai Sun, Souvik Kundu, Eric Xing, and Mikhail Yurochkin. 2023b. Fusing models with complementary expertise. In *Annual Conference on Neural Information Processing Systems*.

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. 2022. Pvtv2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, 8(3):1–10.

Xin Wang, Yujia Luo, Daniel Crankshaw, Alexey Tumanov, Fisher Yu, and Joseph E Gonzalez. 2017. Idk cascades: Fast deep learning by learning not to overthink. *arXiv preprint arXiv:1706.00885*.

Yangyifan Xu, Jinliang Lu, and Jiajun Zhang. 2024. Bridging the gap between different vocabularies for llm ensemble. *arXiv preprint arXiv:2404.09492*.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. 2024. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. 2024. Yi: Open foundation models by 01. ai. *arXiv preprint arXiv:2403.04652*.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2023. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *arXiv preprint arXiv:2311.03099*.

Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A Smith. 2023. How language model hallucinations can snowball. *arXiv preprint arXiv:2305.13534*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

A Benchmarks

This paper uses the lm-evaluation-harness v0.4.1. The task names in the repo corresponding to each of the benchmarks we used are as follows:
MMLU: *mmlu_flan_n_shot_generative*, 5-shots
GSM8K: *gsm8k*, 5-shots.
BBH: *bbh_fewshot*, 3-shots.
TriviaQA: *triviaqa*, 5-shots.
NQ: *nq_open*, 5-shots.

B Hardware Specifications

Models	**Hardware**
Phi-3-mini-4k-instruct	1x A100
Llama-3-8B-Instruct	1x A100
openchat_3.5	1x A100
OpenChat-3.5-0106	1x A100
Qwen1.5-14B-Chat	1x A100
SOLAR-10.7B-Instruct-v1.0	1x A100
Nous-Hermes-2-SOLAR-10.7B	1x A100
Yi-34B-Chat	1x A100
Qwen1.5-32B-Chat	1x A100
Nous-Hermes-2-Mixtral-8x7B-DPO	2x A100
Mixtral-8x7B-Instruct-v0.1	2x A100
Llama-3-70B-Instruct	2x A100
Qwen1.5-72B-Chat	3x A100
Qwen2-72B-Instruct	3x A100

Table 7: Models and Hardware