Dynamic Data Mixing Maximizes Instruction Tuning for Mixture-of-Experts

Anonymous ACL submission

Abstract

Mixture-of-Experts (MoE) models have shown remarkable capability in instruction tuning, especially when the number of tasks scales. However, previous methods simply merge all training tasks (e.g. creative writing, coding, and mathematics) and apply fixed sampling weights, without considering the importance of different tasks as the model training state changes. In this way, the most helpful data cannot be effectively distinguished, leading to suboptimal model performance. To reduce the potential redundancies of datasets, we make the first attempt and propose a novel dynamic data mixture for MoE instruction tuning. Specifically, inspired by MoE's token routing preference, we build dataset-level representations and then capture the subtle differences among datasets. Finally, we propose to dynamically adjust the sampling weight of datasets by their inter-redundancies, thus maximizing global performance under a limited training budget. The experimental results on two MoE models demonstrate the effectiveness of our approach on both downstream knowledge & reasoning tasks and open-ended queries.

1 Introduction

Instruction tuning is a pivotal step for Large Language Model (LLM) alignment (OpenAI, 2022; Anthropic, 2023). To promote the alignment ability, LLMs are typically fine-tuned on a collection of instruction datasets with multiple tasks (Zhou et al., 2023; Mukherjee et al., 2023; Ouyang et al., 2022). However, dense models may be constrained by their fixed model capacities when the number of tasks grows in instruction tuning (Chung et al., 2022). Instead, Mixture-of-Experts (MoE) naturally incorporates multiple experts, which expands the model capacity (Shazeer et al., 2017; Lepikhin et al., 2020), and assigns relevant tokens to specific experts (Fedus et al., 2022).

To perform instruction tuning, multiple datasets are usually combined in practice (MosaicML, 2023). In such a complex scenario, datasets from diverse domains may exhibit redundancies, which requires a prudent design in the dataset selection and combination (Cao et al., 2023; Xie et al., 2023). Recently, MoE models have demonstrated appealing quality on divergent tasks and reach significantly better performance than dense models, attributed to their excellent task scaling properties (Shen et al., 2023a). However, how to decide appropriate sampling weights according to models' internal preferences is still under-explored.

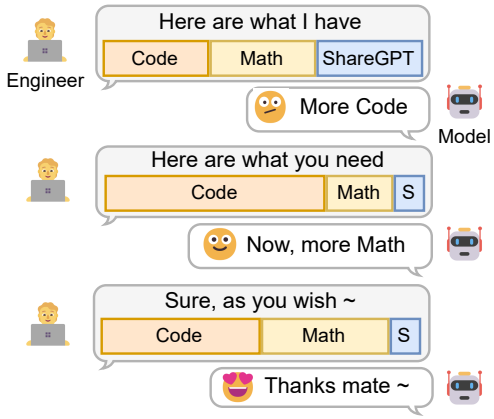


Figure 1: Our proposed dynamic data sampling method for instruction tuning. As the training progresses, the model can dynamically adjust the proportion of data sampling. For comparison, previous works concatenate datasets directly and apply fixed sampling weights.

Most previous studies (Shen et al., 2023a; OpenBMB, 2024; Wang et al., 2023) directly concatenate multiple instruction datasets for supervised fine-tuning (SFT) without considering the sampling weights and task redundancies. Jha et al. (2023) and Chen et al. (2024) take sampling weights as a hyper-parameter and find the best combination by handcraft search, which is laborious and costly to enumerate all the combinations. Thus, it is vital to automatically adjust the sampling weights during the training process with the lowest cost and maximize the alignment abilities.

To this end, we propose a dynamic sampling strategy for MoE models, as illustrated in Figure 1. Our method is based on the hypothesis that *if one dataset is different from the others for the MoE model, there may be less redundancies and the sampling weight should be increased in the next round of training*. Thus, the most important problem is how to identify the differences among datasets considering the model's training state. It is difficult to build such a meticulous dataset-level difference as the model is constantly changing. Inspired by the intrinsic properties of MoE models, we formulate the dataset-level representations resorting to specialized experts and token routing preferences (Zoph et al., 2022). Specifically, we count the number of tokens routed to every expert for each dataset, which refers to the gate load. Afterward, we apply the gate loads as dataset representations and compute L2 distances among them. Since the distances are obtained from token routing preferences, they could represent the model's internal state. Finally, we propose a dynamic algorithm to update the sampling weights according to previous sampling weights and current distances.

We experiment on two MoE models with a combination of four representative instruction datasets. Model performances are evaluated on eight evaluation datasets across knowledge testing, reasoning, and open-ended question answering tasks. The results demonstrate the effectiveness of our dynamic method. To help understand the internal mechanism of our method, we also provide thorough analyses of expert specialization and different data combinations. Our main contributions are summarized as follows:

- To our best knowledge, this is the first work to systematically study different sampling methods for MoE models in instruction tuning. Inspired by the inherent attributes of MoE, we introduce a novel dynamic data mixture for combining different instruction datasets.
- To capture the differences among datasets considering the model's training state, we propose to utilize the routing preferences of MoE models to formulate dataset-level representations.
- We conduct extensive experiments on two MoE models and validate the effectiveness of our method on a wide range of downstream tasks and open-ended questions.

2 Related Work

Mixture-of-Experts. The Mixture-of-Experts (MoE) is a sparsely activated architecture in neural networks with great efficiency (Shazeer et al., 2017; Fedus et al., 2022; Lepikhin et al., 2020). Attributed to its sparsity, MoE has attracted broad attention in the realm of Large Language Models (LLMs) (Fedus et al., 2022; Lepikhin et al., 2020). Subsequent studies follow these model architectures, showing the effectiveness of MoE in dealing with reasoning (Jiang et al., 2024), cross-domain (Li et al., 2023), and multi-modal (Mustafa et al., 2022) problems.

Instruction Tuning. The instruction tuning is an important step for the alignment of pre-trained Large Language Models (LLMs). Wang et al. (2022) devise an automatic prompting method to generate enormous instructions and responses with LLMs. Based on this idea, Xu et al. (2023) and Zhao et al. (2023) further utilize LLMs to generate diverse and complex instructions to enhance the alignment. Different from the data augmentation methods, Tunstall et al. (2023) and Zhou et al. (2023) find a small number of high quality instruction data can boost the alignment performance. Cao et al. (2023) and Liu et al. (2023) further study data patterns to filter out high quality data to help LLM alignment. However, none of these approaches consider using different sampling weights when training on multiple instruction datasets.

Dynamic Data Mixing. Since there is no relevant literature on dynamic sampling for instruction tuning, we introduce the relevant methods in LLM pre-training. Xie et al. (2023) propose DoReMi, a dynamic sampling method for LLM pre-training on multiple domains of data. However, they need to train a proxy model for estimating reference losses on target domains, which introduces additional training computations. Xia et al. (2023) propose to use a series of language models and estimate the reference loss by fitting scaling law curves. They have to fine-tune the model on target datasets to estimate the reference loss of instruction tuning by scaling law curves, which still brings additional training computation. Albalak et al. (2023) introduce an online data mixing method for LLM pre-training via the multi-armed bandit algorithm. However, the exploration stage at the beginning of training takes a huge amount of steps, which is not applicable for instruction tuning.

3 Preliminaries of Mixture-of-Experts

In a typical MoE structure, the layer is composed of N expert networks $\{E_1, E_2, \ldots, E_N\}$ and a gating network G. Different from common networks, the MoE manifests itself in the design of computational strategy, characterized by inherent sparsity. Given an input token x, the gating network computes a vector of routing scores $G(x) \in \mathbb{R}^N$, denoting the importance of each expert network to process the given input. The MoE layer then selectively aggregates the outputs from the top-K experts, which is represented as:

$$y = \sum_{i \in \mathcal{I}_K} G(x)_i \cdot E_i(x), \tag{1}$$

where $\mathcal{I}_K$ is the set of indices with the highest $K \leq N$ scores in $G(x)$, denoted as:

$$\mathcal{I}_K = \{i_1, \ldots, i_K \mid G(x)_{i_1} \geq \cdots \geq G(x)_{i_N}\}. \tag{2}$$

To maintain a balanced computational load among experts, an auxiliary balance loss is typically incorporated during the training process. Given the input dataset $\mathcal{D}_i$, a common practice (Shazeer et al., 2017) is to apply a constraint on the routing scores $G(x)$ for each token $x \in \mathcal{D}_i$, which is defined as:

$$\mathcal{L}_{\text{bal}_i} = \text{CV}(\mathcal{G}_i)^2 + \text{CV}(\mathcal{O}_i)^2, \tag{3}$$

where $\text{CV}(\cdot)$ is the function calculating the coefficient of variation from a given vector, measuring the degree of imbalance upon activation. The CV score would be high if tokens dispatched to experts are off-balance. The aggregation of these two terms ensures a balanced dispatching among experts. The importance score vector $\mathcal{G}_i \in \mathbb{R}^N$ corresponds to the summation of routing scores $\sum_{x \in \mathcal{D}_i} G(x)$. The **gate load vector** $\mathcal{O}_i = \sum_{x \in \mathcal{D}_i} \text{BinCount}(\mathcal{I}_K^{(x)}), \mathcal{O}_i \in \mathbb{R}^N$ is the count of tokens routed to each expert across the entire inputs $\mathcal{D}_i$. For all the datasets $\mathcal{D}$, we could obtain the gate loads $\mathcal{O} \in \mathbb{R}^{|\mathcal{D}| \times N}$, where $|\mathcal{D}|$ denotes the number of datasets.

4 Methodology

In this section, we introduce our dynamic sampling strategy, which automatically adjusts the sampling weights of different instruction datasets. After every m steps of model training, we obtain the gate loads $\mathcal{O}$ as dataset-level representations, then calculate the differences across datasets with $\mathcal{O}$ and update sampling weights accordingly. The dynamic sampling algorithm is presented in Alg 1.

Algorithm 1 DYNAMICSAMPLING

Input: sampling weights of last round $\mathbf{w}_{t-1} \in \mathbb{R}^{|\mathcal{D}|}$, normalized gate loads $\hat{\mathcal{O}} \in \mathbb{R}^{|\mathcal{D}| \times N}$, update step size η, smoothing value c, the number of datasets $|\mathcal{D}|$.

Output: updated sampling weights $\mathbf{w}_t$.

1: // Update L2 distances across datasets.
2: $\delta_{ij} \leftarrow ||\hat{\mathcal{O}}_i - \hat{\mathcal{O}}_j||, \quad \delta \in \mathbb{R}^{|\mathcal{D}| \times |\mathcal{D}|}$
3: // Get the average distance for each dataset.
4: $\Delta_i \leftarrow \left(\sum_j \delta_{ij}\right) / \, |\mathcal{D}|, \quad \Delta \in \mathbb{R}^{|\mathcal{D}|}$
5: // Calculate the updated sampling weights.
6: $\boldsymbol{\alpha} \leftarrow \text{softmax}\left(\log \mathbf{w}_{t-1} + \eta \Delta\right)$
7: $\mathbf{w}'_t \leftarrow (1-c)\boldsymbol{\alpha} + c \, / \, |\mathcal{D}|$
8: // Normalize sampling weights.
9: $\mathbf{w}_t \leftarrow \mathbf{w}'_t \, / \sum \mathbf{w}'_t$
10: **return** $\mathbf{w}_t$

4.1 Dataset Differences via Gate Load

As introduced in § 3, the gate load $\mathcal{O}_i \in \mathbb{R}^N$ is a vector where each element represents the number of tokens routed to that specific expert. Since experts in MoE models are well specialized, the token routing distribution can demonstrate the dataset properties. As discussed in Team (2023) and Jiang et al. (2024), deeper layers have better specializations. Therefore, we calculate the differences among instruction datasets via gate loads in the last layer for each model.

For each dataset $\mathcal{D}_i$, we record the routing tokens and calculate the corresponding gate load $\mathcal{O}_i$. To alleviate the bias, we discard all padding tokens which may overwhelm the differences across gate loads. To align the scale of gate loads of different datasets, we normalize $\mathcal{O}_i$ and obtain the final gate load vector $\hat{\mathcal{O}}_i = \mathcal{O}_i / \sum \mathcal{O}$.

After obtaining the gate loads, we calculate the L2 distance δ_{ij} of each dataset pair $\mathcal{D}_i$ and $\mathcal{D}_j$. As shown in Line 4 of Alg. 1, we further calculate the averaged distance of one dataset $\mathcal{D}_i$ to all the datasets. Overall, we obtain $\Delta \in \mathbb{R}^{|\mathcal{D}|}$, a vector that denotes the averaged distance of each dataset. We further adjust the sampling weights based on the distance vector.

4.2 Dynamic Data Sampling

Based on our hypothesis, if one dataset $\mathcal{D}_i$ is different to the others, the sampling weight of $\mathcal{D}_i$ should be increased since it may contain less redundancies with other datasets.

As presented in Line 6 from Alg. 1, we calculate the updated sampling weights by adding $\eta\Delta$ to the logarithmic weights of the last time step $\log \mathbf{w}_{t-1}$, where η is the update step size that could be regarded as a term similar to the learning rate. We follow Xie et al. (2023) and add $c/|\mathcal{D}|$ to smooth and re-normalize the values as shown in Line 7-9 in Alg. 1, where c is a hyper-parameter.

Based on the above strategy, we update the sampling weights every m steps in the training phase. Following Xia et al. (2023) and Xie et al. (2023), the initial sampling weights $\mathbf{w}_0$ is uniformly distributed to alleviate potential biases (Team, 2023).

5 Experiments

5.1 Instruction Tuning Datasets

We use the following four types of instruction datasets for supervised fine-tuning. In each dataset, we sample 20K instances for training, and 1K instances for gate load evaluation in the sampling weight adjustment.

ShareGPT.[*] Multi-turn dialogues with ChatGPT, containing a wide range of open-ended instructions.

OpenOrca.[†] Flan (Longpre et al., 2023) instructions with responses generated by GPT-4 & GPT-3.5 (Lian et al., 2023), containing multiple task-oriented instructions.

Math-Instruct.[‡] A collection of math instructions with step-by-step solutions (Yue et al., 2023).

Code Instructions.[§] LLM-generated responses with multiple languages to solve code problems.

5.2 Evaluation Datasets

We comprehensively evaluate the ability of models from both Knowledge & Reasoning and Open-Ended instruction following aspects. For knowledge & reasoning, we evaluate the models on 5-shot MMLU, 3-shot BigBench-Hard (BBH), 8-shot GSM8K (Math), MBPP (Code), and 0-shot Question Answering (QA) tasks. Here, MMLU contains 57 sub-tasks, BBH has 13 sub-tasks, and QA consists of 3 datasets (ARC-easy, ARC-challenge, and BoolQ). We report the macro-averaged score of these tasks for comparison. Besides, we also report the open-ended instruction following results on MT-Bench, which is automatically evaluated by GPT-4.

5.3 Baselines

w/o SFT. The foundation model without fine-tuning.

DataSize. Static sampling baseline. The sampling weights are determined by the original data size.

Uniform. Static sampling baseline. The model is fine-tuned with the uniformly distributed sampling weights (all datasets have the same sampling probability).

Random. A dynamic sampling baseline where sampling weights are randomly assigned at each round.

RefLoss. RefLoss is a variant of the dynamic sampling method (Xie et al., 2023), where the distance of each dataset is replaced by the loss differences between current loss and reference loss $\Delta_i \leftarrow (\mathcal{L}^i_{\text{current}} - \mathcal{L}^i_{\text{reference}})$. We use the loss evaluated by **Uniform** to estimate the reference loss $\mathcal{L}^i_{\text{reference}}$ on each dataset $\mathcal{D}_i$. Therefore, **RefLoss** consumes 2 times of training computation than the proposed dynamic method.

5.4 Implementation Details

We test our method on two MoE models: MoLM 700M-4E (activating 4 experts with 700M parameters) (Shen et al., 2023b) and LLaMA-MoE 3.5B-2E (Team, 2023). We freeze the gate parameters and train models with 2K steps under a global batch size of 128. The optimizer is AdamW (Loshchilov and Hutter, 2017) with a learning rate of 2e-5, which is warmed up with 3% steps under cosine scheduling. The maximum sequence length is 2,048 and the model is trained with gradient checkpointing (Griewank and Walther, 2000), ZeRO-1 (Rajbhandari et al., 2019), and FlashAttention-v2 (Dao, 2023) accelerations. For our proposed dynamic method in LLaMA-MoE, the evaluation interval $m = 100$, η is 10.0 and c is 5e-2. In MoLM, $m = 200$ and $c = 8e-1$. Experiments are conducted on 4×NVIDIA A100 (80G) GPUs.

5.5 Main Results

As shown in Table 1, supervised fine-tuning (SFT) is beneficial for models to enhance their overall abilities on downstream knowledge & reasoning (K&R) tasks, no matter what the sampling strategy

[*] https://huggingface.co/datasets/anon8231489123/ShareGPT_Vicuna_unfiltered

[†] https://huggingface.co/datasets/Open-Orca/OpenOrca

[‡] https://huggingface.co/datasets/TIGER-Lab/MathInstruct

[§] https://huggingface.co/datasets/iamtarun/code_instructions_120k_alpaca

Model	Knowledge & Reasoning						Open-Ended
	MMLU	BBH	Math	Code	QA	Average	MT-Bench
			MoLM 700M-4E				
w/o SFT	24.73	**27.89**	1.14	5.76	**47.52**	21.41	-
DataSize	**26.62**	23.94	**2.50**	**10.15**	43.65	21.37	2.59
Uniform	25.76	26.08	1.21	9.60	45.01	21.53	2.63
Random	25.99	25.99	1.74	9.55	45.50	21.76	2.72
RefLoss	25.67	26.52	2.05	9.80	44.86	21.78	2.69
Dynamic	25.83	26.96	1.82	10.12	45.28	**22.00**	**2.73**
			LLaMA-MoE 3.5B-2E				
w/o SFT	27.98	29.67	4.63	5.12	57.45	24.97	-
DataSize	31.44	29.46	1.67	11.84	59.96	26.87	4.81
Uniform	32.48	29.18	5.91	14.52	60.85	28.59	5.07
Random	33.39	29.43	2.73	15.80	61.17	28.50	5.00
RefLoss	**33.75**	29.02	9.63	14.48	60.87	29.55	5.18
Dynamic	33.07	**30.77**	**11.90**	**16.88**	**61.28**	**30.78**	**5.22**

Table 1: Main results. Best and the second best results are denoted in **bold** and underlined, respectively.

is. For static sampling, the performances of **DataSize** are lower than **Uniform**, both in knowledge & reasoning tasks and open-ended MT-Bench. Besides, the averaged K&R score in MoLM **DataSize** (21.37) is slightly lower than the foundation model (21.41), eliminating the advantage of MoE model's capabilities.

For dynamic sampling, the performances of **Random** are not stable. It achieves better scores than **RefLoss** in LLaMA-MoE, while it is worse in MoLM in the averaged K&R scores. **RefLoss** is a strong baseline compared to **Uniform** and boost the foundation models' performances across the K&R tasks by 0.37 (MoLM) and 4.58 (LLaMA-MoE). However, it brings additional training compute due to the reference loss estimation. Our **Dynamic** shows great potential and surpasses **RefLoss** without the additional training cost, which leads to a better and faster convergence. Additionally, **Dynamic** outperforms other baselines in the averaged K&R and MT-Bench scores, validating the effectiveness. Besides, **Dynamic** surpasses **Random**, showing the importance of considering the model's internal state and the dataset properties.

5.6 Analysis

5.6.1 Data Combinations

Q: *How do datasets contribute to the final performance?* We conduct experiments on subsets of the training datasets and present the results in Figure 2. Since math and code tasks have strong correlations with the instruction tuning dataset types, we report the GSM8K (math) and MBPP (code) results here.

As shown in the figure, Math-Instruct and Code Instructions are very task-related, and models trained solely on these datasets could reach the best GSM8K and MBPP performances, respectively. Although the single ShareGPT or OpenOrca is less powerful, it shows great performance when they are combined with Math-Instruct or Code Instruction datasets. **Dynamic** is more balanced comparing to the **Uniform** baseline, where **Dynamic** strengthens the MBPP performance on math-related combination (S+O+M), and improves the GSM8K performance on code-related combination (S+O+C). When all four types of datasets are combined for SFT, **Dynamic** improves both GSM8K and MBPP performances.

5.6.2 Expert Specialization

Q: *Does such an gate-load-based dynamic data sampling strategy hurt expert specialization?* Our method's optimization objective is to make the gate loads more similar across datasets. Although we freeze the gate parameters during training, the middle activation states may still affect the expert specialization property. We report the gate load differences and $\mathrm{CV}(\mathcal{O}_i)^2$ for each dataset to measure the expert specialization variations.

As shown in Figure 3 (abde), we find instruction tuning indeed affects the expert specialization. However, it is not determined by our gate-load-

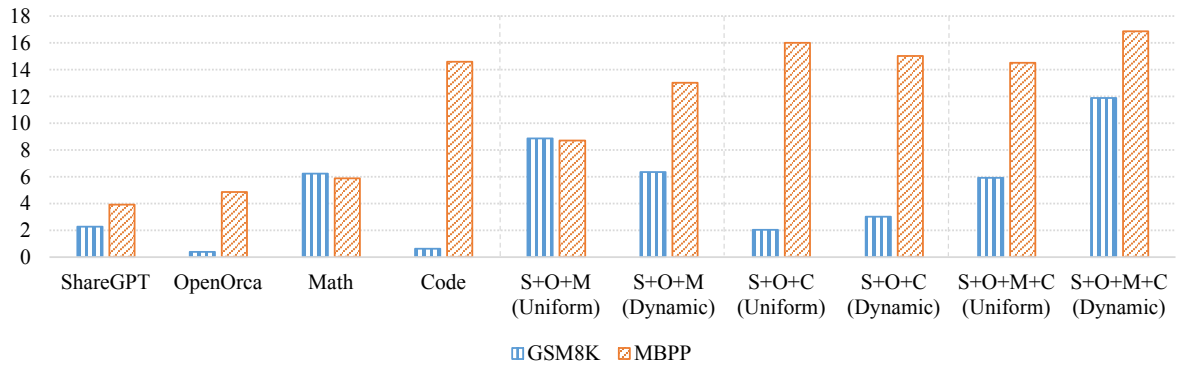
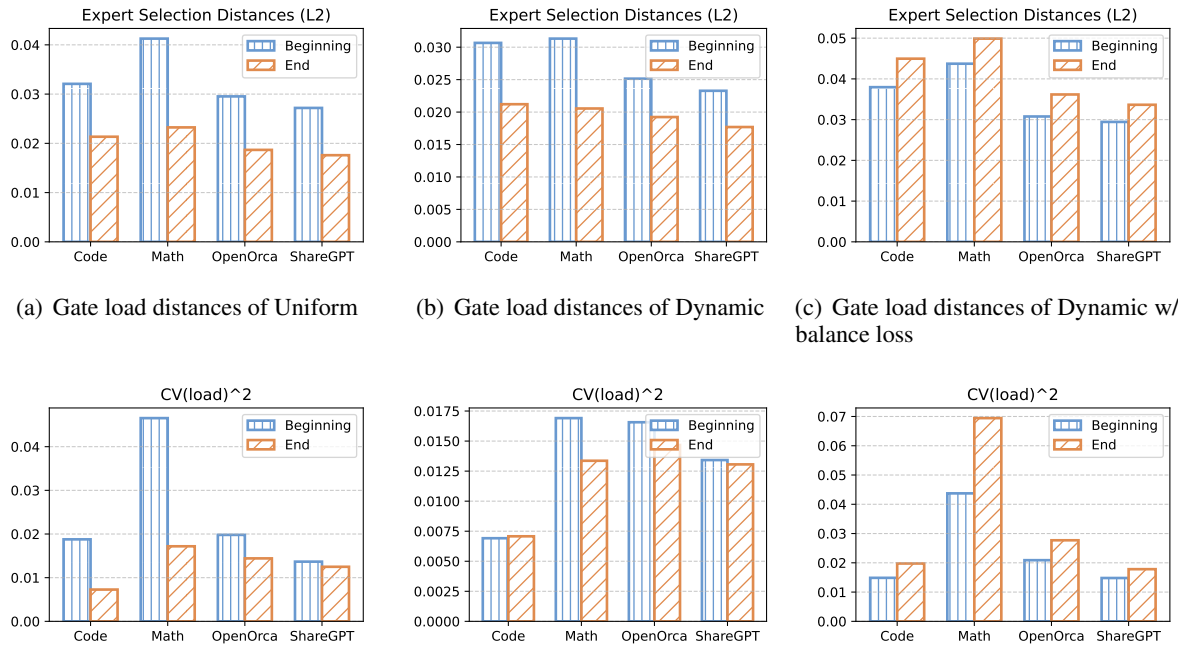


Figure 2: Results on different data combinations. LLaMA-MoE 3.5B-2E is fine-tuned for this experiment. S, O, M, and C denote for ShareGPT, OpenOrca, Math Instruct, and Code Instructions, respectively.



(a) Gate load distances of Uniform

(b) Gate load distances of Dynamic

(c) Gate load distances of Dynamic w/o balance loss

(d) $\mathrm{CV}(\mathcal{O}_i)^2$ of Uniform

(e) $\mathrm{CV}(\mathcal{O}_i)^2$ of Dynamic

(f) $\mathrm{CV}(\mathcal{O}_i)^2$ of Dynamic w/o balance loss

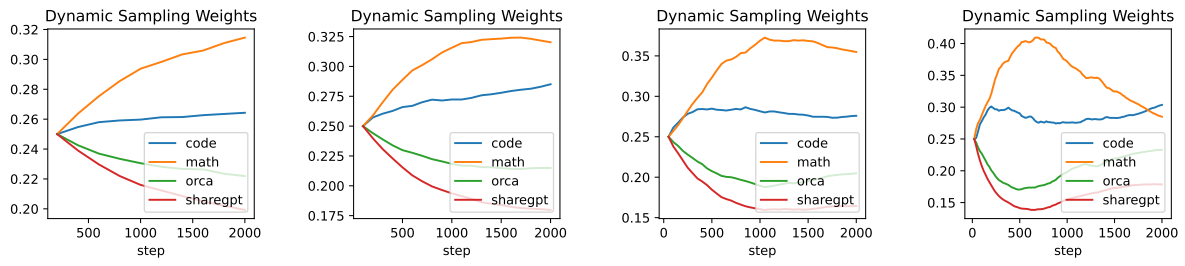
Figure 3: Gate load differences of LLaMA-MoE 3.5B-2E under different training settings. If the experts are less specialized after training, the distances and the $\mathrm{CV}(\mathcal{O}_i)^2$ would go down. For Dynamic and Dynamic w/o balance loss, the "Beginning" stands for the first round of evaluation for easier recording.

based distance calculation and dynamic sampling adjustment. Instead, it is due to the auxiliary balance loss as demonstrated in Figure 3 (cf). If we remove the balance loss during training, it would lead to more specialized experts, but the performance would be lower according to Table 4.

5.6.3 Evaluation Interval

Q: *How does the evaluation interval affect the performance?* Our dynamic sampling weights strategy is applied every m training steps. Here we investigate the effect of the evaluation intervals by conducting experiments with different m values.

As shown in Figure 4, the evaluation interval is crucial to the sampling weights update and may vary a lot with different m values. When $m = 200$, the sampling weights do not converge and monotonically go up or down. However, when $m = 20$, there are more sampling weights adjustments, leading to training instability as the differences in gate loads may have reversals. Comparing to the convergence status in Figure 4 and results in Table 2, we take $m = 100$ as the best practice.



(a) $m = 200$ (b) $m = 100$ (c) $m = 50$ (d) $m = 20$

Figure 4: Performances with different evaluation intervals. Experiments are conducted on LLaMA-MoE 3.5B-2E.

Evaluation Interval	BBH	Math
200	29.21	8.19
100	**30.77**	**11.90**
50	29.04	7.58
20	28.98	5.99

Table 2: Performance on downstream tasks with different evaluation intervals. Experiments are conducted on LLaMA-MoE 3.5B-2E.

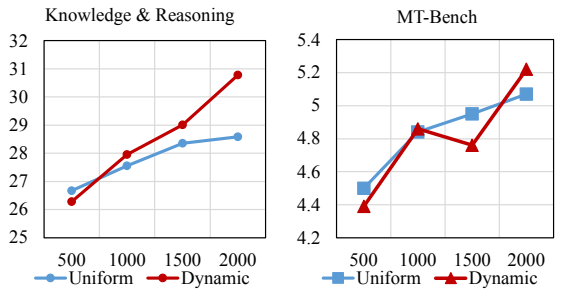


Figure 5: Performances with different training steps. Experiments are conducted on LLaMA-MoE 3.5B-2E.

5.6.4 Learning Efficiency

Q: *How does the number of training steps affect the results?* We change the number of training steps and freeze the other hyper-parameters to observe the trend of performance variation.

From Figure 5, both **Uniform** and **Dynamic** benefits from more training steps, and they consistently improve the performance on knowledge and reasoning tasks. Even 500 steps can make the fine-tuned model outperforms the foundation model (Uniform 26.67 & Dynamic 26.28 vs. w/o SFT 24.97). As the number of training steps grows, **Uniform** seems to reach its performance ceiling, and the gap between these two methods further increases. As to the open-ended performance on MT-Bench, the **Dynamic** method has more fluctuations, but it could outperforms the **Uniform** baseline as more training steps are applied.

5.6.5 Other Sampling Weights

Q: *What if we use the final sampling weights obtained from the proposed **Dynamic** to train the model again?* To find whether the final sampling weights of **Dynamic** provide a good data combination for an MoE model, we conduct the experiments on LLaMA-MoE.

As presented in Table 3, **FinalStatic** is better than **Uniform** and **DataSize** in both K&R tasks and MT-Bench. Surprisingly, compared to the results in Table 1, **FinalStatic** (29.68) is even better than **RefLoss** (29.55) in the averaged K&R score. This indicates that our dynamic method could help find better sampling weights than simple concatenation even on static sampling. In addition, **FinalStatic** is still worse than **Dynamic**, which verifies the model's internal state changing. Thus, dynamic sampling could reach a better performance than static sampling.

Q: *Similar datasets are redundant, how does this assumption hold? What if we use sentence embedding to compute the dataset differences instead of gate loads?* The essence of the distance calculation is to find similarities and differences between each datasets, so why do we need to compute the distance scores from gate load vectors rather than direct sentence embeddings? We conduct this experiment by utilizing SentenceTransformers (Reimers and Gurevych, 2019) to replace the input gate loads $\mathcal{O}$ in Alg. 1 and compute L2 distances afterwards.

As shown in Table 3, **SentEmb** outperforms **Uniform** across the tasks, which indicates the effectiveness of dataset re-weighting by their inter similarities. The averaged **GateLoad** performance

Model	Knowledge & Reasoning						Open-Ended
	MMLU	BBH	Math	Code	QA	Average	MT-Bench
w/o SFT	27.98	29.67	4.63	5.12	57.45	24.97	-
Static Sampling							
DataSize	31.44	29.46	1.67	11.84	59.96	26.87	4.81
Uniform	32.48	29.18	5.91	14.52	60.85	28.59	5.07
FinalStatic	32.84	30.11	9.93	14.61	60.93	29.68	5.11
Static Distances							
SentEmb	33.85	29.70	7.66	16.29	61.75	29.85	5.21
GateLoad	32.75	29.98	6.60	14.07	61.78	29.04	4.98
Initial Sampling Weights							
$\text{Dynamic}_{\text{SentEmb}}$	33.46	29.02	8.95	15.68	61.03	29.63	5.16
$\text{Dynamic}_{\text{Uniform}}$	33.07	30.77	11.90	16.88	61.28	30.78	5.22

Table 3: Other sampling weights. Experiments are conducted on LLaMA-MoE 3.5B-2E.

is lower than **SentEmb** in both the averaged knowledge & reasoning tasks and the open-ended MT-Bench. Nevertheless, **SentEmb** could not be easily applied to make constant improvements in the whole training phase. Although **GateLoad** is worse than **SentEmb**, the model benefits from the iterative sampling weights adjustments, and **Dynamic** surpasses **SentEmb** in both K&R and open-ended performances.

Q: *What about other initial sampling weights rather than the uniform distribution?* Since **SentEmb** has better performance than **Uniform** and **GateLoad**, we wonder if it is better to apply its sampling weights as the initial ones rather than the uniform distribution.

The results in Table 3 show that the uniform initialized $\textbf{Dynamic}_{\textbf{Uniform}}$ outperforms $\textbf{Dynamic}_{\textbf{SentEmb}}$ (30.78 vs. 29.63 in K&R, 5.22 vs. 5.16 in MT-Bench), which is in line with the conclusions in Team (2023). We conjecture that the imbalanced initial weights would make the model hard to convergence.

5.6.6 Ablation Study

There are differences between sparse MoE models and dense models during training due to their specific techniques. Here we investigate the effectiveness of fronzen gate, balance loss, and gate noise for instruction tuning on MoE.

The results are presented in Table 4. Similar to Shen et al. (2023a), we find the frozen gate, balance loss, and gate noise have all positive effects to the model performances. Frozen gate is to freeze the gate parameters when fine-tuning. This leads to better performance as the gate is well trained during the pre-training stage, and SFT may break the specialized token routing property. Balance loss and gate noise are beneficial to model training since they are in line with the pre-training objectives.

Model	Avg. K&R	MT-Bench
LLaMA-MoE	30.78	5.22
w/o frozen gate	28.78	4.91
w/o balance loss	29.38	4.88
w/o gate noise	30.04	4.98

Table 4: Ablation study. Avg. K&R stands for the averaged score of knowledge & reasoning tasks (MMLU, BBH, Math, and Code).

6 Conclusion

To combine different datasets and maximize the MoE model's alignment ability, we assign different sampling weights to corresponding datasets. By incorporating the internal model state and the dataset properties, we propose to use the gate load from MoE models to obtain dataset representations. Based on the representations, we calculate distances between each pair of datasets, indicating the inter-redundancies. We further devise an automatic algorithm to dynamically update the sampling weights. The proposed method outperforms other baselines and demonstrate good performance on knowledge & reasoning tasks and open-ended question answering.

Limitation

More Models. Due to the limit computing resources, we only test the method's effectiveness on two representative decoder-style MoE models. Dynamic sampling on larger models like Mixtral (Jiang et al., 2024) is currently not verified.

Number of Datasets. For a combination of two datasets, there are no differences between the distance vector Δ, so the dynamic sampling method does not take into effect and the sampling weights would stay unchanged.

Final Static Weights on Other Models. Since the proposed dynamic sampling method is strongly combined with the MoE model's internal state, each MoE model may result in different sampling weights. The obtained final static weights may be less useful to extrapolate to other models.

References

Alon Albalak, Liangming Pan, Colin Raffel, and William Yang Wang. 2023. Efficient online data mixing for language model pre-training. *ArXiv*, abs/2312.02406.

Anthropic. 2023. Introducing Claude.

Yihan Cao, Yanbin Kang, and Lichao Sun. 2023. Instruction mining: High-quality instruction data selection for large language models. *ArXiv*, abs/2307.06290.

Shaoxiang Chen, Zequn Jie, and Lin Ma. 2024. Llava-mole: Sparse mixture of lora experts for mitigating data conflicts in instruction finetuning mllms. *ArXiv*, abs/2401.16160.

Hyung Won Chung, Le Hou, S. Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Wei Yu, Vincent Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed Huai hsin Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *ArXiv*, abs/2210.11416.

Tri Dao. 2023. FlashAttention-2: Faster attention with better parallelism and work partitioning.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1):5232–5270.

Andreas Griewank and Andrea Walther. 2000. Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *ACM Trans. Math. Softw.*, 26:19–45.

Aditi Jha, Sam Havens, Jeremey Dohmann, Alex Trott, and Jacob Portes. 2023. Limit: Less is more for instruction tuning across evaluation paradigms. *ArXiv*, abs/2311.13133.

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, Gianna Lengyel, Guillaume Bour, Guillaume Lample, L'elio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts. *ArXiv*, abs/2401.04088.

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

Bo Li, Yifei Shen, Jingkang Yang, Yezhen Wang, Jiawei Ren, Tong Che, Jun Zhang, and Ziwei Liu. 2023. Sparse mixture-of-experts are domain generalizable learners. In *International Conference on Learning Representations*.

Wing Lian, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and "Teknium". 2023. Openorca: An open dataset of gpt augmented flan reasoning traces. `https://https://huggingface.co/Open-Orca/OpenOrca`.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. 2023. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. *ArXiv*, abs/2312.15685.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: Designing data and methods for effective instruction tuning.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

MosaicML. 2023. Introducing mpt-7b: A new standard for open-source, commercially usable llms.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4.

Basil Mustafa, Carlos Riquelme, Joan Puigcerver, Rodolphe Jenatton, and Neil Houlsby. 2022. Multimodal contrastive learning with limoe: the language-image mixture of experts. *ArXiv*, abs/2206.02770.

OpenAI. 2022. Introducing ChatGPT.

OpenBMB. 2024. Minicpm: Unveiling the potential of end-side large language models.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. 2022. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2019. Zero: Memory optimizations toward training trillion parameter models. *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Sheng Shen, Le Hou, Yan-Quan Zhou, Nan Du, S. Longpre, Jason Wei, Hyung Won Chung, Barret Zoph, William Fedus, Xinyun Chen, Tu Vu, Yuexin Wu, Wuyang Chen, Albert Webson, Yunxuan Li, Vincent Zhao, Hongkun Yu, Kurt Keutzer, Trevor Darrell, and Denny Zhou. 2023a. Mixture-of-experts meets instruction tuning:a winning combination for large language models.

Yikang Shen, Zheyu Zhang, Tianyou Cao, Shawn Tan, Zhenfang Chen, and Chuang Gan. 2023b. Moduleformer: Learning modular large language models from uncurated data. *arXiv preprint arXiv:2306.04640*.

LLaMA-MoE Team. 2023. Llama-moe: Building mixture-of-experts from llama with continual pre-training.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. 2023. Zephyr: Direct distillation of lm alignment. *ArXiv*, abs/2310.16944.

Rongsheng Wang, Hao Chen, Ruizhe Zhou, Yaofei Duan, Kunyan Cai, Han Ma, Jiaxi Cui, Jian Li, Patrick Cheong-Iao Pang, Yapeng Wang, and Tao Tan. 2023. Aurora: Activating chinese chat capability for mixtral-8x7b sparse mixture-of-experts through instruction-tuning. *ArXiv*, abs/2312.14557.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. In *Annual Meeting of the Association for Computational Linguistics*.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. *ArXiv*, abs/2310.06694.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. 2023. Doremi: Optimizing data mixtures speeds up language model pretraining. *ArXiv*, abs/2305.10429.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *ArXiv*, abs/2304.12244.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*.

Ying Zhao, Yu Bowen, Binyuan Hui, Haiyang Yu, Fei Huang, Yongbin Li, and Nevin Lianwen Zhang. 2023. A preliminary study of the intrinsic relationship between complexity and alignment. *ArXiv*, abs/2308.05696.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, L. Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. Lima: Less is more for alignment. *ArXiv*, abs/2305.11206.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*.

A Appendix

Here we provide the final sampling weights of the proposed **Dynamic** method across MoE models in Table 5. Table 6 shows the detailed multi-turn results on MT-Bench. For better comparison the **Dynamic** effect on different tasks, we provide the detailed results on BBH subtasks in Table 7.

Model	ShareGPT	OpenOrca	Math-Instruct	Code Instructions
MoLM 700M-4E	28.41	23.51	23.45	24.63
LLaMA-MoE 3.5B-2E	17.98	21.49	32.02	28.51

Table 5: Final sampling weights of **Dynamic** (%). The summation may not equal to exact 100% due to digit rounding. We find the final static weights of different models have many variations. MoLM prefers to accept more ShareGPT, while LLaMA-MoE samples more Math-Instruct.

Rounds	MoLM			LLaMA-MoE		
	DataSize	Uniform	Dynamic	DataSize	Uniform	Dynamic
1st	2.81	2.98	3.10	5.52	5.78	5.96
2nd	2.36	2.28	2.36	4.10	4.36	4.48
Overall	2.59	2.63	2.73	4.81	5.07	5.22

Table 6: Detailed results on MT-Bench. Each question in MT-Bench has two turns of responses. Here we list the results of each turn.

Rounds	MoLM			LLaMA-MoE		
	DataSize	Uniform	Dynamic	DataSize	Uniform	Dynamic
Boolean Expressions	53.20	54.40	55.20	49.20	47.20	46.80
Causal Judgement	36.90	52.94	51.87	52.94	52.41	50.80
Date Understanding	20.80	18.40	19.20	24.40	29.60	36.80
Disambiguation Qa	38.00	38.80	38.80	30.80	31.60	28.00
Dyck Languages	9.20	13.60	15.20	18.40	10.80	15.60
Formal Fallacies	37.60	39.60	21.60	49.20	53.20	52.40
Geometric Shapes	12.00	9.60	10.40	9.60	9.60	22.40
Hyperbaton	48.40	48.40	48.40	51.60	45.60	43.60
Logical Deduction Five Objects	8.40	21.20	22.80	18.40	22.80	20.00
Logical Deduction Seven Objects	10.00	17.20	14.40	15.60	15.60	14.40
Logical Deduction Three Objects	34.00	33.60	34.40	39.20	36.40	38.00
Movie Recommendation	14.80	22.40	19.60	41.60	22.40	26.00
Multistep Arithmetic Two	0.00	0.00	0.00	0.80	1.20	1.20
Navigate	32.40	42.40	46.40	50.80	56.40	50.80
Object Counting	14.80	16.80	13.20	33.20	33.60	38.40
Penguins In A Table	10.27	10.27	22.60	20.55	21.23	26.03
Reasoning About Colored Objects	1.60	7.60	13.20	7.60	14.00	21.60
Ruin Names	20.80	11.60	10.80	21.20	18.00	20.00
Salient Translation Error Detection	20.80	11.60	18.00	22.40	22.40	22.40
Snarks	48.31	51.69	52.25	55.62	46.63	60.67
Sports Understanding	46.00	54.00	54.40	56.00	58.40	57.60
Temporal Sequences	27.60	21.20	25.20	11.60	10.80	12.80
Tracking Shuffled Objects Five Objects	6.80	8.40	18.40	13.60	20.00	16.40
Tracking Shuffled Objects Seven Objects	7.20	14.00	14.00	12.80	15.20	14.80
Tracking Shuffled Objects Three Objects	33.20	32.80	36.00	33.60	33.60	32.00
Web Of Lies	51.20	50.40	49.60	49.60	51.60	53.60
Word Sorting	2.00	1.20	2.00	5.20	7.60	7.60
Average	23.94	26.08	26.96	29.46	29.18	30.77

Table 7: Detailed results on different subtasks of BBH.