MDIT: A Model-free Data Interpolation Method for Diverse Instruction Tuning

Yangning Li[†] **Zihua Lan**[†] **Zihua Lan** **Lv Qingsong**
Yinghui Li **Hai-Tao Zheng**[‡]
Tsinghua University

Abstract

As Large Language Models (LLMs) are increasingly applied across various tasks, instruction tuning has emerged as a critical method for enhancing model performance. However, current data management strategies face substantial challenges in generating diverse and comprehensive data, restricting further improvements in model performance. To address this gap, we propose MDIT, a novel model-free data interpolation method for diverse instruction tuning, which generates varied and high-quality instruction data by performing task interpolation. Moreover, it contains diversity-based clustering strategies to ensure the diversity of the training data. Extensive experiments[1] show that our method achieves superior performance in multiple benchmark tasks. The LLMs fine-tuned with MDIT show significant improvements in numerous tasks such as general question answering, math reasoning, and code generation. MDIT offers an efficient and automatic data synthetic method, generating diverse instruction data without depending on external resources while expanding the application potential of LLMs in complex environments.

1 Introduction

Instruction tuning has enabled large language models (LLMs) to accurately follow human instructions and significantly enhance their performance (Longpre et al., 2023; Zhang et al., 2023; Yi et al., 2024). The diversity of instruction datasets plays an essential role in improving LLM's ability to handle various scenarios (Muscato et al., 2024; Fan et al., 2025). Therefore, recent research focuses on curating high-diversity and wide-ranging instruction datasets (Mukherjee et al., 2023; Chung et al., 2024).

In recent years, numerous studies attempt to increase the diversity of instruction datasets by filtering out simpler and less varied data (Liu et al., 2024; Pan et al., 2024; Tan et al., 2024). However, data selection methods primarily focus on removing low-diversity data and addressing the negative effects of overly simplistic data, but fail to expand the diversity of the original dataset fundamentally.

To overcome the limitations of data selection methods, researchers turn to data synthesis (Xu et al., 2024; Zhao et al., 2024; Chen et al., 2024b), generating diverse instruction data to improve the capacity of LLM for handling complex tasks. For example, Self-Instruct (Wang et al., 2022) uses some human-annotated examples to prompt the model into creating more varied datasets, while UltraChat (Ding et al., 2023) iteratively refines multi-turn dialogues through systematically designed prompts. However, data synthesis heavily depends on external models and extensive human annotation, leading to high labor costs and inconsistent annotation quality.

It leads to a critical question naturally: how to effectively expand the diversity of instruction data and enhance the performance of LLM without relying on additional external models?

To address this challenge, Mixup (Zhang et al., 2018), originally proposed in the computer vision domain, provides a promising approach by linearly blending images and their corresponding labels to improve model robustness and generalization. However, directly applying Mixup to instruction tuning in LLMs is tough due to the fundamental differences between structured image-label pairs and complex, natural language instructions. Traditional Mixup methods primarily perform simple linear interpolations within the same task, which does not naturally extend to the diverse, multi-faceted nature of instruction datasets.

To this end, we propose **MDIT**, a **M**odel-free **D**ata **I**nterpolation method for **D**iverse **I**nstruction **T**uning. Concretely, we (1) apply interpolation on different tasks at the embedding layer to generate more diverse tasks and (2) use clustering to

[1]The code will be open source upon acceptance.

filter out low-diversity data. To achieve this, we first transform samples into hidden states within the model, then perform linear interpolation on the embeddings to create new tasks, thereby fundamentally enhancing data diversity. Next, a clustering step ensures the overall diversity of the training data without relying on additional resources.

The key innovation of our MDIT over existing data synthesis methods is its labor-free nature, as it avoids the need for external resources to minimize costs. By avoiding reliance on pretrained models or manual annotations, our method reduces potential errors and ensures robust and diverse data fusion.

We conduct comprehensive experiments on several benchmarks including ARC Challenge, MMLU-Math, Humaneval, and MBPP, showing that our MDIT significantly enhances LLM performance. Furthermore, it outperforms SOTA data selection and synthesis methods by generating more diverse tasks while discarding external resources.

The key contributions of this paper as follows:

- We analyze current instruction data management strategies systematically, revealing that data selection methods fail to expand diversity basically, while data synthesis methods often rely on additional resources.
- We propose MDIT, a model-free data interpolation method that generates diverse tasks without external resources, improving the overall performance of LLM.
- Extensive experiments across multiple benchmarks show the effectiveness of MDIT, achieving superior results without the need for additional resources.

2 Related Work

2.1 Instruction Data Management for Diversity

Recent research on managing instruction data diversity can be classified into filter-based data selection and generation-based data synthesis methods.

2.1.1 Instruction Data Selection

Data selection methods aim to filter out and remove low-diversity instruction data, including metric-based and model-based methods (Chen et al., 2023b; Qiu et al., 2024). Metric-based selections (Gonen et al., 2022; Zhou et al., 2023; Zeng et al., 2025) use quantitative metrics to identify diverse instruction data. Instruction mining (Cao et al., 2023) uses a linear equation to evaluate instruction quality, while InstructionGPT-4 (Wei et al., 2023) further filters multimodal instruction data (Yu et al., 2024; Chen et al., 2024a) according to CLIP scores (Radford et al., 2021) and instruction length. Model-based selections (Wu et al., 2023; Chen et al., 2023a; Yu et al., 2023; Ge et al., 2024) leverage LLMs as data selectors to identify more diverse instructions (Li et al., 2024a; Liu et al., 2024). INSTAG (Lu et al., 2023) utilizes ChatGPT to annotate instruction data. Active Instruction Tuning (Kung et al., 2023) filters tasks based on prompt uncertainty, while Nuggets (Li et al., 2024b) employs two-stage scoring to select diverse data. However, these data selection methods focus on filtering out low-diversity data and fall short of fundamentally enriching the instruction dataset by adding novel instructions.

2.1.2 Instruction Data Synthesis

Data synthesis methods aim to generate diverse instruction data and improve the robustness of LLMs. Some work leverages the generative capabilities of LLMs to create new instructions (Taori et al., 2023; He et al., 2024; Kou et al., 2024), utilizing semantic parsing (Zhao et al., 2024), transforming simple queries into complex tasks (Xu et al., 2024) and blending model outputs with human-written content (Chen et al., 2024b), effectively enhancing dataset diversity and quality. However, these data synthesis methods typically depend on powerful external models or extensive human annotation, leading to high computational costs and potential data leakage risks. Different from them, our MDIT is entirely labor-free, generating diverse tasks without external resources. By incorporating diversity-based clustering, we further ensure the variety of the training data.

2.2 Mixup Methods in Computer Vision

To enhance data diversity, (Zhang et al., 2018) introduced Mixup for computer vision, which creates new training samples by linearly interpolating pairs of input images and their corresponding labels. Numerous Mixup variants (Yun et al., 2019; Qin et al., 2020; Kim et al., 2020; Chen et al., 2022; Wang et al., 2024; Sun et al., 2024; Islam et al., 2024) show improvements in tasks such as image classification and object detection, highlighting their effectiveness in enhancing data diversity and model robustness. However, these Mixup methods primarily focus on blending samples from similar cate-

gories within the same task. Significantly different from them, our MDIT performs interpolation across multiple tasks, generating more diverse and dynamic training data and enhancing LLM's ability to handle complex challenges.

3 Methodology

We present the framework of **MDIT** in Figure 1. Our method aims to select diverse training data for instruction tuning, consisting of two core phases: embedding-based task synthesis with interpolation (§ 3.2) and diversity-based data selection with clustering (§ 3.3). For the initial phase, we apply embedding interpolation across different tasks to create varied training tasks. Then we combine original and generated tasks and utilize clustering to ensure training data diversity. Finally, selected embeddings are directly used for LLM training, improving LLM's performance and robustness.

3.1 Preliminaries

Mixup (Zhang et al., 2018) is a data augmentation technique originally developed for computer vision, designed to enhance model's generalization capabilities. The core idea of Mixup is to perform linear combinations on both input data and labels during the training process. By linearly combining two distinct training samples and their corresponding labels in specific proportions, new mixed samples are generated, which increases training data diversity. Mixup enables the model to encounter various intermediate states within the data space during training, rather than relying solely on the original data. Specifically, given input data samples x_i and x_j , along with their corresponding labels y_i and y_j , Mixup creates new training samples x_{new} and labels y_{new} by performing a linear combination of the two pairs (x_i, y_i) and (x_j, y_j)

$$x_{\text{new}} = \lambda x_i + (1-\lambda)x_j \tag{1}$$

$$y_{\text{new}} = \lambda y_i + (1-\lambda)y_j \tag{2}$$

where λ is randomly sampled from the beta distribution. The generated samples are then used to train the neural network, which learns a more comprehensive range of data distribution characteristics through data augmentation, thereby improving its performance on unseen data.

3.2 Embedding-Based Interpolation

In this section, we detail our task-level interpolation method for generating diverse training data, aiming to expand the task distribution and improve the generalization ability of LLM, as summarized in Algorithm 1. MDIT performs interpolation between different tasks in a high-dimensional embedding space to create new tasks. These new tasks are generated by blending knowledge from multiple task distributions.

We define the training sets $\mathcal{D}_i$ and $\mathcal{D}_j$ for task i and task j as $\mathcal{D}_i = (\mathbf{X}_i, \mathbf{Y}_i) = \{(x_{i,k}, y_{i,k})\}_{k=1}^{N_i}$ and $\mathcal{D}_j = (\mathbf{X}_j, \mathbf{Y}_j) = \{(x_{j,k}, y_{j,k})\}_{k=1}^{N_j}$. Concretely, a LLM f_θ consists of $\mathcal{L}$ layers, and the hidden representation of samples $x_{i,k}$ at the embedding layer is denoted as $\mathbf{H}^{\mathbf{e}}_{\mathbf{i,k}} = f_\theta(x_{i,k})$. The samples from task i and task j are mapped into the high-dimensional embedding space through the model's embedding layer, while their corresponding labels are encoded as one-hot vectors, getting $\mathcal{D}_i^e = (\mathbf{H}_i^e, \mathbf{Y}_i)$ and $\mathcal{D}_j^e = (\mathbf{H}_j^e, \mathbf{Y}_j)$.

Next, we apply the task interpolation separately in the high-dimensional embedding space. First, an interpolation weight $\lambda \sim \text{Beta}(\alpha, \alpha)$ is randomly sampled from a Beta distribution with hyperparameter α that controls the concentration of the distribution, the probability density function as follows:

$$f(\lambda; \alpha, \alpha) = \frac{\Gamma(2\alpha)}{\Gamma(\alpha)\Gamma(\alpha)} \lambda^{\alpha-1}(1-\lambda)^{\alpha-1} \tag{3}$$

Then, we apply task interpolation to the hidden representations of two samples from different tasks and their corresponding labels $(\mathbf{H}_{i,k}^e, \mathbf{Y}_{i,k})$ and $(\mathbf{H}_{j,k}^e, \mathbf{Y}_{j,k})$ to generate new tasks as:

$$\mathbf{H}_{cr,k}^e = \lambda \cdot \mathbf{H}_{i,k}^e + (1-\lambda) \cdot \mathbf{H}_{j,k}^e \tag{4}$$

$$\mathbf{Y}_{cr,k}^n = \lambda \cdot \mathbf{Y}_{i,k}^n + (1-\lambda) \cdot \mathbf{Y}_{j,k}^n \tag{5}$$

where the superscript "e" means "interpolation in the embedding space" while the subscript "cr" indicates "cross". $\mathbf{H}_{i,k}^e$ represents the hidden representations of the k-th sample in the i-th task, while $\mathbf{Y}_{i,k}^n$ represents the label of the k-th sample in the i-th task, where each label length is n. The generated dataset retains the semantic information from the original dataset while incorporating randomness through the interpolation weights, enhancing the semantic diversity of the dataset. By applying the interpolation operation across multiple tasks, we generate a diverse set of tasks, which can be formally defined as:

$$\mathcal{D}_{\text{cr}}^e = \left\{ \left(\mathbf{H}_{\text{cr,(k)}}^e, \mathbf{Y}_{\text{cr,(k)}} \right) \mid 1 \le k \le N_{\text{cr}} \right\} \tag{6}$$

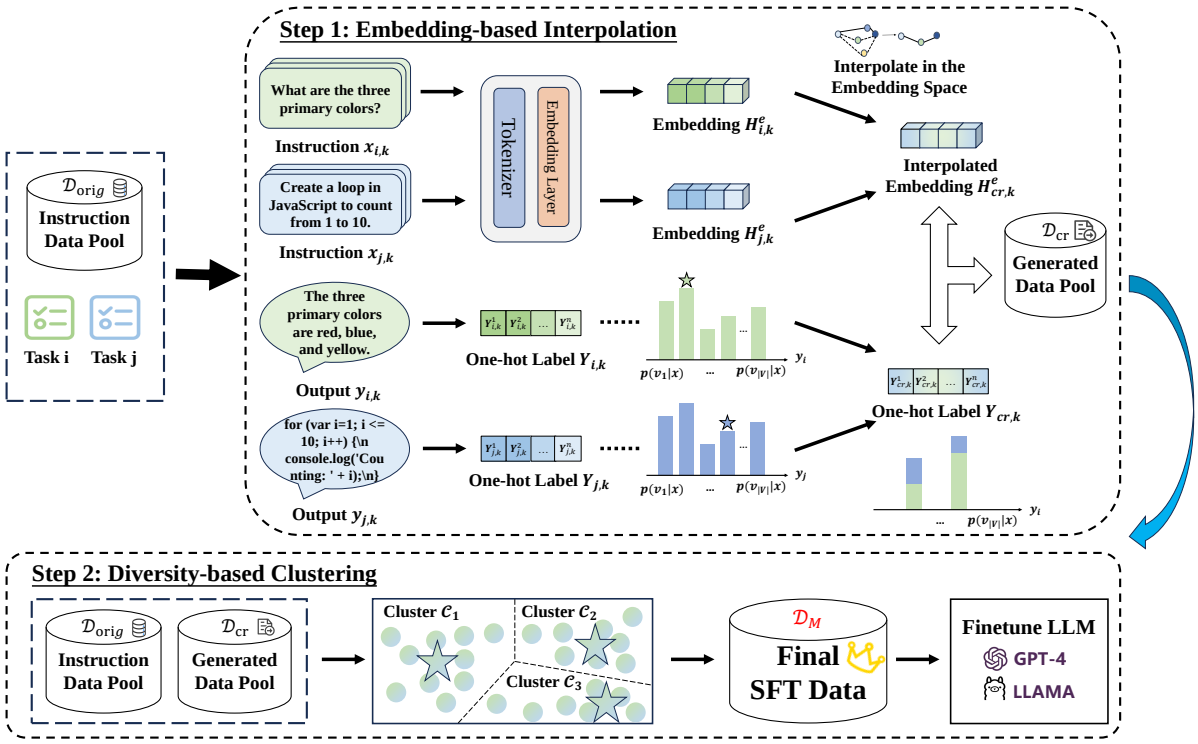


Figure 1: The framework of our method MDIT. MDIT consists of two primary steps: Embedding-based Interpolation and Diversity-based Clustering: In the first step, we perform task interpolation within the high-dimensional embedding space, generating new tasks that capture diverse semantic relationships. The second step involves clustering filtering to the curated set and selecting diverse training data from each cluster for instruction tuning.

where N_{cr} is the number of samples in the generated tasks, and $\mathbf{H}^e_{\text{cr,(k)}}$ represents the hidden representations of the generated samples while $\mathbf{Y}_{\text{cr,(k)}}$ represents the corresponding labels.

Through task-level interpolation, we effectively expand the task distribution, introducing a wider variety of tasks into the training dataset, while improving the robustness and flexibility of LLMs.

3.3 Diversity-Based Clustering

After generating new tasks through embedding-based interpolation, it is essential to implement effective filtering strategies to eliminate low-diversity data from new tasks. Data selection ensures a diverse and high-quality training dataset, providing an optimal foundation for finetuning LLMs.

Concretely, we apply a clustering selection to ensure training dataset diversity. First, we combine the original dataset $\mathcal{D}_{\text{orig}}$ and the generated dataset $\mathcal{D}^e_{\text{cr}}$ to form a total dataset $\mathcal{D}^e_{\text{total}} = \mathcal{D}^e_{\text{orig}} \cup \mathcal{D}^e_{\text{cr}}$. The combined dataset $\mathcal{D}^e_{\text{total}}$ provides a comprehensive pool for clustering.

Then, the K-Means algorithm partitions $\mathcal{D}^e_{\text{total}}$ into m clusters, optimizing the division by minimizing the sum of squared Euclidean distances d between each data point and its corresponding cluster center c_m. The set of clusters $\mathcal{C}$ and the objective function f are defined as follows:

$$\mathcal{C} = KMeans(\mathcal{D}^e_{\text{total}}, m) \tag{7}$$

$$f(\{c_m\}; \mathcal{C}) = \min_{\{c_m\}} \sum_{m=1}^{N_{\mathcal{C}}} \sum_{\mathbf{H}^e_{\text{cr, k}} \in \mathcal{C}_m} \|\mathbf{H}^e_{\text{cr, k}} - c_m\|_2^2 \tag{8}$$

where c_m represents the center of the m-th cluster, $\mathcal{C}_m$ is the set of data belonging to the m-th cluster, and $\mathbf{H}^e_{\text{cr, k}}$ is the hidden representation of the k-th sample from the interpolation task. The dataset $\mathcal{D}^e_{\text{total}}$ is divided into distinct clusters $\mathcal{C}$ based on clustering results. After clustering, we compute the Euclidean distance d from each data point $\mathbf{H}^e_{\text{cr, k}}$ to its respective cluster center c_m, defined as $d(\mathbf{H}^e_{\text{cr, k}}, c_m) = \|\mathbf{H}^e_{\text{cr, k}} - c_m\|_2$. We select data points that $d(\mathbf{H}^e_{\text{cr, k}}, c_m)$ is minimized, focusing on those closer to the cluster centers $\mathcal{C}$ or within densely populated regions of the cluster.

Finally, the total dataset $\mathcal{D}^e_{\text{total}}$ is selected to a new dataset $\mathcal{D}^e_{\text{M}}$, which enhances the coverage and informational value of the training data. The filtered dataset $\mathcal{D}^e_{\text{M}} = (\mathbf{H}^e_{\text{M}}, \mathbf{Y}_{\text{M}})$ serves as the final

Algorithm 1: Embedding-Based Interpolation

Input: Training tasks $\mathcal{D}_i$ and $\mathcal{D}_j$, Interpolation weight α
Output: Augmented task $\mathcal{D}_{\text{cr}}$

Initialize an empty dataset $\mathcal{D}_{\text{cr}}$;
foreach $x \in \mathcal{D}_i, \mathcal{D}_j$ **do**
  Tokenize the instruction x into tokens: $\mathcal{T}_x$;
  Embed the tokens into high-dimensional embeddings: $\mathbf{H}^e$;
  One-Hot Encode the label $y \in \mathcal{D}_i, \mathcal{D}_j$: $\mathbf{Y}$;
  Group samples based on similar input lengths;
  foreach *pair of samples* $(\mathbf{H}^e_{i,k}, \mathbf{Y}_{i,k})$, $(\mathbf{H}^e_{j,k}, \mathbf{Y}_{j,k})$ *with similar input lengths* **do**
    Sample λ from beta distribution with parameter α: $\lambda \leftarrow \text{Beta}(\alpha, \alpha)$;
    Generate Interpolated Embedding: $\mathbf{H}^e_{cr,k} = \lambda \cdot \mathbf{H}^e_{i,k} + (1-\lambda) \cdot \mathbf{H}^e_{j,k}$;
    Generate Interpolated Label: $\mathbf{Y}^n_{cr,k} = \lambda \cdot \mathbf{Y}^n_{i,k} + (1-\lambda) \cdot \mathbf{Y}^n_{j,k}$;
  Add the generated embedding $\mathbf{H}^e_{cr,k}$ and the generated label $\mathbf{Y}_{cr,k}$ to the augmented task $\mathcal{D}_{\text{cr}}$;
return $\mathcal{D}_{cr}$;

training dataset for finetuning LLM, ensuring LLM learns from a diverse and representative set of tasks.

By combining embedding-based interpolation with diversity-based clustering, MDIT greatly expands training data diversity, thereby enhancing the generalization ability of LLMs. Additionally, MDIT provides an automatic data synthetic solution, enriching the diversity of instruction data without relying on external resources.

3.4 Model Training

We use the selected dataset $\mathcal{D}_{\text{M}}$ to finetune LLM. During training, LLM performs forward propagation to generate predictions, which are then compared to the true labels to compute the loss. The loss function $\mathcal{L}$ is defined as:

$$\mathcal{L} = -\frac{1}{\mathcal{N}_{\text{M}}} \sum_{k,n,r} \log \left(P(\mathbf{Y}^n_{\text{M,k}} = r \mid \mathbf{H}^{e,n}_{\text{M,k}}) \right) \quad (9)$$

where N_{M} is the total number of training data, including selected original and generated data. $\mathbf{Y}^n_{\text{M,k}}$ is the true one-hot label and $P(\mathbf{Y}^n_{\text{M,k}} = r \mid \mathbf{H}^{e,n}_{\text{M,k}})$ is the predicted probability for the n-th token in the k-th sample.

Once the loss is computed, the model parameters are updated with the following gradient update rule:

$$\theta \leftarrow \theta - \eta \cdot \frac{1}{\mathcal{N}_{\text{M}}} \sum_{k=1}^{\mathcal{N}_{\text{M}}} \left(P(\mathbf{Y}^n_{\text{M,k}} = r \mid \mathbf{H}^{e,n}_{\text{M,k}}) - \mathbf{Y}^n_{\text{M,k}} \right) \cdot \frac{\partial z^n_k}{\partial \theta} \quad (10)$$

where θ represents the model parameters at the current iteration, η is the learning rate that controls the size of the parameter update, $\frac{\partial z^n_k}{\partial \theta}$ is the gradient of the logit z^n_k relative to the model parameters θ. The gradient update ensures the parameters are adjusted to minimize the loss, allowing LLM to improve its predictions with each iteration.

During finetuning, MDIT utilizes selected data to enhance training efficiency. Training on diverse tasks enables LLM to learn richer expressions, improving its performance on complex tasks.

4 Experiments

4.1 Experiment Setup

Datasets. We use the general question-answering task Alpaca (Taori et al., 2023), the math reasoning task GSM8K (Cobbe et al., 2021), and the code generation task CodeAlpaca (Chaudhary, 2023) for training. To evaluate model performance, we adopt general question answering, math reasoning, and code generation benchmarks for automatic evaluation including ARC Challenge (Clark et al., 2018), MMLU-Math (Hendrycks et al., 2020), Humaneval (Chen et al., 2021), and MBPP (Austin et al., 2021).

Baselines. We compare our MDIT method with several leading data selection and data synthesis methods. We consider the following baselines:

IFD (Li et al., 2024a) selects a balanced subset of instructions by assessing the complexity of instructions through difficulty scores.

DEITA (Liu et al., 2024) combines complexity and quality scoring models to evaluate the diversity and difficulty of each instruction, applying a nearest-neighbor distance threshold to maintain a varied and high-quality training set.

Evol-Instruct (Xu et al., 2024) leverages the gen-

Model	General QA	Math Reasoning	Code Generation		
	ARC Challenge	MMLU-Math	HumanEval	MBPP	Average
Model FineTuned based on Sheared-LLaMa-1.3B					
Zero-Shot	29.10	24.30	0.00	0.20	13.40
IFD (Li et al., 2024a)	30.20	25.70	1.83	0.04	14.44
DEITA (Liu et al., 2024)	29.61	23.60	2.44	0.04	13.92
Evol-Instruct (Xu et al., 2024)	28.67	22.70	7.32	0.28	14.74
MDIT (ours)	26.28	29.20	3.05	4.32	**15.71**
w/o Cluster-Selection	28.33	25.80	3.05	5.55	15.68
Model FineTuned based on LLaMa-2-7B					
Zero-Shot	44.11	30.50	16.46	17.68	27.19
IFD (Li et al., 2024a)	47.10	30.20	21.95	18.59	29.46
DEITA (Liu et al., 2024)	45.31	30.40	20.73	19.93	29.09
Evol-Instruct (Xu et al., 2024)	43.60	24.60	25.61	19.05	28.22
MDIT (ours)	45.40	32.70	23.17	20.76	**30.51**
w/o Cluster-Selection	46.25	31.80	22.56	20.84	30.36
Model FineTuned based on LLaMa-2-13B					
Zero-Shot	50.17	33.90	22.56	16.63	30.81
IFD (Li et al., 2024a)	49.57	35.00	26.22	25.36	34.04
DEITA (Liu et al., 2024)	49.23	31.80	28.66	24.12	33.45
Evol-Instruct (Xu et al., 2024)	49.57	34.00	28.66	25.00	34.31
MDIT (ours)	51.37	34.90	27.44	25.13	**34.71**
w/o Cluster-Selection	50.43	32.30	28.05	26.63	34.35

Table 1: Evaluation Results on the Open LLM Leaderboard. We present the comparison results of our method MDIT with various baselines on SHEARED-LLAMA-1.3B, LLAMA-2-7B and LLAMA-2-13B. We report the results of our MDIT and MDIT w/o cluster selection, the best overall performance in each group is in **bold**.

erative capabilities of LLMs to transform simple instructions into more complex variants.

For the baseline methods, we adopt the best parameters as reported in the original papers.

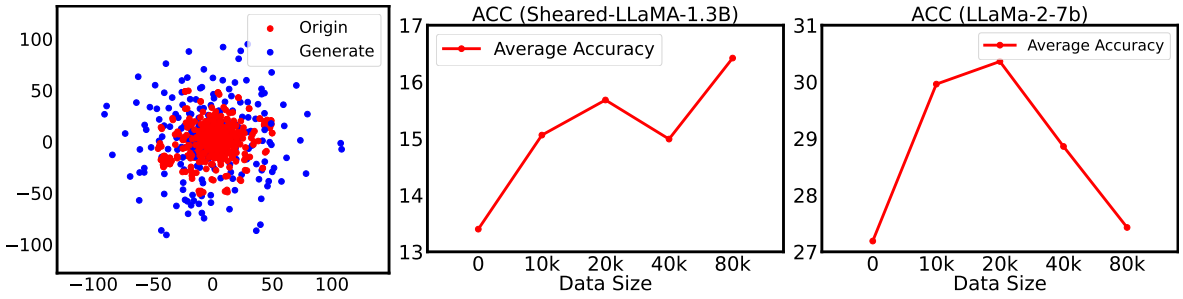
Implementation Details. We perform full-parameter finetuning on the SHEARED-LLAMA-1.3B model (Xia et al., 2023), while using LoRA finetuning for LLAMA-2 7B AND 13B (Touvron et al., 2023). To ensure a fair comparison, we use the same setting for all finetuning experiments. The finetuning process lasts for 3 epochs with learning rate $\eta = 2e - 5$ and a global batch size of 16. For MDIT, we set α = 8 to sample λ from the Beta distribution, and set the number of generated samples per original sample pair $T = 1$.

4.2 Main experiment

The main results are shown in Table 1. Our **MDIT** Supervised Fine-Tuning (SFT) model achieved the best average performance among SFT alignment models across different foundation models. For example, in experiments with the SHEARED-LLAMA-1.3B model, MDIT improves the average accuracy on four test sets by 2.31% compared to the original model and outperforms the baseline methods IFD, DEITA and Evol-Instruct by 1.27%, 1.79% and 0.97% respectively. In experiments with the LLAMA-2-7B model, MDIT achieves an even greater improvement of 3.32% over the original model. Furthermore, on the LLAMA-2-13B model, MDIT improves the average accuracy by 3.9% compared to the original model.

By generating large amounts of diverse tasks using MDIT and finetuning LLM, we observe performance improvements across tasks such as general question answering, math reasoning, and code generation. The model learns richer semantic representations, showing enhanced generalization capabilities when handling more challenging tasks.

We utilize t-SNE visualization to further illustrate the impact of MDIT on data diversity. As depicted in Figure 2, the original data primary cluster



(a) t-SNE visualizations (b) MDIT Scaling(1.3b) (c) MDIT Scaling(7b)

Figure 2: The left figure shows the t-SNE plots of multiple datasets enhanced with MDIT. Red indicates the original data, while blue represents the newly generated data produced by MDIT. The two right figures show the performance scaling of the 1.3B & 7B model with the MDIT under different k values.

Model	General QA ARC Challenge	Math Reasoning MMLU-Math	Code Generation HumanEval	MBPP
MDIT	**26.28**	**29.20**	**3.05**	**4.32**
w/o (Alpaca × GSM8K)	+0.43	-8.40	+2.44	-0.37
w/o (GSM8K × Codealpaca)	+2.39	-6.80	+4.27	-2.15
w/o (Alpaca × Codealpaca)	-1.11	-2.10	+2.44	-1.97

Table 2: Performance of MDIT on the SHEARED-LLAMA-1.3B model with different combinations of task interpolation. The main experiment applies pairwise combinations of the Alpaca, GSM8K and CodeAlpaca tasks. The rows below show the performance with the respective combinations removed. "+" indicates performance improvement over the main experiment, while "-" indicates a decline.

in the central region of the feature space, while the generated embeddings are more widely dispersed, showing that MDIT creates new and diverse instruction data with richer semantic content.

Data Scaling: We investigate the impact of data scaling on the SHEARED-LLAMA-1.3B and LLAMA-2-7B models by finetuning them with data budgets *m* ranging from 10K to 80K samples. Figure 2 shows that our models outperform the original models across all data scales, with performance gains being most notable when the data volume is relatively small. In particular, the 1.3B model improves as the dataset size increases, while the 7B model initially benefits but eventually declines. It suggests that smaller models require larger datasets for better performance, while larger models perform well with a moderate data scale.

4.3 Ablation Study

Effects of Different Tasks Interpolation: To evaluate the impact of task interpolation combinations on MDIT, we selectively remove task pairings. As shown in Table 2, removing Alpaca × GSM8K interpolation improves General QA performance but significantly decreases Math Reasoning, with an 8.40% drop. Removing GSM8K × CodeAlpaca leads to noticeable improvements in General QA (+2.39%) and Code Generation (+4.27%), but harms Math Reasoning (-6.80%), indicating that this combination is beneficial for tasks requiring complex reasoning. The removal of Alpaca × CodeAlpaca causes a slight decline in General QA (-1.11%) and Math Reasoning (-2.10%), but boosts HumanEval by +2.44%, showing its importance for question-answering and reasoning tasks. These results emphasize the importance of carefully selecting dataset pairs for interpolation to achieve a balanced performance across different tasks.

Effects of Different interpolation Parameter α: To explore the impact of the interpolation weight on model performance, we vary the α parameter. The interpolation weight λ follows a Beta(α, α) distribution. As α increases, λ becomes more concentrated around 0.5, causing the interpolated samples to move farther from the original samples. We select α values from $\{1, 2, 4, 8, 12\}$, the results are shown in Table 3. Our observations indicate that $\alpha = 4$ achieves the best performance.

Effects of Different Data Size: To evaluate MDIT in few samples scenarios, we conduct ex-

Method	ARC Challenge	MMLU-Math	HumanEval	MBPP	Average
$\alpha = 1$	28.67	22.30	5.49	5.20	15.41
$\alpha = 2$	27.47	21.40	6.71	5.55	15.28
$\alpha = 4$	26.79	25.00	6.71	3.45	**15.49**
$\alpha = 8$	27.73	21.80	4.88	5.53	14.99
$\alpha = 12$	27.99	25.30	6.10	0.95	15.08

Table 3: Performance of MDIT on SHEARED-LLAMA-1.3B model under various α values.

Size	ARC Challenge	MMLU-Math	HumanEval	MBPP	Average
10000	43.77	27.60	18.90	17.17	26.86
+ MDIT	42.92	32.10	17.59	18.90	**27.88**
20000	44.20	28.50	18.29	17.31	27.07
+ MDIT	44.45	30.80	21.34	17.91	**28.63**
40000	45.48	30.90	16.89	20.12	28.35
+ MDIT	45.48	32.00	20.12	18.84	**29.11**
80000	45.14	30.50	24.39	18.04	29.52
+ MDIT	46.25	31.80	22.56	20.84	**30.36**

Table 4: Performance of MDIT on LLAMA-2-7B model under various data sizes.

tensive experiments on LLAMA-2-7B. The experiments utilize a subset of the dataset, with $N = \{10000, 20000, 40000, 80000\}$, where $N = 80000$ represents the full dataset. As shown in Table 4, with only 10K training samples, MDIT improves accuracy by 4.50% on MMLU-Math and 1.73% on MBPP, resulting in an average accuracy increase of 1.02%. With 20K samples, the average accuracy increased by 1.56%. Notably, training with 10K samples using MDIT outperformed training with 20K samples without augmentation. It shows that even with limited training data, MDIT can effectively generate diverse data, significantly improving LLM performance and maximizing the potential of small-scale datasets.

Effect of Different Numbers of Generated Samples per Original Sample Pair T: To evaluate the impact of the number of generated samples per original sample pair T on model performance, we conduct experiments using subsets of the dataset. The values of T are set to $\{0, 1, 2, 4, 8\}$, where $T = 0$ indicates that only the original data is used. As shown in Figure 3, generating one or two augmented samples per original sample pair leads to improvements in average performance. However, as T increases further, model performance starts to decline, suggesting that there is a limit to the number of useful augmented samples that can be generated from a single original sample pair. Based on these findings, we recommend setting T to no more than 2 for any dataset size.

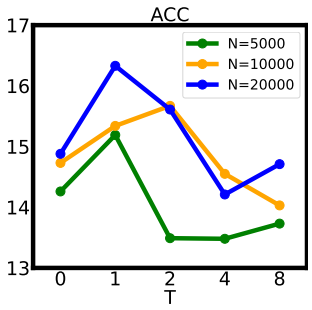


Figure 3: Performance of MDIT on SHEARED-LLAMA-1.3B model under different generation sample number per original sample pair T.

5 Conclusion

In this paper, we propose MDIT, a novel model-free task-level interpolation method that generates diverse tasks for instruction tuning, combined with diversity-based clustering strategies. Extensive experiments show its superior performance that improves the generalization capabilities of LLMs across various tasks. Our method expands the coverage of data within the semantic space, enabling LLMs to learn richer semantic representations. Furthermore, MDIT offers an innovative method to generate diverse instruction data without relying on external resources, providing valuable insights for future research in instruction data management.

Limitations

In this paper, we utilize task interpolation to enhance data diversity for instruction tuning. However, even with the application of clustering-based filtering, some noise is inevitably introduced during the data synthesis process. Moving forward, how to implement more effective filtering strategies as well as improve the transparency of the data generation process leaves for future work.

Ethics Statement

All the data utilized in our work is gathered from the public resources. We have utilized various open-source models including Sheared-LLaMa-1.3B, LLaMa-2-7B, and LLaMa-2-13B, as well as open-source software such as Hugging Face and PyTorch.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Yihan Cao, Yanbin Kang, Chi Wang, and Lichao Sun. 2023. Instruction mining: When data mining meets large language model finetuning. *arXiv preprint arXiv:2307.06290*.

Sahil Chaudhary. 2023. Code alpaca: An instruction-following llama model for code generation. https://github.com/sahil280114/codealpaca.

John Chen, Samarth Sinha, and Anastasios Kyrillidis. 2022. Stackmix: A complementary mix algorithm. In *Uncertainty in Artificial Intelligence*, pages 326–335. PMLR.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. 2023a. Alpagasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Yankai Chen, Yixiang Fang, Qiongyan Wang, Xin Cao, and Irwin King. 2024a. Deep structural knowledge exploitation and synergy for estimating node importance value on heterogeneous information networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 8302–8310.

Yankai Chen, Quoc-Tuan Truong, Xin Shen, Ming Wang, Jin Li, Jim Chan, and Irwin King. 2023b. Topological representation learning for e-commerce shopping behaviors. *Proceedings of the 19th International Workshop on Mining and Learning with Graphs (MLG)*.

Yongrui Chen, Haiyun Jiang, Xinting Huang, Shuming Shi, and Guilin Qi. 2024b. Dog-instruct: Towards premium instruction-tuning data via text-grounded instruction wrapping. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4125–4135.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

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

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*.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*.

Ziqing Fan, Siyuan Du, Shengchao Hu, Pingjie Wang, Li Shen, Ya Zhang, Dacheng Tao, and Yanfeng Wang. 2025. Combatting dimensional collapse in LLM pre-training data via submodular file selection. In *The Thirteenth International Conference on Learning Representations*.

Yuan Ge, Yilun Liu, Chi Hu, Weibin Meng, Shimin Tao, Xiaofeng Zhao, Mahong Xia, Zhang Li, Boxing Chen, Hao Yang, Bei Li, Tong Xiao, and JingBo Zhu. 2024. Clustering and ranking: Diversity-preserved instruction selection through expert-aligned quality estimation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 464–478, Miami, Florida, USA. Association for Computational Linguistics.

Hila Gonen, Srini Iyer, Terra Blevins, Noah A Smith, and Luke Zettlemoyer. 2022. Demystifying prompts in language models via perplexity estimation. *arXiv preprint arXiv:2212.04037*.

Zihao He, Minh Duc Chu, Rebecca Dorn, Siyi Guo, and Kristina Lerman. 2024. Community-cross-instruct: Unsupervised instruction generation for aligning

large language models to online communities. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17001–17019, Miami, Florida, USA. Association for Computational Linguistics.

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*.

Khawar Islam, Muhammad Zaigham Zaheer, Arif Mahmood, and Karthik Nandakumar. 2024. Diffusemix: Label-preserving data augmentation with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27621–27630.

Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. 2020. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *International conference on machine learning*, pages 5275–5285. PMLR.

Jianshang Kou, Benfeng Xu, Chiwei Zhu, and Zhendong Mao. 2024. KNN-instruct: Automatic instruction construction with k nearest neighbor deduction. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10337–10350, Miami, Florida, USA. Association for Computational Linguistics.

Po-Nien Kung, Fan Yin, Di Wu, Kai-Wei Chang, and Nanyun Peng. 2023. Active instruction tuning: Improving cross-task generalization by training on prompt sensitive tasks. *arXiv preprint arXiv:2311.00288*.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2024a. From quantity to quality: Boosting LLM performance with self-guided data selection for instruction tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7595–7628, Mexico City, Mexico. Association for Computational Linguistics.

Yunshui Li, Binyuan Hui, Xiaobo Xia, Jiaxi Yang, Min Yang, Lei Zhang, Shuzheng Si, Ling-Hao Chen, Junhao Liu, Tongliang Liu, Fei Huang, and Yongbin Li. 2024b. One-shot learning as instruction data prospector for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4586–4601, Bangkok, Thailand. Association for Computational Linguistics.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. 2024. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. In *The Twelfth International Conference on Learning Representations*.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*, pages 22631–22648. PMLR.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. # instag: Instruction tagging for analyzing supervised fine-tuning of large language models. In *The Twelfth International Conference on Learning Representations*.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*.

Benedetta Muscato, Chandana Sree Mala, Marta Marchiori Manerba, Gizem Gezici, and Fosca Giannotti. 2024. An overview of recent approaches to enable diversity in large language models through aligning with human perspectives. In *Proceedings of the 3rd Workshop on Perspectivist Approaches to NLP (NLPerspectives) @ LREC-COLING 2024*, pages 49–55, Torino, Italia. ELRA and ICCL.

Xingyuan Pan, Luyang Huang, Liyan Kang, Zhicheng Liu, Yu Lu, and Shanbo Cheng. 2024. G-dig: Towards gradient-based diverse and high-quality instruction data selection for machine translation. *arXiv preprint arXiv:2405.12915*.

Jie Qin, Jiemin Fang, Qian Zhang, Wenyu Liu, Xingang Wang, and Xinggang Wang. 2020. Resizemix: Mixing data with preserved object information and true labels. *arXiv preprint arXiv:2012.11101*.

Zexuan Qiu, Jieming Zhu, Yankai Chen, Guohao Cai, Weiwen Liu, Zhenhua Dong, and Irwin King. 2024. Ease: Learning lightweight semantic feature adapters from large language models for ctr prediction. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 4819–4827.

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

Ke Sun, Bing Yu, Zhouchen Lin, and Zhanxing Zhu. 2024. Patch-level neighborhood interpolation: A general and effective graph-based regularization strategy. In *Asian Conference on Machine Learning*, pages 1276–1291. PMLR.

Shaomu Tan, David Stap, Seth Aycock, Christof Monz, and Di Wu. 2024. UvA-MT's participation in the WMT24 general translation shared task. In *Proceedings of the Ninth Conference on Machine Translation*, pages 176–184, Miami, Florida, USA. Association for Computational Linguistics.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Stanford alpaca: An instruction-following llama model.

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*.

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. *arXiv preprint arXiv:2212.10560*.

Zhicai Wang, Longhui Wei, Tan Wang, Heyu Chen, Yanbin Hao, Xiang Wang, Xiangnan He, and Qi Tian. 2024. Enhance image classification via inter-class image mixup with diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17223–17233.

Lai Wei, Zihao Jiang, Weiran Huang, and Lichao Sun. 2023. Instructiongpt-4: A 200-instruction paradigm for fine-tuning minigpt-4. *arXiv preprint arXiv:2308.12067*.

Shengguang Wu, Keming Lu, Benfeng Xu, Junyang Lin, Qi Su, and Chang Zhou. 2023. Self-evolved diverse data sampling for efficient instruction tuning. *arXiv preprint arXiv:2311.08182*.

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

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. WizardLM: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*.

Zihao Yi, Jiarui Ouyang, Yuwen Liu, Tianhao Liao, Zhe Xu, and Ying Shen. 2024. A survey on recent advances in llm-based multi-turn dialogue systems. *arXiv preprint arXiv:2402.18013*.

Dianzhi Yu, Xinni Zhang, Yankai Chen, Aiwei Liu, Yifei Zhang, Philip S Yu, and Irwin King. 2024. Recent advances of multimodal continual learning: A comprehensive survey. *arXiv preprint arXiv:2410.05352*.

Zhaojian Yu, Xin Zhang, Ning Shang, Yangyu Huang, Can Xu, Yishujie Zhao, Wenxiang Hu, and Qiufeng Yin. 2023. Wavecoder: Widespread and versatile enhanced instruction tuning with refined data generation. *arXiv preprint arXiv:2312.14187*.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032.

Min Zeng, Caiquan Liu, Shiqi Zhang, Li Xie, Chen Sang, and Xiaoxin Chen. 2025. Data quality enhancement on the basis of diversity with large language models for text classification: Uncovered, difficult, and noisy. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 4704–4714, Abu Dhabi, UAE. Association for Computational Linguistics.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. *Preprint*, arXiv:1710.09412.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

Yingxiu Zhao, Bowen Yu, Binyuan Hui, Haiyang Yu, Minghao Li, Fei Huang, Nevin L Zhang, and Yongbin Li. 2024. Tree-instruct: A preliminary study of the intrinsic relationship between complexity and alignment. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 16776–16789.

Daquan Zhou, Kai Wang, Jianyang Gu, Xiangyu Peng, Dongze Lian, Yifan Zhang, Yang You, and Jiashi Feng. 2023. Dataset quantization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17205–17216.