Synthetic Data (Almost) from Scratch: Generalized Instruction Tuning for Language Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

We introduce *Generalized Instruction Tuning* (called GLAN), a general and scalable method for instruction tuning of Large Language Models (LLMs). Unlike prior work that relies on seed examples or existing datasets to construct instruction-tuning data, GLAN exclusively utilizes a pre-curated taxonomy of human knowledge and capabilities as input and generates large-scale synthetic instruction data across all disciplines. Specifically, inspired by the systematic structure in human education system, we build the taxonomy by decomposing human knowledge and capabilities to various fields, sub-fields and ultimately, distinct disciplines semi-automatically, facilitated by LLMs. Subsequently, we generate a comprehensive list of subjects for every discipline and proceed to design a syllabus tailored to each subject, again utilizing LLMs. With the fine-grained key concepts detailed in every class session of the syllabus, we are able to generate diverse instructions with a broad coverage across the entire spectrum of human knowledge and skills. Extensive experiments on large language models (e.g., Mistral) demonstrate that GLAN excels in multiple dimensions from mathematical reasoning, coding, academic exams, logical reasoning to general instruction following without using task-specific training data of these tasks. In addition, GLAN allows for easy customization and new fields or skills can be added by simply incorporating a new node into our taxonomy.

1 Introduction

Large Language Models (LLMs) have enabled unprecedented capabilities to understand and generate text like humans. By scaling up model size and data size [17, 13], LLMs are better at predicting next tokens and prompting to perform certain tasks with a few demonstrations [2]. However, these capabilities do not directly translate to better human instruction following [25]. Instruction tuning [34] bridges this gap by fine-tuning LLMs on instructions paired with human-preferred responses.

Prior work constructs instruction tuning data from seed examples or existing datasets. Initially, natural language processing (NLP) datasets described via instructions are used to fine-tune LLMs and the resulting LLMs can generalize on unseen (NLP) tasks [34]. However, there are only thousands of NLP tasks [33, 19] available, which limits the tuned LLMs to generalize in real-world scenarios [39]. Self-instruct [32] is a cost-effective method for creating synthetic instruction tuning datasets, which starts from a small pool of human-written seed instructions and generates new instructions by few-shot prompting an LLM (e.g., `text-davinci-002`) with randomly selected instructions from the pool. Unfortunately, the diversity of generated instructions is still an issue, since few-shot prompting tends to generate new instructions similar to its demonstrations. In addition, the process of creating high-quality seed instructions requires considerable human effort and expertise. Evolve-Instruct [39] improves self-instruct by augmenting existing instruction tuning datasets with different rewriting operations using LLMs, which is essentially data argumentation. Consequently, the scope of domains

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

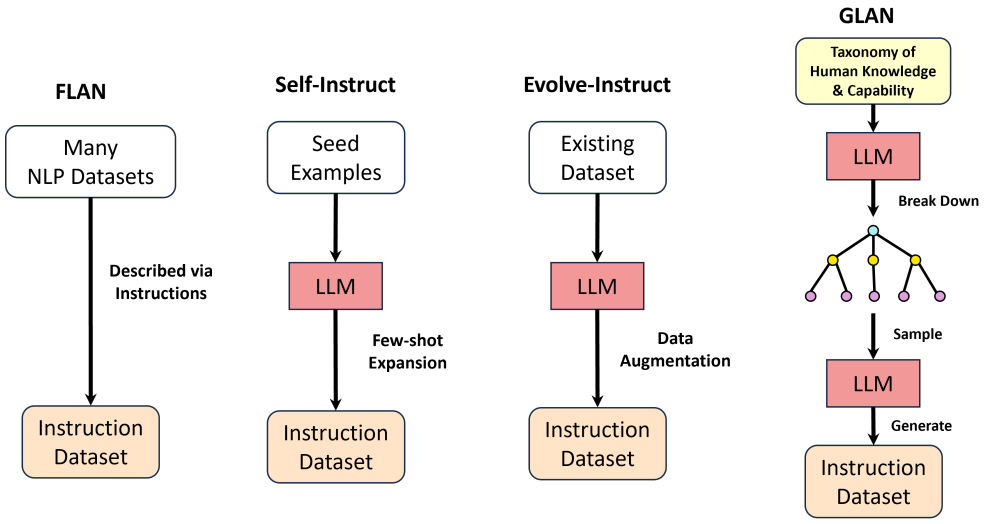

Figure 1: Comparing GLAN with FLAN, Self-Instruct and Evolve-Instruct. The inputs of FLAN, Self-Instrct and Eovlve-Instruct are either seed examples or existing datasets, which limits the scope of domains of instructions that these methods can generate. GLAN takes the taxonomy of human knowledge & capabilities as input to ensure the broad coverage of generated instructions in various domains. This taxonomy is then broken down into smaller pieces and recombined to generate diverse instruction data.

or tasks that these augmented datasets can cover is limited by the original input datasets. See Figure 1 for illustrations of these methods described above. There are also studies concentrated on developing instruction-tuning datasets tailored to particular domains or tasks. For instance, [20] creates datasets targeting mathematical reasoning. In contrast, [3] and [21] focus on coding-related tasks. All of the above methods cannot produce instruction datasets that are generally applicable to a wide range of domains.

How to create a *general* instruction tuning dataset? We draw inspiration from the systematic structure in human education system. The structure of human education includes several levels, starting from early childhood education up to higher education and beyond [37]. Within each level, a student acquires knowledge, skills, and values in a systematic process. The courses a student learns from primary school to college cover a broad range of knowledge and skills, which facilitates the development of a diverse array of abilities. We believe that the systemic framework of the human education system has the potential to help the generation of high-quality and *general* instruction data, which spans a diverse range of disciplinary areas.

In this paper, we introduce a generalized instruction tuning paradigm GLAN (shorthand for **G**eneralized Instruction-Tuning for **L**arge **LAN**guage Models) to generate synthetic instruction tuning data almost from scratch. Unlike existing work [39, 21, 20, 24], GLAN exclusively utilizes a pre-curated taxonomy of human knowledge and capabilities as input and generates large-scale instruction data systematically and automatically across all disciplines. Specifically, inspired by the structure of the human education system, the input taxonomy is constructed by decomposing human knowledge and capabilities to various fields, sub-fields, and, ultimately, distinct disciplines semi-automatically, facilitated by LLMs and human verification. The cost of human verification process is low due to the limited number of disciplines in the taxonomy. As shown in Figure 1, we then further break down these disciplines into even smaller units. We continue to generate a comprehensive list of subjects for every discipline and proceed to design a syllabus tailored to each subject, again utilizing LLMs. With the fine-grained key concepts detailed in every class session of the syllabus, we can first sample from them and then generate diverse instructions with broad coverage across the entire spectrum of human knowledge and skills. The process described above mirrors the human educational system, where educators in each discipline craft a series of subjects for student learning. Instructors then develop a syllabus for each subject, breaking down the content into specific class sessions. These sessions are then further divided into core concepts that students must comprehend and internalize. Based on these detailed core concepts outlined in the syllabus, teaching materials and exercises are subsequently created, which are our instruction tuning data.

GLAN is general, scalable and customizable. GLAN is a general method, which is task-agnostic and is capable of covering a wide range of domains. GLAN is scalable. Similar to [32, 39], GLAN generates instructions using LLMs, which can produce instructions on a massive scale. Moreover, the input of GLAN is a taxonomy, which is generated by prompting an LLM and human verification, requiring minimal human effort. GLAN allows for easy customization. New fields or skills can be added by simply incorporating a new node into our taxonomy. Note that each node of the taxonomy can be expanded independently, which means that we only need to apply our method to the newly added nodes without re-generating the entire dataset. Extensive experiments on large language models (e.g., Mistral) demonstrate that GLAN excels in multiple dimensions from mathematical reasoning, coding, academic exams, and logical reasoning to general instruction following without using task-specific training data of these tasks.

2 GLAN: Generalized Instruction-Tuned Language Models

GLAN aims to create synthetic instruction data covering various domains of human knowledge and capabilities on a large scale. As shown in Algorithm 1, we first build a taxonomy of human knowledge and capabilities using frontier LLMs (i.e., `GPT-4`) and human verification. The taxonomy naturally breaks down human knowledge and capabilities to *fields*, *sub-fields*, and ultimately different *disciplines* (see Section 2.1). The following steps are fully autonomously facilitated by `GPT-4` (or `GPT-3.5`). Then for each discipline, we again instruct `GPT-4` to further decompose it into a list of subjects within this discipline (Section 2.2). Similar to an instructor, `GPT-4` continues to design a syllabus for each subject, which inherently breaks a subject into various class sessions with key concepts students need to master (Section 2.3). With obtained class sessions and key concepts, we are ready to construct synthetic instructions. We prompt `GPT-4` to generate homework questions based on randomly sampled class sessions and key concepts as well as the syllabus (Section 2.4). We recursively decompose human knowledge and capabilities into smaller units until atomic-level components (i.e., class sessions and key concepts). We expect to randomly combine these class sessions and key concepts to ensure the coverage and diversity of synthetic instructions.

Algorithm 1 GLAN Instruction Generation

$\mathbb{D} \leftarrow$ `build_taxonomy()` ▷ build a taxonomy and return a list of *disciplines* (Section 2.1)
$\mathbb{L} \leftarrow \varnothing$
for each discipline $d \in \mathbb{D}$ **do**
  $\mathbb{S} \leftarrow$ `generate_subjects`(d) ▷ Obtain a list of *subjects* in d (Section 2.2)
  for each subject $s \in \mathbb{S}$ **do**
    $\mathcal{A} \leftarrow$ `generate_syllabus`(s, d) ▷ Return syllabus $\mathcal{A}$ for s (Section 2.3)
    $\mathbb{C}, \mathbb{K} \leftarrow$ `extract_class_details`$(\mathcal{A})$ ▷ Extract class sessions and key concepts (Section 2.3)
    $\mathbb{Q} \leftarrow$ `generate_instructions`$(\mathcal{A}, \mathbb{C}, \mathbb{K}, d)$ ▷ Generate instructions by sampling class sessions and key concepts (Section 2.4)
    $\mathbb{L} \leftarrow \mathbb{L} \cup \mathbb{Q}$
  end for
end for
return $\mathbb{L}$

2.1 Taxonomy of Human Knowledge and Capabilities

We build a taxonomy of human knowledge and capabilities to guide the generation of synthetic instructions. Therefore, its coverage is important. On the other hand, it is also essential to make the taxonomy highly extensible, since the preferred capabilities of LLMs may change over time. In the first step, we propose to generate the taxonomy by prompting `GPT-4` with a set of different instructions (e.g., `list all fields of human knowledge and capabilities`). Then, we do human post-editing to ensure its correctness and completeness. Due to the limited number of fields, sub-fields, and disciplines in our taxonomy, the cost of human verification is reasonably low. Another advantage of human post-editing is that we can easily add new fields or disciplines to the taxonomy as needed.

Our taxonomy currently covers a diverse range of knowledge and capabilities in both academic education and vocational training. The top level of the taxonomy contains *fields* such as *Natural Sciences*, *Humanities*, or *Services* (vocational training). These fields branch out to various *sub-fields* and/or *disciplines* such as *Chemistry*, *Sociology* or *Retailing*. We keep breaking down nodes of the taxonomy until *disciplines*, and we leave the breaking down of disciplines to automatic methods described in the following sections. By collecting the leaf nodes of the taxonomy, we obtain a list of disciplines $\mathbb{D} = \{d_1, d_2, \ldots, d_M\}$.

2.2 Subject Generator

As in Algorithm 1, for each discipline d, we aim to extract the list of subjects in it through prompt engineering. Specifically, we instruct `GPT-4` to `act as an education expert of discipline` d `and design a list of subjects a student should learn`. The completion of `GPT-4` contains a comprehensive list of subjects and their meta data (e.g., level, introduction and subtopics of the subject) in unstructured text format, which can not be directly used in subsequent steps. We therefore used another round of prompting to convert the completion to `JSONL` format:

```
Awesome! Transform the above to JSONL format so that it is easier for
a computer to understand. Enclose the JSONL output between two sets of
triple backticks. For each JSONL object, use the keys ‘‘subject_name’’,
‘‘level’’ and ‘‘subtopics’’.
```

It is worth noting that generating a subject list in `JSONL` format using a single prompt is feasible. However, we refrain to do so, because we observe that incorporating additional formatting instructions directly into the prompt can compromise the quality of the resulting subject list. These extracted subjects (as well as their meta data) $\mathbb{S} = \{s_1, s_2, \ldots, s_N\}$ can be subsequently used in next steps. For each $s \in \mathbb{S}$, let `s.name`, `s.level` and `s.subtopics` denote the name, grade level and subtopics of subject s, respectively. We can apply the above prompts multiple times to ensure better coverage of subjects within this discipline.

2.3 Syllabus Generator

For each subject s, we have already extracted its name (`s.name`), grade level (`s.level`), and a small set of included sub-topics (`s.subtopics`) in a structured format. In this section, we aim to further segment each subject into smaller units, making them more suitable for creating homework assignments. We consult `GPT-4` to design a syllabus for this subject. We opt for syllabus generation for the following reasons. Firstly, a syllabus essentially breaks down the main topic of a subject into smaller segments in a hierarchical manner. Specifically, each subject comprises several class sessions, and each session covers a variety of sub-topics and key concepts. Secondly, a syllabus provides an introduction, objectives, and expected outcomes of a subject, which are inherently useful for formulating homework questions. We instruct `GPT-4` to 1) design a syllabus based on its meta data (`s.level`, `s.name` and `s.subtopics`); 2) break the subject into different class sessions; 3) provide details for each class session with a description and detailed key concepts students need to master.

Let $\mathcal{A}$ denote the generated syllabus. The resulting syllabus $\mathcal{A}$ is in unstructured text format. However, class session names and key concepts of each class are required in the instruction generation step (see Algorithm 1). Similar to the process of subject list extraction in Section 2.2, we again extract these meta data of each class session by prompting `GPT-4`. As a result, we obtain a list of class sessions $\mathbb{C} = \{c_1, c_2, \ldots, c_{|\mathbb{C}|}\}$ and their corresponding key concepts $\mathbb{K} = \{\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_{|\mathbb{C}|}\}$. The detailed prompt for syllabus generation is in Appendix A.3.

2.4 Instruction Generator

Given a syllabus $\mathcal{A}$ as well as a list of its class sessions $\mathbb{C}$ and their associated key concepts $\mathbb{K}$, we are ready to generate homework questions and their answers. To generate diverse homework questions, we first sample one or two class session names from $\mathbb{C}$ and one to five key concepts under these selected class sessions. Let $\hat{\mathbb{C}}$ denote the selected class session names and $\hat{\mathbb{K}}$ the selected key concepts. Then we prompt `GPT-4` (or `GPT-3.5`) to generate a homework question given the selected class sessions $\hat{\mathbb{C}}$ and key concepts $\hat{\mathbb{K}}$ as well as the syllabus $\mathcal{A}$. We intend to give `GPT-4/3.5` more

context (e.g., what students have already learned in previous sessions) when creating assignments. Therefore, we additionally instruct GPT to consider that students have learned up to class sessions $\hat{\mathbb{C}}$ when crafting homework and try to leverage multiple key concepts across different class sessions. See details of our prompt for instruction generation in Appendix A.4.

Sampling Class Sessions and Key Concepts In a single syllabus, there are numerous class sessions and key concepts. We have two strategies to sample from them. In the first strategy, we generate assignments from a single class session. Therefore, we have only one class session name. Suppose we have m key concepts in total in this session. We randomly sample one to five key concepts from the m key concepts, which means we have totally $\sum_{i=1}^{5} \binom{m}{i}$ combinations. In this strategy, we focus on creating *basic* homework questions. To make the resulting questions more challenging (combine knowledge from multiple class sessions), we propose a second strategy to combine key concepts from two class sessions in the second strategy. We intend to generate questions leverage knowledge from two different class sessions. Suppose we have m_1 and m_2 key concepts in the first and second class sessions, respectively. We can have $\sum_{i=2}^{5} \binom{m_1+m_2}{i} - \sum_{i=2}^{5} \binom{m_1}{i} - \sum_{i=2}^{5} \binom{m_2}{i}$ different combinations, which is significantly more than that of the first strategy. We use both strategies to ensure our created questions are diverse in difficulty levels.

Answer Generation After we generate questions in previous steps, we simply send these questions to `GPT-3.5` and collect answers. We use `GPT-3.5` for answer generation, because we find the quality of generated answers from `GPT-3.5` is sufficiently good and using `GPT-3.5` is significantly faster than `GPT-4`. The resulting question-answer pairs are our instruction tuning data. With a huge amount of question-answer pairs ranging from different disciplines with various difficulty levels, we expect the resulting LLM can excel in a wide range of tasks.

3 Experiments

3.1 Data Generation

Taxonomy Creation By asking `GPT-4` to create a taxonomy of human knowledge and capabilities, we end up with a set of fields, sub-fields, and disciplines that cover a broad range of domains in human knowledge and capabilities. Next, we ask human annotators to decide whether these elements in the taxonomy should be kept or not in order to reduce the redundancy of the taxonomy while maintaining its correctness. Note that if a field or sub-field is marked as *remove*, we remove its descendant as well. We kept 126 *disciplines* after majority voting (provided in supplementary materials). Note that it is feasible to manually add extra disciplines, sub-fields, or fields whenever necessary.

Subject and Syllabus Generation During the subject list and syllabus generation, we prompt `GPT-4` and employ nucleus sampling [14] with temperature $T = 1.0$ and top-$p = 0.95$ to encourage diversity. We do not use `GPT-3.5-turbo` since some subjects belong to the long-tail distribution which may not be effectively modeled by `GPT-3.5-turbo`. To ensure diversity and completeness of the generated subjects, we query `GPT-4` 10 times for each discipline (Section 2.2). There are 100 to 200 subjects for each discipline on average. It is worth noting that the same subjects may appear in different disciplines. For instance, the subject *calculus* is both in physics and mathematics. We do not de-duplicate those subjects, since it may reflect their importance in human knowledge. Given a subject in a specified discipline, we query `GPT-4` for only one time to design a syllabus (see details in section 2.3). The temperature and top-p are still set to 1.0 and 0.95, respectively. The number of class sessions contained in each syllabus varies from 10 to 30 and each class session contains around five key concepts.

Instruction Generation Each instruction data consists of a question and its answer. We choose to generate questions and answers separately since we observed that separate generations lead to better quality. After question generation with `GPT-4`, each question is then answered by `GPT-3.5-turbo` with temperature $T = 0.7$, top-$p = 0.95$ (we use a lower temperature in order to make the resulting answers more accurate). We use `GPT-3.5-turbo` instead of `GPT-4` for answer generation, because `GPT-3.5-turbo` is significantly faster with reasonably good results. We generate 10 million instruction-response pairs in total and then we do training data decontamination. Specifically, the training instruction-response pairs are decontaminated by removing pairs that contain questions or

Table 1: Main results on Mathematical Reasoning, Coding, Logical Reasoning, and Academic Exam benchmarks. Best results are in boldface, while the second best results are underscored.

Model	$\|\theta\|$	HumanE	MBPP	GSM8K	MATH	BBH	ARC-E	ARC-C	MMLU
GPT-4	–	88.4	80.0	92.0	52.9	86.7	95.4	93.6	86.4
GPT-3.5-turbo	–	72.6	70.8	74.1	37.8	70.1	88.9	83.7	70.0
LLaMA2	7B	12.8	36.2	15.4	4.2	39.6	74.6	46.3	45.9
Orca 2	7B	17.1	28.4	55.7	10.1	42.8	<u>87.8</u>	<u>78.4</u>	53.9
WizardLM v1.2	13B	31.7	47.9	46.8	9.0	48.4	74.2	50.2	52.7
Mistral	7B	28.0	50.2	43.4	10.0	56.1	79.5	53.9	<u>62.3</u>
Mistral Instruct	7B	46.7	31.7	24.4	8.2	46.0	76.9	52.0	53.7
MetaMath Mistral	7B	35.4	48.6	77.7	28.2	55.7	77.3	51.0	61.0
WizardMath v1.1	7B	**51.2**	<u>54.1</u>	**83.2**	**33.0**	<u>58.2</u>	79.8	53.2	60.3
Mistral CodeAlpaca	7B	35.4	50.2	34.6	8.3	56.1	79.1	54.2	60.9
GLAN	7B	<u>48.8</u>	**57.6**	<u>80.8</u>	<u>32.7</u>	**60.7**	**90.7**	**81.1**	**62.9**

input prompts from the test and training (if any) sets of benchmarks we evaluate. We exclude the training set of benchmarks we evaluate to verify the generalization capability of our synthetic data.

3.2 Model Training

We employ Mistral 7B [16] as our base model. During training, we concatenate each instruction and response pair to a single sequence and only compute loss on response tokens. We train our model for 3 epochs with a learning rate of $3e$-6. The batch size is set to approximately 512 instruction-response pairs. We employ a dynamic batch size to ensure a constant total number of tokens per batch. We use a cosine learning rate schedule and we start with a linear warm-up of 1000 steps and the final learning rate is reduced to 0. The training requires approximately 8 days using 32 A100 GPUs.

3.3 Benchmark Evaluation

The instruction data GLAN generated spans a wide range of subjects. We evaluate its effectiveness in mathematical reasoning, coding, logical reasoning, and academic exams.

***Mathematical Reasoning*:** Mathematics is a common subject in many different disciplines. Hence, it is necessary to test the math reasoning ability of GLAN. We choose the two popular benchmarks for evaluation (i.e., GSM8K [7] and MATH [12]). GSM8K [7] is a high-quality math problem dataset that measures the basic multi-step mathematical reasoning ability. It contains around 7k problems for training and 1K problems for test. MATH [12] is a challenging math dataset that contains mathematics competition-level problems from AMC, AIME, etc. The 7.5k training and 5K test problems cover seven math subjects, i.e., Prealgebra, Precalculus, Algebra, Intermediate Algebra, Number Theory, Counting and Probability, and Geometry. Note that GLAN does not use any examples in the training set of GSM8K or MATH. Following [20], we report 0-shot setting results for GLAN. ***Coding*:** To evaluate the coding capability of GLAN, we opt for two coding benchmarks HumanEval [4] and MBPP [1]. We employ 0-shot setting for HumanEval and 3-shot setting for MBPP following prior art [4, 21]. ***BBH*:** The instruction dataset we generated covers many disciplines, which can potentially enhance the reasoning ability of GLAN. Therefore, we evaluate GLAN on the BIG-Bench Hard dataset (BBH [29]), which contains 23 challenging tasks from Big-Bench [28]. We employ the standard 3-shot setting with chain-of-thought demonstrations. ***Academic Exams*:** We also evaluate GLAN on different academic benchmarks to verify whether GLAN is capable of solving exam questions. We choose two benchmarks (i.e., ARC [6] and MMLU [11]). Both benchmarks are composed of multi-choice questions. AI2 Reasoning Challenge (ARC [6]) contains grade-school level, multi-choice science questions. It contains two sub-sets, which are ARC-Challenge (ARC-C) and ARC-Easy (ARC-E). Massive Multitask Language Understanding (MMLU [11]) consists of a set of multiple-choice questions about 57 subjects ranging in difficulty from elementary levels to professional levels. It covers various of domains of knowledge, including humanities, STEM and social sciences. Note that there is a training set for ARC. However, we have excluded it from our

Table 2: Detailed Results on Academic Exam benchmarks.

Model	ARC-E	ARC-C	MMLU			
			STEM	Humanities	Social Sciences	Other
Mistral	79.5	53.9	52.0	56.5	73.3	70.1
GLAN	**90.7**	**81.1**	**60.1**	54.9	71.8	68.6

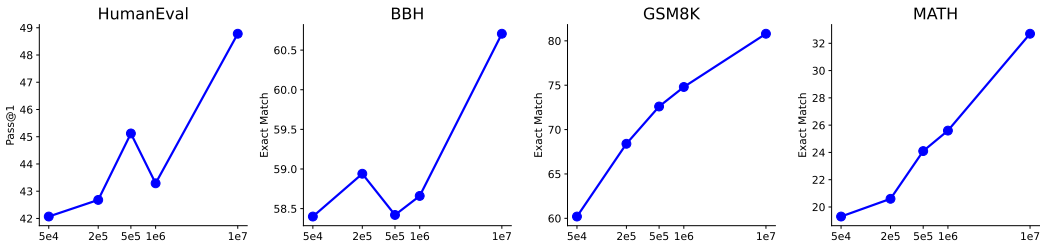

Figure 2: The scaling curve of GLAN on downstream tasks. The x-axis denotes GLAN data size (in $\log_{10}$ scale following [17]), and the y-axis denotes the task performance.

training set during the decontamination process described in Section 3.1. Previous models mostly leverage probability-based methods on ARC and MMLU, which returns the best option based on the probabilities of the four options conditioned on the corresponding multi-choice question. We observe that after training on 10 million instructions, GLAN is able to *generate* its predicted options and analysis of multi-choice questions in plain text as `GPT-3.5` does. We therefore opt for 0-shot setting for GLAN and extract predictions using rules based on its completions as in [22].

Results Our main results are shown in Table 1. We compare GLAN against general domain models (Orca 2 [22], Mistral Instruct [16] and WizardLM [39]), math optimized models (MetaMath [40] and WizardMath [20]) and coding optimized models (CodeAlpaca [3]). We also report results of base LLMs (i.e., LLaMA2 [31] and Mistral [16]) as references. GLAN either obtains the best results or results close to the best across all benchmarks. We observe that capabilities of math or coding optimized models increase on math or coding benchmarks while usually not others. After instruction tuning, GLAN excels on multiple dimensions from mathematical reasoning, coding, reasoning, and academic exams with a systematical data generation approach. Also note that our method does not use any task-specific training data such as training sets of GSM8K, MATH, or ARC as in Orca 2, MetaMath, and WizardMath, which indicates the general applicability of GLAN.

A Closer Look at Academic Exams ARC and MMLU are all multi-choice based benchmarks on academic exams. However, we observe that improvements of GLAN over Mistral on ARC are much larger than these on MMLU (see Table 1). By grouping the 57 subjects in MMLU into four categories (i.e., STEM, Humanities, Social Sciences, and Other (business, health, misc.)), we observe GLAN wildly improves on STEM in MMLU while not in other categories (Table 2). Also note that ARC is composed of high school science problems, which are also STEM questions. GLAN is good at STEM subjects may be because responses of our dataset are from `GPT-3.5-turbo`, which by default generates responses with Chain-of-Thoughts (CoT) reasoning. Indeed, we observe that GLAN generates solutions with CoT for multi-choice questions. CoT may help the multi-step reasoning in STEM multi-choice questions [35], while humanities and social sciences questions involve more memorization and single-step reasoning, where CoT may introduce additional errors.

3.4 Scaling Property of GLAN

We investigate the scaling property of GLAN by training Mistral on different numbers of examples (i.e., 50K, 200K, 500K, 1M, and 10M) we generated. The results on downstream tasks are shown in Figure 2. It can be observed that overall task performance tends to increase as we increase the data size. Notably, the curve has not reached a plateau, indicating the potential for further improvement through the continued scaling of the data size of GLAN. However, we defer further scaling experiments to future work.

Table 3: The evaluation of loss values between the test data and training data. Large positive Δ (or $\Delta(\%)$) indicates task-specific in-domain training data might be exposed to the model during training.

Benchmark/Loss		LLaMA2-7B	Orca2-7B	Mistral-7B-Instruct	WizardLM-13B-V1.2	GLAN-7B
ARC-C	Δ	-0.01	0.05	-0.01	-0.01	-0.03
	Δ (%)	**-0.5%**	2.10%	**-0.43%**	**-0.47%**	**-0.74%**
ARC-E	Δ	-0.02	0.04	-0.03	-0.02	-0.01
	Δ (%)	**-0.95%**	1.61%	**-1.19%**	**-0.91%**	**-0.23%**
GSM8K	Δ	0	0.13	0	0.05	0.02
	Δ (%)	**0%**	11.4%	**0%**	4.39%	**0.92%**
MATH	Δ	-0.03	0.03	-0.03	-0.02	-0.03
	Δ (%)	**-2.70%**	2.54%	**-2.67%**	**-1.63%**	**-1.79%**

3.5 Task-specific Training Data

GLAN is a generalized method to create synthetic data for instruction tuning. In order to evaluate the generalization capabilities of this synthetic data, we deliberately exclude task-specific training sets from all benchmarks on which we conduct our assessments. Similar to [36], we explore whether models have been trained on task-specific in-domain data. We compute the training loss L_{train} and test loss L_{test} on ARC Challenge (ARC-C), GSM8K, and MATH for GLAN and other models in comparison. We choose these datasets because among all benchmarks evaluated in Section 3.3, these benchmarks contain training sets. Intuitively, the larger $\Delta = L_{test} - L_{train}$ is, the more likely the training set is exposed. To make Δ easier to interpret, we additionally compute the relative difference $\Delta(\%) = (L_{test} - L_{train})/L_{test}$. Table 3 shows the losses of the training and test splits for GLAN are nearly identical (or Δ is negative). This suggests that GLAN has not been exposed to in-domain data during training and tuning procedures. Please refer detailed L_{train} and L_{test} losses in Table 8 (in Appendix). Additionally, as shown in Table 8, we observe that GLAN obtains higher losses on both test and training splits on GSM8K, MATH, and ARC compared to other models, while performances of GLAN on these datasets are high (see Table 1). This might imply that synthetic data generated by GLAN is diverse and our resulting model avoids convergence to any specific domain or style present in existing benchmarks.

3.6 Instruction Following Evaluation

IFEval We assess the instruction-following capabilities of GLAN utilizing the Instruction Following Evaluation dataset (IFEval [42]). IFEval consists of a collection of "verifiable instructions", encompassing 25 distinct types of instructions (around 500 prompts in total). Each prompt comprises one or more verifiable instructions. The evaluation involves four types of metrics at both prompt level and instruction level, evaluating strict and loose accuracies. As shown in Table 4, GLAN demonstrates superior instruction-following capabilities in both prompt-level and instruction-level evaluations. However, there is still a considerable gap compared to `GPT-3.5-turbo` and `GPT-4`.

Table 4: Instruction following capability evaluation on IFEval.

Model	Prompt-level strict-accuracy	Instruction-level strict-accuracy	Prompt-level strict-accuracy	Instruction-level loose-accuracy
GPT-3.5-turbo	53.8	64.7	56.6	67.5
GPT-4	77.1	83.7	79.7	85.6
LLaMA2-7B	14.8	27.1	16.6	29.4
Orca2-7B	19.4	28.9	26.1	34.7
Mistral-7B-Instruct-v0.1	32.0	42.8	37.7	48.0
WizardLM-13B-V1.2	23.1	33.5	26.6	37.6
GLAN-7B	**34.0**	**44.8**	**41.2**	**51.6**

Evol-Instruct Test Evol-Instruct testset [39] contains real-world human instructions from diverse sources, and it consists of 218 instances with 29 distinct skills. Each instruction is associated with a difficulty level from 1 to 10. The responses are often open-ended descriptions, and we believe this benchmark is a necessary supplement to IFEval (answers to their instructions are "verifiable"). Following [39] and [5], we adopt a GPT-4-based automatic evaluation method to conduct a pairwise comparison between GLAN and other models. Specifically, GPT-4 is instructed to assign a score between 1 and 10 overall score w.r.t. the helpfulness, relevance, accuracy, and level of detail of

Table 5: Pairwise comparison on various difficulty levels between GLAN and other models on Evol-Instruct testset. The scores are the average gap of scores assigned by GPT-4, calculated as $\texttt{avg\_score(GLAN)} - \texttt{avg\_score}(x)$.

Difficulty	Ratio	LLaMA2-7B	Orca2-7B	Mistral-7B-Instruct	Wizard-13B-V1.2	GPT-3.5-turbo
(1-5) Easy	41.00%	**5.46**	**2.19**	**1.13**	**1.32**	-1.22
(6-10) Hard	59.00%	**5.38**	**2.28**	**1.68**	**0.99**	-0.68

responses generated by two different models for a given input question. A higher score indicates better overall performance. To mitigate potential order bias, we perform bidirectional comparisons for each response pair and determine their average score. The average score difference to GLAN (i.e., $\texttt{avg\_score(GLAN)} - \texttt{avg\_score}(x)$) serves as the final metric. Table 5 presents the results of pairwise comparisons across various levels of instruction difficulty. GLAN showcases superior performance compared to LLaMA-2, Orca 2, Mistral Instruct, and even WizardLM-13B (note that GLAN contains only 7B parameters) on most difficulty levels and overall scores. This suggests that GLAN demonstrates improved ability to process diverse instructions, regardless of their difficulty or complexity. Also, note that GLAN falls behind `GPT-3.5-turbo` as other models in comparison. Additionally, we group Evol-Instruct test according to the 29 skills and observe the same trends. Detailed results are listed in Appendix (Table 9 and 10). GLAN demonstrates strong performance on most skills, especially in Math, Coding, and Reasoning. However, it slightly falls short in common-sense related tasks. We also created GLAN-Test, similar to the Evol-Instruct Test but much larger in size, where GLAN outperforms other models as well (see Appendix A.8).

4 Related Work

Recent literature has extensively explored the collection of various human-made resources for instruction tuning. An intuitive direction is to collect existing NLP datasets and corresponding task descriptions [26, 33, 41], typical LLMs such as BLOOMZ [23] and FLAN [34] are trained on this type of instruction tuning data. However, with only tens to thousands of existing datasets available, the scope and diversity of instruction tuning are inevitably limited. Another common practice is to implement instruction tuning with real-world human user prompts. For instance, InstructGPT [25] was trained on high-quality human prompts submitted by real-world users to OpenAI GPT APIs. Vicuna [5] leverages user-shared prompts along with ChatGPT responses for instruction tuning, and Dolly[8] was trained on simulated human-user interactions written by over 5k employees. Nevertheless, acquiring instructional data from human users typically involves high costs and involves privacy concerns. As LLM capabilities improve, instruction tuning with LLM-generated data exhibits better scalability and potential in addressing the super-alignment problem [27]. Leveraging the in-context learning ability of LLMs, Unnatural instructions [15] and Self-instruct [32] sampled seed instructions as examples to elicit LLMs to generate new instructions. Taking advantage of the rephrasing ability of LLMs, WizardLM [39] and WizardMath [20] were trained using Evol-Instruct. Evol-Instruct iteratively employs ChatGPT to rewrite seed instructions into increasingly complex instructions. Similar to generation from seed instructions, carefully selected seed topics are used for generating textbook-like synthetic data [18] or self-chat multi-turn dialogues [38, 9] for instruction tuning. However, models trained on these LLM-generated data only work well in specific domains such as math [20, 40], dialogue [38, 9] or open-ended question answering [30, 39]. These methods encounter challenges in generalization [10], as the data diversity is restricted by seed instructions or seed topics.

5 Conclusions

We propose GLAN, a general and scalable method for synthesizing instruction data. Experiments show that GLAN can help large language models improve their capabilities in multiple dimensions, from mathematical reasoning, coding, academic exams, and logical reasoning to general instruction following. Currently, our synthetic data are based on the taxonomy of human knowledge and capabilities, and there are other types of useful data that have not been covered. We are interested in designing methods with border coverage. Our current instruction data are mostly question-answer pairs, and in the next step, we plan to generate synthetic data of multi-turn conversations and long documents.

References

[1] J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 2020.

[3] S. Chaudhary. Code alpaca: An instruction-following llama model for code generation. `https://github.com/sahil280114/codealpaca`, 2023.

[4] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[5] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023.

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

[7] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[8] M. Conover, M. Hayes, A. Mathur, J. Xie, J. Wan, S. Shah, A. Ghodsi, P. Wendell, M. Zaharia, and R. Xin. Free dolly: Introducing the world's first truly open instruction-tuned llm, 2023.

[9] N. Ding, Y. Chen, B. Xu, Y. Qin, Z. Zheng, S. Hu, Z. Liu, M. Sun, and B. Zhou. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*, 2023.

[10] A. Gudibande, E. Wallace, C. V. Snell, X. Geng, H. Liu, P. Abbeel, S. Levine, and D. Song. The false promise of imitating proprietary language models. In *International Conference on Learning Representations*, 2024.

[11] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

[12] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the math dataset. In *Advances in Neural Information Processing Systems*, 2021.

[13] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, O. Vinyals, J. Rae, and L. Sifre. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems*, 2022.

[14] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020.

[15] O. Honovich, T. Scialom, O. Levy, and T. Schick. Unnatural instructions: Tuning language models with (almost) no human labor. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023.

[16] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[17] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[18] Y. Li, S. Bubeck, R. Eldan, A. Del Giorno, S. Gunasekar, and Y. T. Lee. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*, 2023.

[19] S. Longpre, L. Hou, T. Vu, A. Webson, H. W. Chung, Y. Tay, D. Zhou, Q. V. Le, B. Zoph, J. Wei, and A. Roberts. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*, 2023.

[20] H. Luo, Q. Sun, C. Xu, P. Zhao, J. Lou, C. Tao, X. Geng, Q. Lin, S. Chen, and D. Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023.

[21] Z. Luo, C. Xu, P. Zhao, Q. Sun, X. Geng, W. Hu, C. Tao, J. Ma, Q. Lin, and D. Jiang. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*, 2023.

[22] A. Mitra, L. Del Corro, S. Mahajan, A. Codas, C. Simoes, S. Agarwal, X. Chen, A. Razdaibiedina, E. Jones, K. Aggarwal, et al. Orca 2: Teaching small language models how to reason. *arXiv preprint arXiv:2311.11045*, 2023.

[23] N. Muennighoff, T. Wang, L. Sutawika, A. Roberts, S. Biderman, T. Le Scao, M. S. Bari, S. Shen, Z. X. Yong, H. Schoelkopf, X. Tang, D. Radev, A. F. Aji, K. Almubarak, S. Albanie, Z. Alyafeai, A. Webson, E. Raff, and C. Raffel. Crosslingual generalization through multitask finetuning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023.

[24] S. Mukherjee, A. Mitra, G. Jawahar, S. Agarwal, H. Palangi, and A. Awadallah. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*, 2023.

[25] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.

[26] V. Sanh, A. Webson, C. Raffel, S. H. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, A. Raja, M. Dey, M. S. Bari, C. Xu, U. Thakker, S. S. Sharma, E. Szczechla, T. Kim, G. Chhablani, N. V. Nayak, D. Datta, J. Chang, M. T. Jiang, H. Wang, M. Manica, S. Shen, Z. X. Yong, H. Pandey, R. Bawden, T. Wang, T. Neeraj, J. Rozen, A. Sharma, A. Santilli, T. Févry, J. A. Fries, R. Teehan, T. L. Scao, S. Biderman, L. Gao, T. Wolf, and A. M. Rush. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022.

[27] T. Shen, R. Jin, Y. Huang, C. Liu, W. Dong, Z. Guo, X. Wu, Y. Liu, and D. Xiong. Large language model alignment: A survey. *arXiv preprint arXiv:2309.15025*, 2023.

[28] A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso, A. Kluska, A. Lewkowycz, A. Agarwal, A. Power, A. Ray, A. Warstadt, A. W. Kocurek, A. Safaya, A. Tazarv, A. Xiang, A. Parrish, A. Nie, A. Hussain, A. Askell, A. Dsouza, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023.

[29] M. Suzgun, N. Scales, N. Schärli, S. Gehrmann, Y. Tay, H. W. Chung, A. Chowdhery, Q. Le, E. Chi, D. Zhou, and J. Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, 2023.

[30] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

[31] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[32] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2023.

[33] Y. Wang, S. Mishra, P. Alipoormolabashi, Y. Kordi, A. Mirzaei, A. Naik, A. Ashok, A. S. Dhanasekaran, A. Arunkumar, D. Stap, E. Pathak, G. Karamanolakis, H. Lai, I. Purohit, I. Mondal, J. Anderson, K. Kuznia, K. Doshi, K. K. Pal, M. Patel, M. Moradshahi, M. Parmar, M. Purohit, N. Varshney, P. R. Kaza, P. Verma, R. S. Puri, R. Karia, S. Doshi, S. K. Sampat, S. Mishra, S. Reddy A, S. Patro, T. Dixit, and X. Shen. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022.

[34] J. Wei, M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022.

[35] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

[36] T. Wei, L. Zhao, L. Zhang, B. Zhu, L. Wang, H. Yang, B. Li, C. Cheng, W. Lü, R. Hu, C. Li, L. Yang, X. Luo, X. Wu, L. Liu, W. Cheng, P. Cheng, J. Zhang, X. Zhang, L. Lin, X. Wang, Y. Ma, C. Dong, Y. Sun, Y. Chen, Y. Peng, X. Liang, S. Yan, H. Fang, and Y. Zhou. Skywork: A more open bilingual foundation model, 2023.

[37] Wikipedia contributors. Education, 2023. Last edited on 24 March 2023.

[38] C. Xu, D. Guo, N. Duan, and J. McAuley. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

[39] C. Xu, Q. Sun, K. Zheng, X. Geng, P. Zhao, J. Feng, C. Tao, and D. Jiang. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*, 2023.

[40] L. Yu, W. Jiang, H. Shi, J. YU, Z. Liu, Y. Zhang, J. Kwok, Z. Li, A. Weller, and W. Liu. Metamath: Bootstrap your own mathematical questions for large language models. In *International Conference on Learning Representations*, 2024.

[41] C. Zhou, P. Liu, P. Xu, S. Iyer, J. Sun, Y. Mao, X. Ma, A. Efrat, P. Yu, L. YU, S. Zhang, G. Ghosh, M. Lewis, L. Zettlemoyer, and O. Levy. LIMA: Less is more for alignment. In *Advances in Neural Information Processing Systems*, 2023.

[42] J. Zhou, T. Lu, S. Mishra, S. Brahma, S. Basu, Y. Luan, D. Zhou, and L. Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

A Appendix

A.1 Limitations

While GLAN presents significant advancements in academic benchmarks. However, there may still have several limitations in real world deployment. The resulting LLMs train on generated data using GLAN may occasionally produce factual incorrect (or even toxic) responses. Further training for refusal, hallucination reduction as well as toxic content reduction should be performed before deployment.

A.2 Broader Impacts

Data synthesizing is crucial for the continual scaling of large language models, especially as we exhaust available human data. GLAN demonstrates the potential to generate vast amounts of synthetic data from scratch, paving the way for even larger-scale data synthesis efforts. While GLAN has shown the effectiveness of synthetic data, we must point out that synthetic data may inherit and even amplify social biases present in the frontier LLMs for generation. Future research should focus on developing techniques to identify and correct biases in the generated datasets and models trained on them.

A.3 Prompt for Syllabus Generator

The prompt template for syllabus generation is in Table 6.

Table 6: Prompt template for Syllabus Generator.

You are an expert in {`s.name`}.

Using the given data, design a syllabus for teaching students at the specified level.
Note that example subtopics or descriptions are just give you an impression of what this class like.
Feel free to add extra subtopics if needed (remember you are the expert in {`s.name`}).

Data:
- Level: {`s.level`}
- Main Topic: {`s.name`}
- Description or Example Subtopics: {`s.subtopics`}

Syllabus Design Guide
1. **Introduction**: Start with an overview of the primary topic for the syllabus.
2. **Class Details**: For each class session, provide:
   - **Description**: Briefly describe the focus of the session.
   - **Knowledge Points**: Enumerate key concepts or topics.
   These will be used to craft homework questions.
   - **Learning Outcomes & Activities**: Offer expected learning results and suggest related exercises or activities.

A.4 Prompt for Instruction Generator

The prompt template for instruction generator is in Table 7.

A.5 Task-specific Training Data

We provide the specific train/test values of different models on different benchmarks in Table 8.

A.6 Evol-Instruct Test Results on Different Difficulty Levels

The concrete Evol-Instruct test results on different difficulty levels are shown in Table 9.

Table 7: Prompt template for Instruction Generator.

Background
- You are an expert in {`s.name`} education and you have designed a syllabus (i.e., '## Syllabus')
- We invite you (again) to design ONE homework question for given class sessions and some knowledge points.
- The student have already learned all class sessions up to the current sessions (i.e., '## Current Session(s)').
- There might be multiple class session in '## Current Session(s)'
- The designed homework question should focus on the topics in '## Current Session(s)' and you should try to cover the given knowledge points in '## Given Knowledge Points'
- We prefer homework questions leveraging multiple knowledge points and across different topics

Syllabus
$\{\mathcal{A}\}$

Current Session(s)
$\{\hat{\mathbb{C}}\}$

Given Knowledge Points
$\{\hat{\mathbb{K}}\}$

Table 8: The evaluation of loss values between the test data and training data. Large positive Δ (or $\Delta(\%)$) indicate task specific in-domain training data may be exposed to the model during training.

Benchmark/Loss		LLaMA2-7B	Orca2-7B	Mistral-7B-Instruct	WizardLM-13B-V1.2	GLAN-7B
ARC-C	L_{test}	2.02	2.39	2.32	2.11	4.03
	L_{train}	2.03	2.34	2.33	2.12	4.06
	Δ	-0.01	0.05	-0.01	-0.01	-0.03
	Δ (%)	**-0.5%**	2.10%	**-0.43%**	**-0.47%**	**-0.74%**
ARC-E	L_{test}	2.10	2.47	2.51	2.18	4.31
	L_{train}	2.12	2.43	2.54	2.20	4.32
	Δ	-0.02	0.04	-0.03	-0.02	-0.01
	Δ (%)	**-0.95%**	1.61%	**-1.19%**	**-0.91%**	**-0.23%**
GSM8K	L_{test}	1.38	1.14	1.26	1.14	2.17
	L_{train}	1.38	1.01	1.26	1.09	2.15
	Δ	0	0.13	0	0.05	0.02
	Δ (%)	**0%**	11.4%	**0%**	4.39%	**0.92%**
MATH	L_{test}	1.11	1.18	1.12	1.22	1.67
	L_{train}	1.14	1.15	1.15	1.24	1.70
	Δ	-0.03	0.03	-0.03	-0.02	-0.03
	Δ (%)	**-2.70%**	2.54%	**-2.67%**	**-1.63%**	**-1.79%**

A.7 Evol-Instruct Test Results on Different Skills

The concrete Evol-Instruct test results on different skills are shown in Table 10.

A.8 GLAN-Test Overall Results

GLAN-Test There are only hundreds of instructions in In IFEval and Evol-Instruct Test and we believe the domains or skills they can cover are rather limited. Therefore, we propose a held-out test set using GLAN data and we call it GLAN-Test. It contains 6,300 instructions on 126 disciplines (50 instructions for each discipline). We further categorize the 126 disciplines to 8 distinct *fields* (i.e., Academic-Humanities, Academic-Social Science, Academic-Natural Science, Academic-Applied Science, Academic-Formal Science, Industry-Manufacturing, Industry-Services and Industry-Agriculture). We believe that the extensive domain coverage of GLAN-Test renders it an effective test bed for the assessment of generalization capabilities in LLMs. We adopt the same GPT-4 based evaluation protocol as in Evol-Instruct Test (previous paragraph). We prompt GPT-4 to do a pairwise ranking of GLAN and other models in comparison. The overall results and results across the 8 fields are presented in Table 11, where GLAN obtains higher GPT-4 scores than Orca2-7B, Mistral-7B Instruct and WizardLM-13B, despite using only 7B parameters. GLAN still

Table 9: Pairwise comparison on various difficulty levels between GLAN and other models on Evol-Instruct testset. The scores are the average gap of scores assigned by GPT-4, calculated as $\texttt{avg\_score(GLAN)} - \texttt{avg\_score}(x)$.

Difficulty	Ratio	LLaMA2-7B	Orca2-7B	Mistral-7B-Instruct	Wizard-13B-V1.2	GPT-3.5-turbo
1	5.1%	5.41	2.23	-0.37	-0.21	-2.41
2	8.7%	5.87	1.74	1.06	1.41	-1.18
3	12.4%	5.72	2.35	1.04	1.37	-1.14
4	10.5%	5.61	1.34	1.52	1.54	-0.92
5	4.1%	4.67	3.31	2.39	2.5	-0.45
6	19.3%	4.43	2.42	0.74	1.54	-1.36
7	11.0%	4.97	1.26	1.62	1.36	-0.41
8	17.9%	6.02	3.58	3.17	1.7	0.15
9	6.0%	6.35	4.2	1.36	0.9	-0.92
10	5.1%	5.14	-0.05	1.53	-0.54	-0.85
(1-5) Easy	41.00%	**5.46**	**2.19**	**1.13**	**1.32**	-1.22
(6-10) Hard	59.00%	**5.38**	**2.28**	**1.68**	**0.99**	-0.68

Table 10: Pairwise comparison on various skills between GLAN and other models on Evol-Instruct testset. The scores are the average gap of scores assigned by GPT-4, calculated as $\texttt{avg\_score(GLAN)} - \texttt{avg\_score}(x)$.

Skill	Ratio	LLaMA2-7B	Orca2-7B	Mistral-7B-Instruct	Wizard-13B-V1.2	GPT-3.5-turbo
Math	8.7%	6.58	2.16	2.41	2.46	-1.42
Code Generation	8.3%	6.16	3.87	4.22	2.59	-0.25
Writting	8.3%	5.2	0.79	-0.22	0.24	-1.1
Computer Science	6.9%	7.1	4.4	0.83	1.22	0.02
Reasoning	6.0%	6.3	2.52	3.38	3.02	0.62
Complex Format	5.5%	3.13	3.5	-0.17	2.41	-1.96
Code Debug	4.6%	5.85	2.3	1.4	0.2	-2.5
Common-Sense	4.1%	6.5	3.19	-1.33	-0.92	-2.78
Counterfactual	3.7%	7.06	2.15	3	1.5	0.72
Multilingual	3.2%	7.35	0.79	1.71	-0.68	-2.75
Roleplay	2.8%	7.08	2.25	3.5	0.92	-0.59
Biology	2.8%	6.66	2.75	1.46	-0.09	1.38
Technology	2.8%	-0.08	2.54	-3	-1.5	-2.75
Ethics	2.8%	6.59	3.38	2.41	5.42	-0.21
TruthfulQA	2.3%	3.1	3.7	-1.05	-1.3	-0.85
Sport	2.3%	4.3	0.55	-0.2	4.8	-0.3
Law	2.3%	7.7	4.65	5.85	1.7	0.2
Medicine	2.3%	3.9	-2.05	1.9	0.15	-1.25
Literature	2.3%	6.3	1.9	0.2	1.45	-0.15
Entertainment	2.3%	4.5	2.7	-3	1.9	-3.2
Art	2.3%	4.9	1	2.9	-0.85	-2.05
Music	2.3%	4.4	4.1	0.5	1.45	-2.3
Toxicity	1.8%	7.25	3.12	3.75	1.63	-1.32
Economy	2.3%	6	0.15	1.9	0	0
Physics	2.3%	6.8	2.5	4.35	3.65	-1
History	1.8%	4.12	-0.56	3.76	-0.31	0.12
Academic Writing	1.8%	6.76	6.37	2.44	1.37	0.62
Chemistry	0.9%	9.5	0.63	5.25	2.5	0.75
Philosophy	0.5%	11	-0.25	0.25	-0.25	0.5
Avg.(29 skills)	100%	5.42	2.24	1.41	1.16	-0.95

lag behind GPT-4. Detailed results for the 126 fine-grained disciplines can be found in Appendix A.9 (see Table 12 for more details). GLAN demonstrates its effectiveness on multiple domains (or disciplines) such as Mathematics, Physics, Chemistry, Computer science, Electrical, Mechanical, etc., indicating that smaller models may yield general improvements on various domains through strategic fine-tuning. Furthermore, it is noted that GLAN demonstrates less-than-ideal performance across distinct disciplines such as American history, Divinity, or Radiology. This observation underscores the potential for further refinement and development of our methodology within these domains.

A.9 GLAN-Test Results on Different Disciplines

Table 11: Pairwise comparison between GLAN and other models on GLAN-Test (the 126 disciplines are categorized into 8 fields for clarity of the illustration). The scores are the average gap of scores assigned by GPT-4, calculated as $\texttt{avg\_score(GLAN)} - \texttt{avg\_score}(x)$.

Field (Ratio)	**Orca2-7B**	**Mistral-7B-Instruct**	**WizardLM-13B-V1.2**	**GPT-4**
Academic-Humanities (15.9%)	0.79	0.25	0.02	-0.62
Academic-Social Science (7.9%)	1.22	0.21	0.09	-0.63
Academic-Natural Science (4.0%)	1.73	1.23	0.53	-0.5
Academic-Applied Science (42.1%)	1.58	0.32	0.08	-0.58
Academic-Formal Science (3.2%)	3.87	2.48	2.32	-0.55
Industry-Manufacturing (12.7%)	2.26	0.56	0.33	-0.43
Industry-Services (11.9%)	1.82	0.23	0.09	-0.5
Industry-Agriculture (2.4%)	1.2	0.46	0.13	-0.33
Overall (100.0%)	**1.61**	**0.43**	**0.19**	-0.55

Table 12: Pairwise comparison across 126 disciplines (or domains) on *GLAN-Test*. The scores are generated from the average gap between GLAN and other model x in assessment scores assigned by GPT-4, calculated as $\texttt{avg\_score(GLAN)} - \texttt{avg\_score}(x)$.

Discipline	**Orca-2-7b**	**Mistral-7B-Instruct-v0.1**	**WizardLM-13B-V1.2**	**GPT-4**
Avg.	1.61	0.43	0.19	-0.55
Advertising	1.92	0.46	0.21	-0.04
Aerospace industry	3.24	1.24	0.6	-0.42
Agriculture	2.44	0.04	-0.05	-0.48
American history	-0.49	-0.27	-0.76	-0.83
American politics	1.23	-0.3	-0.4	-0.87
Anthropology	0.59	0.17	0.06	-0.27
Applied mathematics	3.75	2.6	2.74	-0.47
Archaeology	2.59	-0.11	0.1	-0.56
Architecture and design	2.63	0.34	0.4	-0.37
Astronomy	1.01	0.83	0.03	-0.44
Automotive industry	1.27	0.71	0.46	-0.06
Biblical studies	-0.05	0.33	-0.47	-0.65
Biology	1.09	0.22	-0.09	-0.17
Business	3.61	1.14	0.88	-0.26
Chemical Engineering	3.15	1.6	1.18	-0.77
Chemistry	3.06	2.09	0.8	-0.87
Civil Engineering	1.94	0.74	0.75	-0.25
Clinical laboratory sciences	1.32	0.94	-0.11	-0.47
Clinical neuropsychology	2.15	0.29	0.25	-0.4
Clinical physiology	2.07	0.41	0.51	-0.08
Communication studies	0.3	0.26	-0.15	-0.3
Computer science	4.29	1.45	1.9	-0.33
Cultural industry	3.15	0.44	0.05	-0.36
Dance	2.11	0.21	0.4	-0.47
Dentistry	1.67	0.66	0.48	0.01
Dermatology	2.12	0.55	-0.05	-0.65
Divinity	-0.34	-0.17	-0.48	-0.89
Earth science	0.39	0.44	-0.08	-0.33
Economics	2.62	0.96	0.62	-0.4
Education	2.67	0.42	0.2	-0.84
Education industry	2.19	0.4	0.56	-1.33
Electric power industry	3.23	1.31	0.39	-0.79
Electrical Engineering	3.81	1.26	1.41	-0.34
Emergency medicine	2.04	0.44	-0.18	-0.86
Energy industry	3.59	0.98	0.54	-0.22
Environmental studies and forestry	0.12	0.41	0.1	-0.45
Epidemiology	3.02	0.52	0.33	-0.46
European history	0.14	0.62	0.15	-0.18
Fashion	2.5	0.66	0.47	-0.53
Film	0.76	0.45	-0.16	-0.78
Film industry	1.58	0.46	0.25	-0.59
Fishing industry	1.67	1	0.57	-0.09
Floral	1.92	0.89	0.58	-0.09
Food industry	3.64	0.12	0.14	-0.42
Foreign policy	2.4	0.49	0.16	-0.46
Geography	0.88	0.6	0.28	-0.66
Geriatrics	2.19	-0.32	-0.56	-0.71
Gynaecology	1.05	-0.27	-0.26	-0.67
Healthcare industry	1.62	-0.25	0.14	-0.5
Hematology	0.35	0.32	-0.05	-0.72
History	0.75	0.54	-0.04	-0.38
Holistic medicine	0.85	0.48	0.26	-0.27
Hospitality industry	2.36	0.48	0.28	-0.07
Housing	4.04	0.15	-0.22	-0.62
Industrial robot industry	3.84	1.22	0.84	-0.71
Infectious disease	1.76	0.14	0.18	-0.56
Insurance industry	2.67	0.42	0.61	-0.4
Intensive care medicine	1.11	0.56	0.08	-0.33
Internal medicine	1.02	0.45	-0.01	-0.42
Journalism	2.77	-0.13	-0.21	-0.69
Languages and literature	0.45	0.05	-0.39	-0.84
Law	0.42	0.39	0.04	-0.49
Leisure industry	1.49	0.12	-0.09	-0.49
Library and museum studies	1.52	0.5	0.33	-0.32

Discipline	Orca-2-7b	Mistral-7B-Instruct-v0.1	WizardLM-13B-V1.2	GPT-4
Linguistics	0.39	0.38	-0.12	-0.96
Logic	2.95	1.56	1.62	-0.79
Materials Science and Engineering	1.71	0.97	0.54	-0.91
Mathematics	4.69	3.81	2.73	-0.61
Mechanical Engineering	2.25	1.71	1.15	-0.95
Medical toxicology	0.62	0	0.11	-1.01
Medicine	1.49	0.93	0.36	-0.37
Military sciences	0.42	0.53	0.17	-0.45
Mining	3.17	0.32	0.41	-0.61
Music	2.85	0.38	1.07	-0.05
Music industry	2.05	-0.03	-0.08	-0.8
Nursing	1.49	0.14	-0.12	-0.59
Nutrition	1.15	-0.2	-0.13	-0.65
Obstetrics	1.49	0.08	-0.43	-0.53
Ophthalmology	0.97	0.01	-0.47	-0.97
Otolaryngology	1.51	-0.44	-0.29	-1.11
Pathology	0.23	0.35	0.19	-0.72
Pediatrics	1.62	0.55	-0.34	-0.47
Performing arts	0.38	0.09	-0.36	-1.06
Petroleum industry	3.12	0.44	0.08	-0.54
Pharmaceutical industry	2.75	0.41	0.4	-0.46
Pharmaceutical sciences	0.77	0.19	0.16	-0.8
Philosophy	0.51	0.25	0.49	-0.64
Physics	3.15	2.67	2.05	-0.73
Political science	0.04	-0.05	-0.31	-0.91
Prehistory	0.35	0.19	0.05	-0.41
Preventive medicine	2.69	0.57	0.09	-0.36
Psychiatry	2.93	0.27	-0.07	-0.32
Psychology	0.53	-0.02	-0.3	-0.96
Public administration	0.94	-0.27	0.1	-1.2
Public health	1.21	0.07	0.22	-0.56
Public policy	0.78	-0.06	-0.28	-0.92
Pulp and paper industry	1.13	0.63	0.57	-0.25
Radiology	-0.17	-0.19	-0.82	-0.62
Real estate industry	1.01	0.02	-0.12	-0.5
Religious Studies	0.38	0	-0.32	-0.63
Retail industry	1.1	-0.25	-0.37	-0.6
Semiconductor industry	1.49	0.64	0.71	-0.42
Sexology	1.81	-0.44	-0.37	-0.96
Shipbuilding industry	1.54	0.37	0.42	-0.32
Social work	0.93	-0.42	-0.53	-0.77
Sociology	1.49	0.21	0.76	-0.3
Steel industry	0.88	0.45	0.09	-0.34
Surgery	0.86	-0.02	-0.35	-0.73
Systems science	1.9	0.56	0.41	-0.45
Telecommunications industry	1.81	0.4	0.39	-0.27
Television	0.37	-0.33	-0.69	-1
Textile industry	0.82	-0.26	-0.68	-0.59
Theatre	0.31	-0.27	-0.34	-1.07
Theology	-0.38	0.37	-0.45	-0.54
Tobacco industry	0.59	-0.13	-0.48	-0.67
Transport industry	1.19	-0.33	-0.36	-0.56
Transportation	1.74	0.26	0.17	-0.74
Urology	0.05	-0.29	-0.36	-0.64
Veterinary medicine	-0.14	0.36	-0.31	-0.62
Video game industry	1.67	0.2	-0.24	-0.62
Visual arts	0.98	0.22	0.26	-0.56
Water industry	0.9	-0.11	-0.09	-0.51
Wood industry	1.36	0.5	0.31	-0.25

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See Abstract and Section 1.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Section 5 and Appendix A.1

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: No theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: In Section 2 and 3.1, we provide a detailed description of the data generation process. Although we haven't shared the original prompts yet, they are quite simple and customizable. Besides, we are actively working to gain authorization to release them as soon as possible.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: While we are temporarily unable to provide open access to the data and code, we are actively working to gain the necessary authorization to release these resources. Once obtained, we will ensure that all data and code, along with detailed instructions, are made available to faithfully reproduce the main experimental results.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: See Section 3.2, 3.3

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: We did not include error bars in the experiments due to the high computational demands.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We included compute resources in Section 3.2.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This study strictly adheres to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Appendix A.2

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [No]

    Justification: To ensure future responsible release, we are still in the process of implementing comprehensive safeguards.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All existing assets used in this paper are properly credited. The license and terms of use are properly respected.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Once authorization is obtained, we will ensure that comprehensive documentation is provided alongside the assets to facilitate their proper use and understanding.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.