Synthesize, Partition, then Adapt: Eliciting Diverse Samples from Foundation Models

Yeming Wen*& Swarat Chaudhuri
Department of Computer Science
The University of Texas at Austin

Abstract

Presenting users with diverse responses from foundation models is crucial for enhancing user experience and accommodating varying preferences. However, generating multiple high-quality and diverse responses without sacrificing accuracy remains a challenge, especially when using greedy sampling. In this work, we propose a novel framework, Synthesize-Partition-Adapt (SPA), that leverages the abundant synthetic data available in many domains to elicit diverse responses from foundation models. By leveraging signal provided by data attribution methods such as influence function, SPA partitions data into subsets, each targeting unique aspects of the data, and trains multiple model adaptations optimized for these subsets. Experimental results demonstrate the effectiveness of our approach in diversifying foundation model responses while maintaining high quality, showcased through the HumanEval and MBPP tasks in the code generation domain and several tasks in the natural language understanding domain, highlighting its potential to enrich user experience across various applications.

1 Introduction

Transformer-based foundation models have revolutionized the fields of natural language processing (NLP) and code generation with their remarkable abilities a wide range of understanding and generation tasks (Vaswani et al., 2017; Devlin et al., 2019; Brown et al., 2020; Chen et al., 2021). These models are typically pre-trained on vast amounts of text data and then undergo instruction fine-tuning — a post-training process — to improve alignment with user expectations and enhance the overall user experience (Ouyang et al., 2022). Due to the high cost of human-annotated data, synthetically generated datasets (Wang et al., 2022b) such as OSS-Instruct (Wei et al., 2023) and Alpaca (Taori et al., 2023) have become an important component of instruction tuning, demonstrating strong effectiveness in improving foundation model performance.

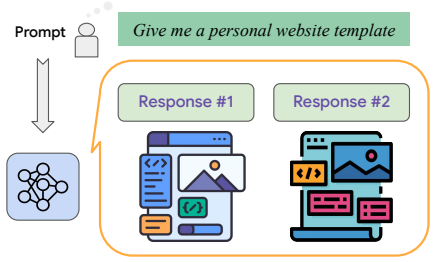

Figure 1: A user is expecting two diverse templates from the foundation model.

To date, these synthetic datasets have been primarily used to align foundation models with instructions or to induce certain preferable behaviors. In this paper, we focus on a different use of synthetic data: in improving the *diversity* of foundation models' outputs. Diversifying the generated responses is crucial for accommodating diverse user preferences and enhancing user satisfaction. Consider the scenario illustrated in Fig. 1, where a user prompts a foundation model with "Give me a personal website template". In this case, we would prefer the model to generate two diverse templates while

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

*vwen@utexas.edu

https://doi.org/10.52202/079017-1911

maintaining good quality, providing users with a variety of styles and layouts. Conventional methods for improving diversity, such as temperature sampling (Ackley et al., 1985; Hinton et al., 2015; Wang et al., 2019, 2023), rely on sampling techniques that anneal the probabilistic distribution of outputs. These methods often trade off diversity for quality, as the generated responses may deviate from the learned distribution and produce hallucination or less coherent outputs (Lee, 2023). Moreover, these techniques are not applicable when using greedy sampling, which is often preferred for its simplicity and precision. This highlights the need for approaches that not only align foundation model outputs with user expectations but also elicit diverse responses without sacrificing quality.

In this paper, we present a framework, *Synthesize-Partition-Adapt* (SPA), that achieves these objectives. The framework partitions the synthetic data and adapts foundation models to these partitions in the post-training stage. By leveraging the inherent diversity in the training data, this approach can generate diverse responses without compromising accuracy. The potential of partition-and-adapt approach is further amplified by the increasing availability of large-scale synthetic datasets because the utility of instruction-tuning a single model on the entire dataset diminishes. In particular, we show that influence function (Koh & Liang, 2017) can be an effective signal to partition synthetic datasets into subsets, each targeting unique aspects that elicit distinct model behaviors. However, SPA is not limited to influence function and can be extended to other partitioning strategies. By training multiple adaptations on these subsets using parameter-efficient fine-tuning techniques, such as LoRA (Hu et al., 2021), we enable the generation of diverse and accurate responses.

To demonstrate the effectiveness of our approach, we conduct experiments on a range of tasks in both the code generation and natural language understanding domains. We evaluate our method on the HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) datasets for code generation, as well as several natural language understanding tasks. The results showcase the ability of our approach to diversify model responses while maintaining high accuracy, highlighting its potential to enrich user experience across various applications.

To summarize, the main contributions of this paper are as follows:

- We propose SPA, a novel framework that leverages synthetic data, data partitioning, and model adaptation to elicit diverse responses from foundation models.
- We demonstrate the effectiveness of SPA in diversifying foundation model responses while maintaining sampling quality through extensive experiments on code generation and natural language understanding tasks.
- We highlight the potential of SPA to leverage the increasing availability of large-scale synthetic datasets for improving the diversity of foundation model responses.

2 Background

2.1 Instruction Fine-tuning

By fine-tuning foundation models on human-annotated data that demonstrates desired behaviors, instruction tuning aims to improve the alignment between the model's outputs and the user's intentions (Ouyang et al., 2022; Wei et al., 2021; Sanh et al., 2022). Let $\mathcal{D} = (x_i, y_i)_{i=1}^{N}$ denote a dataset of input-output pairs, where x_i represents the input instruction and y_i represents the corresponding desired output. The objective of instruction tuning is to minimize the following loss function: $\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N} \log_\theta(y_i|x_i)$ where θ represents the parameters of the foundation model, and $p_\theta(y_i|x_i)$ is the probability of generating the target response y_i given the input x_i.

Classical approaches for instruction tuning typically require a substantial amount of parallel labeled data of NL intents and gold model responses. Collecting large-scale, high-quality annotated datasets is often time-consuming and expensive. To mitigate this issue, researchers have explored the use of synthetic data for instruction tuning. By leveraging techniques such as data augmentation (Wei & Zou, 2019; Sennrich et al., 2016) and back-translation (Edunov et al., 2018), synthetic data can be generated at scale, providing a cost-effective alternative to human-annotated datasets. Furthermore, synthetic instruction-following data can also be generated from the foundation model itself (Wang et al., 2022a; Honovich et al., 2022; Taori et al., 2023; Peng et al., 2023; Wen et al., 2024, *inter alia*).

2.2 Data Attribution and influence function

Data attribution methods aim to quantify the importance or influence of individual training points on a model's predictions. One such method is the influence function (Koh & Liang, 2017). Formally, let $\mathcal{L}(\theta)$ denote the loss function of the model, where θ represents the model parameters. The influence of a training point z on the model's parameters θ is given by $\mathcal{I}(z) = -H_\theta^{-1}\nabla_\theta\mathcal{L}(z,\theta)$. where H_θ is the Hessian matrix of the loss function with respect to the model parameters, and $\nabla_\theta\mathcal{L}(z,\theta)$ is the gradient of the loss function with respect to the model parameters, evaluated at the training point z. Next, the influence of elevating the weight of z on the loss associated with a test point z_{test} is:

$$\mathcal{I}(z, z_{test}) = -\nabla_\theta\mathcal{L}(z_{test}, \hat{\theta})^\top H_{\hat{\theta}}^{-1}\nabla_\theta\mathcal{L}(z, \hat{\theta}) \tag{1}$$

It is impossible to calculate the full Hessian H_θ^{-1} matrix in deep neural networks. Koh & Liang (2017) developed a simple and efficient implementation that requires only oracle access to gradients and Hessian-vector products. This implementation makes it feasible to apply influence function to large-scale models. However, the vast parameter space of foundation models presents an even greater challenge, rendering the direct application of influence function impractical. In response to this, recent advancements in Grosse et al. (2023) have further refined the methodology, enabling the application of influence function to large language models.

3 Problem Formulation

Given a user input $\mathbf{x}$, our goal is to generate a diverse set of high-quality responses $\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_K$ from a foundation model $\mathcal{M}$. One approach to generating diverse responses is to sample from the model multiple times using techniques like temperature sampling: $\mathbf{y}_k = \mathcal{M}(\mathbf{x}; \theta, \tau)$, where $k = 1, 2, ..., K$ and θ represents the model parameters and τ is the temperature hyperparameter. However, this approach often trades off diversity for quality as studied in Chung et al. (2023). An alternative approach is to train multiple model adaptations $\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_K$ and sample one response from each adaptation:

$$\mathbf{y}_k = \mathcal{M}_k(\mathbf{x}; \theta_k), \quad k = 1, 2, ..., K, \tag{2}$$

where θ_k represents the parameters of the k-th model adaptation. By training each adaptation on a different subset of the data that captures unique aspects and yields distinct model behaviors, we can generate diverse responses while maintaining their quality. Moreover, this approach allows us to elicit diverse samples even with greedy sampling, which is often preferred for maximum precision.

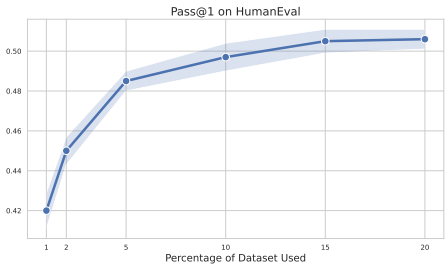

Figure 2: *pass*@1 on HumanEval after fine-tuning on some percentage of OSS-Instruct dataset (Wei et al., 2023) using LoRA. The plot demonstrates the diminishing returns observed with increasing amounts of data used for parameter efficient fine-tuning.

Traditionally, training multiple model adaptations has been considered unfavorable due to the repeated training process, which can be computationally expensive and time-consuming. However, with the increasing popularity of instruction tuning, it has become common practice to go through a post-training stage using instruction data before deploying the model to users. This post-training stage presents an opportunity to train multiple model adaptations without incurring significant additional costs, making the approach more feasible and practical in real-world scenarios.

As the volume of synthetic data grows, the utility of fine-tuning a single model on the entire dataset diminishes due to the diminishing returns in the post-training stage, as demonstrated in Fig. 2. The *pass*@1 accuracy after fine-tuning on the entire synthetic dataset using LoRA is roughly the same as only consuming 15% of the data[2]. This creates an opportunity to leverage the abundant synthetic data to train multiple model adaptations, each specializing in a specific subset of the data. In this work, we propose the Synthesize, Partition, then Adapt (SPA) framework to address the diverse response generation problem. SPA leverages existing synthetic datasets, data partitioning techniques, and parameter-efficient fine-tuning methods to train multiple model adaptations. By sampling from the collection of these adaptations, SPA generates diverse and high-quality responses, enhancing the overall user experience.

[2]This does not suggest full parameter fine-tuning shares the same diminishing return.

https://doi.org/10.52202/079017-1911

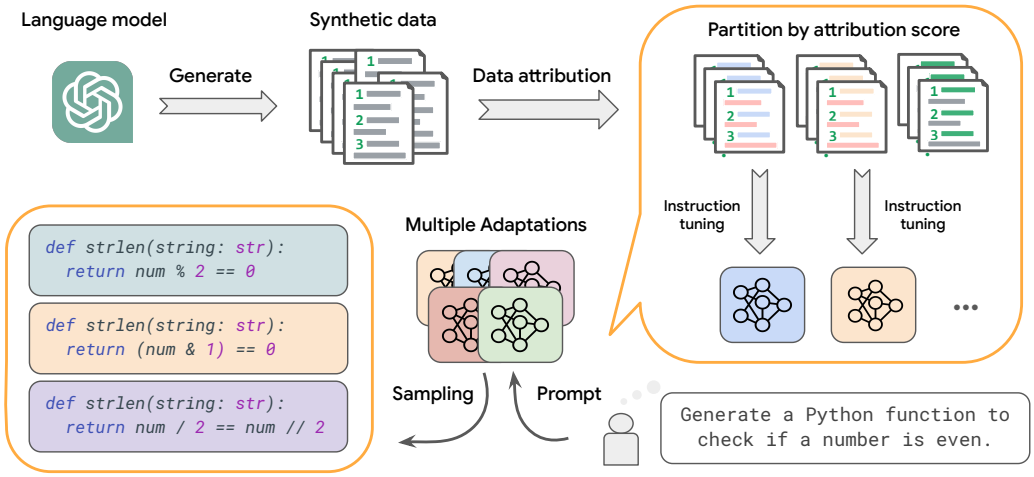

Figure 3: An illustration of the Synthesize, Partition, then Adapt (SPA) framework. SPA partitions synthetic dataset according to data attribution scores, which can be obtained using various methods such as influence function or lexical overlap. Multiple foundation model adaptations are then trained on each subset. Sampling from the collection of these model adaptations can present users with diverse responses. SPA is not limited to a specific attribution method.

4 Partitioning Synthetic Data and Training Adaptations

We present the technical details of our proposed SPA framework for training multiple adaptations. We leverage an existing synthetic dataset $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$ for the purpose of this study. The use of an existing synthetic dataset allows us to focus on the effectiveness of the Partition then Adapt steps in eliciting diverse samples, while demonstrating the flexibility of our framework to work with various synthetic datasets. Fig. 3 provides an overview of the framework. After obtaining the synthetic data, our approach consists of three main steps: (1) computing data attribution scores for synthetic data points, (2) partitioning the synthetic dataset based on these scores, and (3) training multiple foundation model adaptations using parameter-efficient fine-tuning techniques like LoRA.

4.1 Computing Data Attribution Scores

Consider a pre-trained foundation model $\mathcal{M}$ with parameters θ. Our goal is to leverage the synthetic dataset $\mathcal{D}$ to train a set of K foundation model adaptations $\{\mathcal{M}_k\}_{k=1}^K$. Each adaptation focuses on a specific subset of the data that yields similar model behaviors. To partition the synthetic dataset, we employ data attribution methods that measure the importance of each training point to the model's predictions. Although we use influence function as an example to label the data, the SPA framework is not limited to influence function and can be extended to other data attribution methods, such as lexical overlap or TRAK (Park et al., 2023). To calculate the influence function, we first fine-tune the pre-trained foundation model $\mathcal{M}$ on the synthetic dataset $\mathcal{D}$. The fine-tuning process optimizes the model parameters θ to minimize the loss function $\mathcal{L}(\theta)$ on the synthetic dataset using LoRA (Hu et al., 2021): $\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^N \ell(\mathbf{y}_i, \mathcal{M}(\mathbf{x}_i; \theta))$, where $\ell(\cdot, \cdot)$ is a suitable loss function, such as cross-entropy loss for language modeling tasks. This fine-tuning process yields the optimized model parameters $\hat{\theta}$.

Next, we select a set of M test queries $\{(\mathbf{x}_t^{(m)}, \mathbf{y}_t^{(m)})\}_{m=1}^M$, which can be a collection of questions requiring various expertise knowledge to solve. For each test query $(\mathbf{x}_t^{(m)}, \mathbf{y}_t^{(m)})$, we compute the influence score of each synthetic data point $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}$ using Eq. (1):

$$\mathcal{I}((\mathbf{x}_i, \mathbf{y}_i), (\mathbf{x}_t^{(m)}, \mathbf{y}_t^{(m)})) = -\nabla_\theta \ell(\mathbf{y}_t^{(m)}, \mathcal{M}(\mathbf{x}_t^{(m)}; \hat{\theta}))^\top H_{\hat{\theta}}^{-1} \nabla_\theta \ell(\mathbf{y}_i, \mathcal{M}(\mathbf{x}_i; \hat{\theta})). \quad (3)$$

To efficiently compute the influence scores, we employ the stochastic estimation method proposed by Koh & Liang (2017), which approximates the inverse Hessian-vector product using conjugate

gradients. Although even this method is generally infeasible in foundation models due to their vast parameter space, the use of LoRA (Hu et al., 2021) makes it feasible by significantly reducing the number of trainable parameters. The computational cost of estimating the influence of a test query between the entire dataset $\mathcal{D}$ is the same as calculating the gradient of $\mathcal{D}$. Another option to address this issue is to use the K-FAC approximation of the Hessian, as proposed by Grosse et al. (2023). We focus on the LoRA approach and leave the exploration of K-FAC and other approximations for future work.

4.2 Partitioning Synthetic Dataset

After computing the data attribution scores for each synthetic data point with respect to the M test points, we obtain an influence matrix $\mathbf{I} \in \mathbb{R}^{N \times M}$, where $\mathbf{I}_{i,m}$ represents the attribution score of the i-th synthetic data point for the m-th test point. To partition the synthetic dataset $\mathcal{D}$ into K subsets $\{\mathcal{D}_k\}_{k=1}^{K}$, a clustering algorithm can be applied to solve the following objective:

$$\min_{\{\mathcal{D}_k\}_{k=1}^{K}} \sum_{k=1}^{K} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}_k} \sum_{(\mathbf{x}_j, \mathbf{y}_j) \in \mathcal{D}_k} |\mathbf{I}_{i,:} - \mathbf{I}_{j,:}|_2^2 , \tag{4}$$

where $\mathbf{I}_{i,:}$ denotes the i-th row of the influence matrix $\mathbf{I}$, subject to $\bigcup_{k=1}^{K} \mathcal{D}_k = \mathcal{D}$ and $\mathcal{D}_k \cap \mathcal{D}_{k'} = \emptyset$ for all $k \neq k'$. In this work, we assume partitions are disjoint for the simplicity of the study.

The clustering algorithm assigns each synthetic data point $(\mathbf{x}_i, \mathbf{y}_i)$ to one of the K subsets based on the similarity of its influence scores across the M test points. This partitioning ensures that data points within each subset have similar impacts on the model's predictions. The choice of the clustering algorithm may depend on the specific characteristics of the dataset. For simplicity and ease of implementation, in this study, we use a ranking heuristic to partition the synthetic dataset. The details of this heuristic will be explained in the experiment section §5.1. However, it is important to note that our SPA framework is not limited to any specific clustering algorithm.

4.3 Training Multiple Adaptations with LoRA

Once the synthetic dataset is partitioned into K subsets, we train a foundation model $\mathcal{M}_k$ for each subset $\mathcal{D}_k$ using parameter-efficient fine-tuning techniques like LoRA (Hu et al., 2021). LoRA adapts the pre-trained foundation model parameters θ by learning low-rank matrices $\mathbf{A}_k \in \mathbb{R}^{r \times d}$ and $\mathbf{B}_k \in \mathbb{R}^{d \times r}$ for each weight matrix $\mathbf{W} \in \mathbb{R}^{d \times d}$ in the pre-trained foundation model, where $r \ll d$ is the rank of the adaptation matrices.

The adapted weight matrix $\mathbf{W}_k$ for the foundation model adaptation $\mathcal{M}_k$ is computed as: $\mathbf{W}_k = \mathbf{W} + \mathbf{B}_k \mathbf{A}_k$. During the fine-tuning process, only the adaptation matrices $\mathbf{A}_k$ and $\mathbf{B}_k$ are learned, while the pre-trained weights $\mathbf{W}$ remain frozen. This significantly reduces the number of trainable parameters, making it feasible to train multiple foundation model adaptations with limited computational resources. The training objective for each foundation model adaptation $\mathcal{M}_k$ is given by $\min_{\theta_k} \frac{1}{|\mathcal{D}_k|} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}_k} \ell(\mathbf{y}_i, \mathcal{M}_k(\mathbf{x}_i; \theta_k))$ where θ_k represents the parameters of $\mathcal{M}_k$, which include the pre-trained weights θ and the LoRA adaptation matrices $\mathbf{A}_k, \mathbf{B}_k$. By training multiple foundation model adaptations using LoRA, we can efficiently adapt the pre-trained foundation model to different subsets of the synthetic data, each focusing on a specific aspect of the data that yields similar model behaviors. This approach enables the creation of a diverse set of specialized models that capture different knowledge or expertise present in the synthetic data, while leveraging the knowledge acquired during the pre-training phase.

Inference with Multiple Adaptations During inference, given a user input $\mathbf{x}$, our goal is to generate a diverse set of responses by leveraging the multiple foundation model adaptations trained on different subsets of the synthetic data. To achieve this, we randomly sample a foundation model adaptation $\mathcal{M}_k$ from the set of K adaptations $\{\mathcal{M}_k\}_{k=1}^{K}$ and generate the output $\mathbf{y}$ using the selected adaptation. By randomly sampling from the set of adaptations, we can generate a diverse set of responses for the user input $\mathbf{x}$. This approach ensures that the generated responses are not only diverse but also maintain reasonable quality. It is worth noting that this approach is compatible with various sampling techniques, such as temperature scaling, top-k and top-p sampling, which can further enhance the diversity of the generated responses.

https://doi.org/10.52202/079017-1911

To generate multiple diverse responses for the user input $\mathbf{x}$, we can repeat the random sampling process multiple times, each time selecting a different adaptation and generating a response. This allows us to present the user with a set of alternative responses that capture different perspectives or styles, enhancing the overall user experience. Unlike temperature sampling, which can degrade the quality of the generated responses, our approach maintains the quality of each response by leveraging the specialized knowledge captured by each adaptation. Moreover, our approach can generate diverse samples even when greedy sampling is used.

5 Experiments

In this section, we present the experimental setup and results for evaluating the effectiveness of our proposed SPA framework in improving the diversity of foundation model outputs. We conduct experiments on both code generation tasks such as HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) and several natural language understanding tasks.

5.1 Experimental Setup

Base Model and Synthetic Dataset For the code generation experiments, we use CodeLLaMA 7B (Rozière et al., 2023) as the base foundation model. CodeLLaMA is a state-of-the-art language model specifically designed for code-related tasks, pre-trained on a large corpus of code and natural language data. For the synthetic dataset, we utilize the OSS-Instruct dataset (Wei et al., 2023), which consists of 75,000 code-related question-answering pairs generated by GPT-3.5 Turbo (OpenAI, 2023). In the natural language understanding domain, we employ Llama-2 13B (Touvron et al., 2023) as the base foundation model. Llama-2 is a powerful language model trained on a diverse range of web-scale data, demonstrating strong performance across various natural language understanding tasks. For the synthetic dataset, we use Platypus (Lee et al., 2023), which focuses on improving LLMs' STEM and logic knowledge. Platypus consists of a curated sub-selection of public text datasets, comprising approximately 25,000 question-answer pairs.

Data Attribution Scores We compare two methods for computing data attribution scores: influence function and lexical overlap.

For the influence-based method, we hand-write 12 examples that cover a wide range of knowledge for each domain. For each of these examples, we calculate the influence score with respect to each training example in the corresponding synthetic dataset using Equation 3. We then select the top 8 test queries whose distribution of influence scores over the dataset has the highest variance. This ensures that the selected test queries have diverse impacts on the synthetic dataset, capturing different aspects of the domain knowledge. The resulting influence matrices $\mathbf{I}_{code} \in \mathbb{R}^{8\times 75{,}000}$ and $\mathbf{I}_{nlu} \in \mathbb{R}^{8\times 25{,}000}$ are used for partitioning the OSS-Instruct and Platypus datasets, respectively.

For the lexical overlap method, we compute the BM25 score (Robertson et al., 1994) between each training example and the hand-written test queries. The BM25 score is calculated as follows:

$$I(z, z_{query}) = \sum_{t \in z_{query}} \log \frac{N+1}{N_t} \cdot \left(\frac{(k_1+1) f(z,t)}{k_1 \left((1-b) + b \cdot \frac{L(z)}{L_{avg}} \right) + f(z,t)} + 1 \right) \tag{5}$$

where $f(z,t)$ is the overlap count, N is the number of training examples, $L(z)$ is the length of the example, and L_{avg} is the average example length. We adopted the framework and the hyperparameters in Lv & Zhai (2011). While we focus on influence function in this work, exploring the effectiveness of alternative data attribution methods like BM25 could be an interesting direction for future research. More details are provided in Appendix A.

Partitioning the Synthetic Datasets To train multiple foundation model adaptations, we first set the hyperparameter K, which represents the total number of adaptations. We use $K = 8$ for both code generation and natural langauge understanding domain. For each data point in the synthetic dataset, we aim to find the test queries that provides the most influence. Formally, for each synthetic data point $(\mathbf{x}_i, \mathbf{y}_i)$, we assign it to the subset $\mathcal{D}_k^*$ corresponding to the test point with the highest influence score or the BM25 score: $k^* = \arg\max_{k \in \{1,\ldots,K\}} \mathbf{I}_{k,i}$. where $\mathbf{I}_{k,i}$ represents either the influence matrix or the BM25 score matrix. This process partitions the OSS-Instruct dataset

Methods	HUMANEVAL				MBPP			
	pass@1	*pass*@5	diversity	avg. KL	*pass*@1	*pass*@5	diversity	avg. KL
Single ($\tau = 0.1$)	50.02	56.42	0.58	NA	60.15	64.16	0.53	NA
Random ($\tau = 0$)	50.15	63.10	0.69	0.008	**60.65**	70.42	0.64	0.014
Lexical ($\tau = 0$)	**50.30**	66.74	0.78	0.011	60.33	71.17	0.71	0.018
Influence ($\tau = 0$)	50.15	**69.05**	**0.85**	**0.017**	60.46	**73.68**	**0.78**	**0.020**

Table 1: Results on the HumanEval and MBPP. τ denotes the temperature used for sampling. SPA with influence function achieves the best performance in terms of diversity score and avg. KL divergence) while maintaining comparable *pass*@1 performance to the single adaptation baseline. *pass*@5 measures sample quality but also has a positive correlation with diversity.

into K groups for code generation and the Platypus dataset into K groups for natural language understanding. Each group is associated with a specific test example that has the highest influence on the data points within the group.

With the partitioned synthetic dataset, we train K model adaptations using the LoRA technique, as described in §4.3. Each adaptation $\mathcal{M}_k$ is trained on the corresponding subset $\mathcal{D}_k$ of the synthetic dataset, focusing on the specific coding knowledge captured by the associated test point.

Evaluation Metrics We use the following two metrics to assess the diversity:

1. Average KL Divergence: Let P_i and P_j be the probability distributions of the generated responses from two model adaptations i and j, respectively. The KL divergence between P_i and P_j is defined as $D_{KL}(P_i \parallel P_j) = \sum_x P_i(x) \log \frac{P_i(x)}{P_j(x)}$. The average KL divergence is calculated by averaging the pairwise KL divergence between all possible pairs of model adaptations. A higher average KL divergence indicates greater diversity among the model adaptations,

$$\text{Average KL Divergence} = \frac{1}{\binom{K}{2}} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} D_{KL}(P_i \parallel P_j) \tag{6}$$

2. Sample Diversity: The average KL divergence evaluates the diversity at the distributional level. We also consider the sample diversity which measures the uniqueness of individual responses. We calculate the diversity score among K randomly generated samples for each problem. The diversity score is defined as the proportion of unique samples within the generated set. Specifically, it is calculated by taking one minus the ratio of the number of duplicate pairs to the total number of generated pairs.

Baselines We consider two baselines in the evaluation: (1) **Single Adaptation**, where a single model adaptation is trained on the entire synthetic dataset using LoRA, and (2) **Multiple Adaptations (random)**, where multiple adaptations are trained on randomly partitioned subsets of the synthetic dataset using LoRA. Hyperparameters used to train adaptations are provided in Appendix A.

5.2 Code Generation Results

In the code generation domain, we evaluate the performance of our proposed methodology on two popular code generation benchmarks: HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021). HumanEval consists of 164 hand-written programming problems with corresponding test cases, while MBPP contains 399 held-out programming problems collected from online resources[3]. These benchmarks assess the ability to generate functionally correct code.

***pass*@k metric** In addition to the diversity metrics, we also evaluate the sample quality by *pass*@1 and *pass*@5, measuring the percentage of problems for which at least one of the k generated samples passes all the test cases. Note that the *pass*@5 metric has a strong correlation to the diversity of the samples. More diverse samples generally lead to higher *pass*@5 for $k > 1$.

Tab. 1 presents the evaluation results of our SPA framework and the baselines on the HumanEval and MBPP benchmarks. For the multiple adaptation methods, including random partitioning, lexical

[3]We used the evalplus (Liu et al., 2023) framework to evaluate samples.

https://doi.org/10.52202/079017-1911

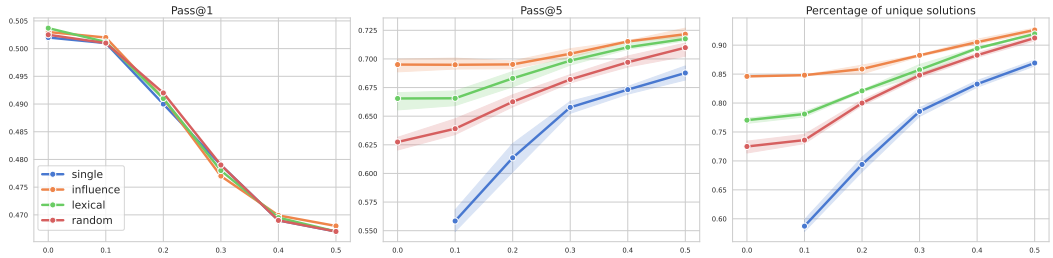

Figure 4: How sampling temperature affects *pass*@1, *pass*@5, and Diversity Score for different methods on the HumanEval benchmark. The results are averaged over 4 checkpoints.

overlap, and influence function, we use greedy decoding ($\tau = 0$) to generate samples. For the single adaptation baseline, we use a temperature of $\tau = 0.1$ to induce some diversity in the generated samples, as greedy decoding would not produce any diversity in this case.

Our primary focus is on comparing the diversity metrics, namely the Diversity Score and the Average KL Divergence (avg. KL), across the different methods. SPA with influence function achieves the highest Diversity Scores of 85% and 78% on HumanEval and MBPP, respectively, indicating that the generated samples are more unique and diverse compared to the other methods. Similarly, SPA with influence function yields the highest Average KL Divergence of 0.017 and 0.020 on HumanEval and MBPP, demonstrating greater diversity at the distributional level.

The random partitioning and lexical overlap approaches also improve upon the single adaptation baseline in terms of diversity metrics, but to a lesser extent than influence function. In particular, the lexical overlap induces more diversity than the random adaptations baseline. This suggests that even simpler data attribution methods can be beneficial for enhancing diversity when training multiple specialized adaptations. It is worth noting that the *pass*@5 scores, while primarily measuring sample quality, also have a positive correlation with diversity. SPA with influence function achieves the highest *pass*@5 scores of 69.05% and 73.68% on HumanEval and MBPP, indicating that the generated samples not only exhibit greater diversity but also maintain high quality.

In summary, these results underscore the effectiveness of our SPA framework in generating diverse code samples without compromising quality. By leveraging influence function for data partitioning and training multiple adaptations using LoRA, SPA enables the generation of diverse and accurate code solutions, even when using greedy decoding. We also showed that training more adaptations than 8 did not lead to more diversity in Appendix B.

Impact of Temperature Fig. 4 presents the impact of temperature on *pass*@1, *pass*@5, and Diversity Score for different methods on the HumanEval benchmark. The first plot shows that all methods, including Single, Random, Lexical, and Influence, exhibit similar patterns in terms of *pass*@1 performance. They achieve maximum accuracy (around 50.2%) when $\tau = 0$ and gradually decrease to approximately 46.5% as the temperature increases to 0.5.

However, both *pass*@5 and Diversity Score improve for all methods as the temperature increases, which is expected as higher temperatures encourage the model to generate more diverse samples. Notably, SPA with influence function (Influence) maintains its advantage over other methods across all temperature values, outperforming Single, Random, and Lexical methods. Although the performance gap between Influence and other methods narrows as the temperature increases due to the inherent diversity promotion of higher temperatures, Influence still maintains a lead at $\tau = 0.5$.

5.3 Natural Language Understanding Results

To demonstrate the effectiveness of SPA in the natural language understanding domain, we evaluate its performance on several diverse tasks, including Big-Bench Hard (BBH) (Suzgun et al., 2022), GPQA (Rein et al., 2023), MMLU (Hendrycks et al., 2020), and WinoGrande (Sakaguchi et al., 2019). For tasks that involve multiple-choice questions, we asked the model to continue generating text even after producing an answer choice for the purpose of measuring sample diversity. As shown in Fig. 5, SPA with influence function consistently achieves higher diversity scores and average KL divergence compared to the lexical overlap and random adaptation across all tasks. Interestingly,

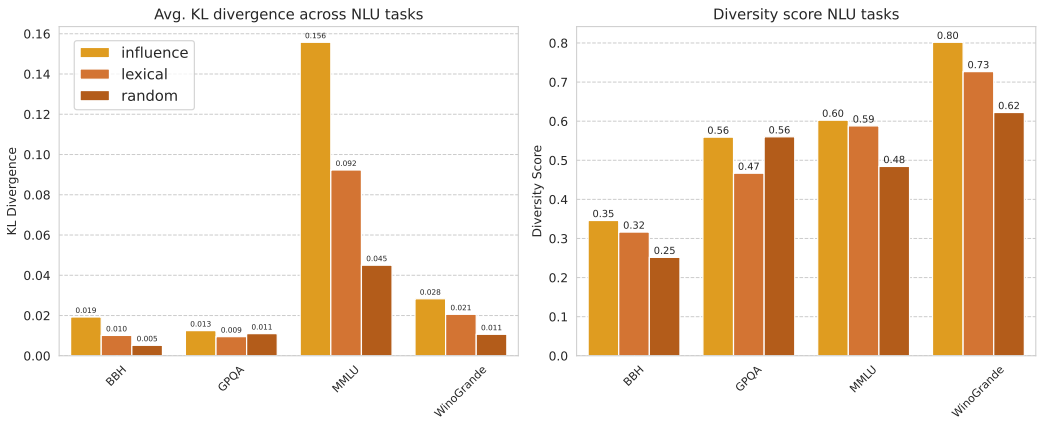

Figure 5: Average KL divergence and diversity score on various natural language understanding tasks. SPA with influence function consistently outperforms the lexical overlap and random adaptations, demonstrating its effectiveness in generating diverse samples across different NLU tasks.

random adaptations achieve better diversity than lexical overlap on the GPQA task, suggesting that the effectiveness of partitioning methods may change depending on the task.

The diversity scores and average KL divergence values vary across tasks, reflecting the inherent differences in the nature and complexity of each task. Tasks like MMLU, which cover a wide range of subjects, tend to yield higher average KL divergence. We also notice that a larger gap in average KL divergence does not necessarily translate to a proportionally greater difference in diversity scores. This suggests that while average KL divergence captures the dissimilarity between the generated sample distributions, it may not always directly correlate with the actual diversity of the samples. Nonetheless, the consistent improvement achieved by SPA with influence function highlights its robustness and adaptability to various natural language understanding challenges.

6 Related Work

Sampling-based methods have been widely explored to generate diverse text from language models. One of the most common approaches is temperature sampling (Ackley et al., 1985; Hinton et al., 2015). Several studies have investigated the impact of temperature on model sampling and its effect on the diversity-quality trade-off (Caccia et al., 2018; Renze & Guven, 2024; Wang et al., 2023). Higher temperatures lead to more diverse but potentially less coherent samples, while lower temperatures produce more conservative and deterministic outputs. When using high temperatures, human interventions can help to correct errors during the sampling process (Chung et al., 2023). Dynamic temperature strategies have also been explored during the model training and inference stages (Lin et al., 2018; Zhang et al., 2018; Wang et al., 2019; Chang et al., 2023).

Besides adjusting temperature, top-k, top-p (nucleus) sampling (Holtzman et al., 2019) and their variants are common sampling methods (Fan et al., 2018; Meister et al., 2022; Hewitt et al., 2022; Ravfogel et al., 2023), which restrict the sampling space or dynamically adjust the number of tokens considered at each step. Another line of works studied how to formulate quality-diversity trade-off as a search or RL problem (Naik et al., 2023; Lim et al., 2024; Mudgal et al., 2023; Bradley et al., 2023; Ji et al., 2023).

7 Conclusion

In summary, we proposed SPA, which that leverages synthetic data, data partitioning, and model adaptation to elicit diverse responses from foundation models. By partitioning synthetic datasets into subsets that capture unique aspects of the data and training multiple model adaptations optimized for these subsets, SPA enables the generation of diverse and high-quality responses.

Limitation One main challenges is the computational cost associated with influence function, which require several extra epochs of backward passes to estimate. Future work could explore more efficient data attribution methods, such as TRAK (Park et al., 2023) and K-FAC (Grosse et al., 2023).

https://doi.org/10.52202/079017-1911

Additionally, the ranking heuristics used to approximate Eq. (4) can be replaced by more advanced clustering algorithms. Additionally, serving multiple LoRA adaptations poses significant computational challenge in real-time serving framework. Recent works such as S-LoRA and FLoRA (Sheng et al., 2023; Wen & Chaudhuri, 2024) can be considered to accommodate this overhead.

References

David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for boltzmann machines. *Cogn. Sci.*, 9:147–169, 1985. URL https://api.semanticscholar.org/CorpusID:12174018.

Naman Agarwal, Brian Bullins, and Elad Hazan. Second-order stochastic optimization for machine learning in linear time. *J. Mach. Learn. Res.*, 18:116:1–116:40, 2016. URL https://api.semanticscholar.org/CorpusID:10569090.

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

Herbie Bradley, Andrew Dai, Hannah Benita Teufel, Jenny Zhang, Koen Oostermeijer, Marco Bellagente, Jeff Clune, Kenneth O. Stanley, Grégory Schott, and Joel Lehman. Quality-diversity through ai feedback. *ArXiv*, abs/2310.13032, 2023. URL https://api.semanticscholar.org/CorpusID:264405960.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Massimo Caccia, Lucas Caccia, William Fedus, H. Larochelle, Joelle Pineau, and Laurent Charlin. Language gans falling short. *ArXiv*, abs/1811.02549, 2018. URL https://api.semanticscholar.org/CorpusID:53208122.

Chung-Ching Chang, D. Reitter, Renat Aksitov, and Yun-Hsuan Sung. Kl-divergence guided temperature sampling. *ArXiv*, abs/2306.01286, 2023. URL https://api.semanticscholar.org/CorpusID:259063711.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, David W. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, Suchir Balaji, Shantanu Jain, Andrew Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew M. Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374, 2021. URL https://api.semanticscholar.org/CorpusID:235755472.

John Joon Young Chung, Ece Kamar, and Saleema Amershi. Increasing diversity while maintaining accuracy: Text data generation with large language models and human interventions. In *Annual Meeting of the Association for Computational Linguistics*, 2023. URL https://api.semanticscholar.org/CorpusID:259096160.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019. URL https://api.semanticscholar.org/CorpusID:52967399.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding back-translation at scale. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 489–500, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1045. URL https://aclanthology.org/D18-1045.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Annual Meeting of the Association for Computational Linguistics*, 2018. URL https://api.semanticscholar.org/CorpusID:44134226.

Roger Baker Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Kamil.e Lukovsiut.e, Karina Nguyen, Nicholas Joseph, Sam McCandlish, Jared Kaplan, and Sam Bowman. Studying large language model generalization with influence functions. *ArXiv*, abs/2308.03296, 2023. URL https://api.semanticscholar.org/CorpusID:260682872.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Xiaodong Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *ArXiv*, abs/2009.03300, 2020. URL https://api.semanticscholar.org/CorpusID:221516475.

John Hewitt, Christopher D. Manning, and Percy Liang. Truncation sampling as language model desmoothing. In *Conference on Empirical Methods in Natural Language Processing*, 2022. URL https://api.semanticscholar.org/CorpusID:253157390.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015. URL https://api.semanticscholar.org/CorpusID:7200347.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *ArXiv*, abs/1904.09751, 2019. URL https://api.semanticscholar.org/CorpusID:127986954.

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. Unnatural instructions: Tuning language models with (almost) no human labor. *ArXiv*, abs/2212.09689, 2022.

J. Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685, 2021. URL https://api.semanticscholar.org/CorpusID:235458009.

Haozhe Ji, Pei Ke, Hongning Wang, and Minlie Huang. Language model decoding as direct metrics optimization. *ArXiv*, abs/2310.01041, 2023. URL https://api.semanticscholar.org/CorpusID:263605885.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, 2017. URL https://api.semanticscholar.org/CorpusID:13193974.

Ariel N. Lee, Cole J. Hunter, and Nataniel Ruiz. Platypus: Quick, cheap, and powerful refinement of llms. *ArXiv*, abs/2308.07317, 2023. URL https://api.semanticscholar.org/CorpusID:260886870.

Minhyeok Lee. A mathematical investigation of hallucination and creativity in gpt models. *Mathematics*, 2023. URL https://api.semanticscholar.org/CorpusID:258768397.

Bryan Lim, Manon Flageat, and Antoine Cully. Large language models as in-context ai generators for quality-diversity. *ArXiv*, abs/2404.15794, 2024. URL https://api.semanticscholar.org/CorpusID:269362584.

 https://doi.org/10.52202/079017-1911

Junyang Lin, Xu Sun, Xuancheng Ren, Muyu Li, and Qi Su. Learning when to concentrate or divert attention: Self-adaptive attention temperature for neural machine translation. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2985–2990, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1331. URL https://aclanthology.org/D18-1331.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=1qvx610Cu7.

Yuanhua Lv and ChengXiang Zhai. Lower-bounding term frequency normalization. In *International Conference on Information and Knowledge Management*, 2011. URL https://api.semanticscholar.org/CorpusID:14029221.

James Martens. Deep learning via hessian-free optimization. In *International Conference on Machine Learning*, 2010. URL https://api.semanticscholar.org/CorpusID:11154521.

Clara Meister, Tiago Pimentel, Gian Wiher, and Ryan Cotterell. Typical decoding for natural language generation. *ArXiv*, abs/2202.00666, 2022. URL https://api.semanticscholar.org/CorpusID:246442062.

Sidharth Mudgal, Jong Lee, Harish Ganapathy, YaGuang Li, Tao Wang, Yanping Huang, Zhifeng Chen, Heng-Tze Cheng, Michael Collins, Trevor Strohman, Jilin Chen, Alex Beutel, and Ahmad Beirami. Controlled decoding from language models. *ArXiv*, abs/2310.17022, 2023. URL https://api.semanticscholar.org/CorpusID:264491118.

Ranjita Naik, Varun Chandrasekaran, Mert Yuksekgonul, Hamid Palangi, and Besmira Nushi. Diversity of thought improves reasoning abilities of llms. 2023. URL https://api.semanticscholar.org/CorpusID:267938465.

OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023. URL https://api.semanticscholar.org/CorpusID:257532815.

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

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale. In *International Conference on Machine Learning*, 2023. URL https://api.semanticscholar.org/CorpusID:257757261.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. Instruction tuning with gpt-4. *ArXiv*, abs/2304.03277, 2023.

Shauli Ravfogel, Yoav Goldberg, and Jacob Goldberger. Conformal nucleus sampling. In *Annual Meeting of the Association for Computational Linguistics*, 2023. URL https://api.semanticscholar.org/CorpusID:258479879.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *ArXiv*, abs/2311.12022, 2023. URL https://api.semanticscholar.org/CorpusID:265295009.

Matthew Renze and Erhan Guven. The effect of sampling temperature on problem solving in large language models. *ArXiv*, abs/2402.05201, 2024. URL https://api.semanticscholar.org/CorpusID:267547769.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at trec-3. In *Text Retrieval Conference*, 1994. URL https://api.semanticscholar.org/CorpusID:41563977.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, I. Evtimov, Joanna Bitton, Manish P Bhatt, Cristian Cantón Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre D'efossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code. *ArXiv*, abs/2308.12950, 2023. URL https://api.semanticscholar.org/CorpusID:261100919.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande. *Communications of the ACM*, 64:99 – 106, 2019. URL https://api.semanticscholar.org/CorpusID:198893658.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=9Vrb9D0WI4.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. In Katrin Erk and Noah A. Smith (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 86–96, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1009. URL https://aclanthology.org/P16-1009.

Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, Joseph E. Gonzalez, and Ion Stoica. S-lora: Serving thousands of concurrent lora adapters. *ArXiv*, abs/2311.03285, 2023. URL https://api.semanticscholar.org/CorpusID:265033787.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

Andrew Trotman, Antti Puurula, and Blake Burgess. Improvements to bm25 and language models examined. *Proceedings of the 19th Australasian Document Computing Symposium*, 2014. URL https://api.semanticscholar.org/CorpusID:207220720.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017. URL https://api.semanticscholar.org/CorpusID:13756489.

 https://doi.org/10.52202/079017-1911

Chi Wang, Susan Liu, and Ahmed Hassan Awadallah. Cost-effective hyperparameter optimization for large language model generation inference. *ArXiv*, abs/2303.04673, 2023. URL https://api.semanticscholar.org/CorpusID:257405357.

Pei-Hsin Wang, Sheng-Iou Hsieh, Shih-Chieh Chang, Yu-Ting Chen, Jia-Yu Pan, Wei Wei, and Da-Chang Juan. Contextual temperature for language modeling. *ArXiv*, abs/2012.13575, 2019. URL https://api.semanticscholar.org/CorpusID:214250287.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language model with self generated instructions. *ArXiv*, abs/2212.10560, 2022a.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language model with self generated instructions, 2022b.

Jason Wei and Kai Zou. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 6382–6388, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1670. URL https://aclanthology.org/D19-1670.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. *ArXiv*, abs/2109.01652, 2021. URL https://api.semanticscholar.org/CorpusID:237416585.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. Magicoder: Source code is all you need. *arXiv preprint arXiv:2312.02120*, 2023.

Yeming Wen and Swarat Chaudhuri. Batched low-rank adaptation of foundation models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=w4abltTZ2f.

Yeming Wen, Pengcheng Yin, Kensen Shi, Henryk Michalewski, Swarat Chaudhuri, and Alex Polozov. Grounding data science code generation with input-output specifications. *ArXiv*, abs/2402.08073, 2024. URL https://api.semanticscholar.org/CorpusID:267637235.

Xu Zhang, Felix X. Yu, Svebor Karaman, Wei Zhang, and Shih-Fu Chang. Heated-up softmax embedding. *ArXiv*, abs/1809.04157, 2018. URL https://api.semanticscholar.org/CorpusID:52193504.

A Experimental Setup Details

This section provides additional details on the experimental setup that were not included in the main content due to space constraints.

Computing Data Attribution Scores For the lexical overlap method, we use a publicly available BM25 (Lv & Zhai, 2011; Trotman et al., 2014) implementation written in Python and released under https://pypi.org/project/rank-bm25/. We used the default hyperparameters.

When calculating the influence function, we employ the conjugate gradient method with LiSSA approximation (Martens, 2010; Agarwal et al., 2016). We leverage a publicly available implementation from https://github.com/alstonlo/torch-influence/. For the OSS-Instruct dataset, we use a damping factor of 0.001, a depth of 120, and 500 repeats, following the guideline that the product of depth and repeats should be roughly equal to the dataset size. For the Platypus dataset, we use a depth of 120 and 200 repeats. It is worth noting that computing the influence function is also intensive with LoRA. Each column of the $\mathbf{I}$ matrix in Eq. (4) requires approximately one epoch of backward passes over the entire synthetic dataset. On the OSS-Instruct dataset, this takes roughly 5 hours using a single A100 80GB GPU. However, this is offline computation which is consumed before deploying the model to users.

After obtaining the data attribution matrix, we observe that using the ranking heuristic presented in §5.1 leads to imbalanced partitions. To achieve more balanced partitions, we normalize the data attribution matrix before applying the heuristics. We leave the exploration of more advanced clustering algorithms, such as k-means, for future work.

Details for Model Adaptations In this section, we provide details on the computing resources and hyperparameters used for training the model adaptations in both the code generation and natural language understanding domains. For the code generation experiments, we use a machine with 3 A100 40GB GPUs and train each partition for 400 steps, which takes approximately 80 minutes (each partition). The hyperparameters are mostly adopted from https://github.com/bigcode-project/starcoder/tree/main. The base model is CodeLLaMA-7B-Python, and we use bf16 precision to accelerate training. The per-device train batch size is set to 1, with a gradient accumulation step of 20. We use a learning rate of 2e-4 with a cosine learning rate scheduler and 20 warmup steps. For the LoRA hyperparameters, we use a rank (r) of 16, an alpha of 16.

In the natural language understanding domain, we train each partition for 400 steps, which takes approximately 40 minutes, using the Llama-2 13B model as the base model. The training time is shorter compared to the code generation domain because the Platypus dataset is much smaller than OSS-Instruct. The hyperparameters are mostly adopted from https://github.com/arielnlee/Platypus. We use a per-device batch size of 1 and a gradient accumulation step of 4. The learning rate is set to 1e-4, with a total of 20 warmup steps.

All the computational costs mentioned in this section, including the time and resources required for computing data attribution scores and training model adaptations, are offline. These costs are incurred before deploying the models to users, and they do not affect the inference time.

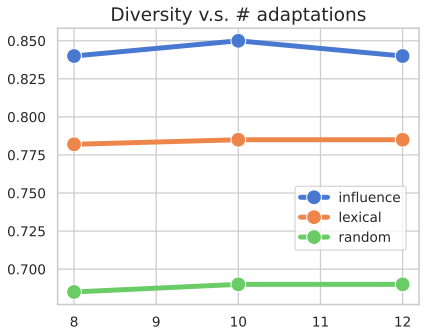

Figure 6: Diversity score as function of the number of adaptations on the HumanEval benchmark.

B Impact of Number of Adaptations

In this section, we investigate the impact of the number of model adaptations on the diversity of the generated responses. We focus on the HumanEval benchmark in the code generation domain and vary the number of adaptations from 8 to 12. The results are presented in Fig. 6.

As shown in Fig. 6, the diversity score remains relatively stable as the number of adaptations increases from 8 to 12, regardless of the partitioning method used. These results suggest that increasing the number of adaptations beyond a certain point may not necessarily lead to an improvement in the diversity of the generated responses.

C Test Queries

In this appendix, we provide the hand-written test queries used in our experiments for both the code generation and text generation domains. These examples were utilized to compute data attribution scores. Most of the examples are generated by GPT-4 (OpenAI, 2023).

C.1 Code Generation Domain

```
1. """Title: Longest Palindromic Subsequence
Query: Write a function to find the longest palindromic subsequence in a given
↪  string.
Solution:
"""
def longest_palindromic_subsequence(s):
    n = len(s)
    dp = [[0] * n for _ in range(n)]
```


https://doi.org/10.52202/079017-1911

```
    for i in range(n):
        dp[i][i] = 1

    for length in range(2, n+1):
        for i in range(n-length+1):
            j = i + length - 1
            if s[i] == s[j] and length == 2:
                dp[i][j] = 2
            elif s[i] == s[j]:
                dp[i][j] = dp[i+1][j-1] + 2
            else:
                dp[i][j] = max(dp[i+1][j], dp[i][j-1])

    return dp[0][n-1]

2. """Title: Nth Fibonacci Number
Query: Implement a function to calculate the nth Fibonacci number using dynamic
↪ programming.
Solution:
"""
def fibonacci(n):
    if n <= 0:
        return 0
    elif n == 1:
        return 1

    fib = [0] * (n + 1)
    fib[1] = 1

    for i in range(2, n + 1):
        fib[i] = fib[i - 1] + fib[i - 2]

    return fib[n]

3. """Title: Sum of Two Largest Elements
Query: Create a function that takes a list of integers and returns the sum of the
↪ two largest elements in the list.
Solution:
"""
def sum_of_two_largest(nums):
    if len(nums) < 2:
        return sum(nums)

    largest = second_largest = float('-inf')

    for num in nums:
        if num > largest:
            second_largest = largest
            largest = num
        elif num > second_largest:
            second_largest = num

    return largest + second_largest

4. """Title: Maximum Subarray Sum
Query: Implement a function to find the maximum subarray sum in a given array of
↪ integers.
Solution:
"""
def max_subarray_sum(nums):
    max_sum = float('-inf')
    current_sum = 0

    for num in nums:
```

```
        current_sum = max(num, current_sum + num)
        max_sum = max(max_sum, current_sum)

    return max_sum

5. """Title: First Non-Repeating Character
Query: Create a function that takes a string and returns the first non-repeating
↪  character in the string.
Solution:
"""
def first_non_repeating_character(s):
    char_count = {}

    for char in s:
        char_count[char] = char_count.get(char, 0) + 1

    for char in s:
        if char_count[char] == 1:
            return char

    return None

6. """Title: Merge Two Sorted Lists
Query: Write a function to merge two sorted lists into a single sorted list.
Solution:
"""
def merge_sorted_lists(list1, list2):
    merged_list = []
    i = j = 0

    while i < len(list1) and j < len(list2):
        if list1[i] <= list2[j]:
            merged_list.append(list1[i])
            i += 1
        else:
            merged_list.append(list2[j])
            j += 1

    while i < len(list1):
        merged_list.append(list1[i])
        i += 1

    while j < len(list2):
        merged_list.append(list2[j])
        j += 1

    return merged_list

7. """Title: Remove Prime Numbers from List
Query: Create a function that takes a list of integers and returns a new list with
↪  all the prime numbers removed.
Solution:
"""
def is_prime(num):
    if num < 2:
        return False
    for i in range(2, int(num ** 0.5) + 1):
        if num % i == 0:
            return False
    return True

def remove_prime_numbers(nums):
    return [num for num in nums if not is_prime(num)]

8. """Title: Longest Common Substring
```


https://doi.org/10.52202/079017-1911

```
Query: Write a function to find the longest common substring between two given strings.
Solution:
"""
def longest_common_substring(str1, str2):
    m, n = len(str1), len(str2)
    dp = [[0] * (n + 1) for _ in range(m + 1)]
    max_length = 0
    end_index = 0

    for i in range(1, m + 1):
        for j in range(1, n + 1):
            if str1[i - 1] == str2[j - 1]:
                dp[i][j] = dp[i - 1][j - 1] + 1
                if dp[i][j] > max_length:
                    max_length = dp[i][j]
                    end_index = i
            else:
                dp[i][j] = 0

    start_index = end_index - max_length
    return str1[start_index : end_index]

9. """Title: Kth Largest Element in an Unsorted Array
Query: Implement a function to find the kth largest element in an unsorted array.
Solution:
"""
def kth_largest_element(nums, k):
    k = len(nums) - k

    def partition(left, right):
        pivot = nums[right]
        i = left - 1

        for j in range(left, right):
            if nums[j] <= pivot:
                i += 1
                nums[i], nums[j] = nums[j], nums[i]

        nums[i + 1], nums[right] = nums[right], nums[i + 1]
        return i + 1

    def quick_select(left, right):
        if left == right:
            return nums[left]

        pivot_index = partition(left, right)

        if k == pivot_index:
            return nums[k]
        elif k < pivot_index:
            return quick_select(left, pivot_index - 1)
        else:
            return quick_select(pivot_index + 1, right)

    return quick_select(0, len(nums) - 1)

10. """Title: Product of Array Elements
Query: Create a function that takes a list of integers and returns the product of all the elements.
Solution:
"""
def product_of_elements(nums):
    product = 1
    for num in nums:
```

```
        product *= num
    return product

11. """Title: Binary Search
Query: Implement a function to perform binary search on a sorted list of integers.
Solution:
"""
def binary_search(nums, target):
    left = 0
    right = len(nums) - 1

    while left <= right:
        mid = (left + right) // 2

        if nums[mid] == target:
            return mid
        elif nums[mid] < target:
            left = mid + 1
        else:
            right = mid - 1

    return -1

12. """Title: Find Missing Number
Query: Create a function that takes a list of integers from 0 to n (inclusive) with one number missing and returns the missing number.
Solution:
"""
def find_missing_number(nums):
    n = len(nums)
    expected_sum = (n * (n + 1)) // 2
    actual_sum = sum(nums)
    return expected_sum - actual_sum
```

C.2 Text Generation Domain

```
1. Title: Economic Impacts of the Black Death
Query: Explain the economic impacts of the Great Mortality in medieval Europe.
Response: The the Great Mortality drastically reduced the population of Europe, leading to severe labor shortages, higher wages, lower prices for land, and a shift in economic power from the feudal lords to the working class and merchants.

2. Title: Photosynthesis Process
Query: Describe the process of photosynthesis and its importance to the Earth's ecosystem.
Response: Photosynthesis is the process by which green plants and some other organisms use sunlight to synthesize nutrients from carbon dioxide and water. It generates oxygen as a byproduct, which is vital for most life forms on Earth.

3. Title: Calculating Travel Distance
Query: If a car travels at 60 miles per hour for 3 hours, how far has it gone? Explain your calculation.
Response: The car has traveled 180 miles, calculated as 60 miles/hour * 3 hours.

4. Title: Utilitarianism vs Deontological Ethics
Query: Discuss the main differences between utilitarianism and deontological ethics.
Response: Utilitarianism focuses on the outcomes or consequences of actions to determine morality, while deontological ethics considers the actions themselves and the adherence to duties or rules as the basis for morality.

5. Title: Advancements in Quantum Computing
Query: What are the key advancements in quantum computing over the last decade?
```


https://doi.org/10.52202/079017-1911

```
Response: Key advancements include the development of quantum supremacy, error correction, and the creation of more stable qubits, enhancing computing power and reliability.

6. Title: Wedding Traditions in India
Query: Compare the wedding traditions of Northern and Southern India.
Response: Northern Indian weddings often feature elaborate rituals like Sangeet and Mehendi, while Southern Indian weddings are marked by rituals like Kashi Yatra and Oonjal. Both have vibrant traditions but differ in cultural practices and attire.

7. Title: Deforestation in the Amazon
Query: What are the primary causes of the Amazon rainforest's deforestation and what measures are being taken to address it?
Response: Primary causes include agriculture, logging, and infrastructure development. Measures to address this include enforcement of laws, satellite monitoring, and international cooperation on sustainable practices.

8. Title: Theme of Ambition in Macbeth
Query: Analyze the theme of ambition in Shakespeare's 'Macbeth'.
Response: Ambition in 'Macbeth' serves as both a driving force and a tragic flaw for the characters, particularly Macbeth, leading to his rise and eventual downfall as he succumbs to the ambition spurred by the prophecy and his wife’s encouragement.

9. Title: Global Impact of Renewable Energy
Query: Discuss the global impact of renewable energy sources on climate change.
Response: Renewable energy sources like solar and wind have a significant impact on mitigating climate change by reducing dependence on fossil fuels, decreasing greenhouse gas emissions, and promoting sustainability. Countries adopting renewable energy contribute to a global reduction in carbon footprints, which can help meet the goals set by international climate agreements like the Paris Agreement.

10. Title: The Role of Artificial Intelligence in Healthcare
Query: Explain how artificial intelligence is transforming healthcare.
Response: Artificial intelligence in healthcare is transforming the industry by enhancing diagnostic accuracy, improving treatment personalization, and optimizing operational efficiencies. AI applications include predictive analytics for patient management, automated imaging and diagnostics, and robot-assisted surgeries, leading to faster, more accurate patient care and reduced healthcare costs.

11. Title: Cultural Significance of Food in Japan
Query: Describe the cultural significance of food in Japan and how it reflects Japanese society.
Response: Food in Japan is deeply intertwined with the nation's culture, reflecting aspects of beauty, seasonality, and regional diversity. Traditional dishes like sushi and bento embody aesthetic principles and social customs, such as respect for nature and meticulous attention to detail. Food rituals, such as tea ceremonies, also highlight the importance of mindfulness and harmony in Japanese society.

12. Title: Economic Effects of Globalization
Query: Analyze the economic effects of globalization on developing countries.
Response: Globalization has both positive and negative economic effects on developing countries. On the positive side, it allows access to international markets, increases capital inflow, and promotes technology transfer, leading to job creation and economic growth. However, it can also lead to economic dependency, cultural homogenization, and the potential exploitation of local resources and labor, which might exacerbate inequalities and social tensions.
```

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: We propose the SPA framework which is the main contribution of this paper.

   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: The limitation is stated in the Section 6.

   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]

https://doi.org/10.52202/079017-1911

Justification: No theoretical result in this paper.

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: Hyper-parameters are given in the appendix.

   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: [Yes]

Justification: In the supplementary material.

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 5.

   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: [Yes]

   Justification: The results are averaged over 4 checkpoints.

   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.

https://doi.org/10.52202/079017-1911

- 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: The information is given in the Appendix.

   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: No violation.

   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: [NA]

    Justification: This work is foundational research on how to generate diverse samples. It is not directly applied in any product.

    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: [NA]

    Justification: We use existing dataset and models.

    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: We cited the used datasets, models and the repos.

    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.

https://doi.org/10.52202/079017-1911

- 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: The code is included in the supplementary material.

    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 work 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 work 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.