

Adversarial methods for LLM alignment on coding tasks

Seminar

Ilyas Oulkadda

January 2025

Introduction

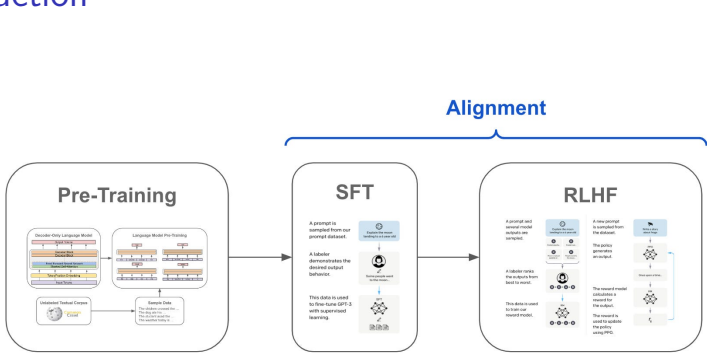


Figure: The whole training cycle of an LLM, alignment step usually refers to the Reinforcement Learning from Human Feedbacks, RLHF, step but Supervised Fine-Tuning, SFT could also be considered as alignment (Wolfe, 2023).

Introduction

Goal

Improve CodeLLMs alignment (Ouyang et al., 2022)

How ?

- Adversarial (OpenAI et al., 2021)
- Self-play (Sukhbaatar et al., 2018)
- Curriculum learning (Sukhbaatar et al., 2018)
- At scale (Bowman et al., 2022)

Code LLMs

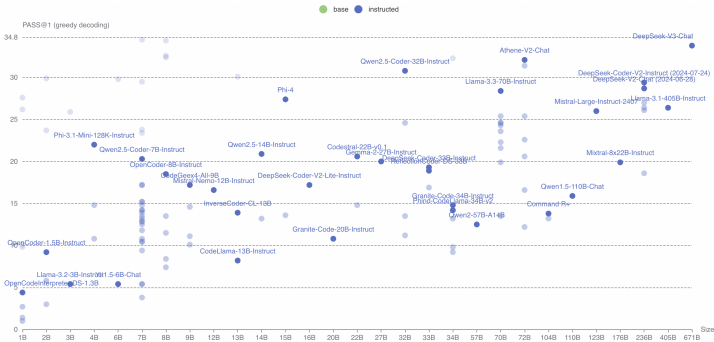


Figure: The open code LLM benchmark (Ben Allal, Muennighoff, et al., 2022) on the BigCodeBench dataset

- **Key focus:** Enhancing the robustness and reliability of Code-LLMs.
- **Techniques:** DPO, adversarial methods, curriculum learning, synthetic datasets.
- **Contributions:**
 - ▶ Dataset-independent training framework.
 - ▶ Quality data generation due to the curriculum and adversarial.
 - ▶ Effective training framework.

AKD : Adversarial Knowledge Distillation For Large Language Models Alignment on Coding tasks

Ilyas Oulkadda¹ Julien Perez¹

Abstract

The widespread adoption of Large Language Models (LLMs) for code generation, exemplified by GitHub Copilot¹ surpassing a million users, highlights the transformative potential of these tools in improving developer productivity. However, this rapid growth also underscores critical concerns regarding the quality, safety, and reliability of the code they generate. As Code-LLMs evolve, they face significant challenges, including the diminishing returns of model scaling and the scarcity of new, high-quality training data. To address these issues, this paper introduces Adversarial Knowledge Distillation (AKD), a novel approach that leverages adversarially generated synthetic datasets to distill the capabilities of larger models into smaller, more efficient ones. By systematically stress-testing and refining the reasoning capabilities of Code-LLMs, AKD provides a framework for enhancing model robustness, reliability, and security. We believe this work represents a critical step toward ensuring dependable automated code generation within the constraints of existing data and scaling limitations.

1. Introduction

The rapid development and deployment of Large Language Models (LLMs), particularly those designed for code generation, such as GitHub Copilot, have started a transformative era in software development. These models hold the promise of significantly enhancing productivity by automating repetitive coding tasks and assisting developers with complex problem-solving. However, this evolution has also introduced pressing concerns regarding the quality, security, and ethical implications of the generated code. As developers increasingly depend on these models for critical applications, ensuring their outputs meet high standards of reliability and security has become imperative.

Traditional methods for aligning LLMs rely heavily on human-annotated datasets, which provide direct feedback

¹A coding extension powered by a Code-LLM to assist in code completion tasks

and corrections. While effective, this approach faces challenges in scalability and adaptability due to the growing size of modern LLMs and the rapidly evolving nature of coding practices. The need for more sustainable and efficient alignment methodologies has become apparent, prompting research into alternative strategies.

On the other hand, generating datasets synthetically using LLMs can be beneficial in terms of scalability and efficiency (Günasekar, Zhang, Anjea, Caio César Teodoro Mendes, et al., 2023; Long et al., 2024). However, it doesn't guarantee that the synthetic data will address all relevant cases. Synthetic data, generated in an open-ended manner, might lack the diversity and complexity of real-world scenarios, leading to gaps in the model's understanding and performance. This can result in models that perform well on synthetic benchmarks but struggle with real-world applications. Therefore, it's crucial to combine synthetic data generation with rigorous validation against real-world data to ensure the model's robustness and applicability.

To address these challenges, we propose Adversarial Knowledge Distillation (AKD), a novel framework designed to align Code-LLMs with functional and performance standards without the extensive reliance on human annotation. AKD integrates Direct Preference Optimization (DPO) (Rafailov et al., 2023) into a curriculum-driven adversarial dataset generation process (Günasekar, Zhang, Anjea, Caio César Teodoro Mendes, et al., 2023; OpenAI et al., 2021). This approach enables distillation of knowledge (Hinton, Vinyals, and Dean, 2015) from a larger, more capable "teacher" model into a smaller, computationally efficient "student" model through an automatically defined curriculum by leveraging adversarially generated synthetic datasets of code exercises.

Concretely, AKD begins with the teacher model generating coding tasks, complete with function descriptions and solution outlines, which serve as a curated curriculum for the student model. Adversarial interactions are then used to stress-test the student model's reasoning and alignment capabilities, systematically exposing weaknesses and guiding improvements. The optimization process relies on DPO loss applied to the adversarial curriculum, allowing the student model to iteratively refine its alignment with the teacher's

Adversarial Knowledge Distillation

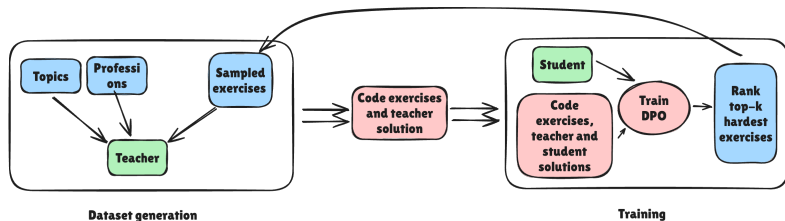


Figure: Adversarial Knowledge Distillation framework involves three components: models (green), training processes (red), and synthetic data generation (blue). Topics and seeds initiate the dataset, with exercises iteratively sampled using margin rewards to guide improvement. Teacher-generated 'chosen' and student-generated 'rejected' solutions form the dataset for Direct Preference Optimization (DPO) (Rafailov et al., 2023), enabling targeted student model enhancement.

Direct Preference Optimization

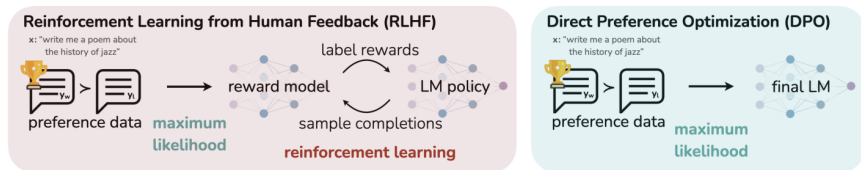


Figure: DPO directly optimizes for the policy best satisfying the preferences with a simple classification objective, fitting an implicit reward model whose corresponding optimal policy can be extracted in closed form. (Rafailov et al., 2023)

DPO : Equations

$$\mathcal{L}_{\text{DPO}}(\pi_{\theta}; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_{\theta}(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \beta \log \frac{\pi_{\theta}(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right) \right]. \quad (4)$$

$$R_{\text{chosen}} = \beta \cdot (\log \pi_{\theta}(y_w | x) - \log \pi_{\text{ref}}(y_w | x)) \quad (1)$$

$$R_{\text{rejected}} = \beta \cdot (\log \pi_{\theta}(y_l | x) - \log \pi_{\text{ref}}(y_l | x)) \quad (2)$$

$$R_{\text{margin}} = R_{\text{chosen}} - R_{\text{rejected}} \quad (3)$$

Synthetic Datasets

Main issues

- Structured outputs
- Control diversity

Solution

- Try/Catch, repeat n times until we get the asked outputs and markdown usage.
- Use a database of seeds, diverse prompts = diverse outputs.

Seeds

Topic seeds

Extracted a list of coding categories from LeetCode. Total of 70 categories. (Gunasekar et al., 2023) (Ben Allal, Lozhkov, et al., 2024)

Example of topics

- Dynamic Programming
- Tree
- Linked List

Professions

We also have a list of professions which allows us to control the semantic of the generated coding exercises.

Adversarial game

Requirements

For DPO, datasets must have a Prompt, Chosen and Rejected

Setup

- An oracle and student models.
- Generate subtopics for initial topics.
- Create a dataset of prompts using seeds combinations.
- Generate a dataset of exercises using the oracle
- Student and oracle both generate their own solutions for the exercises

Adversarial game

Adversarial dataset

At the end of each training, we retrieve the hardest exercise to use as seeds for the next steps. (Sukhbaatar et al., 2018)

$$R_{\text{chosen}} = \beta \cdot (\log \pi_{\theta}(y_w | x) - \log \pi_{\text{ref}}(y_w | x)) \quad (5)$$

$$R_{\text{rejected}} = \beta \cdot (\log \pi_{\theta}(y_l | x) - \log \pi_{\text{ref}}(y_l | x)) \quad (6)$$

$$R_{\text{margin}} = R_{\text{chosen}} - R_{\text{rejected}} \quad (7)$$

Knowledge Distillation (Hinton et al., 2015)

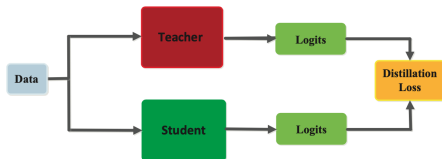


Figure: Knowledge distillation, the loss is usually a combination of soft and hard labels

Issues

The classic KD approach usually requires you to train your student from scratch.

LoRA (Hu et al., 2021)

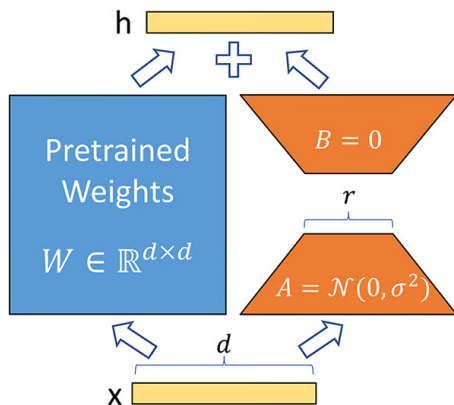


Figure: Low rank adaptation allows us to only train a small amount of parameters. We freeze the rest of the model and inject new lower rank matrices that we train.

LoRA, Which layers ?

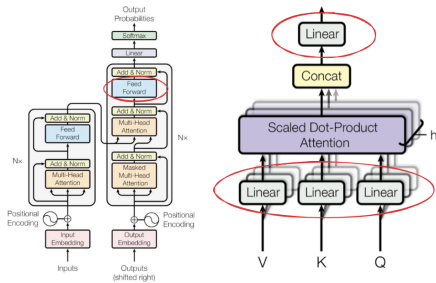


Figure: We apply LoRA to both the MLP module and MHA modules, precisely the up and down projections in the MLP module. In the MHA module, we apply LoRA to all KQV projections and out projections at the end of the module (Vaswani et al., 2023)

HumanEval (Chen et al., 2021)

```
from typing import List
def has_close_elements(numbers: List[float], threshold: float) -> bool:
    """
    Check if in given list of numbers, are any two numbers closer
    to each other than given threshold.
    >>> has_close_elements([1.0, 2.0, 3.0], 0.5)
    False
    >>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3)
    True
    """
```

Figure: The HumanEval dataset contains 164 python coding exercises. The prompt is a function signature with it's arguments and a doc-string describing the expected behavior. We evaluate using a test suite provided for each sample.

Models

- **Qwen 2.5 Coder models** (Hui et al., 2024). Available in multiple sizes (1.5B, 3B, 7B, 14B, 32B). SOTA scores on coding benchmarks for the smaller models. Fine-tuned Qwen 2.5 on coding datasets.
- **Llama 3.2/3.1 models** (Dubey et al., 2024). Available in (1B and 7B). The 1B version is a pruned version using 8B and 70B models. Followed by an instruction distillation using 405B.
- **Phi 1.5/2** (Gunasekar et al., 2023). Around 2B parameters. Pre-trained models only for Phi-1.5. Phi-2 has been trained to respond in chat format. Trained on synthetic data.

HumanEval results

Model Pair	Teacher	Student	AKD
Llama-3.1 8B / Phi-1.5	62	31	39
Qwen2.5 7B / Phi-1.5	88	31	38
Qwen2.5 7B / Llama 3.2 1B	88	32.0	35

Table: Fine-Tuning Results using AKD with Teacher/Student Pairs on HumanEval (%)

Method comparison

Method	Accuracy	Dataset Size
Self-Supervised (AKD)	38	1.6k
Self-Supervised (APPS)	38	5.0k
AKD	38	1.6k

Table: Comparison of Benchmark Performance Between AKD and Self-Supervised Fine-Tuning on HumanEval (%). Models: Qwen2.5 Coder 7B / Llama 3.2 1B

Adversarial steps evaluation

Method	Accuracy	Improvement
DPO Baseline	35.0	—
Adversarial Training	38.0	+3.0

Table: Performance Comparison: Adversarial Training vs. DPO Baseline on HumanEval (%)

Limitations

Family models

We observed no improvement when running AKD on models from the same family (e.g. Llama 1B and Llama 7B) due to being a distilled model or being trained on the same dataset. This limited our evaluation when testing speculative decoding. We think that AKD improves the use of SD. However, this requires us to have the same tokenizer on the assistant and main models.

Generation bottleneck

We noticed during our experiments that generation is the slower part of our framework and can be limiting when trying to generate very large datasets.

Conclusion

Goal : Efficiently improve alignment using adversarial methods

How ?

- Synthetic datasets
- Direct Preference Optimization
- Adversarial methods
- Curriculum learning








Results

- Performance improvements on HumanEval.
- High-Quality Synthetic Dataset.
- Adversarial steps improvements. (AKD $>$ single large DPO)








Limitations

Generation bottleneck and limited number of usable models combinations for now.

Bibliography I

-  Ben Allal, Loubna, Anton Lozhkov, et al. (2024). *Cosmopedia*. URL: <https://huggingface.co/datasets/HuggingFaceTB/cosmopedia>.
-  Ben Allal, Loubna, Niklas Muennighoff, et al. (2022). *A framework for the evaluation of code generation models*. <https://github.com/bigcode-project/bigcode-evaluation-harness>.
-  Bowman, Samuel R. et al. (2022). *Measuring Progress on Scalable Oversight for Large Language Models*. arXiv: 2211.03540 [cs.CL].
-  Chen, Mark et al. (2021). *Evaluating Large Language Models Trained on Code*. arXiv: 2107.03374 [cs.LG].
-  Dubey, Abhimanyu et al. (2024). *The Llama 3 Herd of Models*. arXiv: 2407.21783 [cs.AI]. URL: <https://arxiv.org/abs/2407.21783>.
-  Gunasekar, Suriya et al. (2023). *Textbooks Are All You Need*. arXiv: 2306.11644 [cs.CL].
-  Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean (2015). *Distilling the Knowledge in a Neural Network*. arXiv: 1503.02531 [stat.ML]. URL: <https://arxiv.org/abs/1503.02531>.

Bibliography II

-  Hu, Edward J. et al. (2021). *LoRA: Low-Rank Adaptation of Large Language Models*. arXiv: 2106.09685 [cs.CL].
-  Hui, Binyuan et al. (2024). *Qwen2.5-Coder Technical Report*. arXiv: 2409.12186 [cs.CL]. URL: <https://arxiv.org/abs/2409.12186>.
-  OpenAI, OpenAI et al. (2021). *Asymmetric self-play for automatic goal discovery in robotic manipulation*. arXiv: 2101.04882 [cs.LG].
-  Ouyang, Long et al. (2022). *Training language models to follow instructions with human feedback*. arXiv: 2203.02155 [cs.CL].
-  Rafailov, Rafael et al. (2023). *Direct Preference Optimization: Your Language Model is Secretly a Reward Model*. arXiv: 2305.18290 [cs.CL].
-  Sukhbaatar, Sainbayar et al. (2018). *Intrinsic Motivation and Automatic Curricula via Asymmetric Self-Play*. ICLR 2018. arXiv: 1703.05407 [cs.LG].
-  Vaswani, Ashish et al. (2023). *Attention Is All You Need*. arXiv: 1706.03762 [cs.CL]. URL: <https://arxiv.org/abs/1706.03762>.

Bibliography III



Wolfe, Cameron R. (2023). *Understanding and Using Supervised Fine-Tuning (SFT) for Language Models*.

<https://cameronrwolfe.substack.com/p/understanding-and-using-supervised>. Accessed: 2025-01-11.