TigerCoder: A Novel Suite of LLMs for Code Generation in Bangla

Nishat Raihan, Antonios Anastasopoulos, Marcos Zampieri

George Mason University
Fairfax, VA, USA
{mraihan2,antonis,mzampier}@gmu.com

Abstract

Despite being the 5th most spoken language, Bangla remains underrepresented in Large Language Models (LLMs), particularly for code generation. This primarily stems from the scarcity of high-quality data to pre-train and/or finetune such models. Hence, we introduce the first dedicated family of Code LLMs for Bangla (1B & 9B). We offer three major contributions: (1) a comprehensive Bangla code instruction datasets for programming domain adaptation; (2) `MBPP-Bangla`, an evaluation benchmark for Bangla code generation; and (3) the **`TigerCoder`**-family of Code LLMs, achieving significant 11-18% performance gains at `Pass@1` over existing multilingual and general-purpose Bangla LLMs. Our findings show that curated, high-quality datasets can overcome limitations of smaller models for low-resource languages.

Keywords: Large Language Models (LLMs), Code LLMs, Benchmark

1. Introduction

Recent advancements in LLMs have significantly improved code generation (Touvron et al., 2023; Hui et al., 2024; Team et al., 2025). State-of-the-art models often achieve over 90% `Pass@1` scores on popular coding benchmarks like HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) which led to their widespread adoption in domains such as software engineering (Pasquale et al., 2025) and education (Raihan et al., 2025b). This progress, however, disproportionately benefits high-resource languages (Joshi et al., 2020; Blasi et al., 2022; Ahuja et al., 2023; Wang et al., 2023a). Specifically, Bangla, with over 242 million native speakers[1] still lacks representation in code generation tasks. This deficiency originates from scarce datasets, limited resources, and absent benchmarks (Bhattacharjee et al., 2022; Zehady et al., 2024), leading to poor performance from existing general-purpose Bangla models compared to English counterparts (Bhattacharyya et al., 2023; Uddin et al., 2023).

Current multilingual models like BLOOM (Scao et al., 2022) and LLaMA-3 (Dubey et al., 2024) offer only basic Bangla support, exhibiting code generation performance as low as 20% for Bangla compared to 65-75% `Pass@1` for English (Muennighoff et al., 2023; Raihan et al., 2025a) due to minimal data allocation (Iyer et al., 2022; Ahuja et al., 2024). To address this critical gap, we introduce `TigerCoder`, the first dedicated Bangla code-generation LLM (1B & 9B parameters), finetuned on a large (*300K instruction-code pairs*) Bangla Code Instruction dataset, tailored for the task and language.

We address 2 research questions (RQs):

`RQ1` To what extent do state-of-the-art Code LLMs preserve their code-generation quality when the natural language part of the prompt is written in *Bangla* rather than English?

`RQ2` Does a simple *Bangla* → *English* machine-translation step applied to each coding prompt significantly boost generation quality compared with direct Bangla inference?

With Bangla as our focus, we show how to develop small, open-source, high performance code generation models for low-resource languages. Our key contributions are the following:

`C1` Development of three novel, high-quality Bangla code instruction datasets: `Bangla-Code-Instruct-SI` (100K, self-instruct), `Bangla-Code-Instruct-Syn` (100K, synthetic), and `Bangla-Code-Instruct-TE` (100K, machine-translated).

`C2` Introduction of `MBPP-Bangla`, a benchmark with 974 expert-validated programming problems adopted from MBPP (Austin et al., 2021), designed specifically for evaluating Bangla code generation for 5 different programming languages (PLs) - *Python, C++, JAVA, JavaScript & Ruby*.

`C3` The **`TigerCoder`**[2] model family that outperforms existing models on Bangla programming tasks, achieving strong `Pass@1` scores across all 5 PLs.

[1] `ethnologue.com/`

[2] `huggingface.co/md-nishat-008/TigerCoder-9B`

Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026), pages 3044–3054
11–16 May 2026.

Model	Size	pt	corpora	ft	ft-dataset	Paper?	Reprod.?
titu-Gemma	2B	4.4B	X	X	X	X	X
titu-LLaMA	3B	37B	X	X	X	X	X
Bangla-LLaMA	3B	Y	X	172K	Orca-trans.	Y	X
G2B	9B	X	X	145K	Alpaca-trans.	X	X
Bangla-LLaMA	13B	Y	X	145K	Alpaca-trans.	X	X
TigerLLM	9B	10M	Bangla-TB	100K	Bangla-Inst	Y	Y

Table 1: Comparative analysis of Bangla LLM initiatives and their methodological approaches. The pre-training (*pt*) and finetuning (*ft*) columns indicate corpus size in tokens and instruction count, respectively.

The remainder of this paper is organized as follows: In Section 2, we discuss the current state of Bangla LLMs, then introduce MBPP-Bangla in Section 3; we address the two RQs in Section 4 & 5; then describe the curation of Bangla-Code-Instruct in Section 6 and TigerCoder in Section 7.

2. Related Work

Bangla LLMs Table 1 catalogs the Bangla-centric LLMs released to date. Although these initiatives demonstrate encouraging progress, most models are distributed only as opaque checkpoints with limited details on data provenance, pre-processing, or training hyperparameters. This paucity of documentation hampers independent verification, reproducibility, and downstream adaptation—key pillars of scientific progress.

Bangla Benchmarks To assess Bangla NLP capability, researchers have introduced several task-specific benchmarks—e.g., BanglaRQA for reading comprehension (Ekram et al., 2022), BEnQA for open-domain question answering (Shafayat et al., 2024), and mHumanEval-Bangla for code generation (Raihan et al., 2025a). More recently, Raihan and Zampieri (2025) released TigerLLM, a suite of general-purpose Bangla models trained on a carefully curated corpus, which set a stronger baseline across these datasets.

Code Generation in Bangla Despite the growth in Bangla NLU and NLG resources, code generation remains virtually unexplored. Prior work evaluates multilingual LLMs on Bangla prompts (Raihan et al., 2025a) but no model has been purpose-built for this setting. TigerCoder addresses this gap by (i) constructing the first Bangla code-instruction corpus, (ii) training specialised Bangla Code LLM checkpoints, and (iii) establishing a rigorous evaluation pipeline that disentangles programming competence from Bangla comprehension. Our work therefore extends the TigerLLM lineage into the software-engineering domain and provides a reproducible foundation for future research on low-resource code generation.

3. The MBPP-Bangla Benchmark

Structure To address the absence of standardized evaluation tools for Bangla–code generation, we introduce **MBPP-Bangla**[3]. Derived from the Mostly Basic Python Programs (MBPP) dataset (Austin et al., 2021), MBPP-Bangla comprises **974** programming problems, each presented in Bangla and paired with *canonical reference solutions* in **five languages**: Python, Java, JavaScript, Ruby, and C++. In addition, every problem is assigned to one of five topical classes (*String, Math, Data-structures, Algorithms, File-I/O*), enabling controlled, category-wise evaluation.

Parameter	Specification
Dataset Size	974 problems
Source	Mostly Basic Python Programs (MBPP) (Austin et al., 2021)
Programming Languages	Python, Java, JavaScript, Ruby, C++
Task Focus	Basic–Intermediate coding problems (five topical classes)
Translation Method	Human translation (independent, manual)
Translators	2 native Bangla speakers (worked independently)
Translator Proficiency	Bangla (native), English (TOEFL[??] >100)
Verification Method	Human expert verification (manual)
Verifier	1 native Bangla speaker & polyglot programmer
Verifier Expertise	Proficient in all five languages above
Verification Scope	Linguistic fidelity, Technical correctness, Cross-lang. consistency
Output Components	Task ID, Bangla prompt, 5× reference codes, test cases, topic class

Table 2: Curation parameters for MBPP-Bangla. Both columns use a light-gray background, with a slightly darker shade for the header row.

Curation Process The complete curation phase goes through five key steps -

[3] https://huggingface.co/datasets/md-nishat-008/MBPP-Bangla

Model	mHumanEval						MBPP					
	English			*Bangla*			*English*			*Bangla*		
	P@1	P@10	P@100	P@1	P@10	P@100	P@1	P@10	P@100	P@1	P@10	P@100
GPT-3.5	0.79	0.81	0.84	0.56	0.56	0.59	0.81	0.83	0.89	0.60	0.62	0.62
Gemini-Flash 2.5	0.82	0.85	0.89	0.58	0.61	0.62	0.82	0.85	0.91	0.62	0.62	0.70
GPT-4o-mini	0.74	0.79	0.84	0.56	0.56	0.58	0.77	0.84	0.87	0.51	0.53	0.54
LLaMA-3.2 (11B)	0.73	0.76	0.77	0.15	0.15	0.20	0.78	0.81	0.86	0.22	0.22	0.30
Gemma-3 (27B)	0.69	0.71	0.78	0.64	0.65	0.69	0.71	0.78	0.83	0.69	0.70	0.70
Pangea (7B)	0.61	0.63	0.63	0.10	0.15	0.20	0.64	0.65	0.65	0.09	0.12	0.17
Phi-4 (7B)	0.79	0.81	0.86	0.10	0.17	0.25	0.82	0.84	0.89	0.09	0.15	0.20
Titu-LLM (2B)	0.20	0.20	0.23	0.02	0.02	0.02	0.21	0.23	0.23	0.05	0.05	0.05
Bong-LLaMA (3B)	0.31	0.32	0.34	0.02	0.02	0.02	0.04	0.04	0.04	0.07	0.09	0.11
Bangla-LLaMA (3B)	0.44	0.48	0.50	0.10	0.19	0.21	0.41	0.49	0.55	0.13	0.15	0.15
Bangla-Gemma (9B)	0.47	0.49	0.49	0.10	0.15	0.16	0.45	0.49	0.50	0.08	0.19	0.21
TigerLLM (1B)	0.64	0.66	0.66	0.61	0.64	0.70	0.68	0.69	0.69	0.65	0.68	0.71
TigerLLM (9B)	0.69	0.69	0.71	0.63	0.69	0.72	0.71	0.73	0.75	0.61	0.68	0.73

Table 3: Pass@{1,10,100} comparison on English and Bangla variants of `mHumanEval` and `MBPP`. It shows the consistent subpar performance of most models (except for `TigerLLM`) in Bangla, compared to English in code generation. (The darker the cell color, the better the performance.)

`Step1` We first extract all 974 basic–intermediate problems from the Mostly Basic Python Programs (MBPP) corpus, ensuring coverage of five topical classes.

`Step2` Two native Bangla speakers, each with certified English proficiency (TOEFL *Internet-based — score* > 100), translate every English prompt into Bangla without consulting one another.

`Step3` For each task we curate or port five reference solutions—Python, Java, JavaScript, Ruby, and C++—so that downstream systems can be evaluated language-by-language.

`Step4` A third reviewer, fluent in Bangla and all five programming languages, manually checks every item for linguistic fidelity, technical correctness, and cross-language consistency.

`Step5` The validated records are released with a task ID, Bangla prompt, five reference codes, original MBPP test cases, and a topic label, forming the `MBPP`-Bangla benchmark.

Table 2 summarizes the key parameters of the curation process.

Compared with `mHumanEval-ben`, `MBPP`-Bangla offers complementary benefits. Its larger size yields more statistically robust estimates, and its conversational Bangla prompts—spanning five programming languages—stress a model's ability to *both* comprehend varied natural language specifications and generate syntactically valid code in multiple target languages. Employing the two benchmarks jointly therefore affords a comprehensive assessment of Bangla code-generation capability.

4. RQ1 - Evaluating LLMs on Bangla Code Generation Benchmarks

Several works on LLM evaluation have already revealed that these models often perform better when prompted in English (Rohera et al., 2025; Li et al., 2024; Schut et al., 2025). None of them have investigated this phenomenon for the task of *Code Generation*. In this section, we attempt to shed some light on it.

Benchmarks We consider both `MBPP-Bangla` & `mHumanEval-Bangla` for evaluation purposes for their unique design. Compared with `mHumanEval-ben`, `MBPP`-Bangla offers complementary benefits. Its larger size yields more statistically robust estimates, and its conversational Bangla prompts—spanning five programming languages—stress a model's ability to *both* comprehend varied natural language specifications and generate syntactically valid code in multiple target languages. Employing the two benchmarks jointly therefore affords a comprehensive assessment of Bangla code-generation capability.

LLMs For the evaluation phase, we choose all the released Bangla LLMs - listed in Table 1. We also consider the multilingual open-source models, including LLaMA 3.2 (Dubey et al., 2024), Gemma 3 (Gemma Team, 2025), Phi 4 (Microsoft, 2024), and Pangea (Yue et al., 2024). In addition, we evaluate proprietary LLMs of similar size, like GPT-4o-mini (OpenAI, 2024), Gemini-2.5-Flash (Google, 2024), and GPT-3.5 (Wang et al., 2023c).

Metric We use *Pass@K* as our evaluation metric. For a task with n generated programs, m of which pass all tests, *Pass@K* is defined as -

$$\text{Pass@}K \;=\; 1 - \frac{\binom{n-m}{K}}{\binom{n}{K}}, \qquad 1 \le K \le n.$$

It is the probability that at least one of K *without-replacement* draws is correct.

We adopt 3 variations of *Pass@K*:

Model	mHumanEval						MBPP					
	Bangla → ***English****-MT*			Bangla			***Bangla*** → ***English****-MT*			Bangla		
	P@1	P@10	P@100	P@1	P@10	P@100	P@1	P@10	P@100	P@1	P@10	P@100
GPT-3.5	0.48	0.44	0.53	0.56	0.56	0.59	0.55	0.50	0.67	0.60	0.62	0.62
Gemini-Flash 2.5	0.51	0.63	0.53	0.58	0.61	0.62	0.58	0.66	0.61	0.62	0.62	0.70
GPT-4o-mini	0.45	0.59	0.49	0.56	0.56	0.58	0.43	0.46	0.42	0.51	0.53	0.54
LLaMA-3.2 (11B)	0.10	0.11	0.11	0.15	0.15	0.20	0.19	0.12	0.18	0.22	0.22	0.30
Gemma-3 (27B)	0.59	0.56	0.62	0.64	0.65	0.69	0.64	0.60	0.73	0.69	0.70	0.70
Pangea (7B)	0.00	0.03	0.14	0.10	0.15	0.20	0.03	0.07	0.05	0.09	0.12	0.17
Phi-4 (7B)	0.04	0.06	0.22	0.10	0.17	0.25	0.00	0.03	0.13	0.09	0.15	0.20
Titu-LLM (2B)	0.00	0.00	0.00	0.02	0.02	0.02	0.00	0.01	0.00	0.05	0.05	0.05
Bong-LLaMA (3B)	0.00	0.00	0.00	0.02	0.02	0.02	0.11	0.12	0.02	0.07	0.09	0.11
Bangla-LLaMA (3B)	0.01	0.07	0.12	0.10	0.19	0.21	0.04	0.09	0.12	0.13	0.15	0.15
Bangla-Gemma (9B)	0.04	0.18	0.04	0.10	0.15	0.16	0.03	0.10	0.15	0.08	0.19	0.21
TigerLLM (1B)	0.52	0.58	0.61	0.61	0.64	0.70	0.69	0.57	0.61	0.65	0.68	0.71
TigerLLM (9B)	0.51	0.74	0.60	0.63	0.69	0.72	0.58	0.63	0.75	0.61	0.68	0.73

Table 4: Pass@{1,10,100} comparison on Bangla and Bangla→English-MT variants of mHumanEval and MBPP. We observe a similar or worse set of results across the board with MT prompts compared to Bangla.

$K \in \{1, 10, 100\}$.

- $K = 1$: measures single-shot quality (default user experience).
- $K = 10$: reflects a small, practical shortlist for manual inspection.
- $K = 100$: estimates the model's upper potential under heavy sampling.

Observation The results reveal a consistent performance gap, with most models favoring English over Bangla. While proprietary models excel in English, they are outperformed by TigerLLM in Bangla. This performance drop in Bangla is even more pronounced among the open-source models. Gemma 3 is the exception in this group, showing only a moderate decline.

The Bangla-specific models underperform significantly in English and perform even worse in Bangla. This leaves TigerLLM as the only model with similar performance in Bangla compared to English.

Analysis Most proprietary and large open-source models are trained on corpora dominated by English, so they naturally excel on the English subsets of both benchmarks (Common Crawl Foundation, 2008–2025). Since Bangla appears only sparsely—and often without paired code examples—these models struggle to map Bangla instructions to valid programs, which leads to the sharp drops we see across mHumanEval and MBPP. Gemma 3 (27 B) shows a smaller gap because its greater capacity and multilingual pretraining let it transfer knowledge more effectively, but the English-first bias remains clear. In contrast, TigerLLM is fine-tuned on a purpose-built Bangla–instruction corpus, so it retains almost all of its English-side skill while markedly boosting Bangla performance—even though it is one to two orders of magnitude smaller than the proprietary systems.

Open-source Bangla-specific models (e.g., Bong-LLaMA, Bangla-Gemma) fare poorly because their training data are small, noisy, or text-only; without consistent, executable code pairs, they cannot learn the rigorous syntax and logical structure that Pass@K rewards. TigerLLM's curated data close this gap: its 1 B variant rivals much larger multilingual models in Bangla, and the 9 B version narrows the English gap to within a few points while establishing a clear Bangla lead. These results underscore a practical lesson for low-resource languages—high-quality, domain-specific data outweigh both model size and generic multilingual pre-training when the task demands precise code generation.

5. RQ2 - Does a simple *Bangla* → *English* machine-translation help?

As some of the recent works (Toukmaji and Flanigan, 2025) have shown, LLMs often perform better when the prompts are simply translated to English before feeding them to the model. Again, this is also a scenario that is not explored in the domain of *Code Generation*. In this section, we attempt to provide some careful insights into our second RQ, investigating whether a simple machine translation for the prompts will solve the issue of the performance drop, as shown in Table 3. We adopt a similar experimental setup with the same benchmarks, models, and metrics from the previous experiment, detailed in Section 4.

Prompt Translation We curate two new benchmarks from our two existing ones—mHumanEval-ben & MBPP-Bangla—which are human-generated. We use NLLB (Costa-Jussà et al., 2022) for the translation, as this is the SOTA model for *Bangla* → *English* machine-translation. Thus, we compile mHumanEval-MT & MBPP-MT. We then carry out the similar set of experiments as

Bangla-Code-Instruct	-SI *(Self-Instruct)*	-Syn *(Synthetic)*	-TE *(Translated)*
Size	100,000	100,000	100,000
Method	Self-Instruction	Synthetic Generation	MT + Filtering
Seed/Source	5000 *(Expert)*	Set of Topics	Evol-Instruct
Teacher Model(s)	GPT-4o	GPT-4o & Claude-3.5	—
MT Model(s)	—	—	NLLB-200
Code Validation	Syntax + Execution Check	Syntax + Execution Check	Retained Source Code
Filtering Metric(s)	Cosine Similarities	BERTScore	BERTScore + Comet QE
Prompt Origin	Semi-Natural	Synthetic	Translated
Code Origin	Synthetic	Synthetic	Natural (Source)

Table 5: Comparing details of the three subsets of Bangla-Code-Instruct; **-SI**, **-Syn** & **-TE**.

Bangla	MT (incorrect)	English (Actual)
অক্ষর	Letter	Character
চলক	Clever	Variable
স্ট্রিং	Rope	String
অ্যারে	Row	Array
লুপ	Whirlpool	Loop

Figure 1: Incorrect keywords generated by machine translation.

Section 4, but this time contrast between Bangla and Bangla-MT (Bangla→English) variants of the benchmarks.

Observation Unlike the previous set of experiments, we do not notice any significant performance decline; rather, the results are mostly similar or a bit worse with English-MT variants to Bangla. While the proprietary models do better in general, their performance does not match the original English benchmark (Table 3), the same with all the other families of models. Hence, we can empirically compare the LLMs' performance over different variants of the same prompts as follows:

> Performance when prompted in **English** > Performance when prompted in **Bangla** > Performance when prompted in **English-MT**

Analysis As we investigate the poor results with MT prompts, we notice a very specific trend leading to the generation of unexpected performance. As the coding prompts are translated into English prompts, several code-specific keywords are often translated into words that do not retain the same meaning. A few common examples are shown in Figure 1, which fails to describe to task and misleads the models to generate poor results.

6. Bangla-Code-Instruct

As is evident from the experiments of Sections 4 & 5, recent LLMs perform poorly with Bangla coding tasks, and the issue can not be solved with MT. Hence, we curate three instruction-tuning datasets in an attempt to finetune LLMs for this particular task and language. The datasets are tailored for Bangla: Bangla Code Instruct-SI, -Syn, and -TE. These datasets are specifically designed to capture diverse aspects of code generation and instruction understanding, ensuring a robust training foundation[4] (see Table 5).

6.1. Bangla-Code-Instruct-SI

This dataset consists of 100,000 instruction-code pairs generated via self-instruction (Wang et al., 2023b). The process begins with 5000 seed prompts manually authored in Bangla by programming experts. These are then used to generate a larger set of 'semi-natural' instructional prompts (human-seeded and LLM-evolved using GPT-4o), also in Bangla. The corresponding Python code for each instruction is generated by GPT-4o and also *execution-validated*, signifying that it passed both a syntax check (using ast.parse[5]) and successful execution in a controlled environment (*Python 3.13.0, 10s timeout, 16GB memory*).

6.2. Bangla-Instruct-Syn

This subset provides 100,000 synthetic Bangla instruction-Python code pairs generated by GPT-4o (OpenAI, 2023) and Claude 3.5-Sonnet (Anthropic, 2023). To ensure instructional diversity, new instructions are compared against existing ones; a BERTScore (Zhang et al.) of ≥ 0.7 against any existing instruction results in the new pair being discarded. The LLMs are prompted in Bangla to produce these Bangla instructions and corresponding code for diverse tasks. For this synthetic set,

[4] huggingface.co/datasets/md-nishat-008/Bangla-Code-Instruct

[5] docs.python.org/3/library/ast.html

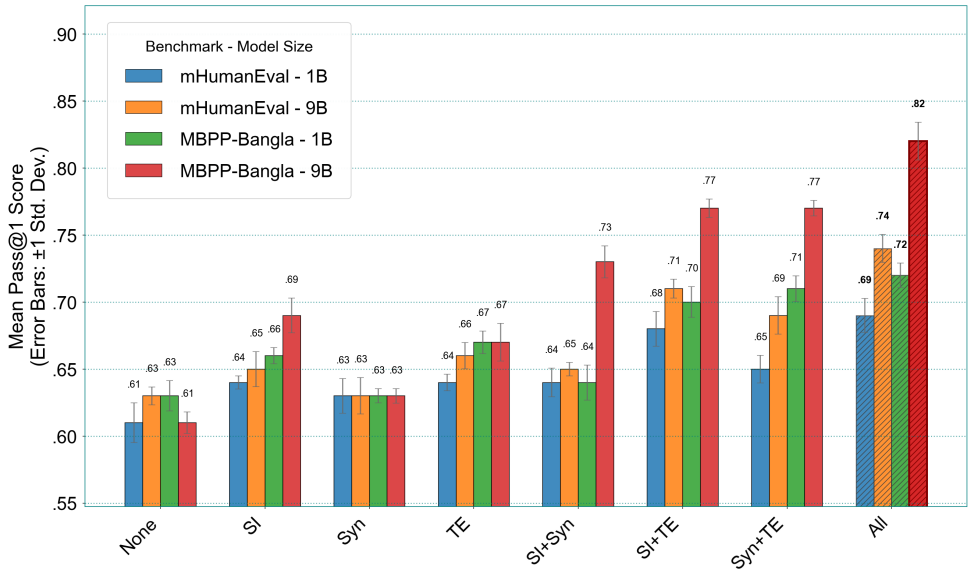

Figure 2: Performance (Pass@1) comparison for different combinations of the SI, Syn, and TE instruction datasets across model sizes (1B vs 9B).

code is validated for syntax (ast.parse) and execution (similar to -SI), aiming to broaden task diversity. This set complements the human-seeded data, though the naturalness of LLM-generated Bangla may differ from expert-authored versions.

6.3. Bangla-Instruct-TE

The final subset contains 100,000 prompt-code pairs by translating English instructions from Evol-Instruct (Xu et al., 2023) using multiple MT models and selecting the best translation based on CometKiwi-22 QE (Rei et al., 2020) (> 0.85) and BERTScore F1 (> 0.95). The original code is retained.

The combined dataset (Table 5) of 300,000 examples provides TigerCoder with diverse training signals from human-seeded (SI), purely synthetic (Syn), and translation-based (TE) sources.

7. TigerCoder

We choose TigerLLM (Raihan and Zampieri, 2025) as our base model, as the results from Table 3 & 4 strongly indicate its efficacy in Bangla code generation. We finetune it using Bangla-Code-Instruct to build TigerCoder family that represents the first dedicated family of Large Language Models specifically optimized for Bangla code generation tasks.

7.1. Experimental Setup

We conduct finetuning on a single NVIDIA A100 (40GB) through Google Colab[6], supported by 80GB RAM and 256GB storage. The process completes in approximately 96 hours, proving sufficient for model adaptation and task-specific optimization with minimal computational overhead.

Hyperparameter	1B	9B
Max Sequence Length	2048	2048
Batch Size (Train / Eval)	16	32
Gradient Accum. Steps	4	8
Number of Epochs	3	3
Learning Rate	1×10^{-5}	1×10^{-6}
Weight Decay	0.02	0.04
Warm-up Steps	10%	15%
Optimizer	AdamW	AdamW
LR Scheduler	Cosine	Cosine
Precision	BF16	BF16
Evaluation Strategy	Steps	Steps
Evaluation Steps	50	250
Save Strategy	Steps	Steps
Save Steps	Varies	Varies
Seed	42	42

Table 6: Empirically selected hyperparameters for fine-tuning the TigerCoder model family.

7.2. Fine-tuning

Given three instruction datasets with distinct curation methods, we fine-tune models on each individually and in all possible combinations for thorough analysis. Figure 2 shows strong base model performance (especially 1B). Fine-tuning on single datasets generally improves results, with SI and TE proving more effective than Syn alone. Dataset combinations exhibit clear synergistic effects: the SI + TE pairing significantly boosts performance, achieving top scores for the 9B model

[6] colab.research.google.com

	mHumanEval *Bangla*						MBPP *Bangla*					
	P@1		P@10		P@100		P@1		P@10		P@100	
	Score	Δ	Score	Δ	Score	Δ	Score	Δ	Score	Δ	Score	Δ
TigerCoder (1B)	0.69	0.05↑	0.73	0.04↑	0.77	0.05↑	0.74	0.05↑	0.74	0.04↑	0.81	0.08↑
TigerCoder (9B)	0.75	0.11↑	0.80	0.11↑	0.84	0.12↑	0.82	0.13↑	0.84	0.14↑	0.91	0.18↑

Table 7: Pass@K scores for TigerCoder models with shaded improvements (Δ) over the strongest prior baseline (Gemma-3 27B or TigerLLM-9B; see Table 3). Darker teal indicates a larger gain; arrow denotes improvement.

on mHumanEval (tying with all three datasets). Using all datasets (SI + Syn + TE) yields the best overall results across both benchmarks and model sizes, pushing the 9B model to 0.82 Pass@1 on MBPP-Bangla. The larger 9B model benefits more from comprehensive fine-tuning, particularly with combined datasets, compared to the 1B model. Performance on MBPP-Bangla is generally higher than on mHumanEval across most configurations.

7.3. Evaluation

We benchmark TigerCoder on two complementary Bangla code–generation suites: mHumanEval-ben, whose tasks resemble traditional docstring-completion, and our conversational MBPP-Bangla. Because these benchmarks differ in prompt style and program length, the performance gaps seen in Table 7 reveal how well a model copes with terse versus chat-like instructions.

7.4. Discussion

Table 7 shows that both TigerCoder variants eclipse the strongest prior baselines (Gemma-3 27B and *TigerLLM* 9B) across every metric. The 1 B model already attains **0.69 P@1** on mHumanEval-ben and **0.74 P@1** on MBPP-Bangla, beating models up to 27× its size by 4–8 percentage points. Scaling to 9 B pushes the frontier further to **0.75** and **0.82 P@1**, with even larger margins at higher K (see the shaded Δ columns). Proprietary APIs score in the mid-0.5 to low-0.6 range on Bangla (cf. Table 3), and existing Bangla-specific models lag far behind, confirming that TigerCoder fills a substantial capability gap.

> Our approach delivers marked efficiency gains: the 1 B model surpasses systems 27 × larger by **4–8%**, while the 9 B variant widens the lead to **11–18%** despite being only one-third their size.

7.5. Ablation Study

Given three instruction datasets with distinct curation methods, we fine-tune models on each individually and in all possible combinations for thorough analysis. Figure 2 shows strong base model performance (especially 1B). Fine-tuning on single datasets generally improves results, with SI and TE proving more effective than Syn alone. Dataset combinations exhibit clear synergistic effects: the SI + TE pairing significantly boosts performance, achieving top scores for the 9B model on mHumanEval (tying with all three datasets). Using all datasets (SI + Syn + TE) yields the best overall results across both benchmarks and model sizes, pushing the 9B model to 0.82 Pass@1 on MBPP-Bangla. The larger 9B model benefits more from comprehensive fine-tuning, particularly with combined datasets, compared to the 1B model. Performance on MBPP-Bangla is generally higher than on mHumanEval across most configurations.

7.6. Performance on other PLs

TigerCoder models also exhibit strong performance for Bangla Code Generation in programming languages other than Python, as we evaluate it on C++, JAVA, JavaScript and Ruby subsets of mHumanEval-Bangla and our MBPP-Bangla.

7.6.1. C++

Table 8 shows that only the Bangla-specialised TigerCoder models break the 0.70 Pass@1 and 0.80 Pass@100 barriers on both mHumanEval and

C++						
Model	**mHumanEval** *Bangla*			**MBPP** *Bangla*		
	P@1	P@10	P@100	P@1	P@10	P@100
GPT-3.5	0.36	0.39	0.41	0.42	0.45	0.43
Gemini-Flash 2.5	0.44	0.40	0.42	0.49	0.42	0.59
GPT-4o-mini	0.43	0.45	0.42	0.37	0.35	0.32
LLaMA-3.2 (11B)	0.00	0.00	0.00	0.06	0.01	0.18
Gemma-3 (27B)	0.25	0.27	0.31	0.15	0.28	0.32
Pangea (7B)	0.00	0.00	0.00	0.00	0.00	0.04
Phi-4 (7B)	0.00	0.05	0.05	0.00	0.00	0.00
Titu-LLM (2B)	0.00	0.00	0.00	0.00	0.00	0.00
Bong-LLaMA (3B)	0.00	0.00	0.00	0.00	0.00	0.00
Bangla-LLaMA (3B)	0.00	0.08	0.03	0.00	0.03	0.00
Bangla-Gemma (9B)	0.00	0.04	0.01	0.00	0.02	0.10
TigerLLM (1B)	0.44	0.52	0.51	0.47	0.45	0.49
TigerLLM (9B)	0.48	0.50	0.57	0.50	0.58	0.56
TigerCoder (1B)	0.64	0.68	0.72	0.66	0.66	0.72
TigerCoder (9B)	0.67	0.73	0.78	0.72	0.79	0.82

Table 8: C++ – Pass@{1,10,100} comparison on Bangla variants of mHumanEval and MBPP. Darker cells indicate better performance.

MBPP, with TigerLLM trailing but still leading all generic systems; every other proprietary or open-source baseline remains below 0.45 Pass@1, underscoring the value of targeted, language-aware training for low-resource code generation.

7.6.2. JAVA

For JAVA code generation, as shown in Table 9, the TigerCoder family demonstrates a commanding performance, establishing a new state-of-the-art. It decisively outperforms proprietary systems, while most other multilingual and Bangla-specific models are rendered ineffective with scores near zero. The significant improvement over its TigerLLM base underscores the impact of specialized fine-tuning, positioning TigerCoder-9B as the top-performing model across all benchmarks.

JAVA						
Model	**mHumanEval**			**MBPP**		
	Bangla			*Bangla*		
	P@1	P@10	P@100	P@1	P@10	P@100
GPT-3.5	0.29	0.31	0.30	0.35	0.33	0.36
Gemini-Flash 2.5	0.34	0.32	0.35	0.38	0.32	0.48
GPT-4o-mini	0.36	0.35	0.33	0.31	0.25	0.22
LLaMA-3.2 (11B)	0.00	0.00	0.00	0.01	0.00	0.07
Gemma-3 (27B)	0.18	0.19	0.21	0.05	0.16	0.22
Pangea (7B)	0.00	0.00	0.00	0.00	0.00	0.01
Phi-4 (7B)	0.00	0.01	0.02	0.00	0.00	0.00
Titu-LLM (2B)	0.00	0.00	0.00	0.00	0.00	0.00
Bong-LLaMA (3B)	0.00	0.00	0.00	0.00	0.00	0.00
Bangla-LLaMA (3B)	0.00	0.04	0.00	0.00	0.00	0.00
Bangla-Gemma (9B)	0.00	0.01	0.00	0.00	0.00	0.04
TigerLLM (1B)	0.35	0.41	0.42	0.37	0.39	0.41
TigerLLM (9B)	0.41	0.44	0.48	0.42	0.49	0.47
TigerCoder (1B)	0.58	0.64	0.67	0.61	0.60	0.66
TigerCoder (9B)	0.62	0.68	0.73	0.67	0.72	0.76

Table 9: **JAVA** – Pass@{1,10,100} comparison on Bangla variants of mHumanEval and MBPP. Darker cells indicate better performance.

7.6.3. JavaScript

In JavaScript code generation (Table 10), the TigerCoder models again deliver a superior performance, setting the benchmark for this language. They substantially outperform proprietary models, while the majority of other open-source and Bangla-specific LLMs struggle, posting scores that are frequently zero. The clear improvement from TigerLLM to TigerCoder validates the effectiveness of our fine-tuning approach, with TigerCoder-9B solidifying its position as the most capable model across all evaluation metrics.

8. Conclusion

In this work, we first carefully address the significant performance gap in LLMs' code generation capabilities in Bangla comapred to English. Based on the gathered results and the follow-up analysis

JavaScript						
Model	**mHumanEval**			**MBPP**		
	Bangla			*Bangla*		
	P@1	P@10	P@100	P@1	P@10	P@100
GPT-3.5	0.22	0.25	0.19	0.28	0.21	0.29
Gemini-Flash 2.5	0.28	0.21	0.25	0.31	0.22	0.39
GPT-4o-mini	0.29	0.28	0.24	0.22	0.18	0.13
LLaMA-3.2 (11B)	0.00	0.00	0.00	0.00	0.00	0.02
Gemma-3 (27B)	0.11	0.12	0.15	0.01	0.08	0.14
Pangea (7B)	0.00	0.00	0.00	0.00	0.00	0.00
Phi-4 (7B)	0.00	0.00	0.00	0.00	0.00	0.00
Titu-LLM (2B)	0.00	0.00	0.00	0.00	0.00	0.00
Bong-LLaMA (3B)	0.00	0.00	0.00	0.00	0.00	0.00
Bangla-LLaMA (3B)	0.00	0.01	0.00	0.00	0.00	0.00
Bangla-Gemma (9B)	0.00	0.00	0.00	0.00	0.00	0.01
TigerLLM (1B)	0.28	0.33	0.35	0.29	0.31	0.33
TigerLLM (9B)	0.33	0.37	0.39	0.35	0.41	0.40
TigerCoder (1B)	0.53	0.59	0.61	0.55	0.54	0.61
TigerCoder (9B)	0.57	0.63	0.68	0.62	0.67	0.71

Table 10: **JavaScript** – Pass@{1,10,100} comparison on Bangla variants of mHumanEval and MBPP. Darker cells indicate better performance.

(Section 4), we shed light on **RQ1:** *To what extent do state-of-the-art Code LLMs preserve their code-generation quality when the natural language part of the prompt is written in Bangla rather than English?*

> **RQ1 Findings:** For *Code Generation*, LLMs exhibit a notable drop in performance with Bangla prompts, often failing to capture the full requirements of the request compared to their English counterparts.

We further show that machine-translating Bangla prompts to English does not improve the results (Section 5) answering **RQ2:** *Does a simple Bangla → English machine-translation step applied to each coding prompt significantly boost generation quality compared with direct Bangla inference?*

> **RQ2 Findings:** For the task of *Code Generation*, *Bangla → English* machine-translation does not help improve the performance.

To bridge these gaps, we introduce TigerCoder as the first family of LLMs to specifically and effectively tackle the critical void in code generation for the Bangla language. We have not only identified a gap but have also built the essential foundation to fill it, constructing three diverse, high-quality instruction-following datasets and the comprehensive MBPP-Bangla benchmark. The results are decisive: our TigerCoder models demonstrate a substantial leap in performance, outperforming existing systems and setting a new standard for the field. This research results in a crucial finding:

> *Carefully curated, high-quality datasets empower smaller, efficient models to overcome low-resource limitations*, decisively challenging the prevailing notion that *'scale alone drives perfor-*

mance' (Kaplan et al., 2020; Hoffmann et al., 2022).

Our work confirms that targeted data curation is a powerful and resource-efficient path toward true language comprehensiveness in NLP. By demonstrating that a 1B parameter model can surpass general models more than 27 times its size on specialized tasks, we establish an effective and replicable blueprint for the future of efficient, high-performance LLM development for Bangla and other low-resource languages.

Acknowledgments

We are thankful to the anonymous reviewers for their feedback. This work has been partially supported by the National Science Foundation under CAREER award 2439202. Computational resources for experiments were provided by the Office of Research Computing at George Mason University (URL: https://orc.gmu.edu) and funded in part by grants from the National Science Foundation (Awards Number 1625039 and 2018631).

Ethics Statement

We adhere to the ethical guidelines outlined in the LREC 2026 CFP[7]. Our benchmark creation involved careful translation and verification by qualified native speakers. Instruction datasets were generated using various methods, including expert input and automated filtering for quality and diversity. While acknowledging the inherent challenges in mitigating all potential biases from source data or generation models, we promote transparency through the open-source release of our models, datasets, and benchmark. We encourage responsible downstream use and community scrutiny.

9. Bibliographical References

Kabir Ahuja, Anirudh Das, Sandipan Das, Ashwini Deshpande, Sebastian Gehrmann, Anup Gopinath, Arya Guha, Pooja Kumar-Jois, Prem Mani, Ashwin Paranjape, et al. 2023. Mega: Multilingual evaluation of generative ai. *arXiv preprint arXiv:2310.10567*.

Kabir Ahuja, Karan Sikka, Madhusudana Nallasamy, and Chandrika Singh. 2024. Few-shot learning for low-resource languages with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.

Anthropic. 2023. Claude: The anthropic ai language model. *Online documentation*. Available at: https://www.anthropic.com.

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

Abhik Bhattacharjee, Tahmid Hasan, Wasi Ahmad, Kazi Samin Mubasshir, Md Saiful Islam, Anindya Iqbal, M Sohel Rahman, and Rifat Shahriyar. 2022. Banglabert: Language model pretraining and benchmarks for low-resource language understanding evaluation in bangla. In *Findings of the Association for Computational Linguistics: NAACL 2022*.

Pramit Bhattacharyya, Joydeep Mondal, Subhadip Maji, and Arnab Bhattacharya. 2023. Vacaspati: A diverse corpus of bangla literature. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*.

Damián Blasi, Antonios Anastasopoulos, and Graham Neubig. 2022. Systematic inequalities in language technology performance across the world's languages. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

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

Common Crawl Foundation. 2008–2025. Common crawl. https://commoncrawl.org.

Marta R Costa-Jussà, James Cross, et al. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Adam Afflerbach, Adriane Abramowitz, Aditya Singhal, Aditya Balaji, Akshat Agrawal, Akshat Ravinuthala, Akshay Ramakrishnan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Syed Mohammed Sartaj Ekram, Adham Arik Rahman, Md Sajid Altaf, Mohammed Saidul Islam,

[7] https://lrec2026.info/calls/second-call-for-papers/

Tareq Mahmood Jamil, Shadman Sakib Alam, Irfan Kabir, Mohammad Nasim, Enamul Hossain, and Nawshad Akhter. 2022. Banglarqa: A benchmark dataset for under-resourced bangla language reading comprehension-based question answering with diverse question-answer types. In *Findings of the Association for Computational Linguistics: EMNLP 2022*.

Google Gemma Team. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Google. 2024. Gemini 2.5: The next generation of multimodal ai models.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, et al. 2022. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Sai Iyer, Jiawei Baek, Daisy Li, Zoekook, and Spencer Weiss. 2022. Optimizing tokenization for low-resource languages in machine translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the nlp world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Zihao Li, Yucheng Shi, Zirui Liu, Fan Yang, Ali Payani, Ninghao Liu, and Mengnan Du. 2024. Quantifying multilingual performance of large language models across languages. *arXiv preprint arXiv:2404.11553*.

Microsoft. 2024. Phi-4: A 14b-parameter model with instruct-following and multi-modal capabilities.

Niklas Muennighoff, Alex Wang, Alena Fenogenova, Fangyu Huang, Francesca Toni, Adina Williams, and Colin Wang. 2023. Cross-lingual generalization through multitask finetuning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

OpenAI. 2024. Gpt-4o system card.

Liliana Pasquale, Antonino Sabetta, Marcelo d'Amorim, Péter Hegedűs, Mehdi Tarrit Mirakhorli, Hamed Okhravi, Mathias Payer, Awais Rashid, Joanna CS Santos, Jonathan M Spring, et al. 2025. Challenges to using large language models in code generation and repair. *IEEE Security & Privacy*, 23(2):81–88.

Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. 2025a. mHumanEval - a multilingual benchmark to evaluate large language models for code generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*.

Nishat Raihan, Mohammed Latif Siddiq, Joanna CS Santos, and Marcos Zampieri. 2025b. Large language models in computer science education: A systematic literature review. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*, pages 938–944.

Nishat Raihan and Marcos Zampieri. 2025. TigerLLM - a family of bangla large language models. In *Proceedings of ACL*.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. Comet: A neural framework for mt evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702.

Pritika Rohera, Chaitrali Ginimav, Gayatri Sawant, and Raviraj Joshi. 2025. Better to ask in english? evaluating factual accuracy of multilingual llms in english and low-resource languages. *arXiv preprint arXiv:2504.20022*.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Lisa Schut, Yarin Gal, and Sebastian Farquhar. 2025. Do multilingual llms think in english? *arXiv preprint arXiv:2502.15603*.

Sheikh Shafayat, H Hasan, Minhajur Mahim, Rifki Putri, James Thorne, and Alice Oh. 2024. BEnQA: A question answering benchmark for Bengali and English. In *Findings of the Association for Computational Linguistics: ACL 2024*.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Christopher Toukmaji and Jeffrey Flanigan. 2025. Prompt, translate, fine-tune, re-initialize, or instruction-tune? adapting llms for in-context learning in low-resource languages. In *Proceedings of the ACL GEM Workshop*. ArXiv:2506.19187.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Md Nafis Uddin, Masum Khan, Nabila Hasan, and Mahmudul Hossain. 2023. Exploring code-mixed bangla text in large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Tianyi Wang, Yang Ye, Panupong Pasupat, Aohan Wan, Grant Friedman, Jiacheng Tu, Maya Schaar, Jason Wei, Suriya Gunasekar, Matthew Richardson, et al. 2023a. Babelcode: Llm as a polyglot programmer. *arXiv preprint arXiv:2303.03845*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, et al. 2023b. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*.

Yuhan Wang, Xuanhe Zhou, Ruoxi Chen, et al. 2023c. A comprehensive capability analysis of gpt-3 and gpt-3.5 series models.

Canwen Xu, Ruqing Wang, Yeyun Gong, et al. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Xiang Yue, Yueqi Song, Akari Asai, et al. 2024. Pangea: A fully open multilingual multimodal llm for 39 languages. In *The Thirteenth International Conference on Learning Representations*.

Abdullah Khan Zehady, Safi Al Mamun, Naymul Islam, and Santu Karmaker. 2024. Bongllama: Llama for bangla language. *arXiv preprint arXiv:2410.21200*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.