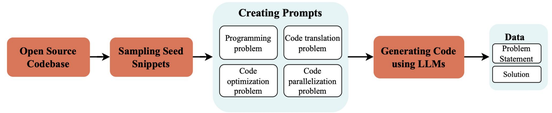
Creating Code LLMs for HPC: It's LLMs All the Way Down

Aman Chaturvedi, Daniel Nichols (advisor), Siddharth Singh (advisor), Abhinav Bhatele (advisor)
Department of Computer Science, University of Maryland

Abstract

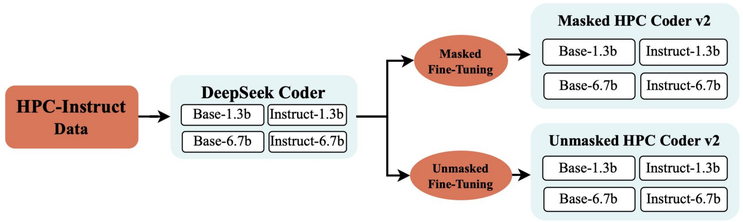
Large language models (LLMs) are being used increasingly by software developers, researchers, and students to assist them in coding tasks. While newer LLMs have been improving their coding abilities with regards to serial coding tasks, they consistently perform worse when it comes to parallelism and HPC-related coding tasks. Bridging this gap and creating HPC-capable code LLMs could significantly improve developer productivity. The current poor performance of LLMs on HPC-related problems can be partially attributed to the lack of significant HPC data in their training datasets, which is what we address in this poster. We present HPC-Coder v2, a new LLM created by fine-tuning a previous code LLM using synthetic HPC code data. We demonstrate that it is one of the most capable open-source LLMs for generating parallel code to date.

Generating Synthetic HPC Code Data



- Previous work [1] has shown that smaller LLMs can be improved by fine-tuning on LLM-generated synthetic coding samples
- We generate four types of samples with an LLM: programming, translation, optimization and parallelization problems
- The LLM is asked to generate a problem statement and a solution
- To generate more diverse data samples, we use seed snippets from open source codebases to inspire the LLM
- Our final dataset, HPC-Instruct, has 120k data samples generated with Gemini-Pro, DBRX, Llama-3-70B, and Mixtral-7B

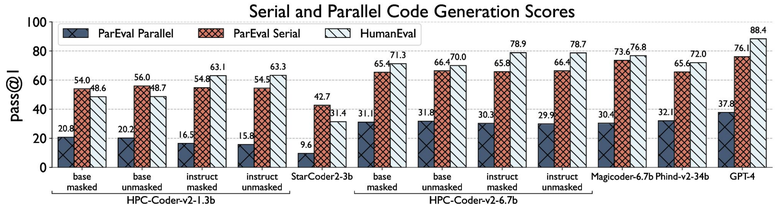
HPC-Coder v2: Fine-tuning DeepSeek Coder



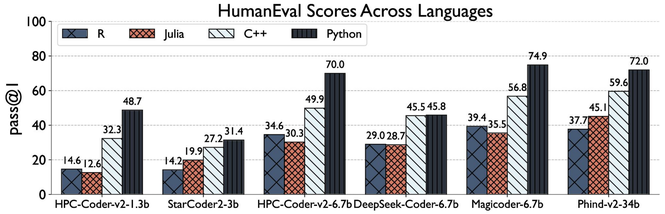
- Base models used: DeepSeek Coder 1.3 billion and DeepSeek Coder 6.7 billion
- Fine-tuned using the AxoNN framework [4] on 4 Perlmutter nodes with 4 80GB A100 GPUs
- Four variants of each model are created: the base and the instruct model versions, with and without instruction masking
- In instruction masking, gradients corresponding to instruction tokens are masked during fine-tuning
- Models fine-tuned in bfloat16 with a batch size of 128 and sequence length of 8192 for 2 epochs
- Total fine-tuning time was 208 node-hours for all 8 models

Evaluation Results: Generating Serial and Parallel Code

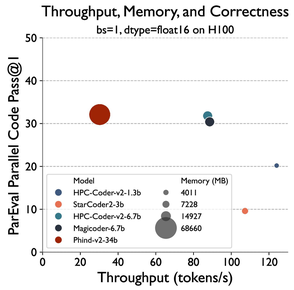
- Used HumanEval [2] and ParEval [3] to evaluate the new models on serial and parallel code generation
- No consistent trend between masked and unmasked models
- Fine-tuning the base models gives the best ParEval scores, but the worst HumanEval scores
- HPC-Coder v2 6.7B is the best open-source LLM under 30B parameters at parallel code generation



HumanEval Performance for Different Languages



- Base, unmasked variants selected as the final HPC-Coder v2 model
- The 1.3B model performs better than StarCoder2 3B model on all languages except Julia
- The 6.7B model performs better than DeepSeek Coder 7B on all languages and is comparable to Phind v2 34B for Python and R
- HPC-Coder v2 models are faster and use less memory than other models while having comparable or better parallel code generation



Conclusion and Future Work

- HPC-Coder v2 6.7B is the best performing open source model for parallel code generation under 30B parameters.
- Future work
  - Generate more data and fine-tune bigger models
  - Investigate inconsistent HumanEval results
  - Evaluate HPC-Coder v2 with more benchmarks

HPC-Coder v2 on HuggingFace

Acknowledgements

This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231 using NERSC award m2404 for 2024.

References

[1] Y. Wei, et. al. "Magicoder: Empowering Code Generation with OSS-Instruct," ICML 2024
[2] M. Chen, et. al. "Evaluating Large Language Models Trained on Code," arXiv 2107.03374
[3] D. Nichols, et. al. "Can Large Language Models Write Parallel Code?" HPDC 2024
[4] S. Singh, et. al. "AxoNN: An asynchronous, message-driven parallel framework for extreme-scale deep learning," IPDPS 2022