# This is the Title

*Your Name*


Master of Science
School of Informatics
University of Edinburgh
2023

# Abstract

# Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

# Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Your Name*)

# Acknowledgements

Any acknowledgements go here.

# Table of Contents

# Chapter 1

# Introduction

In recent years LLM have become very useful for a variety of tasks. One of the tasks that they excel in the code intelligence. The structured syntax and rule-based nature of programming nature are perfect for the application of LLM in contrast to natural language.

The theoretical match along with the success of training very large models on huge lead to the development of very capable models that can perform a variety of tasks involving code generation and understanding (Code Intelligence Models). These code intelligence models can perform code search, vulnerability detection and more tasks as described by code search net.

The feature that we will focus on is the ability of these models to operate in a variety of programming languages. Codex perform with non-trivial results in 8 different languages. The ability of understanding and generating code in a verity of languages, especially underrepresented language is crucial to the ongoing goal democratizing AI as described in section ZZ.

However, this ability of understanding and generating code in multiple programming language is limited by several factors that makes it not accessible to anyone.

The First of these factors is that these multilingual code language models are normally very large (Billions of Parameters) and require huge amounts of computational resources to train which are not accessible to most people. This then makes the only valid option is using pretrained models through APIs (Copilot) which are normally put behind a paywall, or in cases of open-source models like StarCoder, the inference cost is also high for commercial devices.

Some solutions to the inference cost have been suggested like distributed inference, 4-bit quantization and model distillation. While these solutions have proven useful in

a lot of cases, they are still limited to the source model. That is, these solutions are customizable to any programming language of need, but instead, only these seen by the original model.

On the other hand, another solution is using smaller language models which are trained on a single programming language and then fine-tuning them to different programming language. The literature and our experiments ZZ confirm that such small models only perform in acceptable manner when trained on a single programming language and that they can be fine-tuned to achieve good performance in other languages.

In this work we propose using efficient fine-tuning methods to fine-tune small monolingual models to language of choice. We hyposesize that using PEFT will have great benefits that align with the goal of AI democratization. The first and most prominent feature of PEFT is the ability of doing transfer learning in huge quality only by training a small number of parameters. We prove this also applies in our context in experiment ZZ. A follow-up benefit of this also comes up when sharing or saving trained parameters, we make use of this by training and deploying three PEFT in less popular programming language.

The second benefit of using PEFT methods for access to code LLM over Q-bit methods is that PEFT methods are more autonomous and can be used for any programming language of choice. In experiment ZZ we prove that given enough data and time one can fine-tune small language models for his own programming language in less than a day on commercially available GPUs. To further this objective we perform extensive ablation studies to find the optimal parameters and the effect of each on the model training, and then provide our own recommendation and training pipelines and dataset along with the released models.

Finally, we perform an extensive qualitative analysis on our trained models, discuss their shortcomings and provide insights on systemic forms of failure of such models. We also perform a cost-benefit trade off analysis to further simulate real situations in which we expect developer would choose between different hyperparameters in a time and memory constrained environment. We conclude our study with an overall discussion of the finding, the impact and alignment of our work with the goal of AI democratization along with analysis of potential concern and risks in the same context.

# Chapter 2

# Experiments

In this experiment we try to measure the extent to which we can adapt a monolingual code language model form a source programming language (pretraining dataset) to a target programming language (fine-tuning dataset). We design our experiment in order to compare full fine-tuning and parameter efficient tuning in therms of efficiency and performance.

We find that using parameter efficient fine-tuning is effective in learning new programming language and require significantly less memory resource. This low resource requirement gives us the freedom of training for more steps using longer sequences given limited resource. We believe these two features are important for the quality of generation as will be discussed in later section.

In this we will go through our setup highlighting the decision we made and why we made the. Particularly, we are either following best practices , optimizing for our constrained environment or maximizing the code generation from prompt capabilities.

## 2.1 Dataset

The quality of the dataset is an important factor affecting the performance of the trained or fine-tuned model in all machine learning areas. In the context of code generation and understanding, we need to also consider our datasets in terms of legal and safety measures.

For this experiment and most of the rest we curate a small java dataset consisting of github repositories. We start from the stack dataset which is a large crawl of github repositories done on 2018 and cover 47 programming languages. We use the deduplicated and cleaned from harmful code version. We then download the Java subset

of this crawl which is more than 1M repositories. We extract text used for fine-tuning for each repositories and and up with a dataset of total 133000000 files.

Prepossessing is then performed to remove all files with length more than 1000 words. We also filter files with average line length ¡ 5 or ¿ 10. These steps ensure we end up with reasonable code that represent the average java developer. Finally, the data split into train, val and test split as described in table ZZ. The final split and processed versions are available on public for future continuations and experiments.

# Chapter 3

# Conclusions

## 3.1 Final Reminder

The body of your dissertation, before the references and any appendices, *must* finish by page 40. The introduction, after preliminary material, should have started on page 1.

You may not change the dissertation format (e.g., reduce the font size, change the margins, or reduce the line spacing from the default 1.5 spacing). Be careful if you copy-paste packages into your document preamble from elsewhere. Some LaTeX packages, such as `fullpage` or `savetrees`, change the margins of your document. Do not include them!

Over-length or incorrectly-formatted dissertations will not be accepted and you would have to modify your dissertation and resubmit. You cannot assume we will check your submission before the final deadline and if it requires resubmission after the deadline to conform to the page and style requirements you will be subject to the usual late penalties based on your final submission time.

# Bibliography

# Appendix A

# First appendix

## A.1 First section

Any appendices, including any required ethics information, should be included after the references.

Markers do not have to consider appendices. Make sure that your contributions are made clear in the main body of the dissertation (within the page limit).

# Appendix B

# Participants' information sheet

If you had human participants, include key information that they were given in an appendix, and point to it from the ethics declaration.

# Appendix C

# Participants' consent form

If you had human participants, include information about how consent was gathered in an appendix, and point to it from the ethics declaration.