Table of Contents

Background

There has been existing work on automating tests e.g., Git [Figure 6] and Jenkins. While the existing tools are great at tracking and managing source code change and enabling teamwork on a project, they aren't effective at setting benchmarks and conduct tests for the Machine Learning code. Machine Learning has additional data requirements because it consists of training and testing code.

When applying the project, the system should be able to train the datasets based on the training and the benchmarks before moving onto the testing data to determine the model's performance.

Establishing the goals of your project

What is your project fundamentally about?

When building machine learning projects, additional functionalities of training and benchmarking are needed for testing during the code changes. In my project, the goal is to automate the training of the ML code to select the best ML model [1]. My project is divided into two sections: the input containing the ML (training and benchmark) code, and the data. Whenever there is a change made to the repository, the ML system will be retrained to compare with the previous tests. The comparison will help to determine which version of code to become the updated version. Through automation, the project aims to create an iterative process to improve the input machine learning model.

What are you intending to design/build/investigate?

The target is to build a machine learning continuous integration system. The project input will be divided into the code (training and benchmark) and the data sections. I will fit the data onto the model to determine its performance. Then I will produce the visualised performance data e.g., model accuracy, sensitivity, and whether they follow the user's requirements.

Whenever there are changes within the dataset or the ML (training and benchmark code), the program must retrain the existing ML model. Consequently, the project simplifies the testing process and ensure that the code will take less time to reach the deployment stage. Figure 5 visualises the MLCI repository branch when two users attempt concurrent edits. After the project has met its objectives, the code will be moved onto the stage and production stage where the code will be deployed.

What do you intend to deliver as the project results?

Figure 4 is the project architectural diagram providing an overview of the requirements. Using the figure, I aim to combine existing version control system with automated training on the machine learning system. In additional to the backend programming, I intend to host the programs on a browser platform, which will allow the user to conduct the testing from different platforms. The browser will contain a user-friendly menu that leads to training and displaying the model score, which will help the user to effectively apply new changes to the ML repository. If the program is examining large datasets, then I will develop or use a suitable existing data version control system to track the changes in the repository.

What would constitute, in your own and your supervisor's eyes, a 100% satisfactory solution?

A perfect solution would be a web-based platform running the machine learning continuous integration tool. Web browser is portable and can be applied for development on different devices from computer to tablet.

Resolve commit conflicts made by different users as show in Figure 1.

Design

Objective	Stage
Outline the software development cycle.	⏳

Front-end

Objective	Stage
Produce a menu for the different functionalities	⏳
For each function, produce a separate webpage for displaying the outputs.	⏳

Develop a UI for configuring the code for testing.	⏳

Back-end

Objective	Stage
Develop version control and suitably handle the commit conflicts.	⏳
Merge function runs tests on the training and the datasets.	⏳
Include external servers to run and test the program in different environment. [Figure 3]	⏳
Create an iterative process to train and benchmark the ML code.	⏳
Calculate a prediction accuracy score on the training and testing datasets.	⏳

Model

Objective	Stage
Automate pushing code	⏳
Automate the ML models and the data	⏳
Train the Machine learning system in a way that the code ca be run on different machines.	⏳

Data Control

Objective	Stage
Design my own data version control system if there are no suitable existing data version control systems.	⏳
Allowing the user to determining the training and validation sample size.	⏳

Display

Objective	Stage
Using the merge results, generate and display performance statistics.	⏳

Testing

Objective	Stage
Set up a test system to compare the new model with the existing model using a common test score.	⏳

Benchmarking

Objective	Stage
Use the benchmark to perform other operations e.g., push code, model, and data onto the repository.	⏳

In the worst case what is the minimum that needs to be completed to achieve a pass?

My goal is developing the basic frameworks for the program as the minimal requirement. The minimal program will be a working version of the testing and running of the machine learning code on a command prompt system. My further purpose would be to produce a visual framework to run the code and will be finished when the time constraints permit the addition of the supplementary functions.

Planning

Objective	Stage
Build the diagram outlining the program structure.	⏳

Front-end

Objective	Stage
Create a web interface for the continuous integration platform	⏳

Back-end

Data

Objective	Stage
Load the datasets into the repository.	⏳
Apply existing data version control system to keep the data up to date.	⏳

Data Processing

Objective	Stage
Split the data into training and testing groups.	⏳

Code function

Objective	Stage
Detect changes within the repository.	⏳
Link existing source control systems with the developed code.	⏳
Enables the ML code to run different versions of local code.	⏳

Testing

Objective	Stage
For each failed build and test, identify the bugs for the users.	⏳
Provide an outcome for each training process.	⏳
Identify and select a version control system to ensure that the program is up to date.	⏳
I can successfully run the back-end code using the command prompt platform.	⏳
The user can run the ML code (training and benchmarking) as inputs to the software.	⏳

Display

Objective	Stage
Present the benchmark statistics in a visually aesthetic format.	⏳
Generate a probabilistic result (probability in which a result is valid). [3]	⏳

External Libraries

Connect the ML-CI tool with an existing source control (e.g., GitHub or Jenkins)	⏳

What are your personal aims that you hope to achieve?

Design

1. Learn and apply the ML development cycle into the current CI system. 🎯

Technical

1. Improve and demonstrate Python skills at creating the coded solutions. 🎯
2. Advance understanding in continuous integration for helping with future teamwork. 🎯
2. Improve upon the existing automated testing strategies and improving automated training and benchmarking mechanisms for the ML system. 🎯

Testing

1. Automated testing aims to improve the understanding of the software development cycle [Figure 2]. Within the cycle, improve the training for the testing stage and identify the differences between the traditional software testing with testing ML software. 🎯
2. Determine a score (probability) for the validity of the test and a confidence interval. 🎯

Error calculation

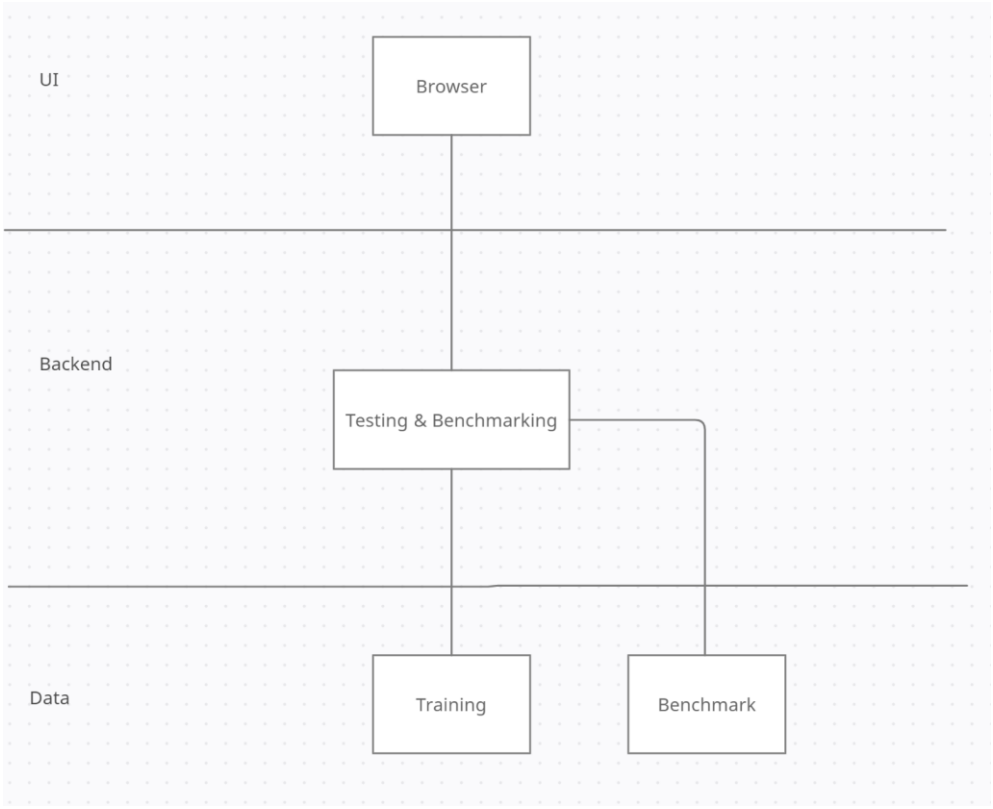
1. Demonstrate building machine learning to minimize the errors from overfitting. 🎯
2. Make some progress within the CI development for the machine learning type of code. 🎯

Planning

1. Apply my research skill into planning and coding the project. 🎯

Optimisation

1. Perform optimisation operations on the machine learning code testing conditions. 🎯


UI
Browser
Backend
Testing & Benchmarking
Data
Training
Benchmark

Figures

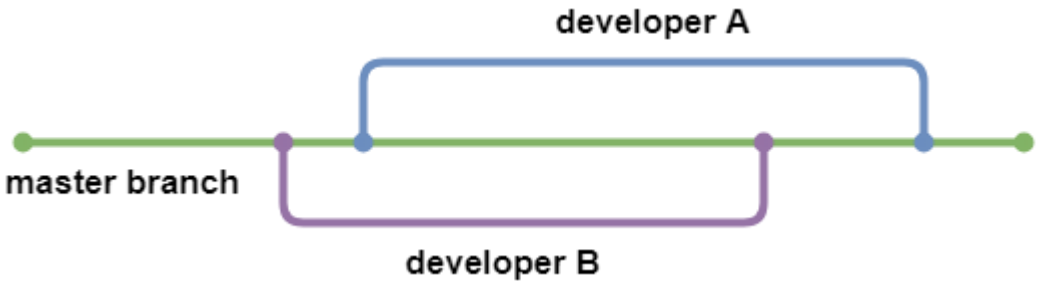


Figure 1

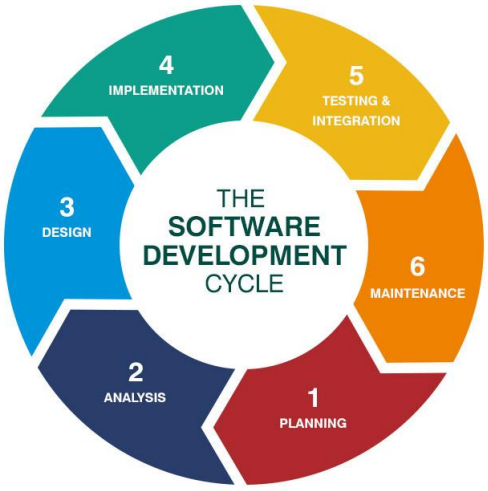


Figure 2

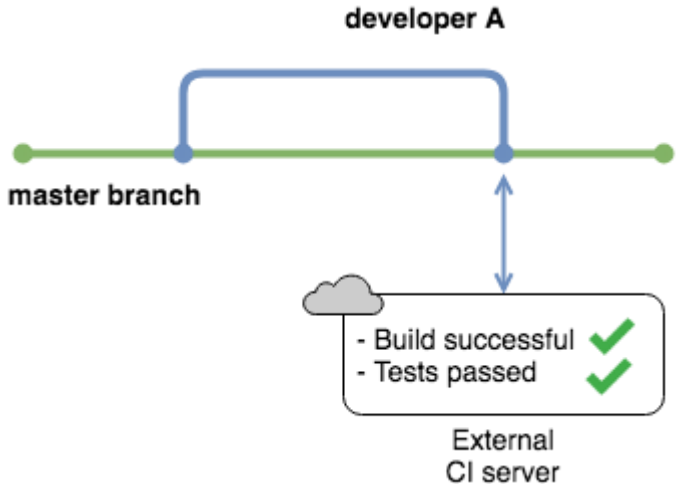


Figure 3

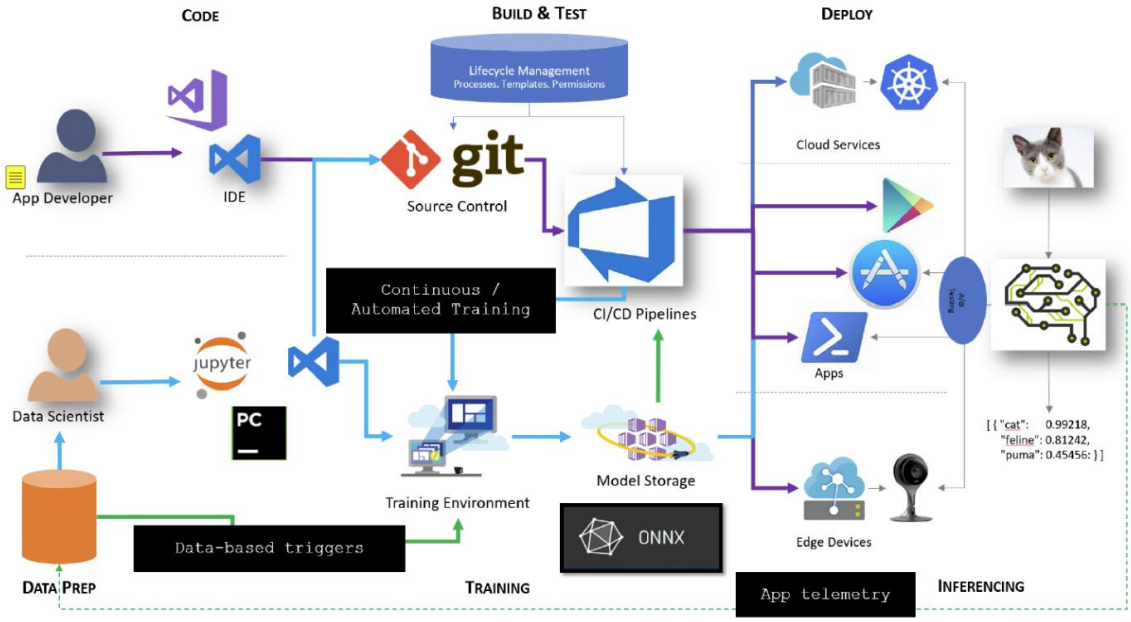


Figure 4

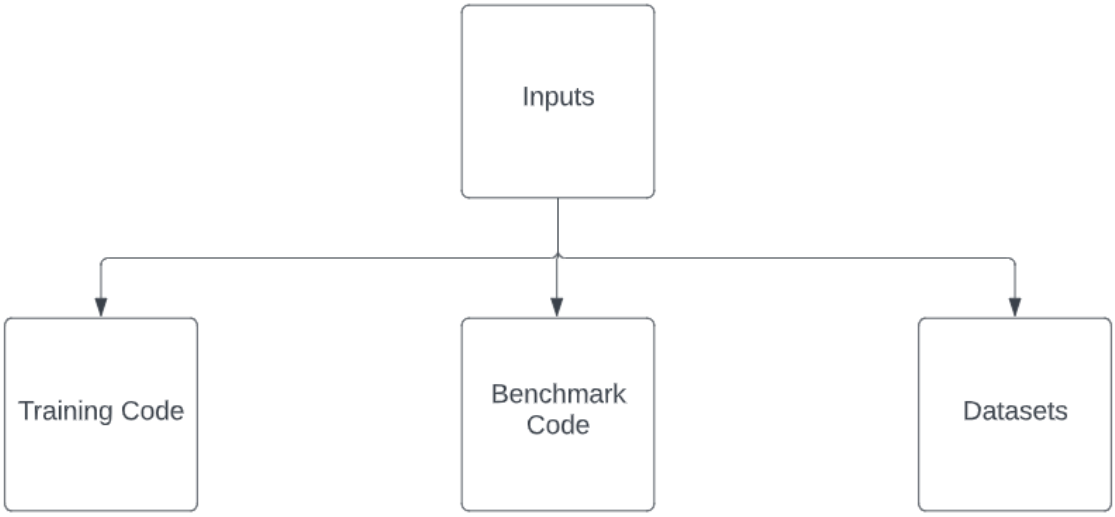


Figure 5

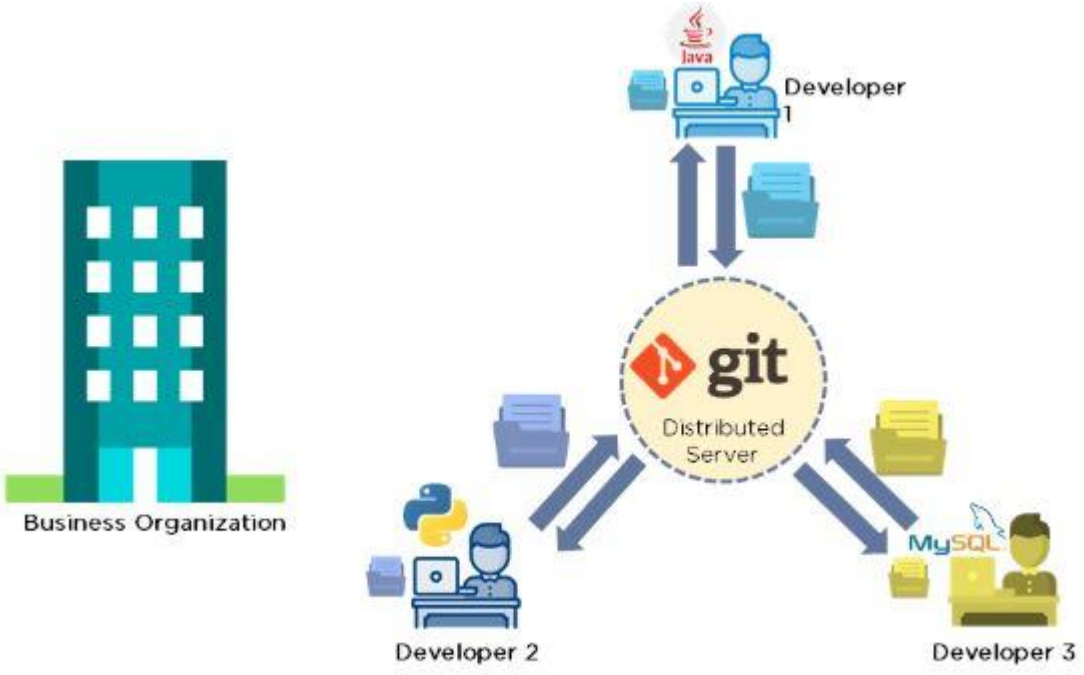


Figure 6

Sources

1. Karlaš, Bojan, et al. "Building continuous integration services for machine learning." Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2020.
2. Danglot, Benjamin, et al. "An approach and benchmark to detect behavioral changes of commits in continuous integration." Empirical Software Engineering 25.4 (2020): 2379-2415.
3. Renggli, Cedric, et al. "Continuous integration of machine learning models with ease. ml/ci: Towards a rigorous yet practical treatment." Proceedings of Machine Learning and Systems 1 (2019): 322-333.