FYP 2019-20 Detailed Project Plan
Building new AI applications for a distributed AI serving system

Waqas Ali
Supervisor: Dr. Heming Cui
Mentor: Shixiong Zhao

October 21, 2019

1 Background

Artificial intelligence is an area of computer science which focuses on granting machines the ability to act intelligently [3]. It's a vast field with limitless applications and each application has its own unique solution. Machine learning, specifically, is a subset of artificial intelligence which learns from data. [4] Nowadays we see artificial intelligence everywhere from spam filters [1] to movie recommendations [2] to virtual assistants to self-driving.

Customers are demanding smarter and smarter capabilities in their machines which leads to its own set of software development challenges; Nvidia summarises them with the PLASTER [5] framework:

- Programmability
- Latency
- Accuracy
- Size of Model
- Throughput
- Energy Efficiency
- Rate of Learning

These challenges arise during the development lifecycle of any machine learning application. A machine learning application has two main stages:

1. Training (Learning from data)
2. Inference (Given an input, predicting an output)

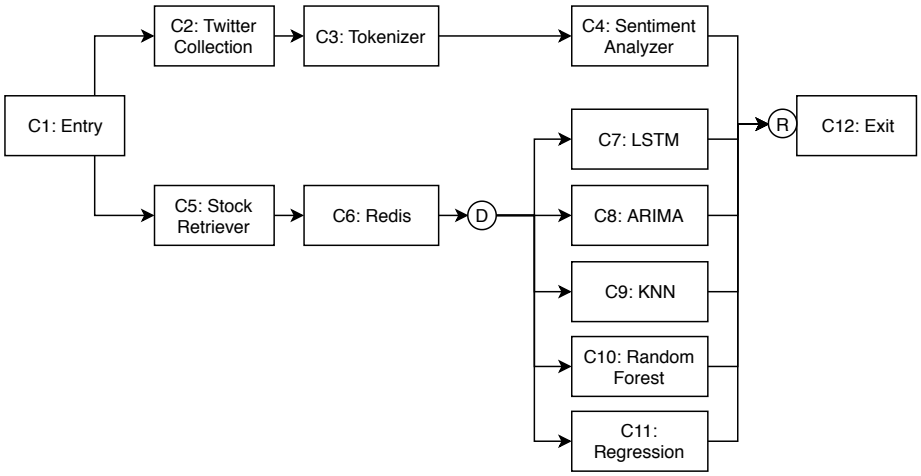

Figure 1: Inference pipeline of a basic stock price prediction service. C2 and C5 are data retrievers. C6 is a caching layer. C7, C8, C9, C10 & C11 are ML models which compete against each other. C4 is a sentiment analyzer whose results are taken into account for price prediction.

For example, for virtual assistants, training the voice recognition model involves exposing the program to hundreds of voice recordings and letting it figure out what sounds correspond to which words. Inference, however, is inputting a new voice recording and transcribing it using the trained voice recognition model.

End-users of machine learning applications are only concerned with inference and they expect real-time response.

As AI application pipelines get complex, inference can take a long time. For instance, a stock price prediction service (Figure 1) involves several steps. On a monolith architecture, all tasks have to be done sequentially (even if they are independent of each other). This could take a long time and hence increase the latency. Moreover, until all tasks for a specific request have finished, processing for a new request cannot be started. Thus, the service cannot handle high number of requests in a given time period i.e. throughput.

This is exactly the kind of situation where a distributed system could shine. By deploying several servers which could independently accomplish the pipeline tasks above, theoretically we can improve the latency and throughput of an AI application (concerns which were highlighted in the PLASTER framework [5]).

2 Objective

The project's aim is to improve latency and throughput of AI applications by implementing and deploying them on a distributed system. As a proof of concept, an AI application will be chosen which has a complex inference pipeline

and it will be developed on a distributed system.

To prove that a distributed architecture can indeed improve latency and throughput, performance will be compared across systems of various specifications:

1. 1 node for n tasks (baseline)
2. less than n nodes for n tasks
3. n nodes for n tasks (optimum)

3 Methodology

1. Choose AI Application i.e. Smart Driving, Stock Price Prediction, etc.
2. Develop and run basic inference pipeline on a monolith architecture.
3. Convert the pipeline to run using RPC (remote procedure call) on one server.
4. Deploy the above on multiple servers.
5. Devise a method to programmatically instantiate cloud resources and deploy model.
6. Compare latency and throughput of model on distributed systems of various specifications.

4 Schedule & Milestones

2019	
October	Choose & Design AI application to work on Develop a basic inference pipeline on a monolithic architecture
November	Convert pipeline to work on a distributed system using RPC
December	Manually deploy pipeline to distributed architecture on cloud
2020	
January	Programmatically instantiate cloud resources and deploy model First Presentation Detailed interim report
February	Vary deployments by cloud resources and measure latency/throughput on each
March	Enhance AI inference pipeline by adding more steps
April	Finalize implementation Final Presentation Final Report

References

[1] Ion Androutsopoulos et al. "Learning to Filter Spam E-Mail: A Comparison of a Naive Bayesian and a Memory-Based Approach". In: *CoRR* cs.CL/0009009 (2000). URL: http://arxiv.org/abs/cs.CL/0009009.

[2] George Lekakos and Petros Caravelas. "A hybrid approach for movie recommendation". In: *Multimedia tools and applications* 36.1-2 (2008), pp. 55–70.

[3] John McCarthy. *What Is Artificial Intelligence?* Tech. rep. Stanford University, 2007.

[4] Thomas Mitchell. *Machine Learning (McGraw-Hill Series in Computer Science)*. McGraw-Hill Education, 1997.

[5] David A. Teich and Paul R. Teich. *PLASTER: A Framework for Deep Learning Performance*. Tech. rep. TIRIAS Research, 2018.