GridAI: Cloud-Based Machine/Deep Learning for Power Grid Data Analytics

Design Document

sdmay21-23
Client/Advisor: Dr. Gelli Ravikumar

Team Members:
Karthik Prakash
Abir Mojumder
Abhilash Tripathy
Justin Merkel
Patrick Wenzel
Zhi Wang

sdmay21-23@iastate.edu
sdmay21-23.sd.ece.iastate.edu

Revised: 09/27/2020 V1

Executive Summary

Development Standards & Practices Used

- Agile work methodology
- Standard Software Development Lifecycle
- Code should be fully tested
- Push code to Git repository often
- Utilizing VM testbeds set up for us
- Using meaningful naming conventions.

Summary of Requirements

- Develop machine/deep learning algorithms to apply to power grid data
- Create a frontend dashboard to display the power grid data after having those algorithm(s) applied to it
- Test and validate the application with power grid simulators like OPAL-RT.
- The project will run on the Google Cloud Platform
- The system will allow for real-time power grid data as input
- The frontend will be implemented as a web application

Applicable Courses from Iowa State University Curriculum

- COM S 228
- COM S 309
- COM S 311
- COM S/S E 319

New Skills/Knowledge acquired that was not taught in courses

List all new skills/knowledge that your team acquired which was not part of your Iowa State curriculum in order to complete this project.

Table of Contents

List of figures/tables/symbols/definitions (This should be the similar to the project plan)

1 Introduction

1.1 Acknowledgement

Special thanks to Dr. Gelli Ravikumar for providing us with the opportunity to work with him on this project, providing us with resources and ideas to complete this project, and for always being available to answer questions when needed.

We would also like to thank the research students for providing us the PowerCyber Testbed so that we will be able to test our application.

1.2 Problem and Project Statement

General Problem Statement

Power Grids serve several homes and businesses and over time, the functioning of a power grid can have anomalies and instabilities. Operators in the power grids must monitor large amounts of data to detect these anomalies. The issue that our team aims to address is the execution of efficient and accurate analyses of real-time power grid data.

General Solution Approach

To resolve the issue, our team will develop a Machine Learning algorithm that can learn to analyze the power grid data and display these analytics in the form of graphs and plots. The Machine Learning algorithm will be deployed on the Google Cloud Platform so that it can run continuously and provide the visuals to the Dashboard that a power grid operator can access. The goal of our application is to predict anomalies within power grid nodes at different locations to enable prompt response and to minimize potential outages.

1.3 Operational Environment

The client side operation for this project will be run in a docker container so that all of the libraries and dependencies are included. The machine learning and backend side of the project will be run on the GCP (Google Cloud Platform) which will handle the processing of power grid data and servicing the client docker containers. There will be no extreme or hazardous conditions that will affect this project.

1.4 Requirements

- User Interface to view analyzed data
    - The UI should be able to parse json data from the backend.

    - UI should be able to display meaningful insights about the power grid data in forms of graphs and statistical data.
    - UI should be in the form of a web application
    - The UI should show real-time data using a moving window
- The power grid data will be streamed into the backend through json format.
- The backend server will use a trained ML model to detect anomalies in the power grid data.
- The ML model will be created using tensor flow and trained on the OPAL-RT power grid simulator.
- The backend server will send information to the frontend about potential anomalies and metrics through json
- The backend server will maintain a database of previous data in an ORM.
- The backend will be migrated from the cyber-testbed VMs to the Google Cloud Platform

1.5 Intended Users and Uses

The intended audience for this project is power grid managers/operators who will need to know real time analytics for the power grid. The application will provide them with visualizations of the power grid nodes and alerts if anomalies are present.

1.6 Assumptions and Limitations

Assumptions:

- Clean power grid data will be supplied to us.
- Machine/deep learning algorithms will work as expected.
- Frontend will provide a clean interface to display data from the backend.

Limitations:

- Google Cloud Platform only allows us $300 worth of trial credits per account.
- Currently, the team has a very limited knowledge on machine/deep learning algorithms.
- Testing will not be performed with real power grid data, or power grid operators, therefore the validity of testing is only from the power grid simulators (OPAL-RT).

1.7 Expected End Product and Deliverables

- A trained Machine Learning algorithm for analyzing power grid data will be implemented (Deadline: April 2021)
    - The algorithm will be created using Python and Google's Tensorflow machine learning libraries. This allows for relatively easy development of ML or DL algorithms. The algorithm will first be trained using simulated data.
- The application will be connected to a live stream of power grid data (Deadline: February 2021)
    - One of the expected requirements is to provide the user a live stream of analyzed data. Therefore it is necessary to receive a livestream of power grid data for the ML algorithm to analyze.
- A web server created with Python to handle HTTP requests (Deadline: March 2021)
    - The backend of this project will be developed using the Flask framework. This component will handle all communication with the client-side of our application.

It will also be connected to the database through which we can feed the live power grid data.

- The frontend will be implemented as a web application (Deadline: March 2021)
    - Our team has decided to design the frontend in Javascript using ReactJS libraries. React is a rich library for creating user interfaces. The application will then be built into a Docker container that we can deploy to our cloud platform.
- The frontend will provide some type of geographical visualization of the power grid (Deadline: April 2021)
    - The user will be able to view the various nodes generating power in the grid. Each node will provide analytics during its operation.

2 Project Plan

2.1 Task Decomposition

In order to solve the problem at hand, it helps to decompose it into multiple tasks and subtasks and to understand interdependence among tasks.

Frontend

- Set up ReactJS template for the Dashboard
- Dockerize ReactJS for PowerCyber testbed
- Migrate from PowerCyber testbed to GCP
- Design the UI for dashboard
- Start building the dashboard modules
- Test communication with backend to display data
- Prototype a working version of the dashboard

Backend

- Web server and Database
    - Deploy web server
    - Setup database config
    - Complete URL mapping to database
- Machine Learning Development
    - Setup Tensorflow
    - Integrate test power grid data
    - Configure ML real-time data
    - Rough training of ML model (Narrow algorithms)
    - Finalize ML model (Testing accuracy and functionality)

2.2 Risks And Risk Management/Mitigation

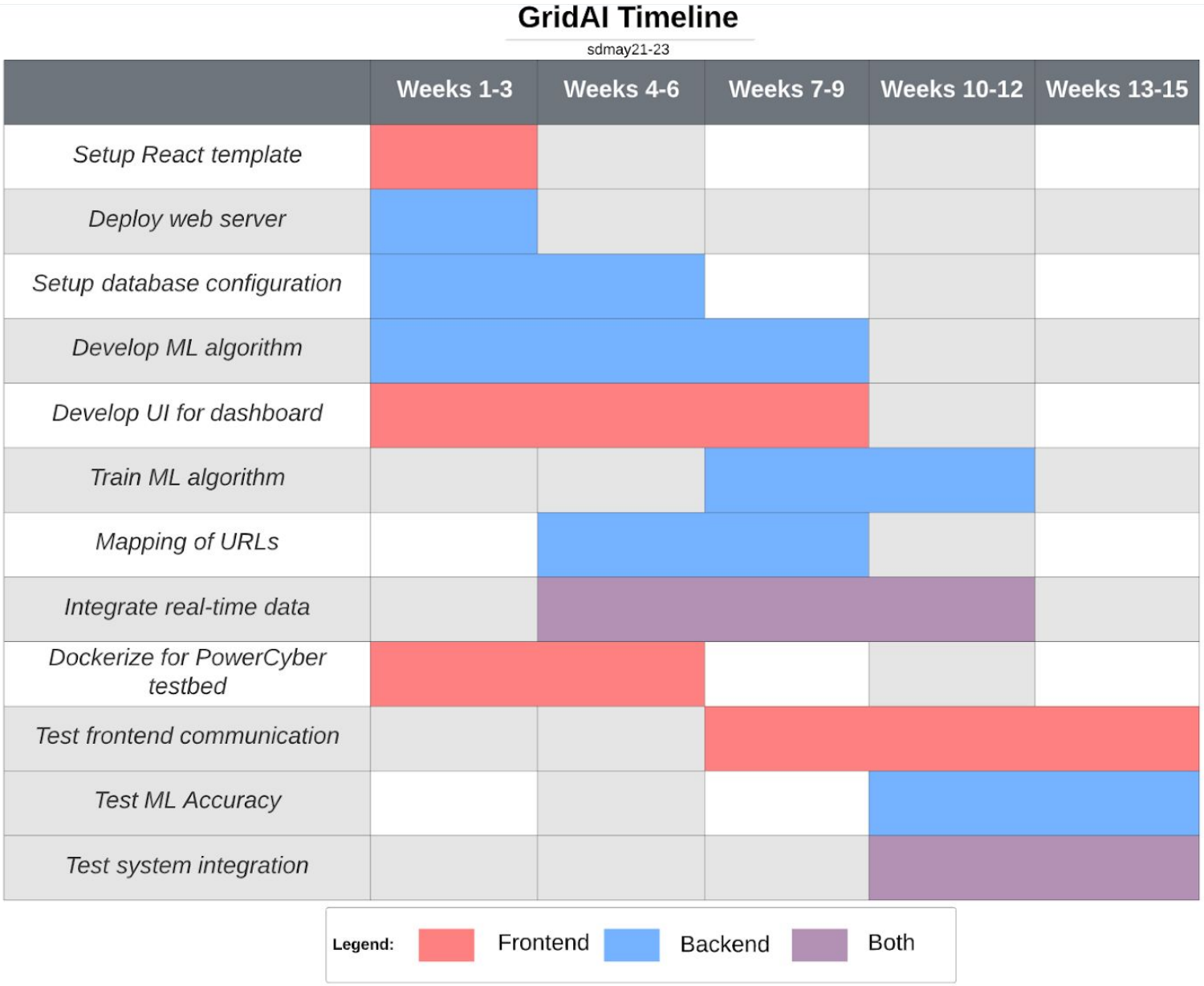
- The accuracy of the machine learning algorithm to detect anomalies will only be trained on OPAL-RT
    - This means that the machine learning model will only have simulated data to predict anomalies in real-world data
- There is a spending requirement of 300$ for using the Google Cloud Platform for the backend

    - This means that the migration will occur later into the project timeline so if there are any problems with the migration there will not be a lot of time to diagnose.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

- The ML algorithm should be able to classify anomalies with 85% accuracy.
- The lag between the real-time data and analysis shown on the frontend should be less than 3 seconds.

2.4 Project Timeline/Schedule



2.5 Project Tracking Procedures

Our team plans to use Git issues to track specific tasks that will be assigned to team members. Git's functionality allows our team to see what tasks are actively being worked on as well as what still needs to be done. Also, members will be able to review the work of other members when an issue is

completed. Additionally, our team is using a Discord server for all internal text and voice communication.

2.6 Personnel Effort Requirements

Task	Description	Person-hours required
React template setup	The frontend dashboard uses ReactJS.	4-5
Deploy web server in Google Cloud Platform	Primary site for frontend and backend/ML will be setup via GCP	6-10
Dockerize ReactJS for PowerCyber testbed	ReactJS will be dockerized for use in PowerCyber and later in GCP server.	2-3
Setup database configuration	Setting up tables for storing input and output data for the ML algorithm	6-8
Design UI	Based on client requirements, a suitable UI needs to be designed	10+
Setup Tensorflow as the primary ML framework	Tensorflow initial setup to accommodate input/output data type(JSON) for training models.	8-12
Connect database repository to the backend framework	Sending and retrieving data from the ML framework will be done through a backend software. This software will communicate with the database	3-5
Building UI modules	After initial design of the UI, the components will be put into development by the frontend team	10+
Integrate power grid data	Configure Tensorflow to take JSON data input	3-5
Connect backend with the frontend	Setting up communication link between backend and frontend.	2-3

Training of ML model	After initial experimentation of the framework, training algorithms will be used on the provided data to detect anomaly and other analyses	30+
Configure frontend and backend for real-time data	The dashboard should show real-time data analyses for this project. Configuration of ML/backend with frontend to transmit data in real-time	6-8
Testing Dashboard (frontend) prototype	The prototype version of Dashboard UI should meet major client requirements	5
Testing/Finalize ML model	Assuming the ML has trained its model, test the accuracy and functionality of its anomaly detection capability	8

2.7 Other Resource Requirements

Since this project is completely built through software, there are no additional resources required such as materials or parts. The project will be using Google Cloud Platform for backend and frontend. Resources will be provided by the client. We will be using open source software including Tensorflow, Flask, Docker, and MySQL.

2.8 Financial Requirements

Our one financial requirement is to ideally stay within $300 worth of Google Cloud Platform credits. Other than that, we will be utilizing open-source tools, so our team should not incur any monetary costs.

3 Design

3.1 Previous Work And Literature

Include relevant background/literature review for the project

– If similar products exist in the market, describe what has already been done

– If you are following previous work, cite that and discuss the **advantages/shortcomings**

– Note that while you are not expected to "compete" with other existing products / research groups, you should be able to differentiate your project from what is available

Detail any similar products or research done on this topic previously. Please cite your sources and include them in your references. All figures must be captioned and referenced in your text.

3.2 Design Thinking

Detail any design thinking driven design "define" aspects that shape your design. Enumerate some of the other design choices that came up in your design thinking "ideate" phase.

3.3 Proposed Design

Include any/all possible methods of approach to solving the problem:

- Discuss what you have done so far – what have you tried/implemented/tested?
- Some discussion of how this design satisfies the **functional and non-functional requirements** of the project.
- If any **standards** are relevant to your project (e.g. IEEE standards, NIST standards) discuss the applicability of those standards here
- This design description should be in **sufficient detail** that another team of engineers can look through it and implement it.

3.4 Technology Considerations

Highlight the strengths, weakness, and trade-offs made in technology available.

Discuss possible solutions and design alternatives

3.5 Design Analysis

- Did your proposed design from 3.3 work? Why or why not?
- What are your observations, thoughts, and ideas to modify or iterate over the design?

3.6 Development Process

Discuss what development process you are following with a rationale for it – Waterfall, TDD, Agile. Note that this is not necessarily only for software projects. Development processes are applicable for all design projects.

3.7 Design Plan

Describe a design plan with respect to use-cases within the context of requirements, modules in your design (dependency/concurrency of modules through a module diagram, interfaces, architectural overview), module constraints tied to requirements.

4 Testing

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, or software.

1. Define the needed types of tests (unit testing for modules, integrity testing for interfaces, user-study or acceptance testing for functional and non-functional requirements).
2. Define/identify the individual items/units and interfaces to be tested.
3. Define, design, and develop the actual test cases.
4. Determine the anticipated test results for each test case
5. Perform the actual tests.
6. Evaluate the actual test results.
7. Make the necessary changes to the product being tested
8. Perform any necessary retesting
9. Document the entire testing process and its results

Include Functional and Non-Functional Testing, Modeling and Simulations, challenges you have determined.

4.1 Unit Testing

– Discuss any hardware/software units being tested in isolation

4.2 Interface Testing

– Discuss how the composition of two or more units (interfaces) are to be tested. Enumerate all the relevant interfaces in your design.

4.3 Acceptance Testing

How will you demonstrate that the design requirements, both functional and non-functional are being met? How would you involve your client in the acceptance testing?

4.4 Results

– List and explain any and all results obtained so far during the testing phase

- Include failures and successes
- Explain what you learned and how you are planning to change the design iteratively as you progress with your project
- If you are including figures, please include captions and cite it in the text

5 Implementation

Describe any (preliminary) implementation plan for the next semester for your proposed design in 3.3.

6 Closing Material

6.1 Conclusion

Summarize the work you have done so far. Briefly re-iterate your goals. Then, re-iterate the best plan of action (or solution) to achieving your goals and indicate why this surpasses all other possible solutions tested.

6.2 References

List technical references and related work / market survey references. Do professional citation style (ex. IEEE).

6.3 Appendices

Any additional information that would be helpful to the evaluation of your design document.

If you have any large graphs, tables, or similar data that does not directly pertain to the problem but helps support it, include it here. This would also be a good area to include hardware/software manuals used. May include CAD files, circuit schematics, layout etc,. PCB testing issues etc., Software bugs etc.