GridAI

FINAL PROJECT REPORT

sdmay23-38
Gelli Ravikumar - Advisor
Josh Clinton - Frontend Lead
Tanay Parikh - Data Lead
Ryan Herren - Machine Learning Lead
Elvis Kimara - Voice Assistant Lead
Rolf Anderson - API, DevOps & Architecture Lead
sdmay23-38@iastate.edu
sdmay23-38.sd.ece.iastate.edu

Revised: April 29, 2023

Executive Summary

Development Standards & Practices Used

Practices Used

- Agile methodology
- Git Version Control
- GitLab CI/CD
- Remote VM testbeds
- Cloud integration

Engineering Standards

- IEEE Data Standards
- HTTP
- IEEE P2840™ - Standard for Responsible AI Licensing
- IEEE P2841™ - Framework and Process for Deep Learning Evaluation
- ISO/IEC 21778:2017 - JSON
- React Architectural Standards

Summary of Requirements

1 Functional Requirements

1.a Program must run locally and be deployable.

1.b Display all necessary power grid data in an interactive grid, such as the amount of power each node uses and how each node is connected.

1.c Capable of running all parts of the program from our speech recognition software.

1.d Utilize machine learning to predict and display future power grid anomalies.

1.e Highlight specific problem areas to help field technicians, such as current power outages.

1.f Allow for both broad and specific scaling of the displayed grid so that users can look at the power grid at both a micro and macro level.

2 Aesthetic Requirements

2.a Display all the grid information on a map clearly and concisely so that the data is easy to read and understand.

2.b Display all the grid information on different types of maps, such as terrain maps, overhead maps, and satellite maps.

3 Security Requirements

3.a Limits accessibility to only those authorized, such as the local government and authorized employees.

3.b Securely receive data without leaks for displaying data about future anomalies, current issues, and current power levels.

Applicable Courses from Iowa State University Curriculum

COM S 227: Object-oriented Programming

COM S 228: Intro to Data Structures

COM S 309: Software Development Practices

COM S 363: Intro to Database Management Systems

COM S 409: Software Requirements Engineering

COM S 572: Principles of Artificial Intelligence

COM S 574: Intro to Machine Learning

DS 201: Introduction to Data Science

DS 201: Data Acquisition and Exploratory Analysis

DS 301: Applied Data Modeling and Predictive Analytics

DS 303: Concepts and Applications of Machine Learning

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 to complete this project.

- Machine Learning
- Reinforcement Learning
- Data Pipelines
- Python development practices + environments
- Component-Based JavaScript
- Frontend Integration

Table of Contents

1 Team

1.1 Team Members

Rolf Anderson

Josh Clinton

Ryan Herren

Tanay Parikh

Elvis Kamara

1.2 Required Skill Sets for Your Project

Machine Learning/Statistics

Advanced Programming Techniques

Database Administration

Cloud Computing

1.3 Skill Sets covered by the Team

Machine Learning/Statistics - Ryan, Elvis

Cloud Computing/DevOps - Ryan, Tanay

Front End Development - Josh, Rolf, Elvis

Back End Development - Josh, Rolf, Tanay

1.4 Project Management Style Adopted by the team

Agile Scrum. Weekly scrum meetings during development cycles (sprints) to make progress iteratively. Sprints are three weeks long, using the first two weeks for development and the final week for testing and verification. A sprint retrospective will happen at the first meeting of each new sprint, along with a sprint planning meeting for the new sprint. Meetings happen twice a week, on Mondays and Wednesdays, to provide time for updates on progress and to allow for feedback to be given throughout the group.

1.5 Initial Project Management Roles

Product Owner - Ravikumar Gelli

Scrum Master - Ryan

Development Team - Rolf, Josh, Tanay, Elvis

2 Introduction

2.1 Problem Statement

The power grid is growing more complex and adding more infrastructure each day. With this growth comes a drastic need for power companies to increase monitoring and gain insight into the health of the grid. To improve our insights and allow for predictive analytics to keep the power grid functioning, we will build a platform that intakes and analyzes power grid data to provide insights into failures to the responsible maintenance teams. Additionally, to prevent further outages, we will build machine learning applications to evaluate current power grid conditions and predict when potential outages or anomalies may occur to decrease response time for responders and keep the grid functioning to its fullest capacity.

2.2 Intended Users and Uses

Energy Companies

The primary intended users for our project are energy companies. The energy sector relies on robust operations and uninterrupted service to customers. As our project provides visibility and analytics of power grids, their specific operators are our most important clients. As grid complexity increases yearly, it is increasingly important for these companies to utilize every tool available to proactively address emerging threats.

i. Headquarters

The power companies will need to monitor the grid on a large scale to make important decisions and fix issues before they happen.

ii. Repair and Maintenance Technicians

Our project will benefit field technicians by giving them visibility into specific issues and confirmation when corrected.

Government Organizations

Local, regional, and national governments have a vested interest in maintaining power grid uptime. Issues and resulting downtime can be major headaches for governments at every level. While not directly responsible for power grids and their upkeep, recent history has shown that incompetent power companies can fail to responsibly manage their domain. Thus, it would be prudent to provide governments with the tools to keep energy companies accountable.

*Note: Due to the sensitive nature of the data we collect and display, we limit who can access our product. Only the power companies, and potentially the government with jurisdiction, will have access. No other users are allowed.

2.3 Requirements & Constraints

List all requirements for your project. Separate your requirements by type, including functional requirements (specification), resource requirements, physical requirements, aesthetic requirements, experiential user requirements, economic/market requirements, environmental requirements, UI requirements, and any others relevant to your project. When a requirement is also a quantitative constraint, separate it into a list of constraints or annotate it at the end of the requirement as "(constraint)." Ensure your requirements are realistic, specific, reflective, in support of user needs, and comprehensive.

2.4 Engineering Standards

What Engineering standards are likely to apply to your project? Some standards might be built into your requirements (Use 802.11 ac wifi standard) and many others might fall out of design. For each standard listed, also provide a brief justification.

- IEEE Data Standards
    - Our project and included services will use various IEEE data standards. As it is a project centered on data storage and analysis, it is imperative that each piece of data uses the most relevant formatting standard.
    - i. int
    - ii. string
    - iii. float64
- HTTP
    - HTTP requests are integral to data transfer in our project. Our APIs utilize HTTP methods such as GET and POST to transfer data between services and from external sources.
- IEEE P2840™ - Standard for Responsible AI Licensing
    - Our project and included services will use various IEEE data for responsible AI licensing Standardized definitions for referring to components, features, and other elements of AI software, source code, and services.
- IEEE P2841™ - Framework and Process for Deep Learning Evaluation

    - This document defines best practices for developing and implementing deep learning algorithms and defines a framework and criteria for evaluating algorithm reliability and quality of the resulting software systems.

- ISO/IEC 21778:2017 - JSON
    - When transferring data between services and from external sources, our project heavily relies on data marshaling in JSON. This almost universal industry standard facilitates painless integration with external APIs and data sources.
- React Architectural Standards
    - Only include one React component per file, favor functionless components, do not use mixins, no unneeded comments.

3 Project Plan

3.1 Project Management/Tracking Procedures

We used Agile for development. We progressed weekly on our already determined plan for the year and presented it to our client every Monday. The decomposition of tasks allowed for incremental development with easy integration to existing code when features were complete.

We used Gitlab to track our progress and assign tasks to our team members. All of our tasks were organized on a kanban-style board. Discord was the primary method used to communicate with teammates and conduct online team meetings.

3.2 Task Decomposition

Epics:

General:

- ➔ Explore the project codebase from last year.
- ➔ Set up development environments.
    - ◆ Set up 5 VMs for developers.
    - ◆ Run project services locally.

Data:

- ➔ Integration of .dss files into the data upload system.
- ➔ Simulate .dss files with python dss API
- ➔ Configure Neo4J and Influx databases
- ➔ Add relationship between all the node data in Neo4j
- ➔ Configure data pipelines for ML.
- ➔ Update databases to reflect ML changes.

Machine Learning:

- ➔ Verify sources.
- ➔ Build data acquisition and ingestion pipelines.
- ➔ Clean data.
- ➔ Create a model.
  - ◆ Train model.
  - ◆ Re-evaluate the model as needed.

Frontend:

- ➔ Implement Machine Learning Algorithms and Voice Assistance
  - ◆ Communicate with backend
  - ◆ Design and Implement UI
- ➔ Redesign UI System
  - ◆ Design new look
  - ◆ Implement new look
  - ◆ Add graph layers for different data
  - ◆ Add hover and click functionalities
  - ◆ Create a line chart on a node click
- ➔ Task: Implement Filesystem
  - ◆ Design the page

- Link with UI
- Push Data to Backend
- Retrieve Data from Backend

Voice Assistant

- ➔ Deploy and analyze code.
  - Remove deprecated methods.
- ➔ Implement new voice commands.
- ➔ Manage dependencies.
  - Update or remove deprecated dependencies.
  - Review dependencies for security flaws.
- ➔ Integrate with frontend.

CI/CD:

- ➔ Remove all hard coded links in code, and replace them with Gitlab CI/CD variables.
  - Replace hard coded links in the frontend.
- ➔ Update pipelines to deploy to reflect new environments.

Testing:

- ➔ Achieve full testing code coverage.
  - Full voice asst. code coverage.
    - Write voice asst. unit tests.
  - Full data platform code coverage.
    - Write data platform unit tests.
  - Full frontend code coverage.
    - Write frontend unit tests.
  - Full machine learning code coverage.
    - Write machine learning unit tests.

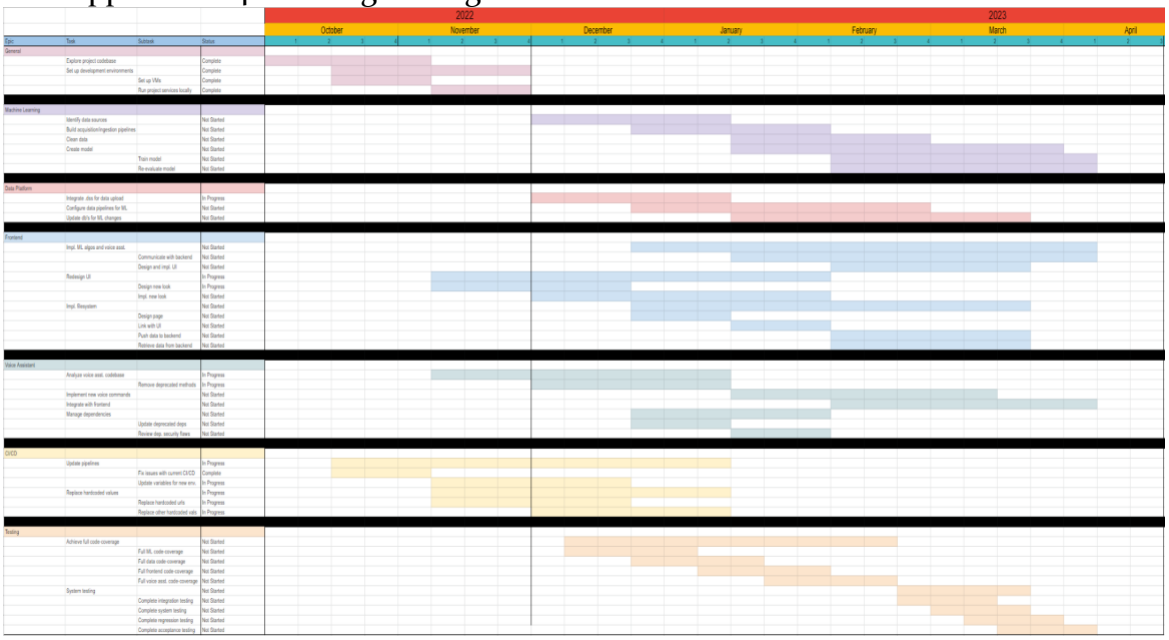
- ➔ Complete integration testing.
- ➔ Complete system testing.
- ➔ Complete regression testing.
- ➔ Complete acceptance testing.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

TIMEFRAME	MILESTONE	Metrics	Evaluation Criteria
10/22 to 12/22	Explore Project Codebase	N/A	Team members understand the codebase.
12/22 to 03/23	Voice Asst. MVP	Voice asst. coverage of available functions.	100% coverage.
10/22 to 01/23	Data platform completed	DSS Files	Stores and processes all values in DSS files.
10/22 to 03/23	Enhance Front End	Relevant QGIS Features	50% relative parity.
11/22 to 04/23	Complete and Integrate ML	Anomaly and Prediction	Able to detect 75% of anomalies and predictions are at least 80% accurate.

3.4 Project Timeline/Schedule

See Appendix 8.4.1 for larger image



3.5 Other Resource Requirements

Due to our project being completely software-based, we only require a few resources. To host and deploy our project, we will need a Google Cloud Platform (GCP) account and credits. We will also require Iowa State's self-hosted version control system, Gitlab. For development and testing purposes, we will also require example data to simulate what contexts our system would experience in production.

4 Design

4.1 Design Context

4.1.1 Broader Context

We are designing our product for use by local governments in combination with utility/energy companies.

The community using our product is limited to people with access to power grid information who are responsible for performing maintenance on the power grid to prevent/restore outages.

The communities affected by our design are endless. Nearly everyone in the United States relies on electricity provided to them by the power grid, so any impact our product can make on ensuring that the power grid stays healthy will impact millions of people.

Our project addresses the need for more oversight and visibility into the health of the power grid, which is nearly nonexistent right now. The more data you have to see how the grid is doing, the better you can address the issues and prevent them in the future.

The list below outlines relevant considerations related to our project in the following areas:

Area	Description	Examples
Public health, safety, and welfare	There are innumerable public safety benefits to ensuring normal power grid operation. While our project would not solve problems directly, early detection of developing issues in the grid could be the difference between a normal winter storm and the infamous 2021 Texas power crisis.	Hospitals and the patients within depend on the many systems functioning correctly. While they have on-premises generators, these are only stop-gap solutions to short-term outages. Also, summer air conditioning units are critical for public safety, especially for the elderly. Heat waves can be deadly, and badly-timed grid disruptions in the past have shown this.
Global, cultural, and social	Our project reflects the values and practices of the cultural groups very well. It's not limited to nations, workplaces, etc., because it touches everything. Our project would predict power outages and stop them from happening, thus giving societies the freedom to carry on with their existing norms and practices.	During the Chinese New Year celebrations, our project will help ensure abrupt power outages shan't occur, thus allowing for this cultural tradition to go on.

Environmental	This project will decrease energy usage from nonrenewable sources. Catching anomalies early can reduce the need for fossil fuels to fill energy gaps caused by them.	For example, power disruptions may lead to businesses and organizations activating inefficient diesel power generators hosted on-premises. Disruptions may also affect the ability of private solar power generators to provide excess power to the grid, negating any positive effects of these systems.
Economic	This project will save energy companies and local governments money by preventing power outages and other grid issues and increasing the speed at which issues can be resolved. Our project could cause energy companies to downsize if it is successful enough Because they will need fewer technicians.	For example, when the power grid goes down, local coffee shops will not be able to process their credit card transactions. This is just a small example, but it shows how many places are impacted economically by outages of the power grid.

4.1.2 Prior Work/Solutions

OpenDSS is currently a software used to provide insights into the power grid, but it is only available as a desktop application and is a very bulky program. It allows you to see the grid based on existing datasets, toggle through different types of electricity(single phase vs. three phase), and filter by types of users.

The previous work is a great start, but it's hard to access and inefficient. It is only available as a desktop application, so it can't be accessed remotely via any web browser, which is a goal of our project. A website that implements that kind of functionality would be a great asset to all utility companies and local governments. Second, since it's not web-based, it requires a great amount of processing power and local machine usage to run. A cloud-based web application takes all of the load off of the user and puts it on the backend, making it easier to use for the end user.

There's work being done by Camus Energy, along with the Pacific Northwest National Laboratory and Kit Carson Electric Cooperative, that are building a machine learning

model that will fix gaps in its grid data. They have received over $750,000 in funds from the US Department of Energy (Camus, Feb. 14, 2022).

PROS

- Our project is sponsored by Iowa State University, and we have a powerful server (one of the best in the US) meant just for us. We also have great resources like Virtual Machines with 4TB space, GCP, an experienced client/professor, and many others.
- We have a solid team of senior software engineering students compared to other projects that might just have one programmer who only works in his/her free time
- We are building on a two-year-old working repository and thus not starting from scratch.

CONS

- We are students with limited expertise in this field, so other teams without such a problem are way ahead.
- Our motivation is slightly above or below meeting a class requirement. This might not be as high as someone motivated to do this to get a job, start a business, or get income for his family. Thus, they have more reason to put in more effort than we do.
- We have a short timeframe to work on this project. As a senior design project, we have a year and after that are finished. This is a con, as other (non senior design) teams could have many years to complete a project.

4.1.3 Technical Complexity

1. Our design consists of multiple components/subsystems. We use APIs, hosts like firebase, GCP, environments in GitLab, webhooks for the google voice assistant, react framework for the frontend and neural networks for machine learning, and neo4j for the influx database.
2. Our project contains many challenging requirements, including creating our own machine learning models. We will use neural networks instead of the old linear learning machine learning models. Another challenging requirement of our project will be using GEOMap to display different layers in the map using the nodeID from the database.

4.2 Design Exploration

4.2.1 Design Decisions

1. Neural network RL model via A2C algorithm provided by Stable-baselines3, an open-source library of reinforcement learning algorithm implementations. This choice gives us the performance and accuracy necessary for our anomaly detection and the ability to use different actions in response to anomaly detection in order to create a relevant and working Reinforcement Learning algorithm.

2. Voice assistant. This feature will enable quicker and more natural interaction with our project for our users. With the ability to interact with your phone with your voice, it allows utility workers to check on the status of the grid or specific substations hands-free.

3. Use GEOMap to display node data to the frontend. This allows us to display data in different layers on a grid. This will allow users to filter unnecessary data.

4.3 Design

4.3.1 Overview

We have three major components in the current design: the backend, the frontend, and the database. The database is where all the data will be stored, and it has its own request handler to handle all the requests. The backend is where all the logic of the application is performed. For example, all of the machine learning models and APIs live here, and it also has its own request handler, so it can send and receive requests to and from the frontend and the database. The frontend is where all the visuals are processed. It has its own request handler, so it can request data from the database and display it.

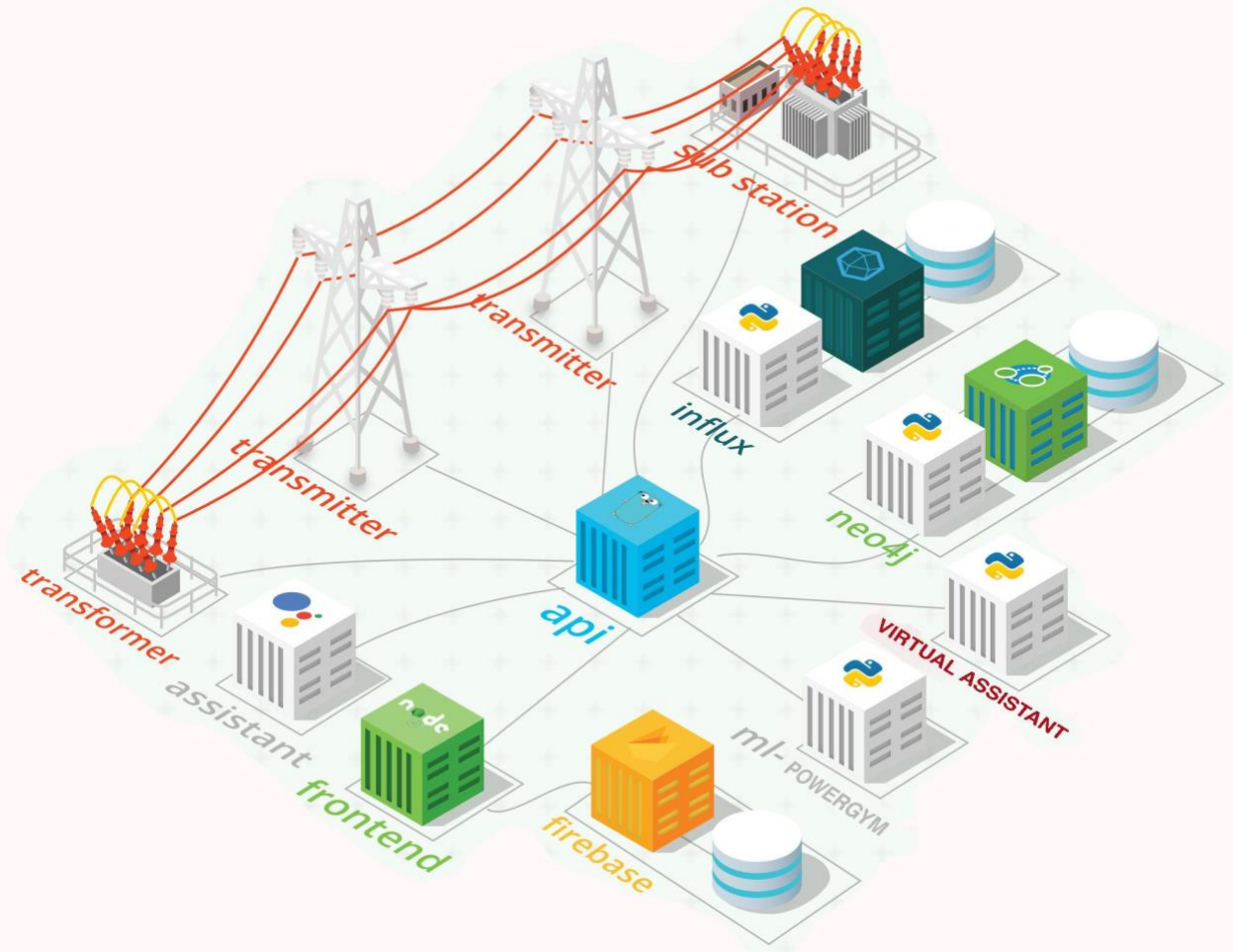


Fig. 1, Design Components Overview

Fig. 2, UI Basic View

4.3.2 Detailed Design - Front End

Our project is a system consisting of a React web app frontend and several backend services connected through an API, as seen in Fig. 1. The frontend allows users (power company personnel, government monitoring agents, etc.) to view the current state of the system, as seen in Fig. 2. Users are able to examine individual nodes which will include their past and predicted future values. Furthermore, the frontend will alert users to current or predicted future anomalies which may indicate dangers to the power grid's health. An additional feature of the frontend is the implementation of a chat assistant, allowing more natural and flexible interaction with the system. The chat assistant is integrated with the web app frontend. The backend will consist of several microservices deployed to Google Cloud Platform (GCP). Influx and Neo4j will be used for backend data storage but currently we only connect to Neo4j and don't have access to live data. The frontend uses GCP Firebase for data storage specific to the frontend, such as profile information. Our system will leverage Tensorflow on Python through two machine learning (ML) services designed to predict future values and detect anomalies respectively but currently, because influx isn't working, we only display static data. The frontend and backend services communicate with each other through a RESTful API implemented in Go (also known as Golang). External data sources, shown as the transmitter nodes in Fig. 1, will push data to the system through this API. This is how data in the system at large is updated. The frontend itself is laid out into three main sections. The components, pages, and libraries. All of the technical code is in the components, the pages create, format, and display the components, while the lib contains the API calls, creates the node layers, and

has other helper methods and hooks. We use DeckGL and MapBox to create the world map and display the grid data.

4.3.3 Detailed Design - API

Our implementation includes a discrete, centralized API that all external and inter-service communication is routed through. It is a simple RESTful API build using the Go programming language. It serves to decouple services from each other, decreasing coupling and making maintenance and evolution easier.

4.3.4 Detailed Design - Data

Our Data Platform consists of two discrete database systems. Neo4j, a graph-based database, stores the static information of power grid components as nodes and edges. Static information includes the geographical location of the components and busses which are stored as nodes and lines which are stored as edges. InfluxDB, a time-series database where we store voltages for buses, current, active power/reactive power for the lines or edges. We also store capacitor, transformer and PV data, like kV, kVar in a time series fashion from the present and past dynamic data of each grid component. We use these databases in combination to allow visualization of the grid, its components and their relationships from Neo4J, as well as storing the temporally dynamic information of the grid. Information is uploaded by the given .dss files and using a python package like DSS-Python to simulate the dss file and using Influx and Neo4j queries to upload all the large amounts of data. The data is accessed by other components of the project like the front end, Machine Learning and Virtual assistant through our separate API service.

4.3.5 Detailed Design - Machine Learning

One of the most exciting aspects of the project is the use of cutting-edge machine learning and artificial intelligence tactics to provide volt-var control for power systems. The main goal of this section is to use live, constantly-updating data from the grid to better learn its behaviors over time and provide automated actions to prevent outages of the grid.

To accomplish this, we implemented a reinforcement learning environment–a kind of machine learning where an intelligent (computer) agent interacts with an environment and learns how to best act to achieve a goal state. This agent is trained on rewards and penalties based on the results of their actions, and can learn how to effectively act in given states to work towards its goal.

Our environment for this project was based on PowerGym [2], a 2018 implementation of a Gymnasium environment built specifically for reinforcement learning for volt-var control.

This allowed us to run training models on pre-built systems within the environment. To pair with the virtual environment, we used algorithms from StableBaselines [6], an open-source set of improved implementations of reinforcement learning algorithms based on the OpenAI Baselines [7]. Our primary algorithm is the Advantage Actor Critic (A2C) algorithm, which utilizes two function approximations (neural networks) that has the actor perform an action (policy) and then get feedback (reward/value) from the critic on its effectiveness. The policy is analyzed by the critic and given a reward based on the effect of the policy to progress towards a goal state. Once the model is trained (policy/reward stage) it can then be used to choose actions on real or simulated environments. This testing stage allows you to evaluate the effectiveness and accuracy of the model- whether it successfully provides voltage regulation or if it doesn't fix fluctuations.

The outcome of the reinforcement learning modules is a significant one- it has the potential to allow for completely autonomous control of increasingly complex power systems. With thousands of different nodes in a system, performing preventative actions is incredibly difficult, but with an automated control system trained on advanced algorithms, a computer can stop outages before they even happen.

Our design for the 'Machine Learning' part of the project changed drastically from the beginning stages of design to where we're at now. The last team left us with little resources and less implementation of 'anomaly detection,' which was used to recognize where outages, surges, or bottlenecks were happening. This was originally marked as a point of improvement, but was left in the backlog for the duration of the project, as most of the effort went into solidifying the data platform before spending large amounts of time implementing a reinforcement learning environment.

Another main point of focus was making sure that the reinforcement learning infrastructure was beneficial to the end-user, which meant that a lot of work needed to go into presenting the data on the front end. The main product of the reinforcement learning module is a time-series chart showing voltage variations over time. In this system, you will be able to see where actions are taking place in order to regulate voltage and the effectiveness of those actions. These charts currently provide information to the operators on courses of action to take when voltages are trending in directions that are too far away from the optimal value. In the future, this system could be completely automated, with the model providing best choice actions to enact on the system to provide further variations.

We ran into many problems while building out the initial foundation for a reinforcement learning system. Getting the environment configured correctly to run independently of the platform proved to be a challenge, primarily due to the old age of the PowerGym

library and its dependence on old versions of Python, which caused build and integration issues with other Python libraries. The combination of using PowerGym, which is based on the broader Gymnasium library, Stable-baselines3, and other Python libraries like Tensorflow and DSS, required a very specific concoction of versions and dependency installations, which took a lot of time and effort to configure correctly before actual reinforcement learning development could begin.

4.3.6 Detailed Design - Virtual Assistant

Our project's Virtual Assistant component leverages SpaCy Natural Language Processing (NLP) to improve accessibility of information. We use a React component similar to a chat box to communicate with a SpaCy based backend service. When a user enters text, it is sent to our assistant service, which uses NLP techniques to process the user's input and return the appropriate response. Some user commands result in database queries to get real time data about the power grid like system health, and current voltage. All these are returned as a json response and rendered by the frontend chatbot that currently hovers on the bottom right of the homepage.

5 Testing

5.1 Unit Testing

We used unit testing extensively in our projects for all of our components, as well as integration testing all the components with the API to see if the system components communicate well with each other. For this, we will use tools like Pytest and jest.

5.2 Interface Testing

There are a lot of components in our project that will require a great amount of testing in regard to the interfaces. First, most of our project runs through an API that takes data from the power grid and integrates it into our environments. We will have to test that interface to ensure that the data is being ingested correctly. Next, once the data is in our environments, we will need to test the interface between the databases and our Google Cloud Platform environment, which is where most of the computation will happen. GCP controls the hosting of all aspects of our project, so it will be important to ensure that it interfaces correctly with the databases. Our methods of interface testing will be workflow and performance tests. We will be testing the workflow to make sure that it is able to ingest live data and support our real-time analytics, which will be part of the functionality of the site. Second, we will have to test the performance of our infrastructure to make sure that it can efficiently integrate high volumes of data.

6 Future Work

6.1 Front End

There are many things that can be enhanced on the frontend to provide better functionality. The biggest thing is to implement a file-system that allows a user to upload and download files into the database from the frontend and store them on their accounts. This improvement will allow for greater functionality and allow users to use this program on a broader scale. An issue they will need to solve is how to set the initial view state based on the position of the data.

Another improvement to be made will be to create a page containing several line charts for data such as consumption and voltage that allows a user to select which nodes they would like to see. This can be an entirely separate page dedicated to a large line graph as opposed to the one on the main page that is quite small. The larger line charts will be able to show more than one node at a time as well. Lastly there are always visual changes that can be made. Over the course of this project I focused much more on the functionality and the logic to make things work than I did about the visuals.

6.2 Data Platform

There remains some work to be done in the database domain like adding a functionality where the data is coming directly from the power grid and the utility companies are getting live predictions of Potential faults from the machine learning section back to the database and up to the frontend . Furthermore having a file management center on the frontend where the users are able to download their data after logging in to their account, where then the frontend would call API to call all the .dss files saved in the database for that user . Also some other functionality where time is also being simulated, Currently when we simulate the master .dss file it only gives the data for that one step. I would have liked to add a functionality where after that step or the time frame is complete, the step would increment to the next step or time frame and upload the simulated data.

6.3 Machine Learning

There remains much work to be done in the Machine Learning domain, but the foundation is set for more efficient progress in the future stages of GridAI. The future of the reinforcement learning module developed throughout this iteration of the project falls into three major objectives: frontend enhancements, live data platform integration, and algorithmic enhancements.

On the current frontend, the full scope of the reinforcement learning work is underrepresented. In the future, it would be great to add sections to see the accuracy of different training algorithms outside of just A2C (see third paragraph) and be able to

toggle different reinforcement learning graphs based on what you're looking to learn. The front-end for the current project is primarily focused on just the analytics and less on the predictive modeling. As the reinforcement learning module continues to be developed, the frontend will have to grow to reflect the greater capabilities.

Secondly, and likely most importantly, the reinforcement learning module still needs to be incorporated into the live data coming from the API. This was a goal from the beginning of our iteration of the project, but was something that was not a priority while working on getting the foundation for reinforcement learning paved. In the current system, all of the training is done on pre-defined environments provided by the PowerGym library, which restricts the amount of different training environments available and doesn't give us the most live, real-life data to train on. In the future, the RL algorithms and environment can be connected to the live data, which then can be trained on and simulated using platforms to provide the most accurate simulated environments for reinforcement learning.

Last, there is a lot of room for improvement in the algorithms and methods being used to perform reinforcement learning to create models for volt-var control. Right now, there is only one working algorithm (A2C), but there are plenty of other options for algorithms that would provide supplemental learning on variation control, like PPO1. Once these algorithms have been further developed, there is also lots of room for improvement in respect to tuning of each individual algorithm's hyperparameters. Right now, there is little hyperparameter tuning outside of general changes to iterations, gammas, activation functions. In the future, working hyperparameter tuning into regular re-trainings of the models would be an effective way to ensure maintenance of the algorithms and increased accuracy over time. We did a little research on implementing a grid search algorithm to tune hyperparameters, but the time crunch left that beyond the scope of this iteration of the project.

6.4 Virtual Assistant

In the future, we'd need to have an automated testing environment for future assistant deployments, analytics, and reporting. Ths would help us track our training process and analytics. We also plan on expanding assistant functionality, scalability, training, and data collection so that it's more robust. Other ideas involve adopting speech to text, more commands, and API calls. Plus more requirements gathering needs to be done in the future iteration of the project so that a well defined long term NLP goal, product, and technology is established.

6.5 Architecture and Platform Operations

For Continuous Integration/Continuous Delivery (CI/CD) and infrastructure deployment, we used Gitlab CI/CD and GCP. We published and deployed Docker containers using

Cloud Run on GCP. While Cloud run is an efficient system, it lacks the flexibility and power needed for more complicated full stack systems like ours. Switching to using the orchestrator Kubernetes with Google Kubernetes Engine (GKE) would benefit GridAI in the long term. Using Cloud Run for production deployments is not common in professional environments, whereas Kubernetes and GKE are extremely common in professional environments. Furthermore, using Terraform to provision and manage infrastructure would stand to give even more positives to GridAI. Infrastructure as Code (IaC) languages like Terraform have become very popular in modern cloud systems. It gives benefits such as repeatability, centralized state storage, and the ability to manage the system from one interface, rather than using the GUI.

7 Closing Material

7.1 Conclusion

While our project has not completely met the requirements set forth, we made important contributions and design changes that successfully furthered GridAI. Our project is a multi-year one, with difficult concepts and implementations. While we wish we could've seen the project to completion, we are satisfied with the results. This project has been at times grueling, but we all learned much, and gained important skills that we will leverage during our careers.

7.2 References

[1] Overcoming Imperfect Grid Data with Machine Learning | Camus Energy. https://www.camus.energy/blog/overcoming-imperfect-grid-data-with-machine-learning. Accessed 21 Oct. 2022.

[2] PowerGym: A Gym-like environment for Volt-Var control in power distribution systems. | Github https://github.com/siemens/powergym. Accessed 21 Nov. 2022.

[3] QGIS: A Free and Open Source Geographic Information System | https://www.qgis.org/en/site/. Accessed 21 Nov. 2022.

[4] Ravikumar Gelli | Research Assistant Professor | Iowa State University Department of Electrical and Computer Engineering.

[5] sdmay22-35: GridAI | Site http://sdmay22-35.sd.ece.iastate.edu. Accessed 21 Nov. 2022.

[6] Stable-Baselines3: Reliable Reinforcement Learning Implementations. | Github https://github.com/DLR-RM/stable-baselines3. Accessed 28 Apr. 2023.

[7] OpenAI Baselines. | Github https://github.com/DLR-RM/openai/baselines. Accessed 28 Apr. 2023.

7.3 Acknowledgments

This research is funded partly by US NSF Grant # CNS 2105269 and US DOE CESER Grant DE-CR000016.