ENS 491 – Graduation Project (Design)

Progress Report I

Sensing As A Service

Group Members:

Ali Özgün Akyüz - 27907

Sermet Özgü - 28479

Supervisor(s):

Kürşat Çağıltay

Utku Günay Acer

Date: 07/01/2024

1. PROJECT SUMMARY

This project aims to create a platform that facilitates the connection between sensors, machine learning algorithms, and developers by providing a user-friendly web application. It focuses on managing and processing data for use in machine learning algorithms in IoT environments. Our motivation is to ensure that data, which is easily accessible, can be seamlessly processed by complex machine learning models, and that developers can effortlessly utilize this data in their projects through APIs. The project centers on creating a full-stack web application, which encompasses backend, frontend, machine learning models, sensors, and Docker containers. After developers select their preferred data providers and machine learning algorithms, they will be able to use the data with the APIs they have installed. The main objective of this project is to create a user-friendly interface that includes compatible data, efficient data processing, and ease of management for sensing projects. This project enables the use of advanced sensing technologies and machine learning algorithms without the need for deep technical knowledge, while considering social, privacy, and data security constraints.

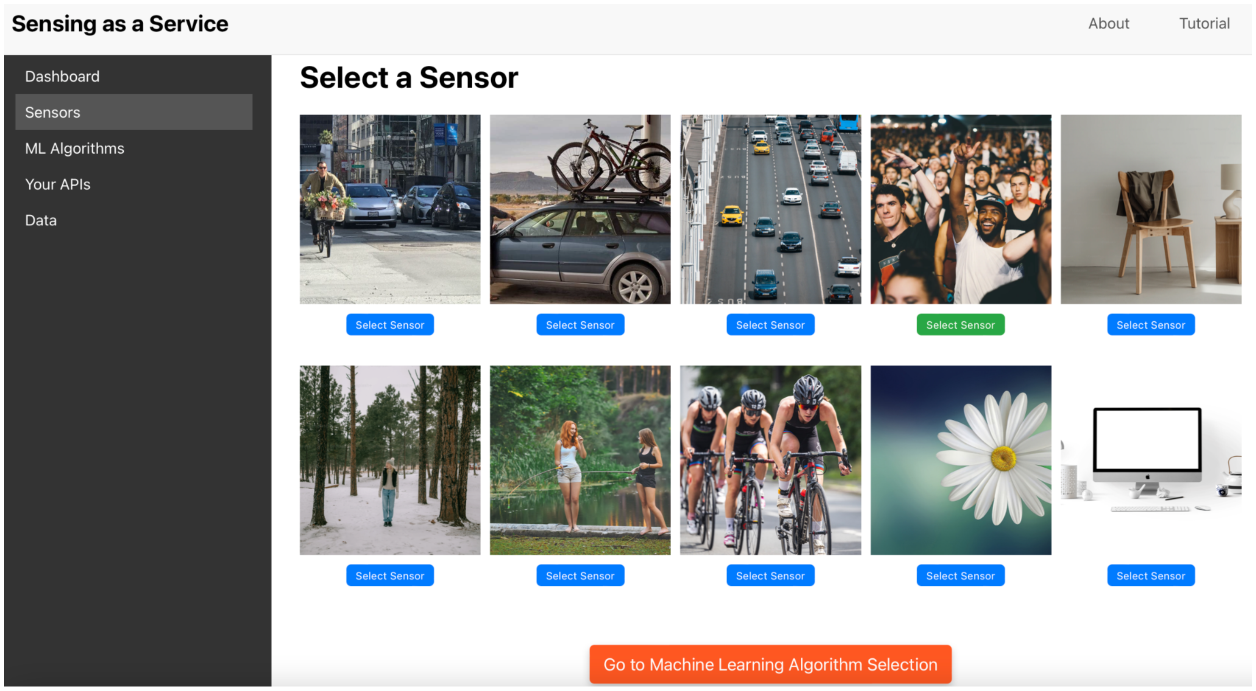
There exist systems that process machine learning algorithms with cloud-based systems and create pipeline systems for these algorithms. Examples of these include KubeFlow (KubeFlow, 2021), Google's TensorFlow(Baylor et al., 2019) and BentoML (BentoML, 2021). In addition to these, SensiX++ also enables machine learning algorithms to run on edge devices(Min et al, 2023). These systems process machine learning algorithms but they do not allow the user to upload the API and access the processed sensor data. With our project, application developers no longer need to concern themselves with model development, service deployment, etc. Our system handles deployment in a cloud environment and their functions are hosted on our serverless platform. They can simply plug-in the model results into their custom functions. These functions are invoked API calls with specific URLs provided to the developers when they register their functions.

In Perera's (2014) research, the role of the Sensing as a Service model in the development of smart cities is thoroughly discussed. It predicts that future smart cities will be embedded with countless sensors, enhancing daily life through the interpretation of data gathered by Machine Learning (ML) algorithms. Our project represents a significant step in this interpretation phase, offering developers a service to easily access and utilize these insights. As previously mentioned, developers can seamlessly integrate model results into their custom functions. We believe our project will significantly contribute to the integration of intelligent systems into everyday life, accelerating the evolution of smart cities.

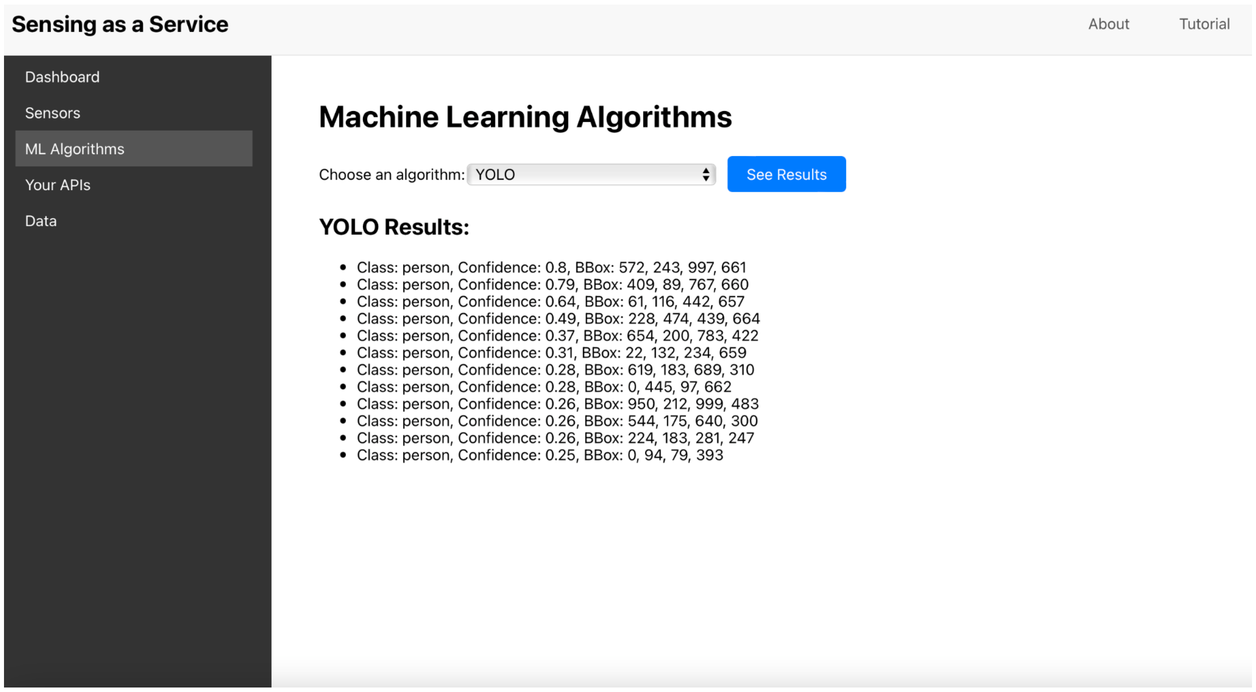
2. TECHNICAL DEVELOPMENTS

Since the last report, our project, "Sensing as a Service" has seen significant technical advancements. We have delved into literature review, focusing on the researching methodologies planned in the proposal draft. This review enriched our understanding of how these methods perform under different conditions and their performance. We've also created the prototype pipeline to bring the conceptual project into tangible form. On the other hand, in terms of commercialization, our patent search revealed no similar existing patents, indicating no Freedom-to-Use (FTU) issues for our project.

As for testing our project, we have designed a series of user test cases. For example, in our user interface, the first step involves selecting the desired sensor, followed by choosing the applicable machine learning algorithms. This process is a crucial step for integrating these choices into a cohesive and dynamic system. The pipeline we have developed, including the YOLOv8 algorithm in a Docker container, serves as the preliminary model for our design. Through this pipeline, we're testing how the chosen sensor data is processed with ML algorithms and delivered back results to the user. The UI, developed using React and NodeJS. This phase of the project allows us to see the practical application of our theoretical designs and make necessary adjustments before the implementation phase (ENS492).



(Fig.1, Sensors Page)



Machine Learning Algorithms

Choose an algorithm: YOLO

YOLO Results:

- Class: person, Confidence: 0.8, BBox: 572, 243, 997, 661
- Class: person, Confidence: 0.79, BBox: 409, 89, 767, 660
- Class: person, Confidence: 0.64, BBox: 61, 116, 442, 657
- Class: person, Confidence: 0.49, BBox: 228, 474, 439, 664
- Class: person, Confidence: 0.37, BBox: 654, 200, 783, 422
- Class: person, Confidence: 0.31, BBox: 22, 132, 234, 659
- Class: person, Confidence: 0.28, BBox: 619, 183, 689, 310
- Class: person, Confidence: 0.28, BBox: 0, 445, 97, 662
- Class: person, Confidence: 0.26, BBox: 950, 212, 999, 483
- Class: person, Confidence: 0.26, BBox: 544, 175, 640, 300
- Class: person, Confidence: 0.26, BBox: 224, 183, 281, 247
- Class: person, Confidence: 0.25, BBox: 0, 94, 79, 393

(Fig.2, ML Algorithms Page)

To test our Machine Learning algorithms and the underlying logic of our pipeline, we have designed a sample user interface (fig.1 & fig. 2) that fulfills only the basic functionalities of our project. These interfaces are not the final version of the project, but provides a preliminary glimpse into the appearance of the project. From the sensors tab in the application(Fig. 1), user selects one or more camera footage then when user clicks the Go to Machine Learning Algorithm Selection button, user is redirected to ML algorithms page(Fig. 2). User selects the algorithm that he wants to use, then he sees the results.

It is important to mention that this current user interface is not the completed version. Our goal is to enhance user experience within our application and ensure error-free pipeline. To this end, we have created a set of sample use cases, elaborated below. These use cases have informed the development of a task flow chart. The article by Hartson (2012) emphasizes the significance of task flow creation as a crucial phase in UX design. He articulates how developing a detailed task flow is instrumental in aligning the design process with user needs, thereby enhancing the overall user experience. Therefore, the flow chart (fig. 4) outlines each step of the application process for various scenarios, serving as a guide for the development of our final UI for the following phases of our project.

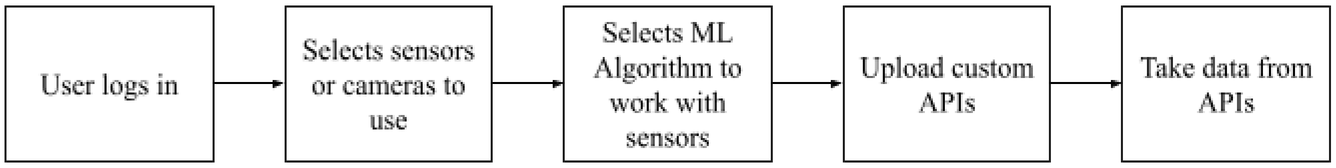
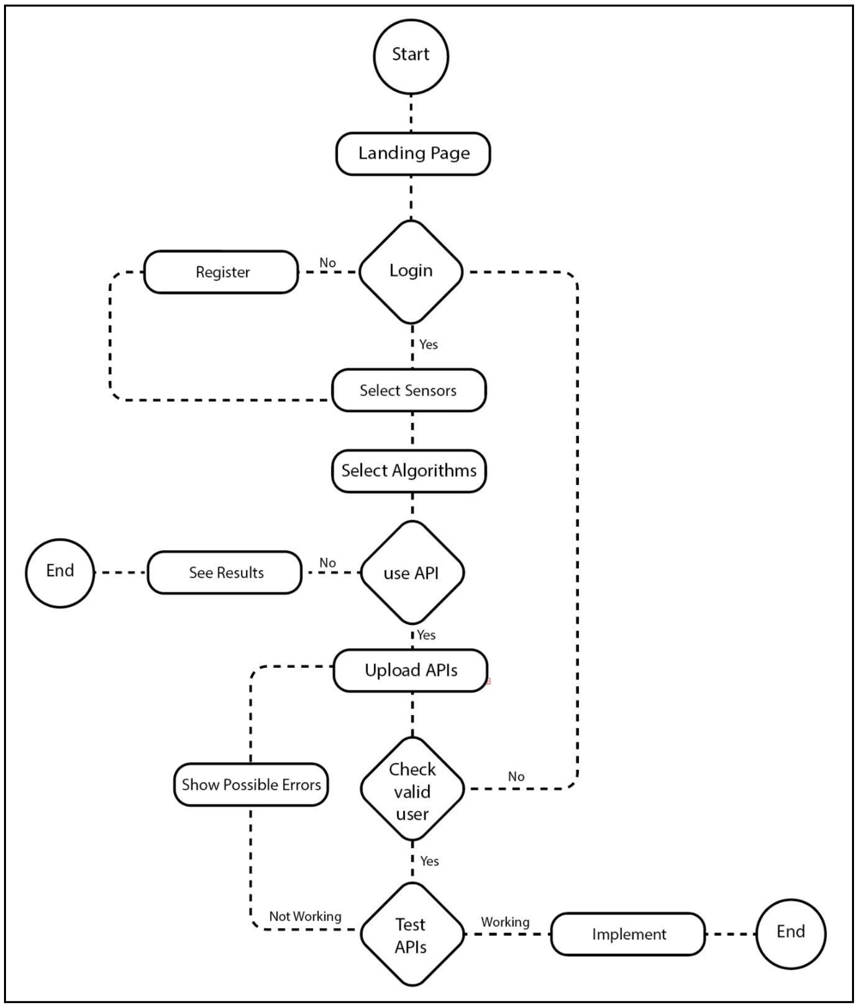

User logs in
Selects sensors or cameras to use
Selects ML Algorithm to work with sensors
Upload custom APIs
Take data from APIs

Fig. 3 Full Pipeline

Sample Use Cases

- **Traffic Management:** A platform which takes data from security cameras into a traffic flow detection system can be developed by developers. With the help of machine learning algorithms, the system can inspect and analyze traffic flow patterns to optimize traffic flow during busy hours. This includes processing large data, using real time data processing and implementing efficient algorithms. Our application performs this for developer to make it easier for them.

- **Availability for cafes:** A system can be designed by developers which use camera feeds of the cafes to analyze the occupancy levels, especially at busy times such as holidays and weekends. Developers can easily take the result of analyzed camera footage with their APIs to use in their system.
- **Parking Lots:** Developers can write an application which tells which parking lot has free space by implementing their APIs to our application.
- **Public Transportation Utilization:** A system can be developed using the cameras and the machine learning algorithms in our application to detect peak hours can be helpful for optimizing transit schedules. While developing this system the developer should not know the usage of machine learning algorithms or IoT device usage. The developers can take the results by implementing his APIs to our application.



(Fig. 4, Task Flows)

3. ENCOUNTERED PROBLEMS

The first problem was learning to use docker. We knew a little bit about using Docker, but we had to learn it in more detail for this project. For this, we first read Docker's own documentation files, tried to develop small projects and watched training courses on YouTube. Our second problem we're facing in this project stems from its design for cloud services. Currently, we don't have direct access to such cloud environments. To handle this problem, we are simulating cloud functionalities on our personal laptops. However, since the docker images took up a lot of space on our computers, we exceeded the data storage capacity of our computers, so we had problems recreating them after the changes. We solved this problem by deleting our old image files and applications that we do not constantly need on our computers. When we create a new image, if this image works smoothly, we delete the old image. Thirdly, we had problems getting live footage. The school did not provide live footage for us to use. We will test using 15-minute videos provided by the school instead of live images.

These problems we experienced did not cause any changes in the steps of the project or caused any problems in the project's timetable. The project is progressing in accordance with the timetable

4. TASKS TO BE COMPLETED UNTIL PROGRESS REPORT II

Until progress report II, we will finish all aspects of the project completely. Key tasks include completing the pipeline structure to take data with user's own APIs and enhancing user interface for efficient usage in the web application. The pipeline will include various machine learning algorithms, different sensors and basic functionalities like login and registration. Additionally, the web application will include some camera feeds to test the accuracy, efficiency and speed of algorithms and connection to given APIs. A database will be created to store user and sensor data.

5. REFERENCES

Baylor, D., Breck, E., Cheng, H. T., Fiedel, N., Foo, C. Y., Haque, Z., ... & Zinkevich, M. (2017, August). Tfx: A tensorflow-based production-scale machine learning platform. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1387-1395).

BentoML. (2021). Retrieved January 7, 2023 from https://www.bentoml.ai

Hartson, R., & Pyla, P. S. (2012). *The UX Book: Process and guidelines for ensuring a quality user experience*. Elsevier.

KubeFlow. (2021). Retrieved January 7, 2024 from https://www.kubeflow.org

Min, C., Mathur, A., Acer, U. G., Montanari, A., & Kawsar, F. (2023). SensiX++: Bringing MLOps and Multi-tenant Model Serving to Sensory Edge Devices. ACM Transactions on Embedded Computing Systems, 22(6), Article 98 (November 2023), 27 pages. https://doi.org/10.1145/3617507&#8203

Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. (2014). Sensing as a service model for smart cities supported by internet of things. *Transactions on emerging telecommunications technologies*, *25*(1), 81-93.