Object Recognition through Machine Learning

A Project Report
Presented to
The Faculty of the Computer Engineering Department

San Jose State University
In Partial Fulfillment
Of the Requirements for the Degree
Bachelor of Science in Software Engineering

by

Mohamed Albgal, Sanford Huynh, Alexander Len, Aaron Warren

02/2020

APPROVED FOR THE COLLEGE OF ENGINEERING

Dr. Wencen Wu, Project Advisor

Professor Rod Fatoohi, Instructor

Dr. Xiao Su, Computer Engineering Department Chair

Abstract

Image Recognition

By Mohamed Albgal, Sanford Huynh, Alexander Len, Aaron Warren

This project enables users to upload their imagery for storage and retrieval. The backend component will then take the uploaded imagery and apply a machine learning model to analyze and report on the content of the images. With further training of our model, the system will be able to then recognize images with a higher degree of accuracy. Extending the capabilities further, we would be able to gain domain level expertise to recognize images related to driving conditions as well as process video imaging. As proof of concept, the interface our project has is web-based. Optimally, the backend would be able to serve recognition capabilities to secure clients with a wider scope of hardware requirements.

A self-driving vehicle must be aware of obstacles or potential hazards in its surroundings in order to safely and effectively navigate its surroundings. Without the implementation of this crucial task with a high degree of accuracy, the vehicle will fail to meet the standard of an autonomous vehicle. Onboard hardware such as sensors and LIDAR systems should also be paired with other means of object detection. Furthermore, the trajectories of objects in motion can be predicted by gaining insights into certain external factors. An autonomous system should be able to leverage this information to be informed of those trajectories.

Our project will address the problem by first creating an algorithm that can detect objects in uploaded images and then, in the future, apply this capability to video content as well. This should help the vehicle detect objects in its surroundings which will then be able to come up with a method to attack the situation at hand. With the data provided from a version of our program, a car should have an easier time figuring out the optimal moves to make by knowing which objects it is surrounded by.

Table of Contents

Chapter 1. Introduction

1.1 Project Goals and Objectives

The objectives of our Object Recognition program is to be able to detect all objects in an image, classify them and also calculate its position with a balance of speed and accuracy.

The goal of our project is to train our AI model in order to detect different objects in images, calculate the bounding borders and display outputs on a webpage. The algorithm should be able to detect multiple instances of an object and also be able to classify different objects under various environments.

1.2 Problem and Motivation

Object Detection technology has a wide range of usage ranging from autonomous driving to face recognitions. This technology is very beneficial in being able to track and analyze images that helps us better understand what happens in the image or video. Our group believes that object recognition technology is important and provides a lot of value thus we were motivated to use machine learning technology with pre-trained models to produce the outcomes of images that any user inputs.

The problem behind the project is finding out how to maintain a high accuracy recognition while producing low error rate results that can potentially be applied in a number of ways for real world applications.

1.3 Project Application and Impact

Application:

This project will be used as a visual recognition system to detect objects in the background of images. The system will allow users to create an account through means provided on our website, and then upload an image of their choice. We will then process the image and return it with bounding boxes around any objects that are detected.

The ability to have a ubiquitous image recognition service that is managed on the cloud enables users to explore the depths of modern technological advancements. As a domain entity, this project is the first step to enabling the user to access information about their images. An enhanced version of this application would allow the user access the artificial intelligence model via any system that has already been authorized, enabling the utility of this application to be extended to a wider scope.

1.4 Project Results and Deliverables

The result of our project is a web interface that allows uploaded imagery to be processed by a machine learning algorithm on the backend. The backend would read imaging inputs from the client, analyze the image for recognizable objects, and then deposit the image to a shared image storage repository to be consumed by the client, showing recognized objects in the image. To ensure secure resource usage, only authorized clients will be allowed to use the service, in this proof of concept prototype, web users must register through their email to use the service.

Chapter 2. Background and Related Work

2.1 Background and Technologies

Necessary knowledge required to complete this project includes machine learning, deep learning, image recognition, neural networks, and Python. Concepts and technologies that are key and may be used would be Long short-term memory, Keras, Google's Vision AI, and AWS Cloud Computation Services.

2.2 State-of-the-art

Current state-of-the-art techniques involve a few different methods. One of the methods that we have researched involves the idea of "visual saliency". The idea behind this method is to simply detect objects that look out of place with its surroundings and then classify this object. The way things are found as to whether they stand out is based on an idea called visual attractiveness. The idea behind visual attractiveness is very similar to how the human eye works in which we take notice in more "active" objects faster than others. This may be in relation to how fast an object is moving, the color of an object, or bright the object is. A group of researchers iterated on top of this idea and added the idea of segmenting an image to allow the image recognition to easily and quickly find multiple objects in a single picture rather than just the single most "active" object.

Another state-of-the-art method is through combining deep learning, object detection, and object recognition into a single topic. The benefits to this is that it reduces overhead by requiring less manual modification of training data therefore allowing easier access to large amounts of data. Another benefit is that transfer learning is possible via pre-trained weights. This makes it easier to train a deep learning AI as long as data is obtainable, however, if the overall data is not obtainable then the AI's accuracy will be severely impacted. On top of this, as more data is collected then eventually the AI will get to the point where inputs and outputs cannot be related to one another and inevitably will become a black box.

The last state-of-the-art that we will be going through is the KAZE descriptor. KAZE makes use of something called the Gaussian scale space as well as a Gaussian kernel. The idea behind image recognition in this manner is to combine the original image with a Gaussian kernel whose standard deviation is higher to get what is known as the Gaussian scale space. This is then able to be used to represent the scale of the space in the original image. There are a few advantages and

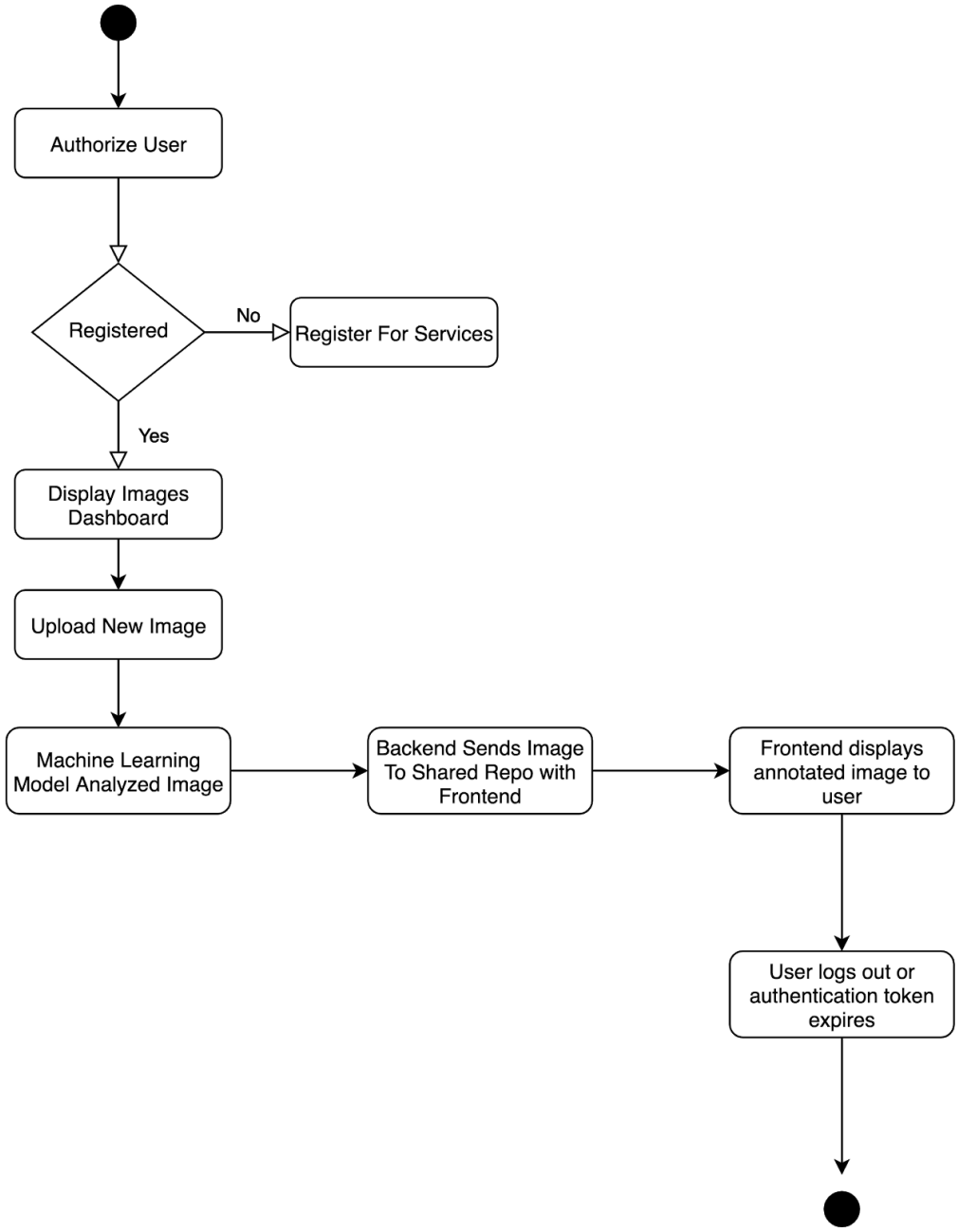
disadvantages to this method. Some advantages may include, a reduction of noise in the image, a greater emphasis on active objects in the image and this method is one of the simplest method for linear diffusion

The disadvantages of this method are that the finer details and edges of an image will become more blurred which makes it harder to designate what objects are located in an image. This makes KAZE less useful in our case due to it potentially being unable to specify certain objects on the roads such as traffic lights, however, it may still be a viable option due to it’s fast processing speeds.

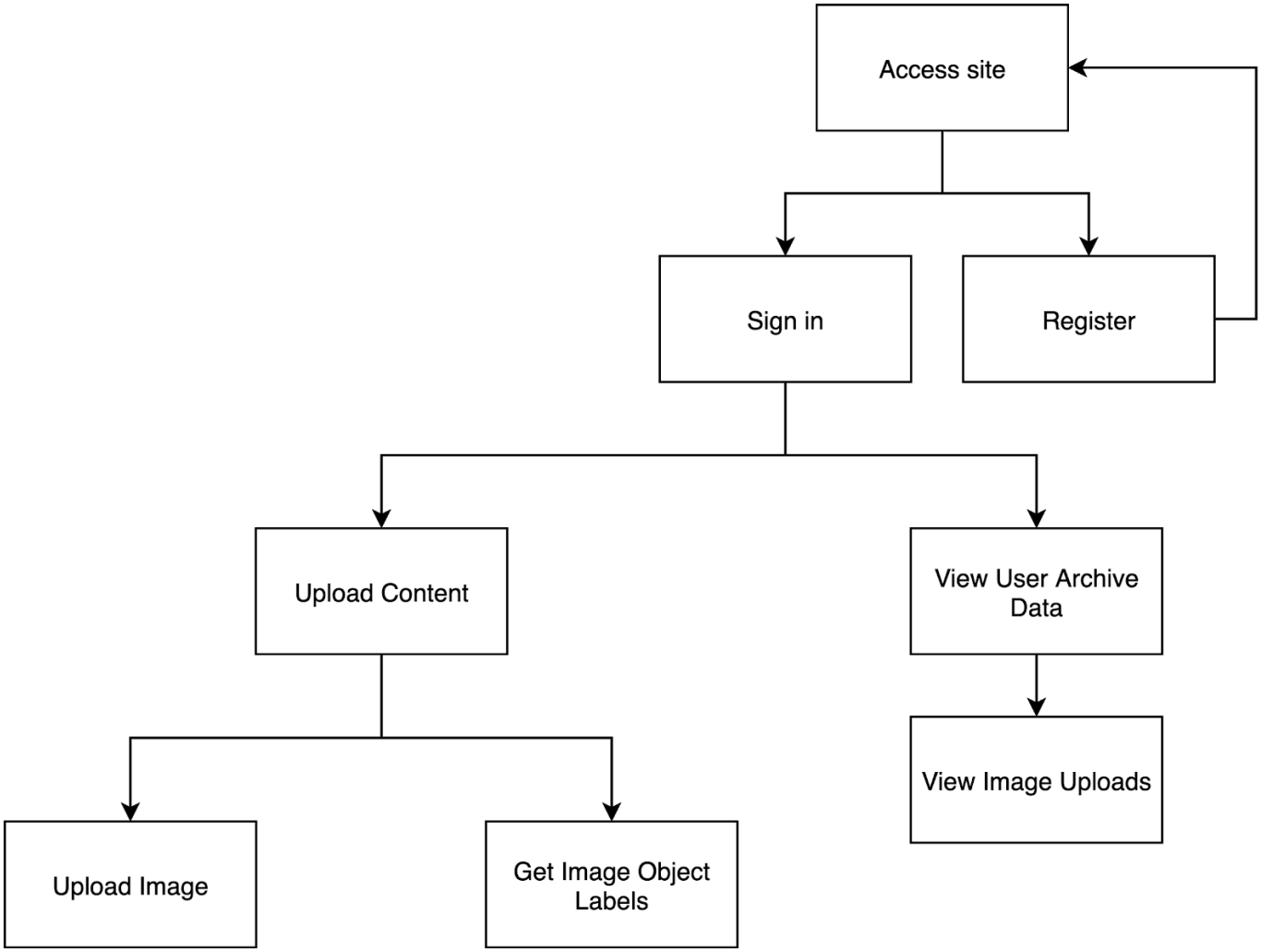
Chapter 3 Project Requirements

3.1 Domain and Business Requirements

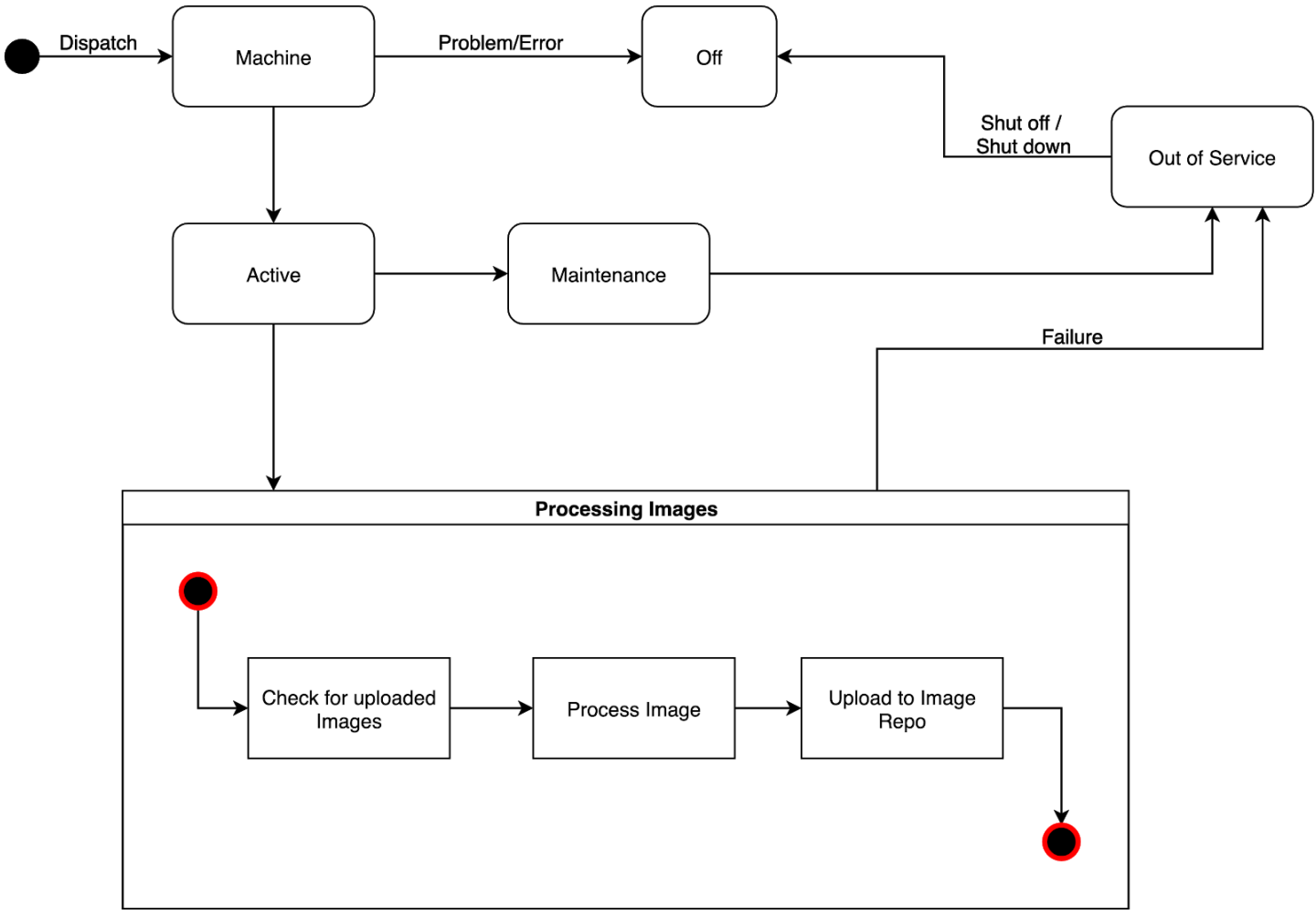
UML Activity Diagram



Decomposition Diagrams



State Machine Diagrams



3.2 System (or Component) Functional Requirements

- The application shall detect an object within uploaded images
- The application shall be public facing, they must register an account to use the service
- The application shall categorize detected objects
- The application shall output a frame of the image with a bounding box
- The application should accept images as input
- The application shall show all user inputted images, data must persist.

3.3 Non-functional Requirements

- The application should be reasonably fast in that the time taken should not be over a few minutes for an average image file
- The application should be able to recognize objects and show what objects it had detected
- The application should be able to not fail most of the time
- The interface should be intuitive and straightforward for users

3.4 Context and Interface Requirements

- Project shall be developed and hosted on AWS
- Project shall be developed using AWS amplify
- The project is meant to deployed as a web application
- Operating system shall be Linux and have the latest Python, TensorFlow, and Keras libraries installed

Interface Requirements:

- User shall use software through a web browser by accessing a http site.
- User shall be able to use the software on iOS and Android mobile devices
- Users shall be able to sign up and login to the application
- Logged in Users may submit images to the application for the model to look through

3.5 Technology and Resource Requirements

- The application will be run on modern browsers, i.e. Google Chrome, Safari, and FireFox
- Images and user data is to be stored using AWS server and database
- YoloV3 model is running on the server for objects to be detected
- Javascript is the language used for the frontend, with Python used in the backend

Resource	Requirement
Site client run on browser	ReactJS frontend
Machine Learning Component	TensorFlow, YoloV3 model
Site hosting, deployment, testing	Amazon Web Services, AWS Amplify

Chapter 4. System Design

4.1 Architecture Design

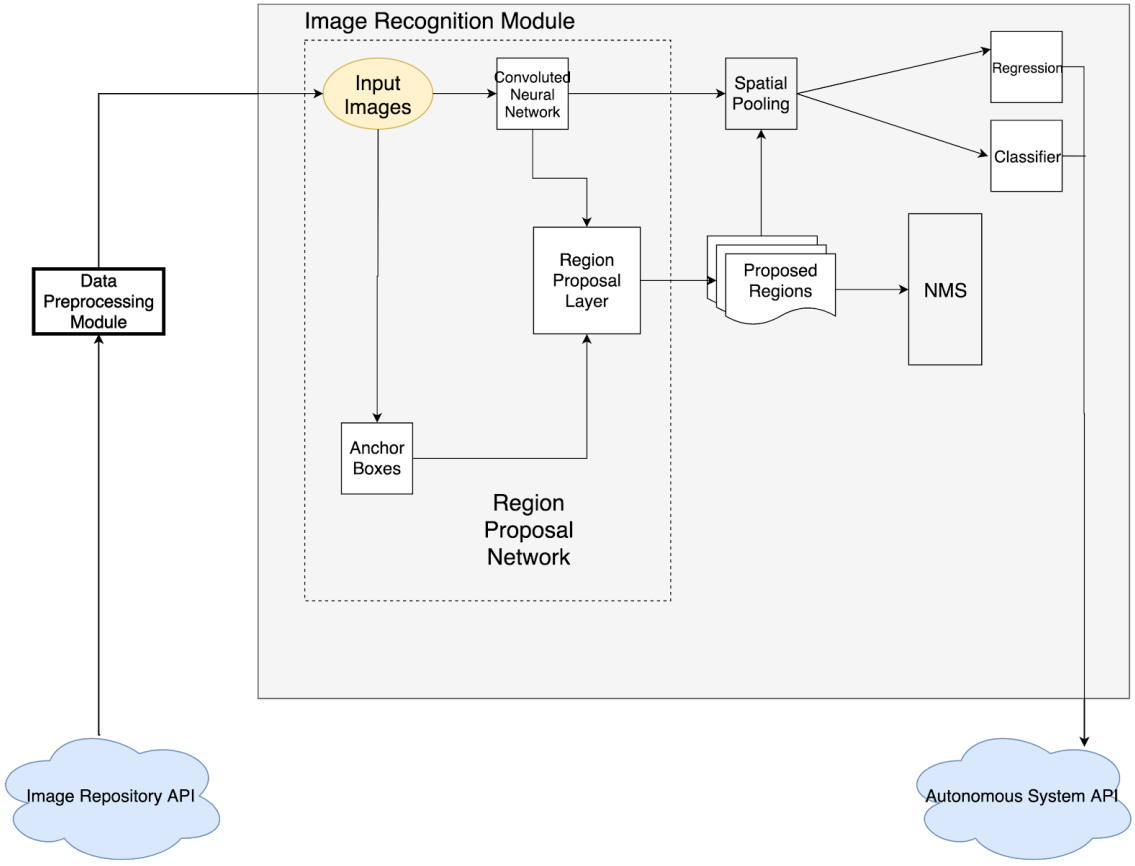


Figure 1. Proposed architecture for object recognition module

Our project's data source is a stream of images from an online repository. We will collect batches of frames and do any preprocessing needed since the images will be frames from video footage. We may need to adjust the spacing of each frame as needed.

A convoluted neural network is a deep learning algorithm that takes an input image and assigns a relative weight or importance value to components of that image in order to compare and contrast the content in each region, based on the color and

pixel density. The primary focus of a CNN is to reduce the image to a form that is less computationally intensive, while still maintaining its main features.

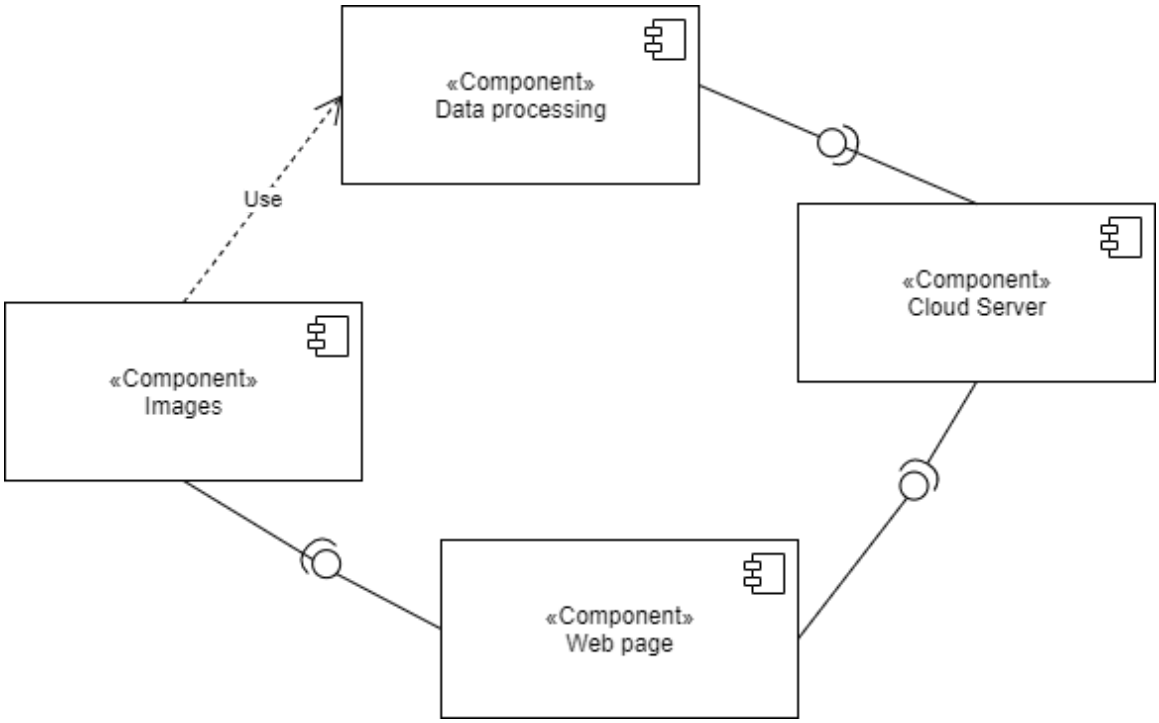
We will then pass the images to the first module of our image recognition system. The first step is to obtain the anchor boxes needed for use in the Region Proposal Network.

Once the CNN has formed layers to consider, the region proposal layer as a whole outputs those regions, or detection boxes with the highest probability of containing an image. This sub-module will output the potential layers to the Non-Maximum Suppression (NMS) component. The NMS sorts all of the detection boxes based on their scores. The detection box with the maximum score is selected and all other detection boxes with a significant overlap are suppressed.

The proposed regions will also go through the Spatial Pooling module. This module is similar to the convolutional layer. It further reduces the spatial size of the information encoded in the proposed region. To do this, it has a kernel that traverses subsections of the proposed region and reduces the information into a more concise representation, usually taking the max value.

The final modules of the system are able to acquire a regression value that represents the probabilities of objects being matched using a dataset API as the model training set. The system also outputs a classifier value that denotes whether the recognized image is a subset of some other set of objects of interest. This information is sent back upstream to the API for use with autonomous vehicle systems or any other system of that domain.

4.2 Interface and Component Design



F*igure 1. Interface and Component diagram*

There are 4 components in our object detection project consisting of Web page, images, data processing AI, and AWS cloud server. The data processing component is dependent on the images for processing. There are also required dependencies from the data processing component to the cloud servers.

4.3 **Structure and Logic Design**

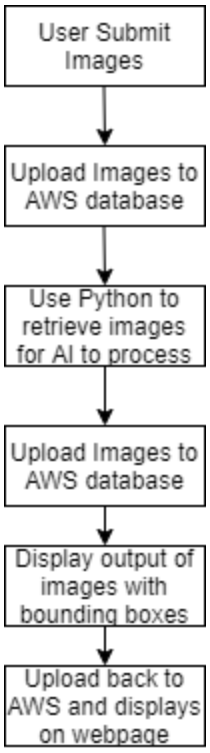


Figure 2. Logic of Finished Program

As shown in the figure, the logic behind the design of our object detection model can be seen through these seven steps. We have built a front end web application using ReactJS that allows a user to submit their files of images. These images are then stored on Amazon S3 Bucket. The backend uses python scripts to retrieve all the images and our AI using tensorflow will process and generate border bounding boxes for all the identifiable objects. These images are uploaded back to Amazon S3 bucket and then displayed on the webpage for the user.

4.4 Design Constraints, Problems, Trade-offs, and Solutions

4.4.1 Design Constraints and Challenges

Since training a machine learning model for computer vision requires a lot of processing power, this would be a large design constraint. Utilizing the required

amount of computing power would mean finding dedicated equipment for this task or using cloud computing.

The input data that our system processes may be stored on a cloud repository that has specific access requirements. Properly retrieving this data would be a potential constraint since each cloud repository has a specific way to access that information that requires some knowledge of that system.

The subject of any image may have its imagery taken from an angle that varies over time, thus dealing with these variations would require a robust training model. The information we pass on should be accurate and meaningful in quantity.

4.4.2 Design Solutions and Trade-offs

Training the AI models and processing images for recognition required a large amount of computer processing power. That is why we went with a pre trained YoloV3 model.

To ensure that the images can be processed efficiently, we required that the image files are in a few set image formats for processing. This allowed us to worry less about how to process different formats of images and instead focus on a few set types. The tradeoff for this was that our application would not be able to process any image unless it is specifically using the formats that we choose to adopt for our project.

To ensure the highest quality possible on an image, we have considered using various API such as Google's Vision AI or Amazons Rekognition AI to do the image processing. The problem with this would be incredibly high costs as well as using a service online rather than a model that we have the code on hand for.

Chapter 5 System Implementation

5.1 Implementation Overview

The current goals of our project are to create an AI that can detect road obstacles and their trajectories. Assuming we do not accomplish the part with the trajectories we plan on creating an application where a user can upload a picture and then be able to download a picture with every object we can detect. Either way, the scope will remain similar on the AI end, on the end with the application it will change a bit. For the AI, we are using TensorFlow 2 implemented with Python. The current testing has been done on using various Nvidia GPUs from the group's various computers. For the application we plan on creating the front end using ReactJS, a server using Python or JavaScript running on an AWS (Amazon Web Services) server, and MySQL for any database needs also on AWS. The primary dependencies for our project will include AWS access, a sufficiently powerful GPU, and an understanding of Python and/or JavaScript.

5.2 Implementation of Developed Solutions

Our current implementation is going on assuming that we are creating an application for users on top of our AI. This means that we are going to create a website for user's to upload an image to the database and then obtain a new image with bounding boxes around each detected object. The AI is currently done using YOLOv3 pre-trained AI via COCO training data. As for the application, it is done using ReactJS for the front end so we can better modularize the site into smaller and easier to handle modules to handle uploading, downloading, and user accounts in an easier to handle manner. We are using a JavaScript library known as Crypto-JS for all encryption for user authentication and then a web server that parses input coded in Python to help prevent SQL injections. We also have a cookie system set up so users can maintain a logged-in state through a unique key for their computer or mobile device.

5.3 Implementation Problems, Challenges, and Lesson Learned

Our major implementation problems are all AI-related. The reason for this is because we all have a lot of experience in web design but next to none in machine learning. While we did study various online courses and guides, it did not help much with our inexperience. Currently, we have a simple AI that can detect most general objects in either static

screenshots or through a live video feed using a webcam. Through this, we learned that machine learning is a relatively complex challenge to get started but with tough work and perseverance, we have managed to obtain relatively good results. Another problem that we thought we may encounter was trying to get the JavaScript front end and Python back end to respond to each other. This was a very minor issue that had a very simple solution we used in the end which was to communicate through HTTP protocols using string values.

Chapter 6 Tools and Standards

6.1. Tools Used

The tools that we are using for our project includes a ReactJS framework frontend, JavaScript using NodeJS for backend, MySQL for database management, and then TensorFlow using python scripts for data processing. The reason we chose ReactJS, NodeJS, and MySQL for our full stack development was due to the high popularity and ease to set-up an entire website. The high popularity allows us to solve relatively common problems easily with the help of the large community. On top of this, our group has a high level of familiarity with these languages and frameworks which allows ease in development for more focus onto the data processing.

For data processing, we chose to use TensorFlow frameworks with python due to TensorFlow being highly renowned for its excellent functionality and flexibility. Python was chosen due to it being a very high-level language which allows for ease in coding any necessary functions.

6.2. Standards

The primary standards that we followed were in relation to full stack development to allow a fully functional website that can be used by almost anyone. The codes also have documentation so that the code and quality stay consistent so ensure good coding development. We will also be using git for version management as well as separating our website into different modules to reduce the number of bugs on our website as a whole. For our backend, we are following standards to parse data that is sent from the frontend to try and stop SQL injection into our database to leak anything stored for higher security standards. We also created the backend to be a restful API to allow others to make use of our data processing for their own personal use.

Chapter 7 Testing and Experiment

7.1. Testing and Experiment Scope

As we are using YOLOv3 which is a pre-trained model, we are going to focus on its consistency and how much the model can handle. The bigger focus is testing the final output the model can provide for our usage in real-time environments. The model should provide the bounding box and trajectory of the vehicle in a consistent manner.

- Test Processes
    - System Testing/Black Box Testing
        - The application used should work in the full product in the full stack with ReactJS, MySQL, and TensorFlow.
        - The full stack should not fail or distort data to and from the model.
    - Stress Testing
        - The model should be able to work in a variety of file sizes and rate of input. Stress Testing will see how big the file input can be and how many it can handle at once.
    - Unit Testing
        - The model should be able to recognize objects with few input variables. The frontend application should be able to function in itself before integration with the model.
- Test focuses and objectives
    - We focus on the consistency and accuracy of the output when given specific inputs. Output should remain consistent given data and accuracy of results should be above a defined percentage.
    - UI application for the model should be functional allowing users to upload and should not fail when given typical input.
- Selected Test Criteria
    - Accuracy Percentage of results
    - Consistency of results
    - Reliability of frontend application

7.2. Testing and Experiment Approach

In addition to the pre-trained YOLOv3 data, we used new inputs and many images from the web in order to verify the accuracy of the AI system. We ran a lot of testing on

different images in order to test and detect different objects. Examples of some of the testing include streets, parking lots, houses, stores, and parks since these are all parts of the road where an autonomous car will encounter. We also tried out different training regimes to test and see which model produces the best outcome. The results from different testing verify that the model indeed works and the results of the object detection are very consistent.

```
I0415 18:32:52.767627 25276 detect.py:35] weights loaded
I0415 18:32:52.768624 25276 detect.py:38] classes loaded
I0415 18:32:53.468519 25276 detect.py:55] time: 0.672966480255127
I0415 18:32:53.468519 25276 detect.py:57] detections:
I0415 18:32:53.472540 25276 detect.py:61]       person, 0.9927860498428345, [0.48121342 0.53539205 0.53943235 0.7205671 ]
I0415 18:32:53.473505 25276 detect.py:61]       bicycle, 0.9893728494644165, [0.00294197 0.67261803 0.1278964  0.885651  ]
I0415 18:32:53.474503 25276 detect.py:61]       person, 0.987955629825592, [0.8411941  0.59022176 0.97005767 0.782845  ]
I0415 18:32:53.475531 25276 detect.py:61]       person, 0.980888843536377, [0.64710873 0.58879596 0.7783466  0.77255577]
I0415 18:32:53.476496 25276 detect.py:61]       chair, 0.9627692699432373, [0.72545874 0.66155595 0.79952323 0.77026135]
I0415 18:32:53.477529 25276 detect.py:61]       person, 0.951276957988739, [0.5676229  0.5748621  0.63427913 0.7228936 ]
I0415 18:32:53.477529 25276 detect.py:61]       person, 0.9290788173675537, [0.52887225 0.5971681  0.5886619  0.7197207 ]
I0415 18:32:53.480485 25276 detect.py:61]       chair, 0.8350304365158081, [0.8386051 0.6635252 0.9021171 0.7732545]
I0415 18:32:53.487532 25276 detect.py:61]       person, 0.830353856086731, [0.359318   0.5744997  0.41999254 0.7052051 ]
I0415 18:32:53.488530 25276 detect.py:61]       bicycle, 0.8081040382385254, [0.35342014 0.62453735 0.4655499  0.7413868 ]
I0415 18:32:53.491521 25276 detect.py:61]       person, 0.52339768409729, [0.64574134 0.59797615 0.68569756 0.68257076]
I0415 18:32:53.536432 25276 detect.py:66] output saved to: ./output.jpg
PS C:\Users\huynh\cmpe195\yolov3-tf2-master>
```

```
I0415 17:27:27.372287 35248 detect.py:35] weights loaded
I0415 17:27:27.373249 35248 detect.py:38] classes loaded
I0415 17:27:28.040141 35248 detect.py:55] time: 0.6209814548492432
I0415 17:27:28.040141 35248 detect.py:57] detections:
I0415 17:27:28.044133 35248 detect.py:61]       car, 0.9966800212860107, [0.26270664 0.34669653 0.40587908 0.5401914 ]
I0415 17:27:28.044133 35248 detect.py:61]       bus, 0.9952293038368225, [0.3228875  0.15145975 0.6138201  0.506682  ]
I0415 17:27:28.050979 35248 detect.py:61]       bus, 0.9951198101043701, [0.67014587 0.17855005 0.9937171  0.5290145 ]
I0415 17:27:28.052977 35248 detect.py:61]       person, 0.9875381588935852, [0.6138846  0.34505445 0.6877058  0.77529997]
I0415 17:27:28.053974 35248 detect.py:61]       person, 0.9875273704528809, [0.6745359  0.36693066 0.76731586 0.80656105]
I0415 17:27:28.061423 35248 detect.py:61]       person, 0.9751366376876831, [0.22922698 0.34322935 0.26619264 0.52385163]
I0415 17:27:28.063435 35248 detect.py:61]       person, 0.9731233716011047, [0.10603387 0.29673994 0.1410992  0.52050036]
I0415 17:27:28.065445 35248 detect.py:61]       person, 0.8406072854995728, [0.73152274 0.19712627 0.76029474 0.2559415 ]
I0415 17:27:28.073545 35248 detect.py:61]       person, 0.7661858201026917, [0.14938346 0.30133063 0.20066255 0.5780297 ]
I0415 17:27:28.076538 35248 detect.py:61]       person, 0.6201698780059814, [0.8069899  0.19955894 0.82683456 0.24689198]
I0415 17:27:28.077536 35248 detect.py:61]       traffic light, 0.5584408640861511, [0.2863981  0.26624355 0.29791862 0.28726056]
I0415 17:27:28.136381 35248 detect.py:66] output saved to: ./output.jpg
PS C:\Users\huynh\cmpe195\yolov3-tf2-master>
```

7.3. Testing and Experiment Results and Analysis

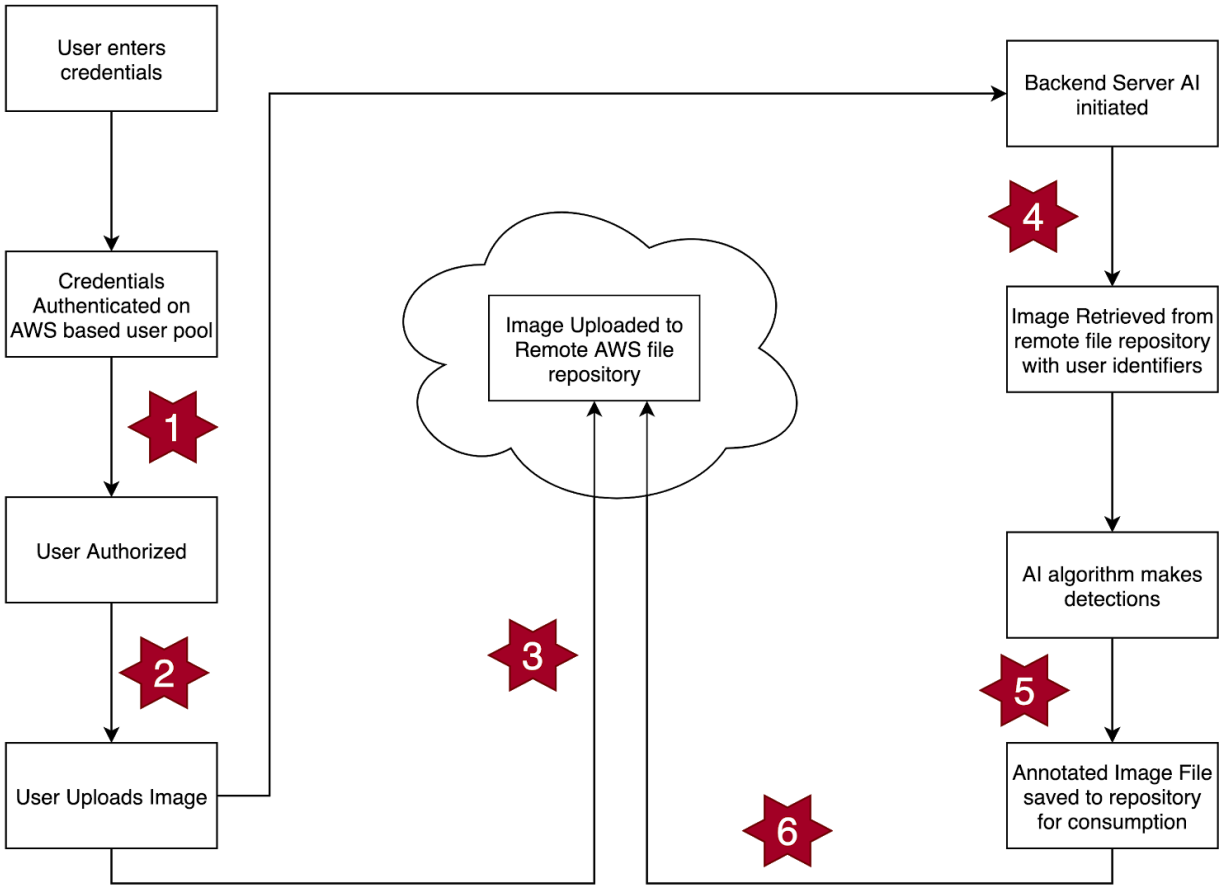
System Testing

For system testing we test the system as a whole, incorporating backend and frontend systems. The data flow will begin from the user interface and the result will be reflected in the user's account page showing all images they have uploaded with the tags the AI has detected. Our test cases are primarily based on this flow of information and we have focused on a number of points by adding mutations to see if the system will be tolerant to the bugs introduced.

Data Flow Branch	Test Cases	Test Type	Expected Result	Success Rate	Number of test runs
1	enter created user	basic	success	100%	5
	enter foreign user	mutation	error	100%	5
	enter username capitalization differences	mutation	error	100%	5
2	user can upload	basic	success	100%	10
	user can view own images	basic	success	100%	1
	user can access others' images	mutation	error	100%	1
	user can view AI recognition tags	basic	success	100%	1
3	approved image file types uploaded	success	success	100%	1
	any file type uploaded	mutation	error	100%	1
	multiple files uploaded	mutation	error	100%	1
4	server runs algorithm on selected image	basic	success	100%	1
	server runs algorithm on null image	mutation	error	100%	1
5	AI shows image tags on image	basic	success	100%	1
	AI gives indicates no tag detection if unable to detect	basic	success	100%	1
	AI gives runs on same image consecutively	mutation	error	100%	1
6	Image stored in user's portion of repository	basic	success	100%	1

	image file name incorrectly named	mutation	error	100%	1

Mutations Introduced in System Data Flow



Integration Testing

We conduct integration testing using the Amazon Web Services console. The console shows input data that any component should receive from the SDK. Using this method, we can easily verify our own modules based on their ability to communicate with these external components.

Unit Testing

Our unit testing is done with the backend system on the terminal and done locally. Using the built-in modules for our recognition algorithm makes verifying the model's accuracy very straightforward as the information is shown after every run of the model.

The front end portion is tested through a javascript testing framework. In this way, we can isolate each function that will handle data and verify that it runs according to the requirements.

Chapter 8 Conclusion and Future Work

In summary, we created an application that allows users to submit photos and receive the analyzed image using our AI through the YoloV3 model. The front end web application is built using ReactJS and images sent by the users are connected to an AWS cloud server. Our object detection model retrieves all images from the server and runs through the image recognition program. This project proved to be different than we had originally envisioned as it changed as the months passed by but we are confident that it was a fulfilling endeavor.

Although we are unsure if we would use this particular application of image recognition in the future, the field for it seems like a promising one and we may continue into it after this. Image recognition has many applications and the uses that our project could lead to is a path that we may like to follow. Working with the different tools in our disposal to accomplish this project together allowed us to learn and to improve while giving us insight into what we could do in the future.

References

M.-R. Hsieh, Y.-L. Lin, and W. H. Hsu, “Drone-Based Object Counting by Spatially Regularized Regional Proposal Network,” *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

H. Madokoro, A. Kainuma, and K. Sato, “Non-Rectangular RoI Extraction and Machine Learning Based Multiple Object Recognition Used for Time-Series Areal Images Obtained Using MAV,” *Procedia Computer Science*, vol. 126, pp. 462–471, 2018.

D. M. Ramík, C. Sabourin, R. Moreno, and K. Madani, “A machine learning based intelligent vision system for autonomous object detection and recognition,” *Applied Intelligence*, vol. 40, no. 2, pp. 358–375, 2013.

S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, Jan. 2017.

S. I. Schwartz, “How Autonomous Cars Will Reshape Our World,” *Wall Street Journal*, 27-Oct-2017. [Online]. Available: https://www.wsj.com/articles/how-autonomous-vehicles-will-reshape-our-world-1539871201.

"Self-driving cars in the EU: from science fiction to reality," *Self-driving cars in the EU: from science fiction to reality | News | European Parliament*, 14-Jan-2019. [Online]. Available: http://www.europarl.europa.eu/news/en/headlines/economy/20190110STO23102/self-driving-cars-in-the-eu-from-science-fiction-to-reality.