Ben Fernandez, Kyle Rood, Suraj Shah

11/12/19

**AmberVision**

AmberVision is a tool intended to assist law enforcement by finding people involved in active Amber Alerts. The final product will be a web-based application, a back end for the API and database, machine learning algorithms to analyze the data, and a front end User Interface to display that data. The structure of the application is not very complex, and that is on purpose. AmberVision is only going to be effective if it is fast, so simplicity is at the core of most of our design decisions.

The key to making our system run efficiently is using docker images. Docker allows you to containerize code and run this specific code as if you are running it on a virtual machine. Our team has access to a GWU server that we will use to host our Database docker images for our image downloader, machine learning algorithms, as well as our front end website. Our image downloader will be one container, and will constantly be sending information about each image to the database. Both the front-end and machine learning algorithms will need to be their own docker containers, also sharing information using the API. The machine learning algorithms will output their bounding boxes and classifications, send that to the database using a POST API call, and the front end will receive those classifications and display them to the user.

Docker will allow our machine learning algorithms, image downloader, and front end to run on the same server, and it will help with the overall development cycle. If something isn’t working, we can simply rebuild those docker images in a matter of minutes. Docker will allow us to improve the efficiency of the development and overall operation of the site.

Essentially, there are three main components of AmberVision, the API, the machine learning component, and the front end web framework. As shown in the diagram, the core component that connects the other two pieces is the self-built API. In layman terms, this allows the other components to handle transactions with the database with ease by using the API. The API will be a RESTful service that talks to the database which will have information about the images we store such as latitude and longitude and the file path of the image. The machine learning component will then use the GET requests of the API to get images from cameras and run the transfer learning models to get all the cars in the image. After the bounding boxes are found, the values of these bounding boxes are POSTED to the API to be stored in the database. After this component, the front-end Web App will call the API and get the file path for the image and the bounding boxes for that specific image and then draw the bounding boxes on the image.

Since there are three major components, there is a clear separation of the parts, but there will be an integration throughout the entire development process itself. Suraj is responsible for the API construction and the TrafficLand data collection. This part is tasked with collecting TrafficLand camera data, storing this in the database, and creating a REST API that will be used for the other two components. Kyle will be in charge of the machine learning and the combination of the canned models to create an overall deep learning approach that can detect cars in these traffic camera images. Ben will be in charge of the front-end Web App written in Javascript. He will use the data from the API and the machine learning outputs to display the results for law enforcement to use. There is a separation of parts, but there is an overlap of work at certain points. For example, with the machine learning, there will be continuous development to train the algorithms to capture cars better in these images, which will require a team effort to brainstorm algorithms that can better detect objects in images. For example, this includes preprocessing data, which will require a group effort.



**Figure 1:** Initial Design of the full pipeline

This shows the overall design of the project

**Responsibilities:**

**Suraj:** API and data engineering with TrafficLand and database design

**Kyle:** Machine learning and deep learning algorithms for object detection of the cars

**Ben:** Front-end Javascript web app