Matt Williams
4/7/2022
Computer Vision
Lab 5

Part 2: Explain how your Image Project app works. Extract the code base and get the model to run outside the web page. This may help you understand the underlying implementation and help with your explanation. Your can use code snippets to support your explanation of the model.

- First for our project, we need to download the html file and the zip file that contains the necessary information to run the model from Teachable Machine. Next we need to make a folder called “my_model” and extract all files in the zip file into that folder. The HTML file is expected to be in the same directory as the “my_model” folder.

- Init Function(Asynchronous function that allows the init function to continue running while functions within its body are still running.)
  1. First we need to load in the model structure and metadata that are now inside our “my_model” folder.
  2. Next we make an await call to load the model based on the information we just loaded in. This method call will likely take a few seconds to complete, hence why we use the await keyword so the init function can continue to run.
  3. The script then connects to our webcam. Afterwards 2 await calls are made to set up and play the webcam.
  4. After the webcam is set up, we start our asynchronous method “loop”.
  5. After the loop has started, some other HTML properties are made to show prediction percentages.

- Loop Function (Asynchronous function that allows the loop function to continue running while functions within its body are still running.)
  1. First we update the webcam to get the most up to date frame.
  2. We make an await call to our method “predict”.
  3. After the “predict” method has been called asynchronously, we make another call to continue our “loop” method.

- Predict Function (Asynchronous function that allows the predict function to continue running while functions within its body are still running.)
  1. First, we make an await call to our model’s “prediction” method based on the input from the webcam.
  2. After our model has made its predictions, HTML components are updated to show the percentage distribution for the prediction.

Part 3:

For my project idea, I wanted to tackle an object detection task since I have done image classification in the past. Since Self-Driving Cars are such a popular Computer Vision research topic, I'm interested in using Deep Learning in order to detect objects on the road from the drivers perspective. I want to build a model that can detect: traffic lights, pedestrians, cars, cyclists, cars, and trucks. Not only do I want the model to detect these objects, but I also want the model to produce bounding boxes that surround the objects based on their location in the image.

For the model itself, I plan on training the model using Tensorflow in Python. In order to visualize the results of the model and present the model to the class, I will be using the technologies we use in class including: Tensorflow.js, OpenCV, WebGL, Javascript, and HTML.

- Data Set: https://www.kaggle.com/datasets/alincijov/self-driving-cars