Detecting Pedestrian Intent

Hyunil Yoo

Overview

1. Why detecting pedestrian intent?
2. How I’m going to approach this topic?
3. Results
4. What are the limits and future work

1. Why I chose this topic?

In Autonomous driving field, the most important thing to consider is public safety.

Therefore, prediction on other cars' and pedestrian's behaviors are one of the key features for public safety. Predicting the other cars' behaviors is an important task, but it might be more important to focus more on pedestrian's behaviors because when an accident involves a pedestrian, the probability of a fatal accident is high.

If we can predict what a pedestrian's intent on the public road, it will be easier to make a self-driving car maneuver around pedestrians.

2. How I’m going to approach this topic?

1.Getting data

   a. drive view videos on YouTube

   b. Split the videos into frames

   c. Resize the images to appropriate size

   d. Label the images using LabelImg library

2. Using Tensorflow Object Detection API

   a. Setting up the environment

   b. Choose a neural network model

   c. Train the model

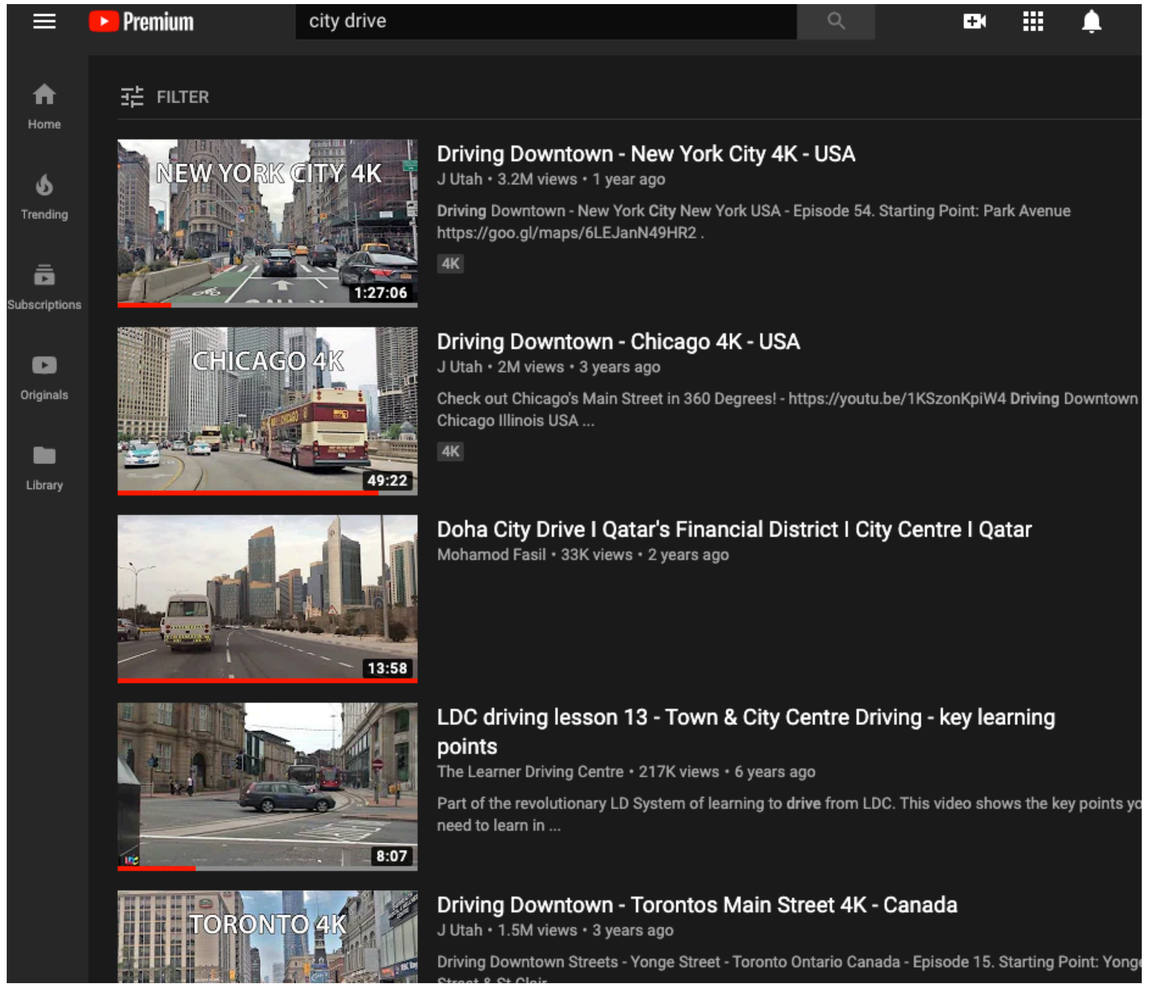
3. Predict the pedestrian intent on unseen images and videos

   a. Using OpenCV to detect pedestrian intent on images and videos

2.1. Getting data

- There are many YouTube videos that have drive views.

- Clip the videos that are interacting with pedestrians.



2.1. Getting data - split the videos into frames and resize the images to appropriate size

Using OpenCV

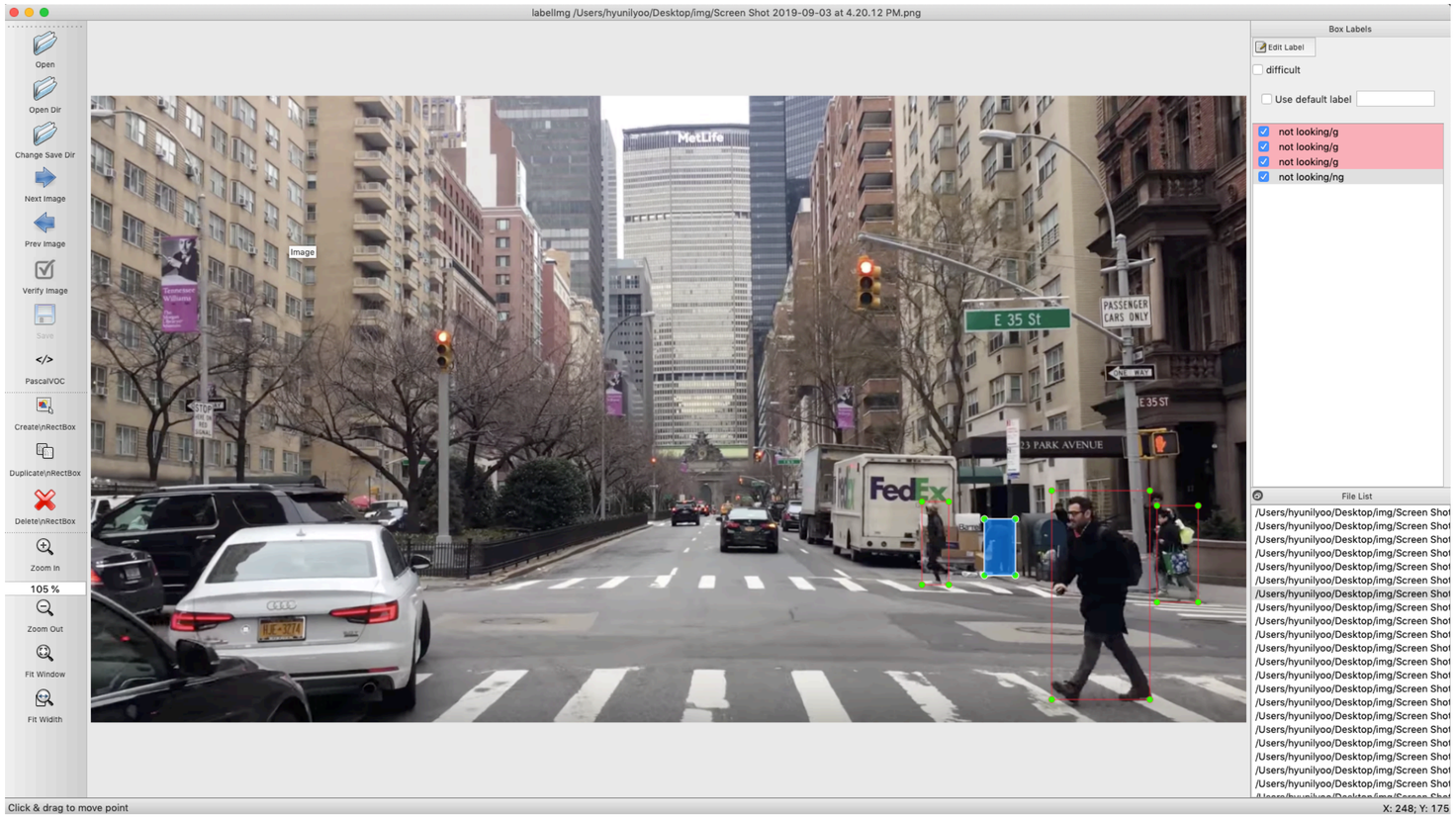
- Split the video into frames to label pedestrian intent.
- Resize the images to 800 x 600 because the original size of videos are too big to train the model (Too slow to train the model).

2.1. Getting data - label the images using LabelImg library

Using LabelImg to label the images

Labels:

1. looking_g (Looking and crossing the road)
2. looking_ng (Looking and not crossing the road)
3. n_looking_g (Not looking and crossing the road)
4. n_looking_ng (Not looking and not crossing the road)



2.2. Using Tensorflow Object Detection API - Setting up the environment

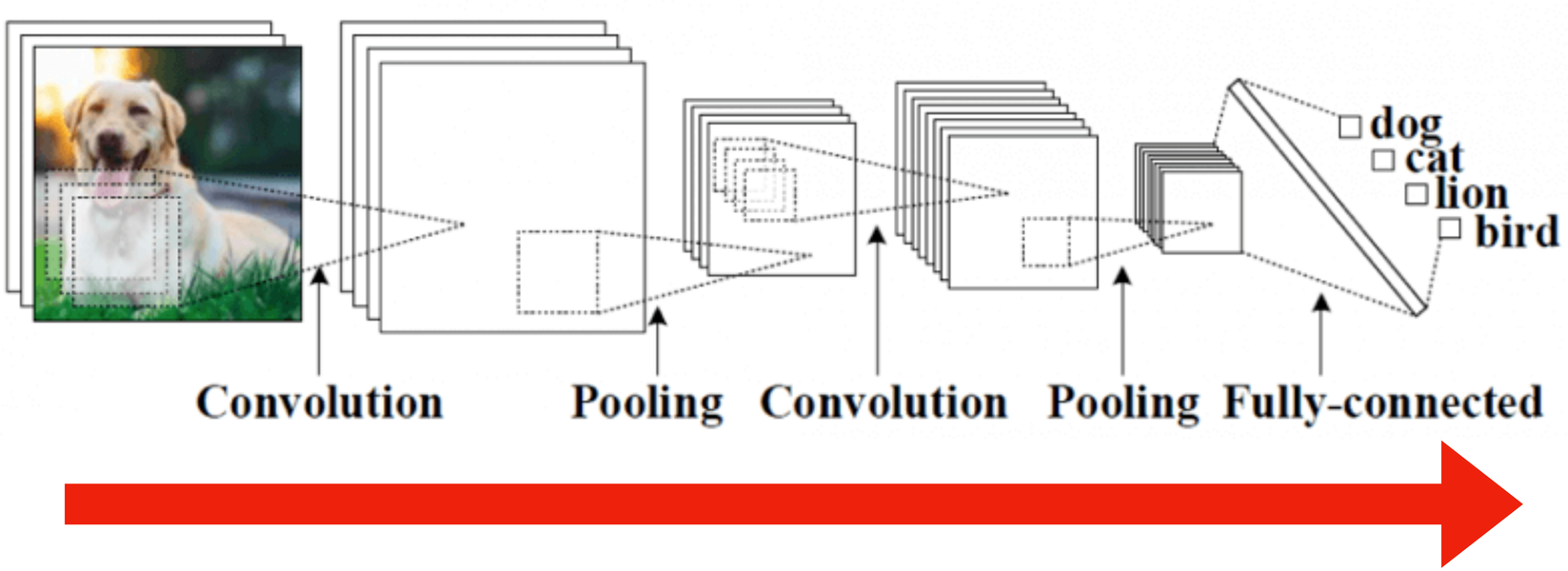
- Since I do not have a powerful gpu on my local machine, I need a cloud powered gpu.
- There are many resources such as AWS, GCP, Azure, and Colab
- I chose Google Colab because it’s free and has enough gpu power that I need (Each session: 13g RAM, Nvidia Tesla K80).

2.2. Using Tensorflow Object Detection API - Choosing a neural network model

How Convolution Neural Network works?

The technique I'm using in this problem is Convolution Neural Network (CNN).

CNN is commonly used in computer vision to identify objects.



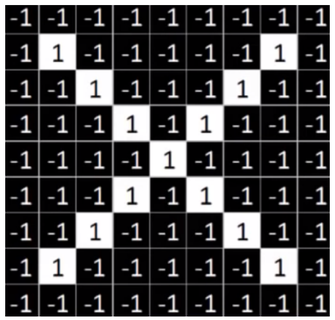
*The illustration above is overview CNN steps.

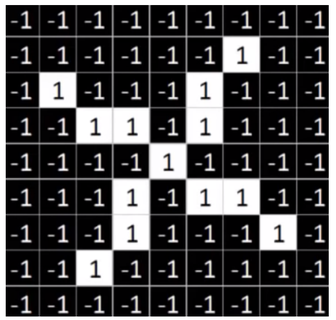
2.2. Using Tensorflow Object Detection API - Choosing a neural network model

- The input for CNN is a vector(pixels) with a color channel (if it’s a black and white image channel equal to 1; if it’s a color image, channel equal to 3 - RGB).

- Instead of comparing every pixel, CNN compares parts of the pixels so that it can classify tricky images.

Ex: For computer, if we compare all the pixels on these images, it is hard to classify that these images mean the same thing.



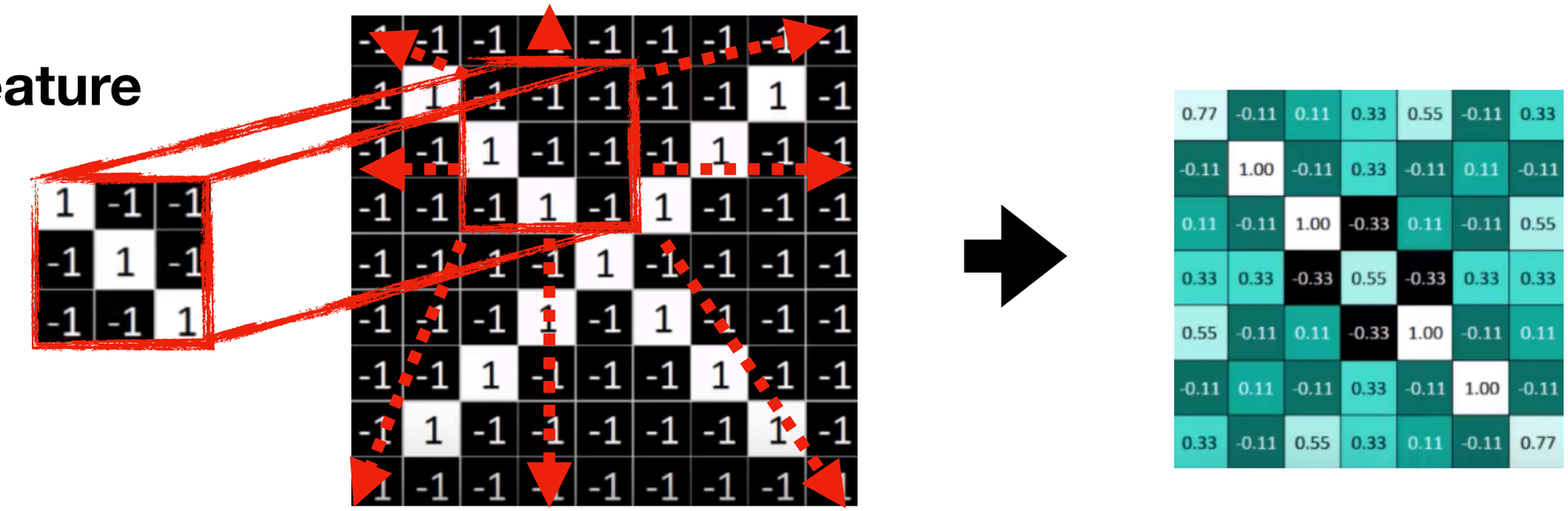


2.2. Using Tensorflow Object Detection API - Choosing a neural network model

- Therefore, we choose features and go over every pixel to match pattern with the features (filtering; this also called Conv).

EX:

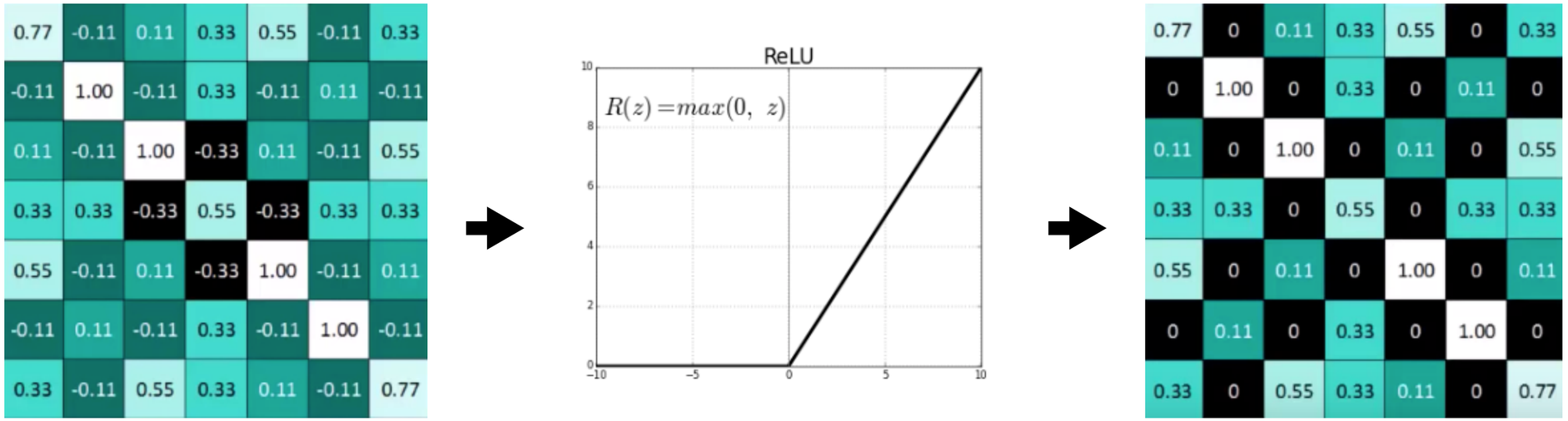
Feature



- After the filtering, we can find overall matches on an image
- CNN uses multiple feature to find out overall matches

2.2. Using Tensorflow Object Detection API - a neural network model

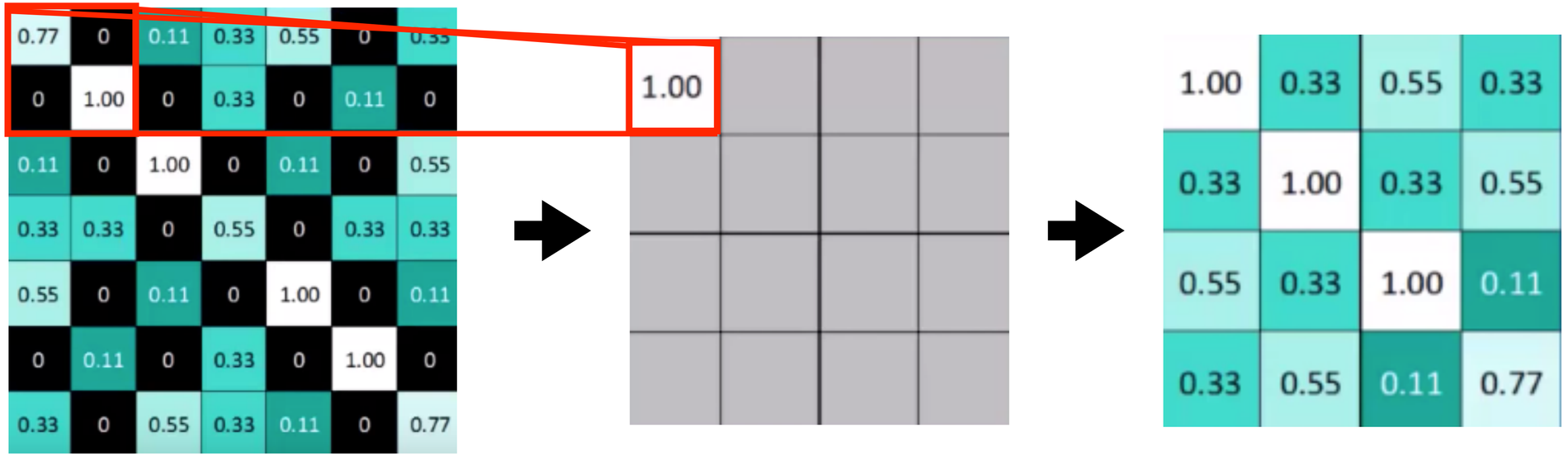
- Normalize the values by ReLU (Rectified Linear Units) activation function.
- This changes negative values to zero but all the positive values have same values.



- This simplify the calculation.

2.2. Using Tensorflow Object Detection API - a neural network model

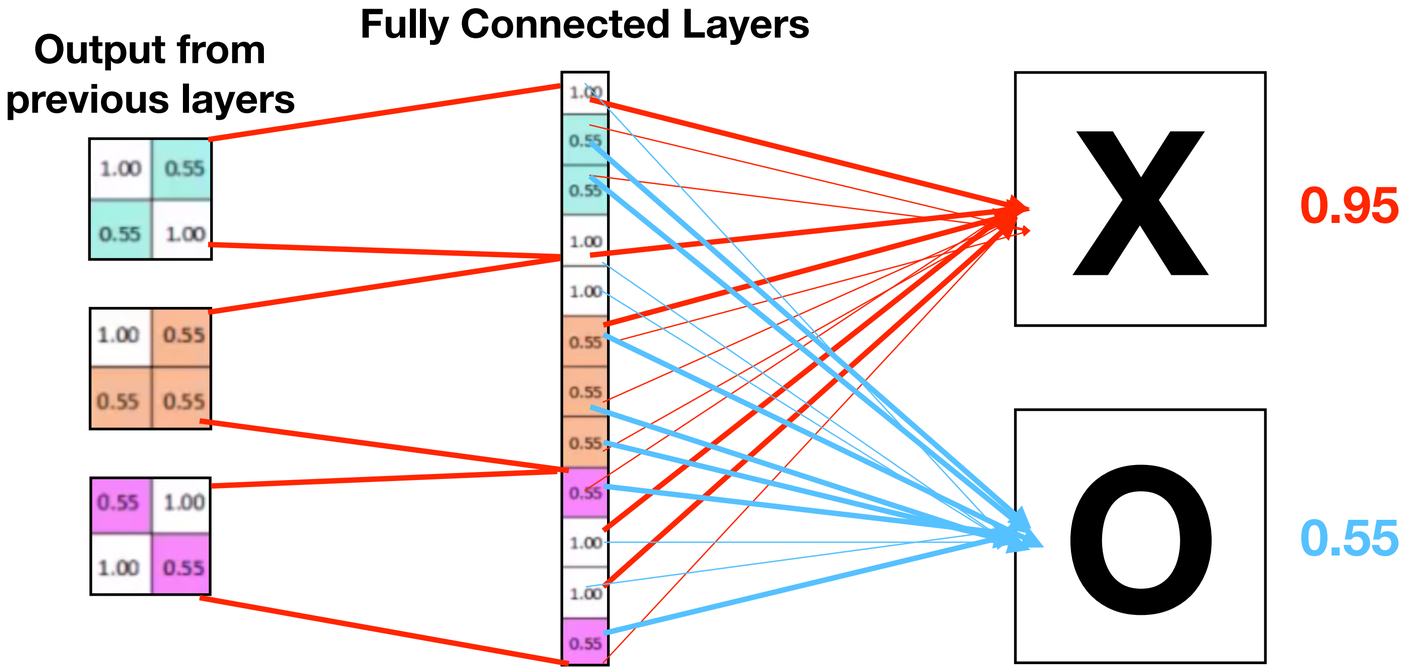
- After normalization, next thing is pooling.
- This Shrinks the image.



- From each window (you have to choose window size that how many pixels you want to see, and a stride that decides how much the window is going to move on the image) take the maximum value.

2.2. Using Tensorflow Object Detection API - a neural network model

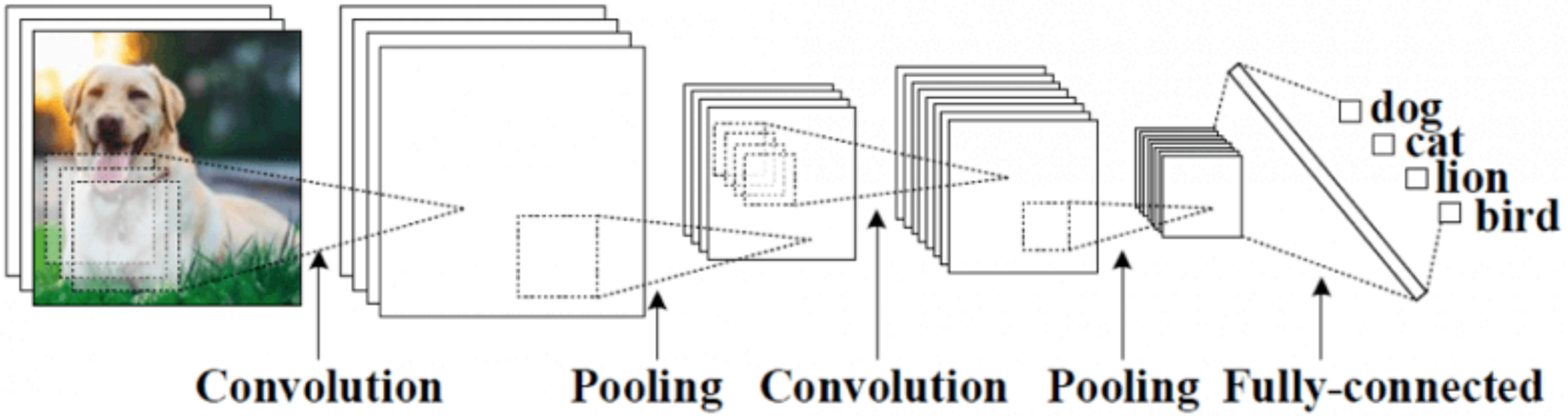
- We can put previous layers whatever we want, and this define how model is deep.
- After those layers (Conv, ReLU, Pooling), now we need Fully Connected Layer to predict the outcome.



- According to Fully Connected Layers, the one that gets most votes will be the final prediction.

2.2. Using Tensorflow Object Detection API - Train the model

- Using a previous structure, train the model to detect the pedestrian intent.



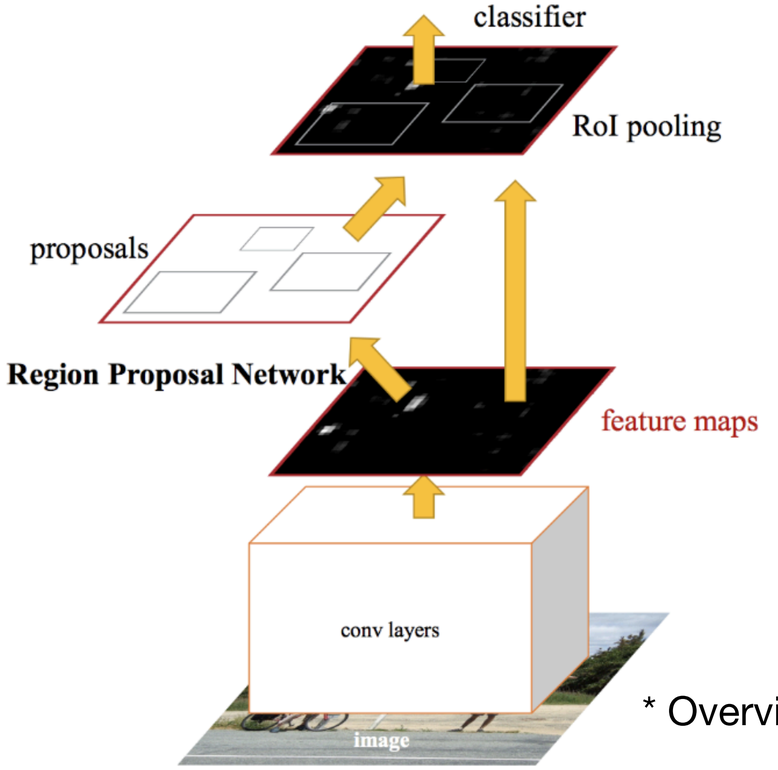
Forward Propagation

Backward Propagation

- Iterate Forward and Backward Propagation to find a global minimum point (reduce error).

2.2. Using Tensorflow Object Detection API - Train the model

- Using a previous structure, train the model to detect the pedestrian intent.
- To make the training process faster, Faster RCNN Inception V2 COCO models from Tensorflow's model zoo is used to train the model.



* Overview of Faster RCNN

2.3. Predicting the pedestrian intent on unseen images and videos

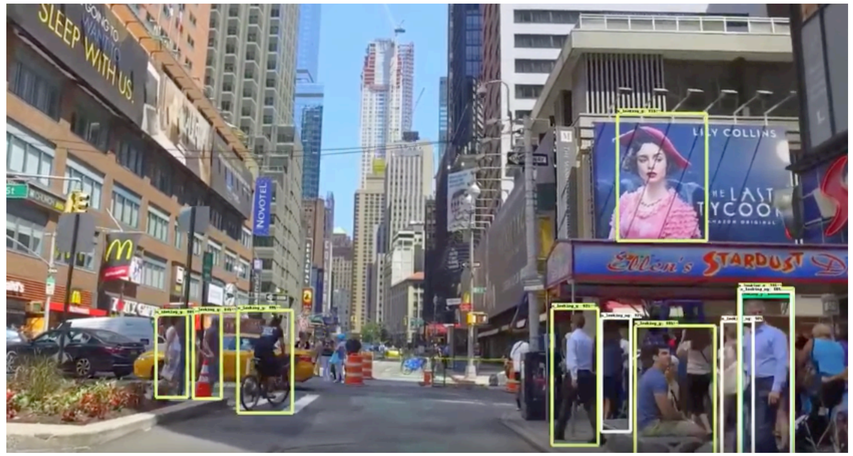
- Training time: ~ 4 hours and 30mins

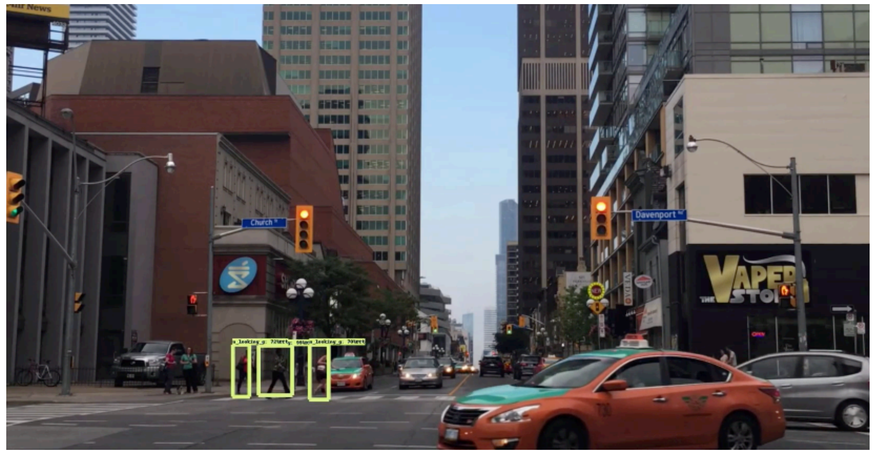
Detecting on images:

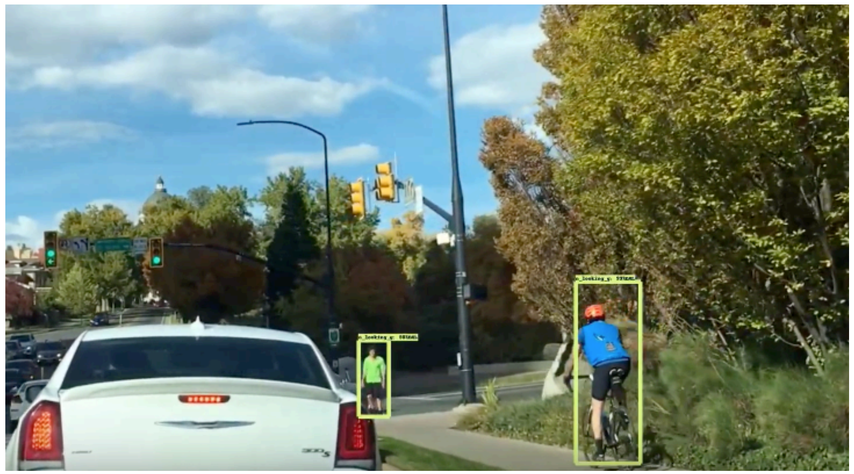
*Github link to view the images: https://github.com/hyunilyoo/pedestrian_tracker/blob/master/test_imgs/

3. Result

Detecting on videos:







*Github link to view videos: https://github.com/hyunilyoo/pedestrian_tracker/tree/master/test_videos

3. Result

Insights

- The model generally good at predicting pedestrian intent with a few pedestrians, but if the model tries to predict a large number of pedestrians, the accuracy fairly drops.
- On the one of the test videos, there is an advertisement display on the building with a face and upper body, and the model detect that display as a pedestrian.
- To solve this problem, the model needs more data to train. Currently this model only trained on 350 images; therefore, with a large dataset, the model will perform much better than now.

4. What are the limits and future works

Limits

- Time constraint: Since the labels were customized labels (ex. Not looking going, looking going, etc…), I needed to make my own dataset and labeled it. It took about 6 hours to make my own dataset with the labels for 350 images.

Future works

- Make an application after the model predicted objects. For example, if the model predicts there are pedestrian not looking and crossing the road, output the signal that proceed to pedestrians more cautiously because they don’t know the car is coming toward them and could come toward to the car with whatever reasons.