Vehicle detection with CNNs - Part 1

July 11, 2017

Marc Chan
marc.chan89@gmail.com

Overview

The SG gvmt hosts an open repository of datasets on https://data.gov.sg.

Developers have access to a realtime API feed that taps into a network of traffic cameras around the country.

What if?
We could use these traffic images to alleviate urban congestion...

Pieces of the puzzle

Scope of Project

Scope of Part 1

Part 2....

Classification	Localization	Detection	Coordination
Does the image contain a vehicle?	Where is the vehicle located within the image?	How many vehicles are in the image?	How does the traffic condition in one image affect the condition in another image?

1. The Data

The Raw Data

Every 2 minutes, a new JSON file is generated. This file contains the latest links to 70 traffic cameras around SG.


```
{
  "items": [
    {
      "timestamp": "2016-12-12T09:44:52+08:00",
      "cameras": [
        {
          "timestamp": "2016-12-12T09:44:12+08:00",
          "image": "https://images.data.gov.sg/api/traffic-images/2016/12/2f4cd31c-909a-4f12-8a1a-455a3cd30d66.jpg",
          "location": {
            "longitude": 103.8587802,
            "latitude": 1.323604823
          },
          "camera_id": "1701",
          "image_metadata": {
            "height": 480,
            "width": 640,
            "md5": "860f8652c8168c51d63b7f5ec24c5cca"
          }
        },
        {
          "timestamp": "2016-12-12T09:44:12+08:00",
          "image": "https://images.data.gov.sg/api/traffic-images/2016/12/895bb8fc-5a4c-49ab-9fff-f7a5e0e7abc0.jpg",
          "location": {
            "longitude": 103.862203282048,
            "latitude": 1.32814722194857
          },
          "camera_id": "1703",
          "image_metadata": {
            "height": 480,
            "width": 640,
            "md5": "a3131f8768555cb8ddfb7e748975032c"
          }
        },
        {
          "timestamp": "2016-12-12T09:44:12+08:00",
          "image": "https://images.data.gov.sg/api/traffic-images/2016/12/b30f5181-814f-43c2-8114-9019e11ed303.jpg",
          "location": {
            "longitude": 103.837524510188,
            "latitude": 1.28569398886979
          },
          "camera_id": "1704",
          "image_metadata": {
            "height": 480,
            "width": 640,
```

Data Preprocessing:

1. Padding

Original: 320 x 240

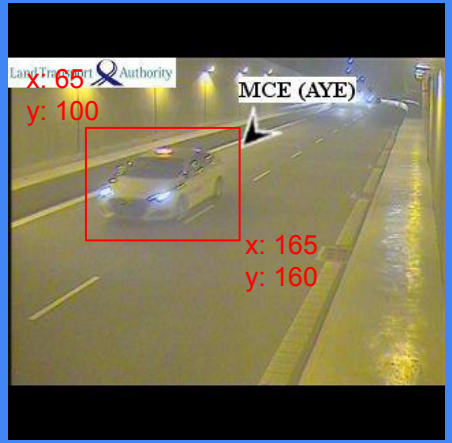
Padded: 320 x 320

Preprocessing Step 2:

2. Labelling

Vehicle Present: 0

Vehicle Present: 1



Data Augmentation:

1. Mirroring

Before: 1000 images

After: 2000 images

Data Augmentation:

2. Random Brightness, Contrast

Original

After: +20% B, C

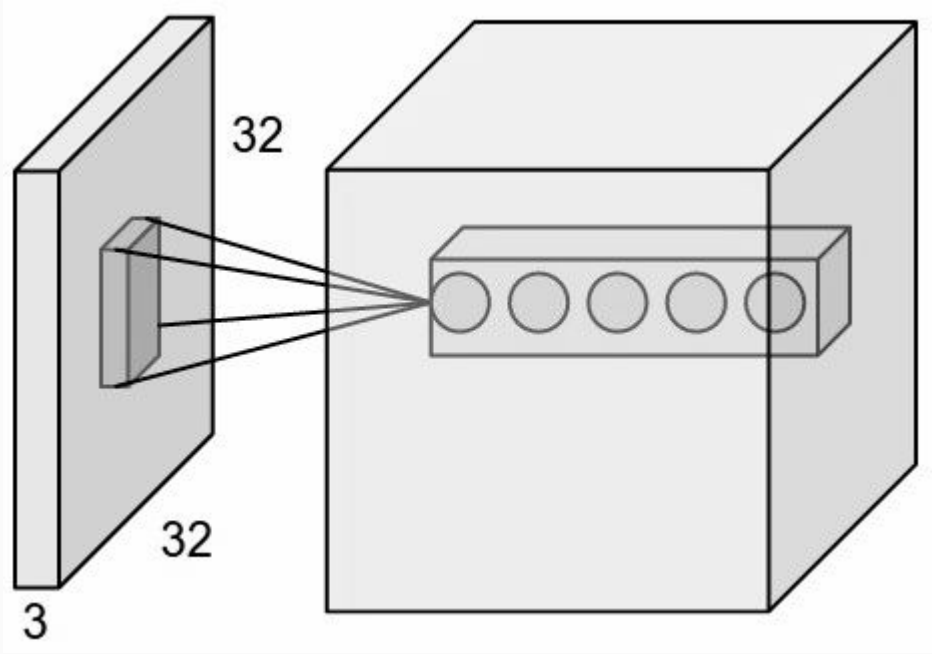
2. Basic building blocks of a CNN

Layer:

Convolution

A convolution operation involves sliding a 3D filter of weights across a 3D (RGB) input image.

Each “pixel” of the output represents the dot-product of input window and the filter weights.

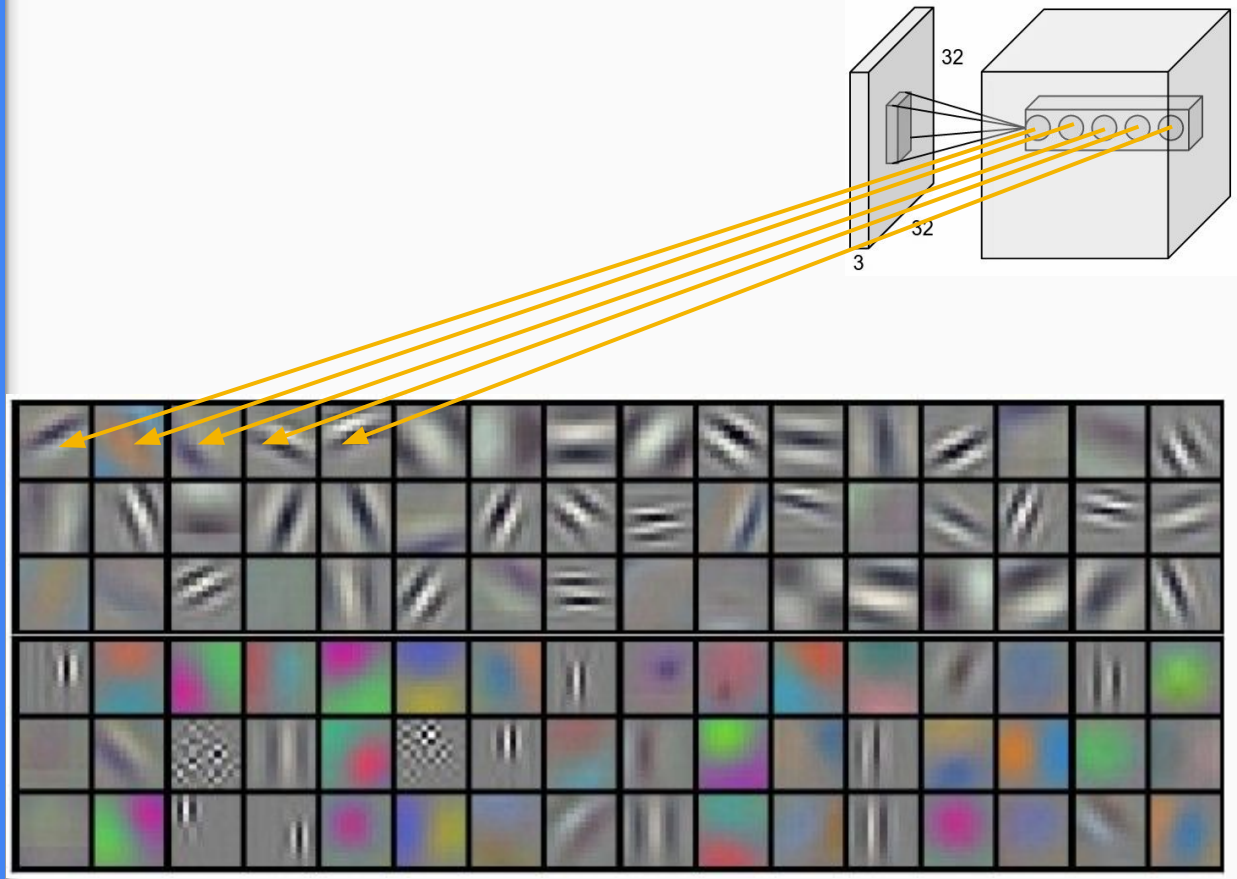


Example: 3 input channels maps to 5 output channels

Layer:

Convolution

Convolutions serve as feature extractors, detecting edges, lines, colors at the lower levels and more complex shapes at the higher levels.

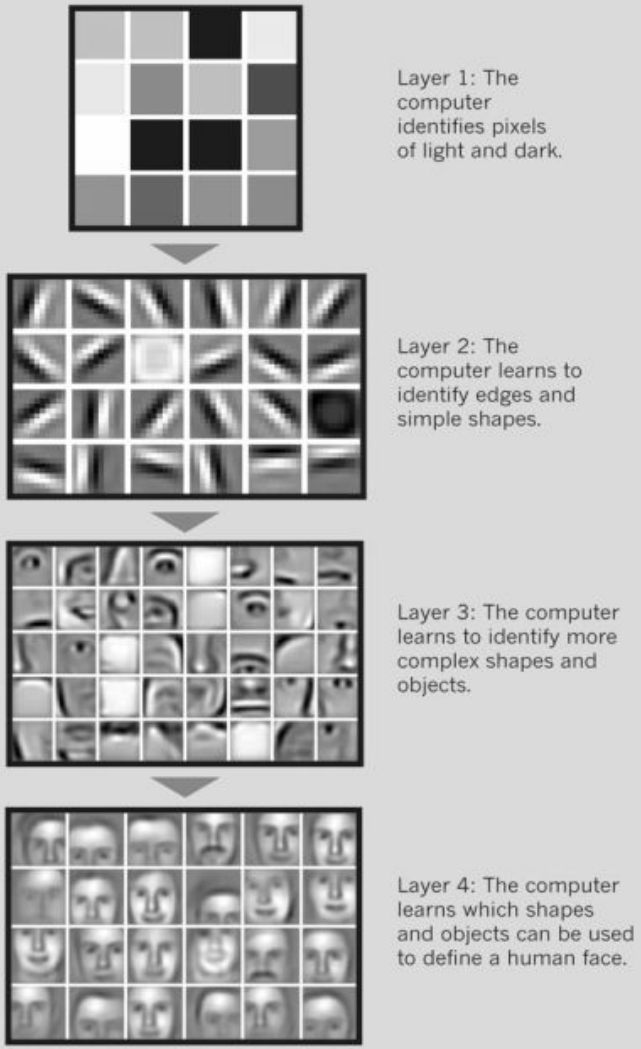


Example of weights in the first convolutional layer of a CNN

Layer:

Convolution

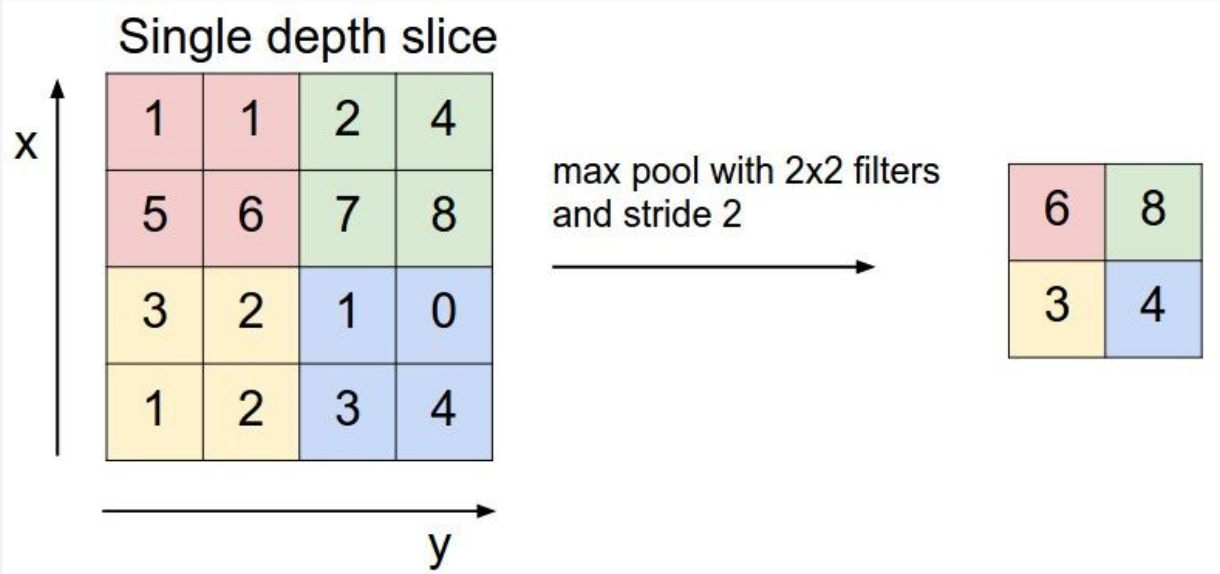
Convolutions serve as feature extractors, detecting edges, lines, colors at the lower levels and more complex shapes at the higher levels.



Layer:

Pooling

A max-pooling operation involves sliding a 2D filter window over each input channel, and retaining only the max-value at each stride.



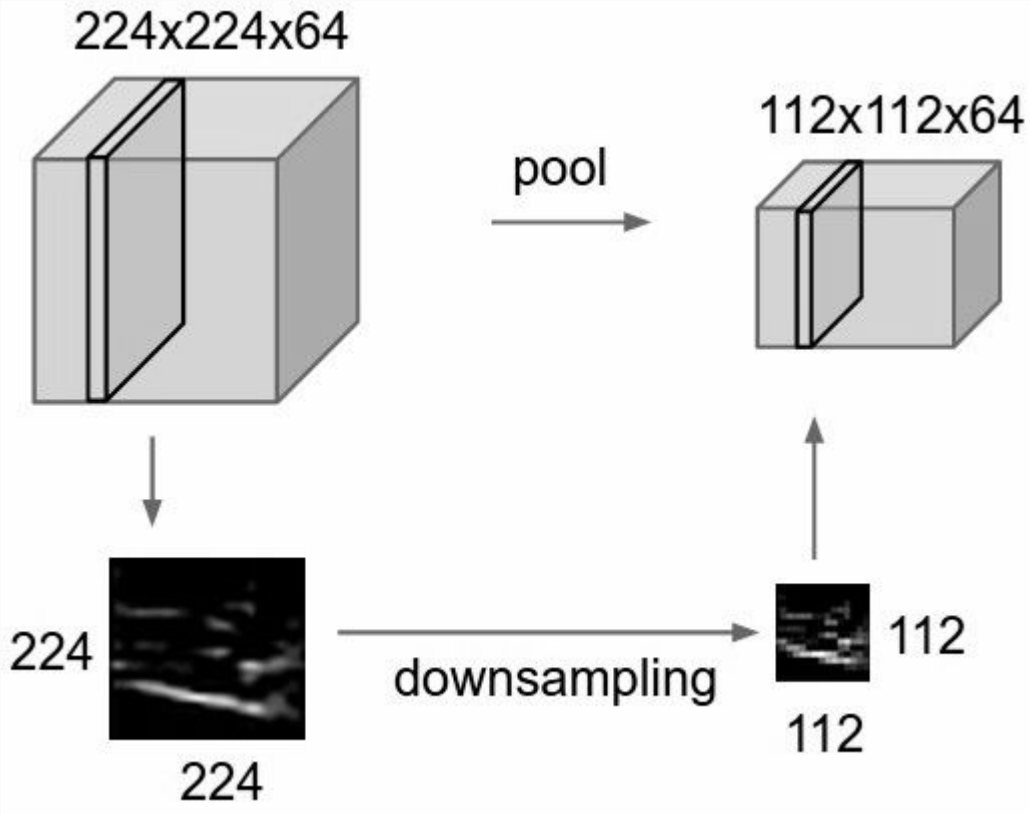
Example: a Max-Pooling operation

Layer:

Pooling

This has a “zoom-out” effect and reduces the spatial dimensions of the input.

Pooling downsamples the quantity of information flowing to the next layer and is used to control overfitting.



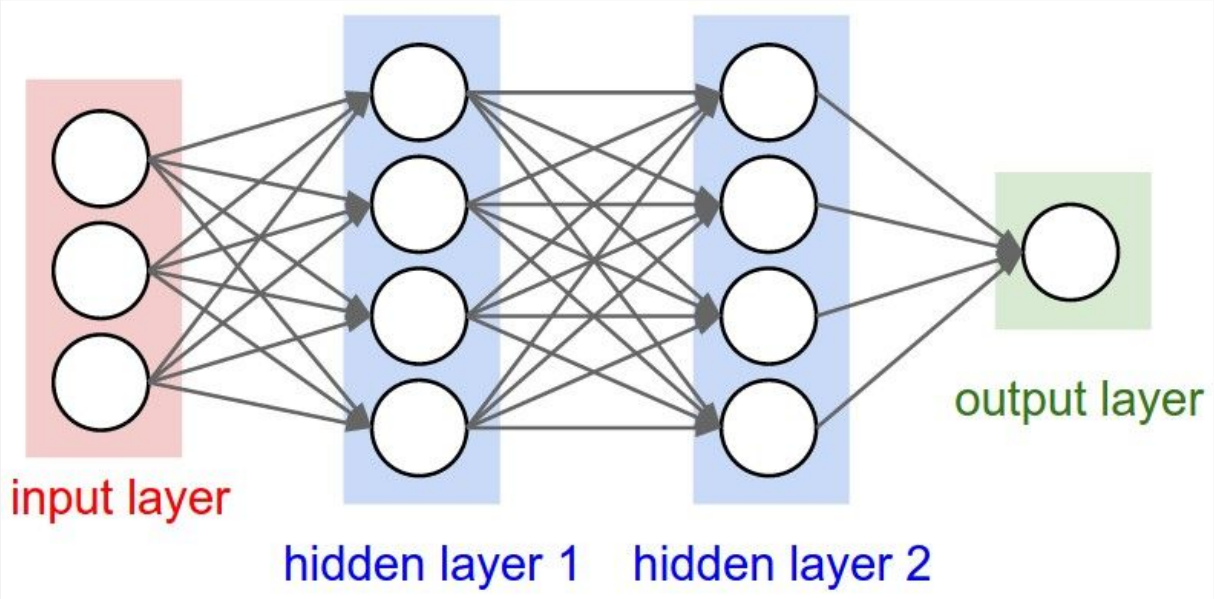
Pooling reduces the spatial dimension of the input

Layer:

Fully-Connected

In a fully-connected (FC) layer, a dot-product is computed against each input and a corresponding weight matrix.

FC layers are typically used in the final layers of a CNN, to transform extracted features (from convolutional layers) into outputs.



Neural Network with 2 fully-connected hidden layers and 1 output layer

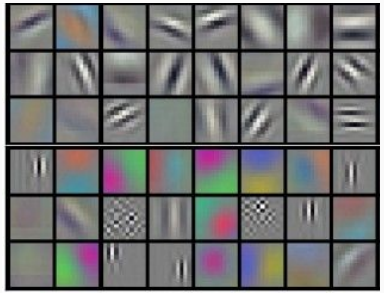
Influential CNNs in Computer Vision

LeNet-5 (1998)

- Yann LeCun
- Developed to recognize handwritten digits on bank cheques
- 5 Layers

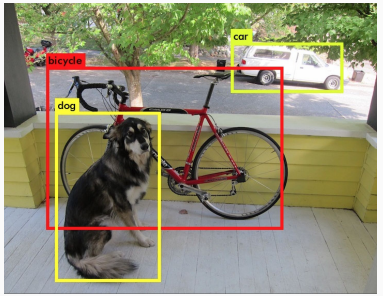
AlexNet (2012)

- Alex Krizhevsky
- Won ImageNet, first CNN to win (15.4% error, vs next best 26.2%)
- 8 Layers



VGG Net(2014)

- Visual Geometry Group, Oxford
- ImageNet 2014, top in classification and localization, 7.3% error
- 19 Layers



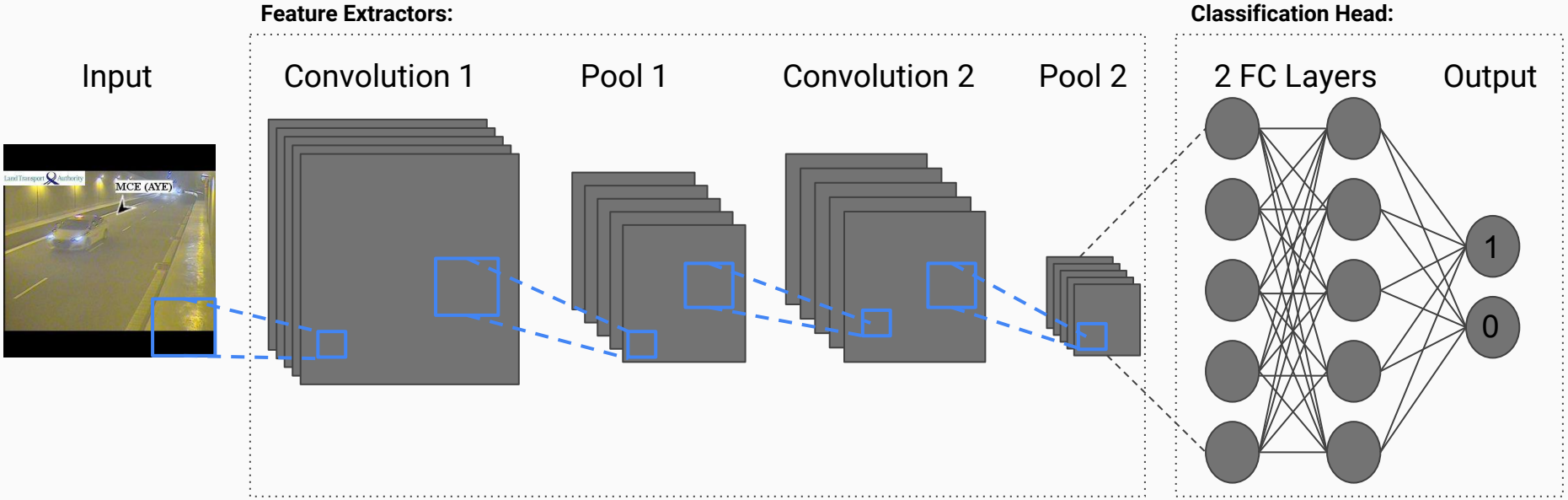
3. Classification Task

Vehicle Present: 0

Vehicle Present: 1

CNN Architecture (LeNet-5)

Classification:



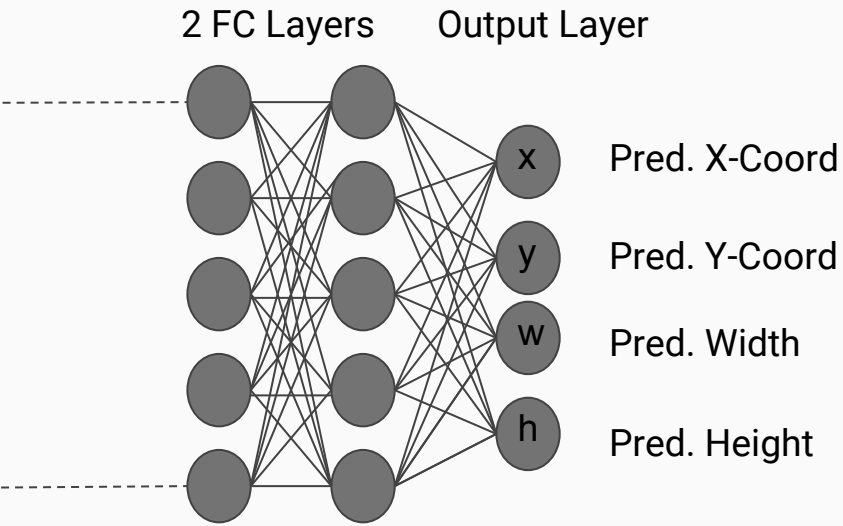
4. Classification + Localization

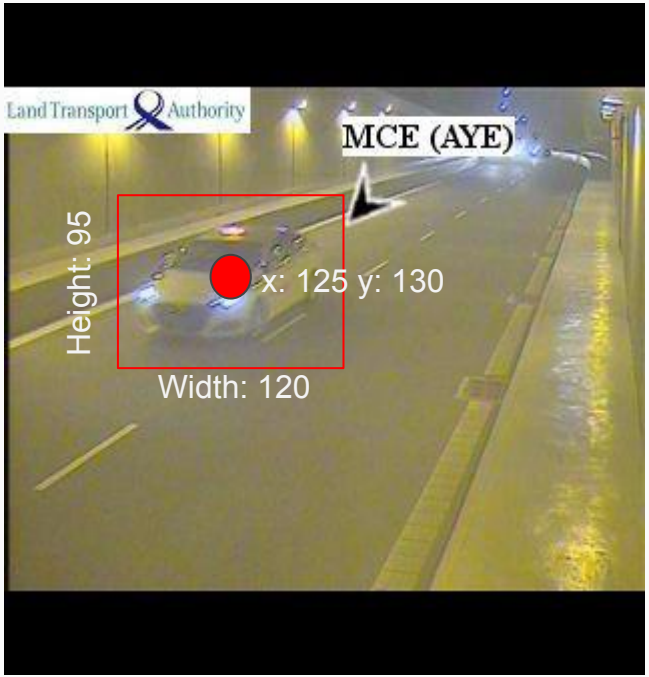
Vehicle Present: 0

Vehicle Present: 1

CNN Architecture (LeNet-5++)

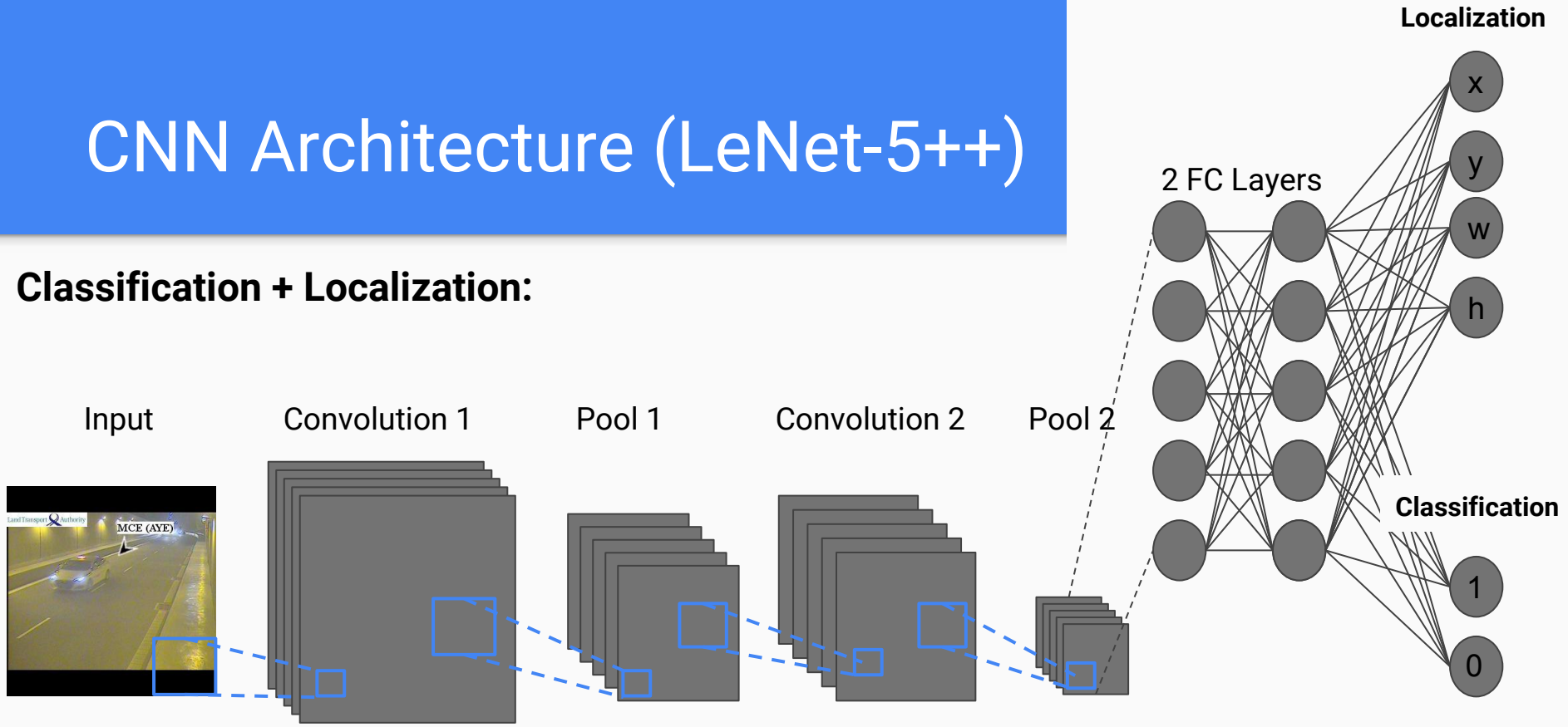
Localization Head:





CNN Architecture (LeNet-5++)

Classification + Localization:



5. iPython, Tensorboard Demo

Conclusion

This is a relatively simple model, but for the purpose of this demo, works reasonably well. In practice, use transfer learning.

Not Discussed:

- Overfitting:
  - Dropout,
  - Regularization (L1, L2)
- Vanishing/Exploding Gradients:
  - Activation Functions (RELU, tanh)
  - Batch Norm
- Initialization (Xavier)

Acknowledgements

- CS231n: Convolutional Neural Networks for Visual Recognition, http://cs231n.github.io/
- LeNet-5, convolutional neural networks, http://yann.lecun.com/exdb/lenet/
- CS1699: Introduction to Computer Vision, https://people.cs.pitt.edu/~kovashka/cs1699/
- Traffic Images API, https://developers.data.gov.sg/transport/traffic-images