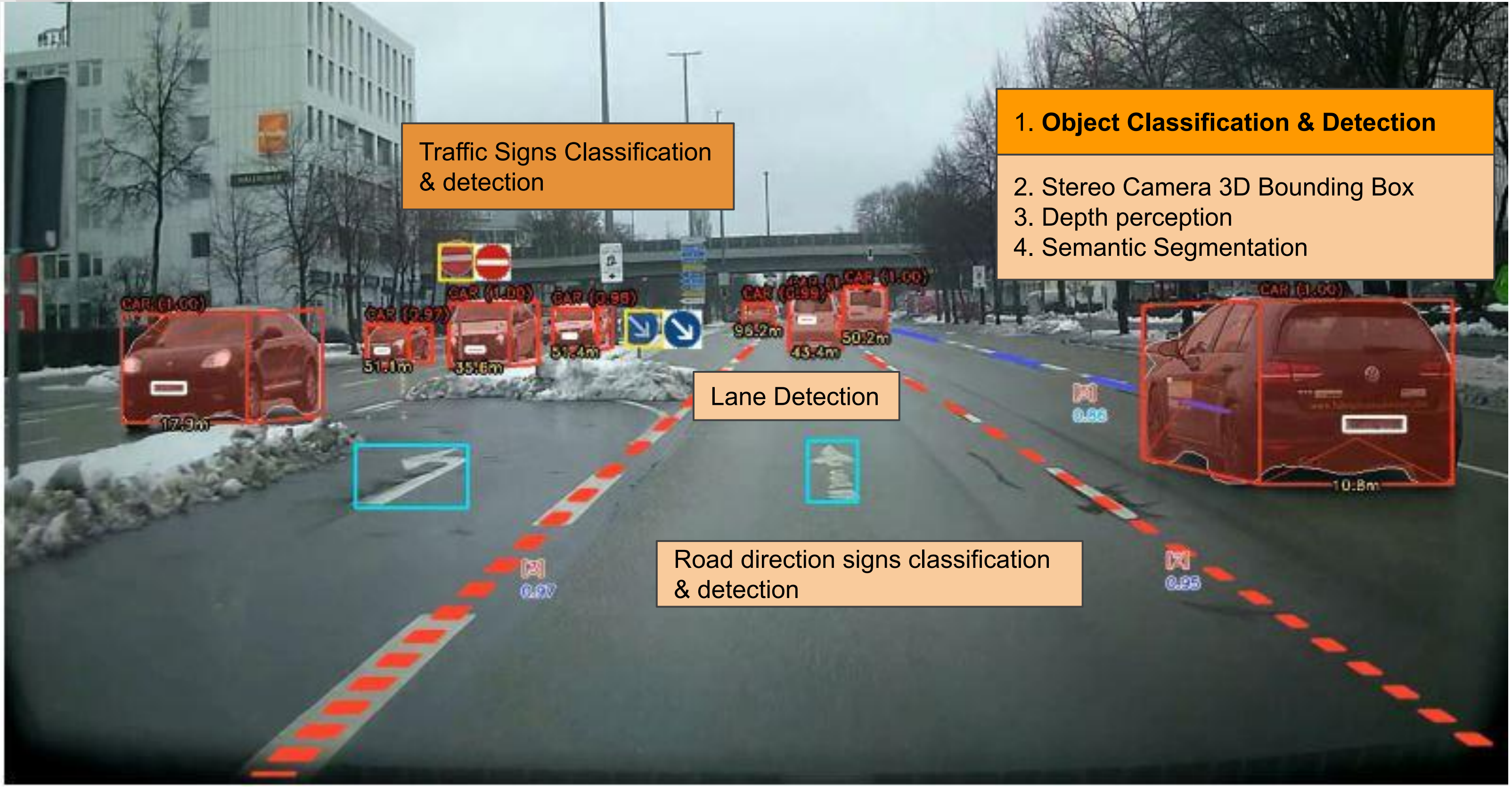
Capstone

Concepts of ADAS - Advanced Driver Assistance System

1. Traffic signs classification & Grad-CAM (Gradient-weighted Class Activation Mapping)
2. Object localization
3. OpenCV with DNN - YOLOv4 bounding box trajectory tracking

Example of ADAS [18]



Problem Statement

Accidents on the road are not acceptable. Regardless whether rates are increasing or decreasing, no lives should be lost through accidents.

With the development of Stereo cameras and LiDAR, it is now far more possible to prevent road accidents through vision systems and deep learning. Object tracking and traffic signs detection methods can potentially be expanded to detect hazards on the road, and provide audible or haptic feedback.

With the development of pre-trained models like YOLO, SSD, Faster-RCNN and frameworks like Tensorflow, object detection has been a lot more reliable than before. [1]

Scope

Scope of the project:

1. Traffic signs classification

2. Activation region visualization - GradCAM

3. Object localization

4. Object tracking and detection algorithm using trajectory tracking and deep learning models.
    a. OpenCV with DNN module - YOLOv4

Traffic Signs Classification

Data Collection - Google Street View

Data Set


0.jpg


1.jpg

2.jpg

3.jpg


4.jpg

5.jpg

6.jpg

7.jpg


8.jpg

9.jpg

10.jpg


11.jpg


12.jpg


13.jpg


14.jpg


15.jpg

16.jpg

17.jpg

18.jpg


19.jpg


20.jpg

21.jpg

22.jpg

23.jpg

24.jpg


25.jpg


26.jpg

27.jpg

28.jpg

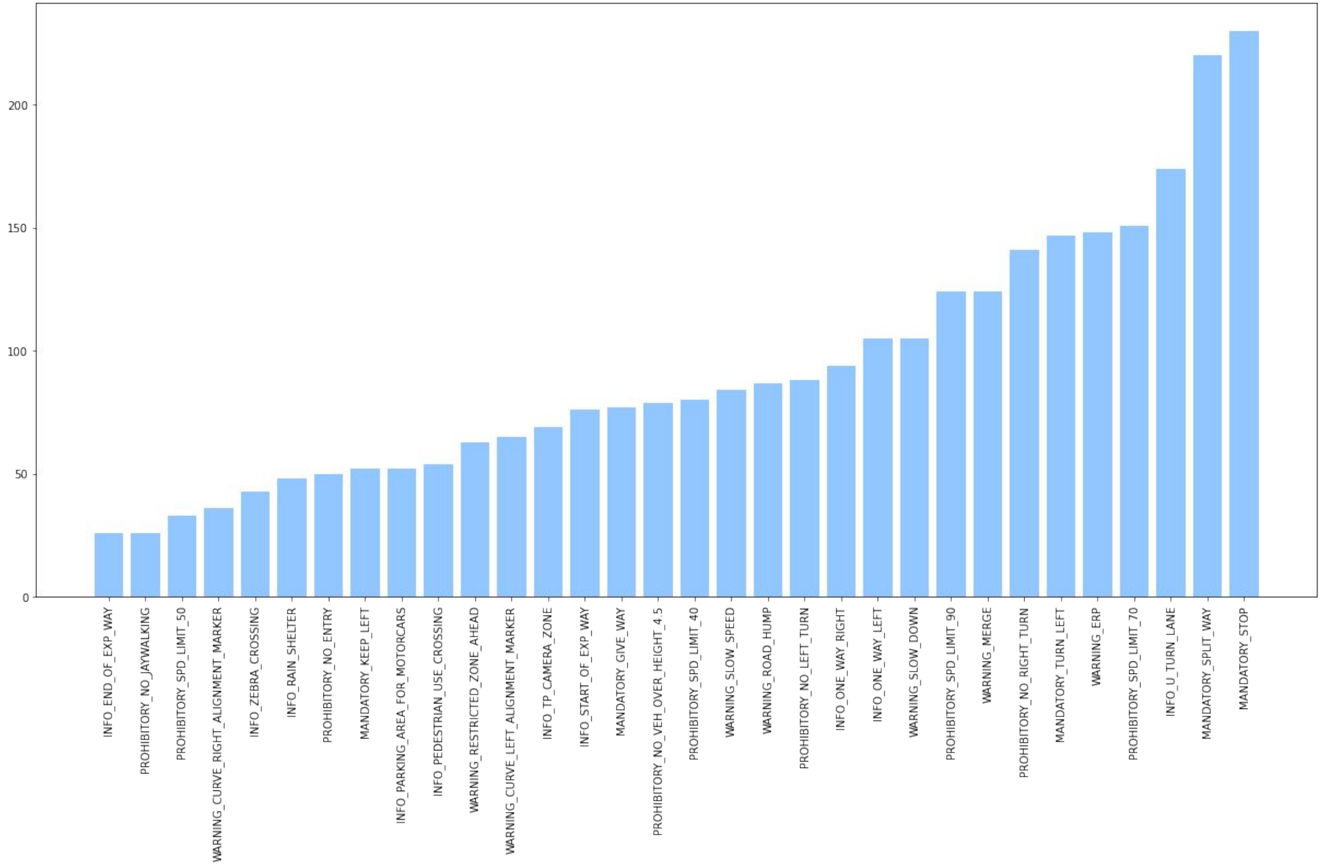
29.jpg

30.jpg


31.jpg

Data Distributions

Data Distribution



Conventional ML - XGBoost

Conventional ML - XGBoost

XGBoost was used to get an idea of how conventional ML model scores on image datasets.

Accuracy score of 0.83

	precision	recall	f1-score	support
0	0.80	0.80	0.80	15
1	0.87	0.83	0.85	71
2	1.00	0.82	0.90	11
3	0.88	0.78	0.82	45
4	0.73	0.57	0.64	14
5	0.83	0.81	0.82	31
6	0.86	0.60	0.71	10
7	0.69	0.85	0.76	26
8	0.81	0.81	0.81	48
9	0.95	0.83	0.88	23
10	0.72	0.65	0.68	20
11	0.87	0.91	0.89	22
12	0.76	0.89	0.82	56
13	0.73	0.86	0.79	22
14	0.88	0.90	0.89	39
15	0.82	0.72	0.77	25
16	0.81	0.87	0.84	30
17	0.83	0.88	0.85	43
18	0.88	0.74	0.80	19
19	0.60	1.00	0.75	9
20	0.79	0.79	0.79	14
21	0.90	1.00	0.95	18
22	0.94	0.67	0.78	43
23	0.60	0.38	0.46	8
24	0.94	0.95	0.94	76
25	0.81	0.89	0.85	28
26	1.00	0.71	0.83	7
27	0.77	0.85	0.81	27
28	0.93	0.88	0.90	16
29	0.77	0.92	0.84	39
30	0.88	0.54	0.67	13
31	0.83	0.88	0.86	17
accuracy			0.83	885
macro avg	0.83	0.80	0.80	885
weighted avg	0.84	0.83	0.83	885

https://stackoverflow.com/questions/61573260/reshape-3d-numpy-array-of-images-to-2d-numpy-array-for-xgboost-dmatrix-input

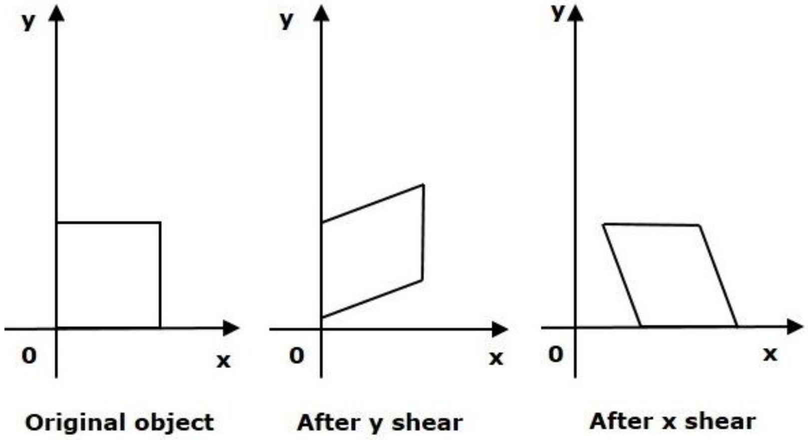
Convolutional Neural Network

Creating variations in dataset

https://www.analyticsvidhya.com/blog/2020/08/image-augmentation-on-the-fly-using-keras-imagedatagenerator/
https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator
https://stackoverflow.com/questions/57301330/what-exactly-the-shear-do-in-imagedatagenerator-of-keras

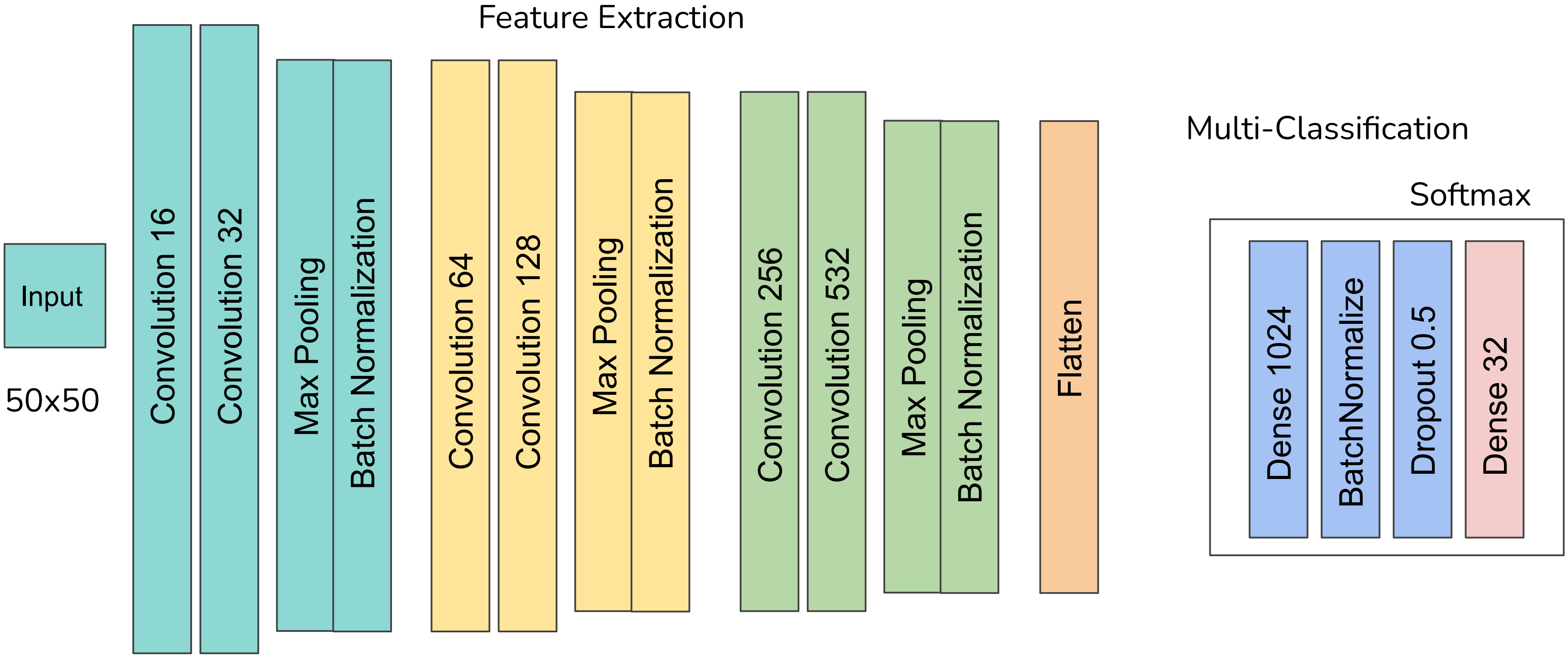
Keras Image Data Generator:

1. Image Generator was used to replace the training images with random:
   a. Shifts
   b. Rotations
   c. Random magnification
   d. Image Shear/Distortions

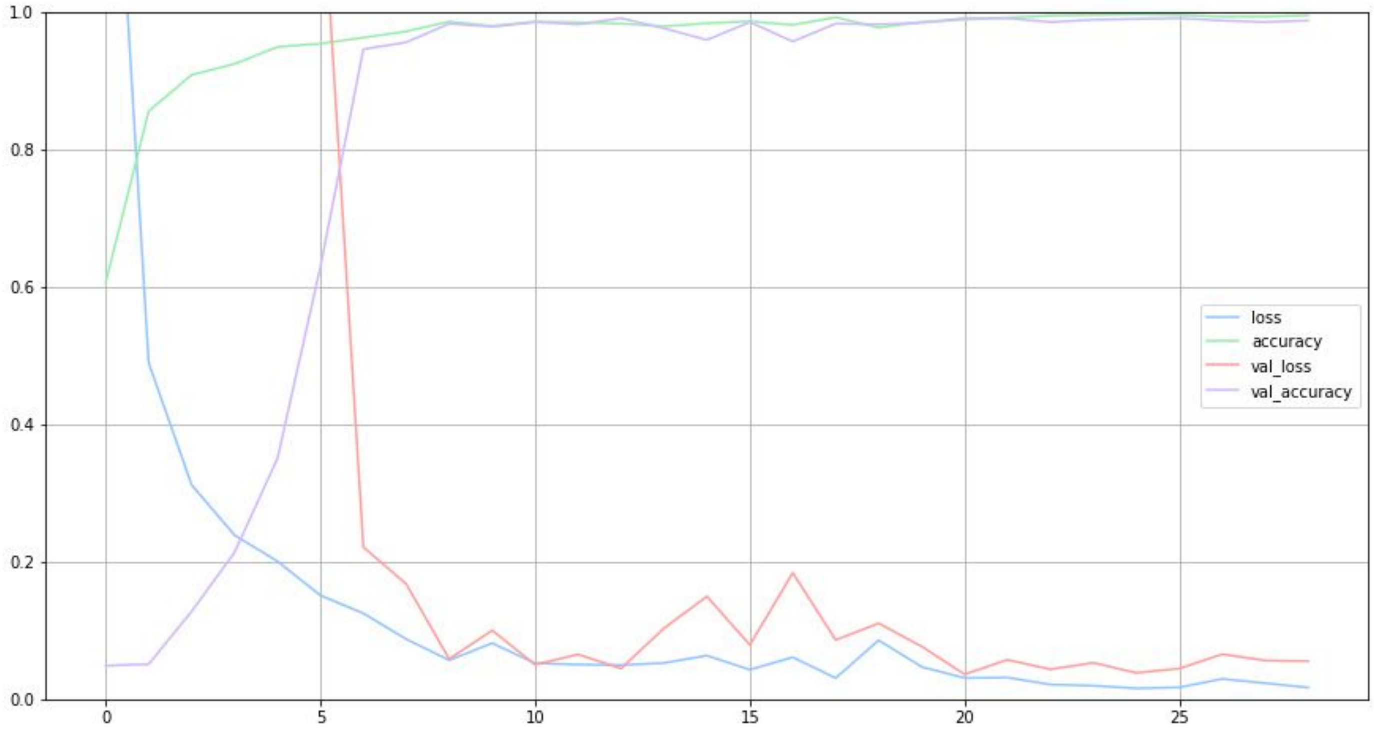


https://stackoverflow.com/questions/57301330/what-exactly-the-shear-do-in-imagedatagenerator-of-keras

Sequential


Feature Extraction
Input
50x50
Convolution 16
Convolution 32
Max Pooling
Batch Normalization
Convolution 64
Convolution 128
Max Pooling
Batch Normalization
Convolution 256
Convolution 532
Max Pooling
Batch Normalization
Flatten
Multi-Classification
Softmax
Dense 1024
BatchNormalize
Dropout 0.5
Dense 32

Training History



Keras - Results

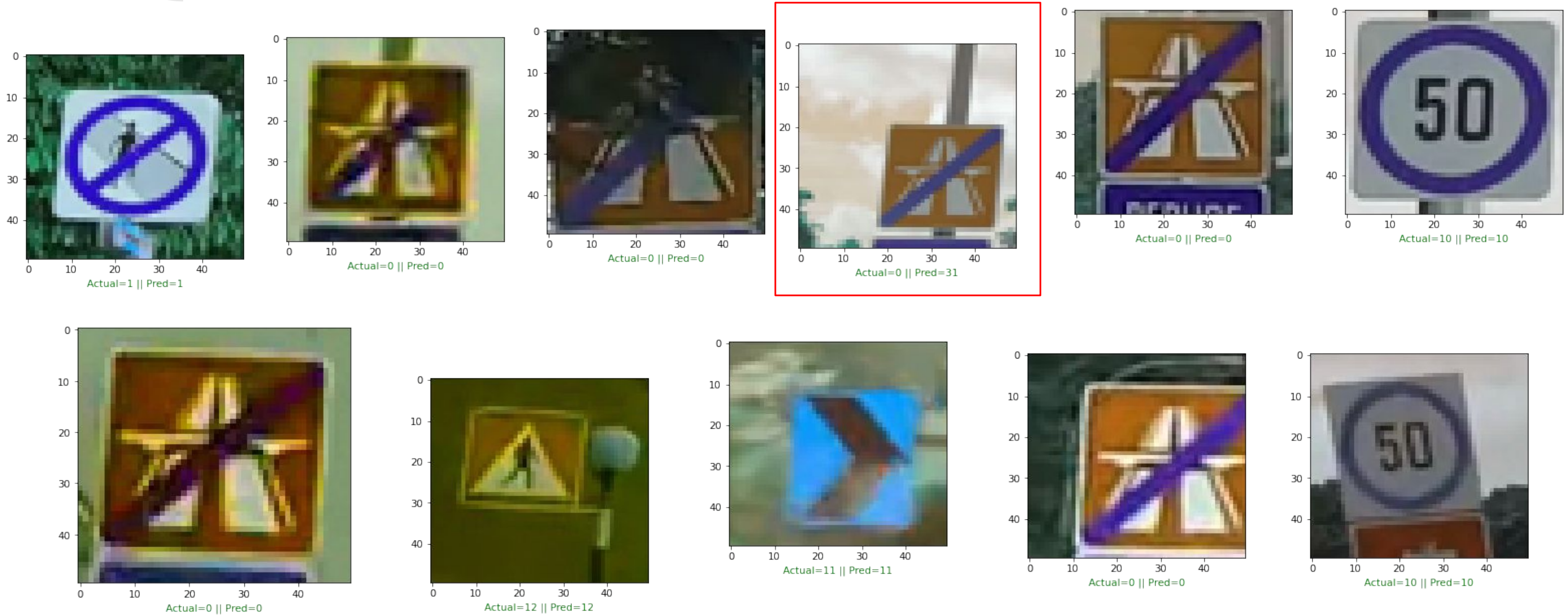
Model is fairly accurate in predicting traffic signs:

Accuracy:
0.96

	precision	recall	f1-score	support
0	0.90	0.90	0.90	10
1	0.85	1.00	0.92	11
2	1.00	1.00	1.00	8
3	0.85	0.92	0.88	12
4	1.00	1.00	1.00	8
5	1.00	0.90	0.95	10
6	1.00	1.00	1.00	10
7	1.00	1.00	1.00	11
8	1.00	1.00	1.00	14
9	1.00	0.80	0.89	15
10	1.00	1.00	1.00	13
11	1.00	1.00	1.00	12
12	0.81	1.00	0.90	13
13	1.00	1.00	1.00	10
14	1.00	1.00	1.00	14
15	1.00	0.92	0.96	12
16	1.00	1.00	1.00	11
17	0.92	1.00	0.96	12
18	1.00	1.00	1.00	10
19	1.00	0.90	0.95	10
20	0.89	0.73	0.80	11
21	1.00	0.91	0.95	11
22	0.87	1.00	0.93	13
23	1.00	1.00	1.00	7
24	1.00	1.00	1.00	13
25	1.00	0.90	0.95	10
26	1.00	1.00	1.00	6
27	1.00	1.00	1.00	9
28	1.00	1.00	1.00	11
29	1.00	1.00	1.00	12
30	0.91	0.91	0.91	11
31	0.92	1.00	0.96	12
accuracy			0.96	352
macro avg	0.97	0.96	0.96	352
weighted avg	0.96	0.96	0.96	352

Classification Results

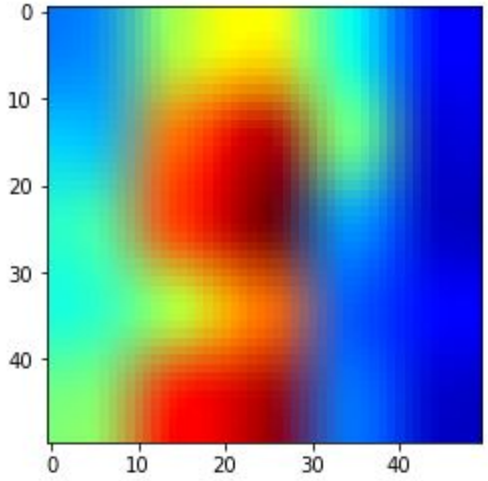
Image Classification



Interpretation & Failure Analysis

https://github.com/kolesman/SEC
Seed, Expand, Constraint, paper for weakly supervised image segmentation

Image Interpretation - Gradcam

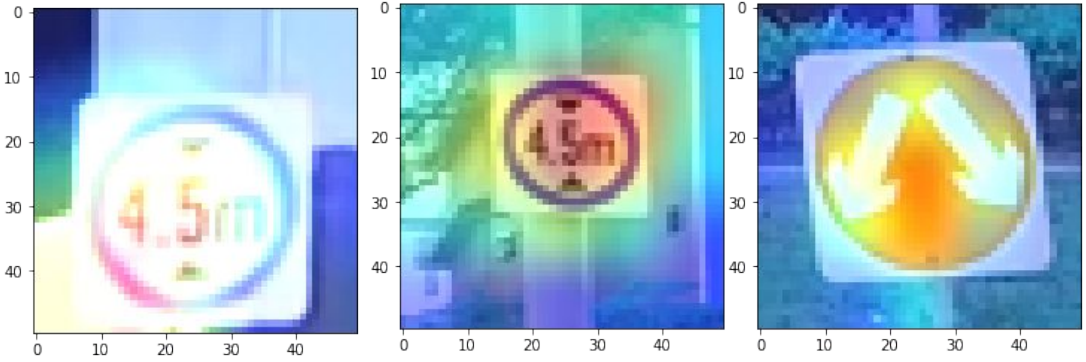


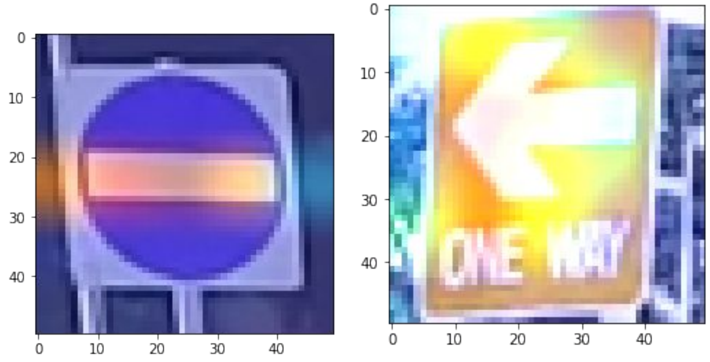
GradCAM's activation heatmap

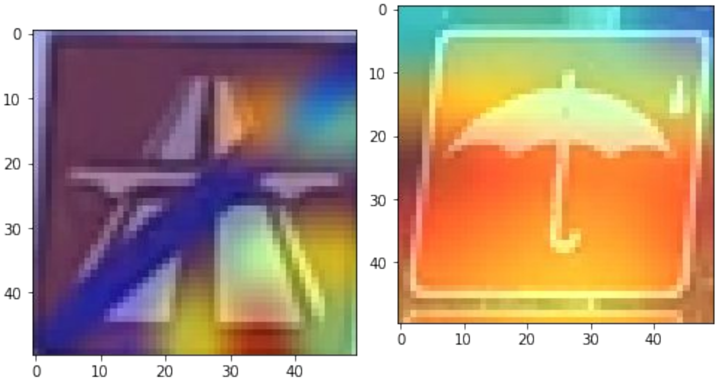
Image Interpretation - Gradcam

Activation Heatmap

Correct classification images



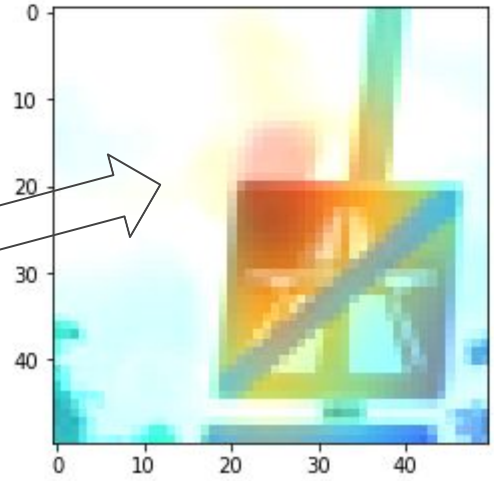


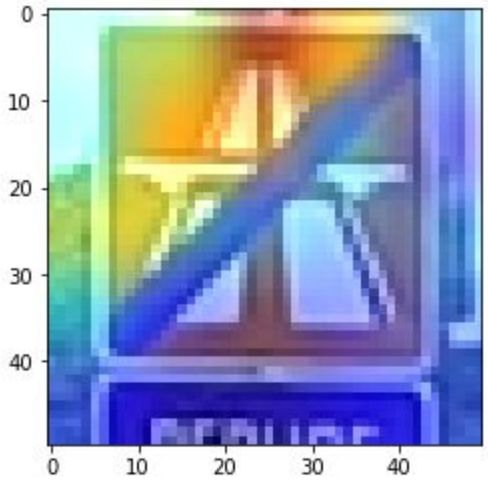


Analyze failed classification for improvements

Wrongly classified images

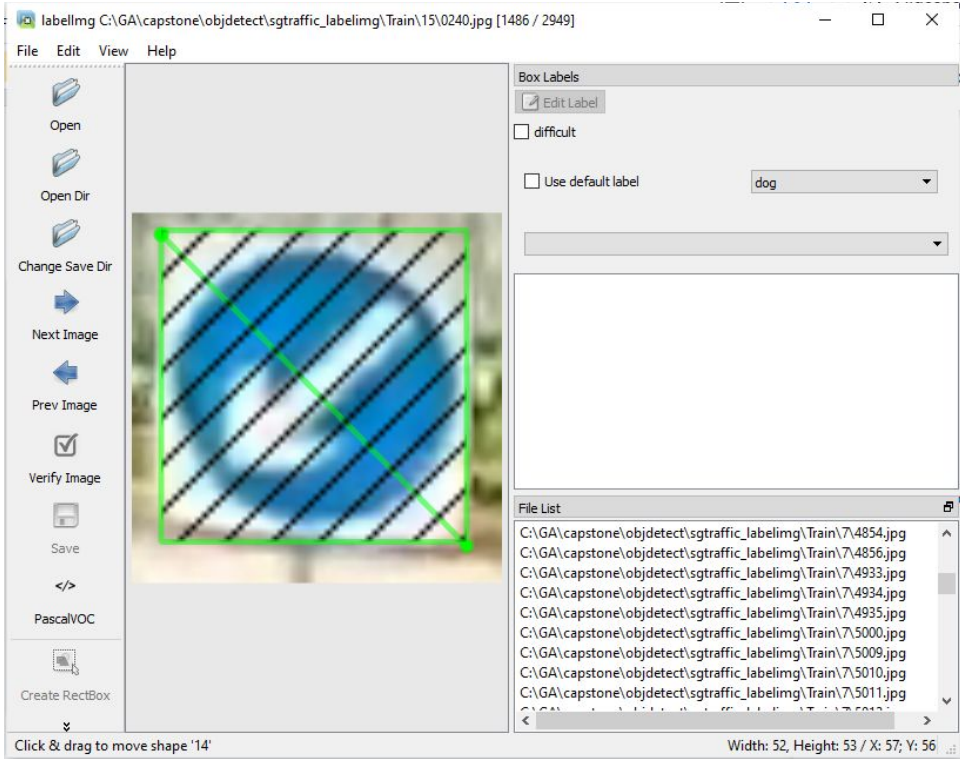
Wrong region activated





Object Localization

LabelImg



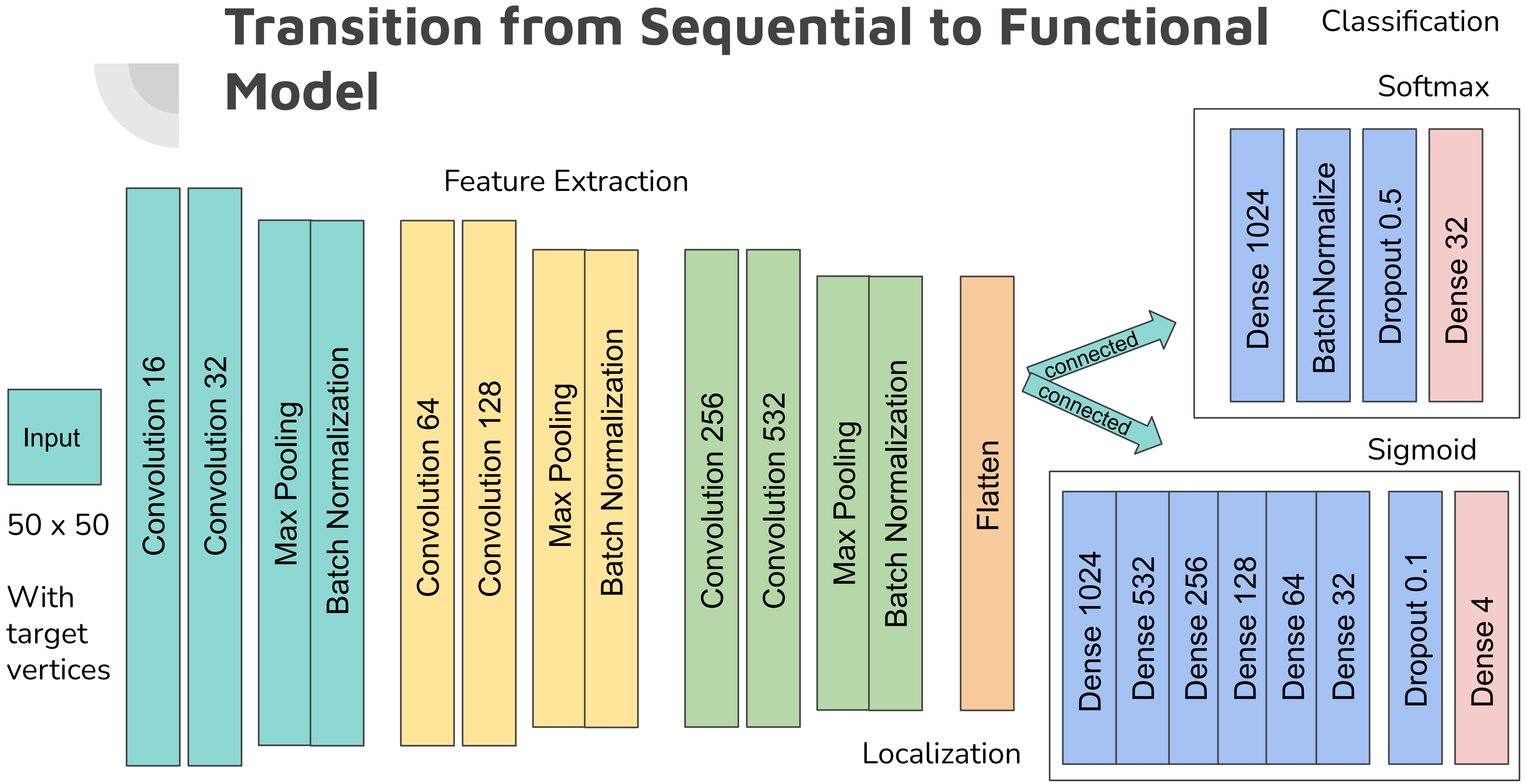
Hand Labeling images in PascalVOC format to generate XML files.

Contains:

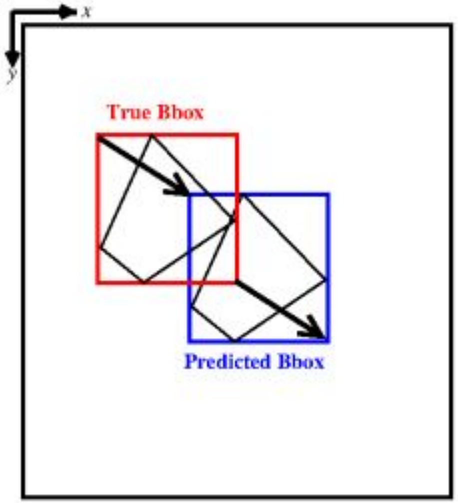
1. Bounding box coordinates
2. Image class
3. Path
4. Image size

Model Architecture

Transition from Sequential to Functional Model



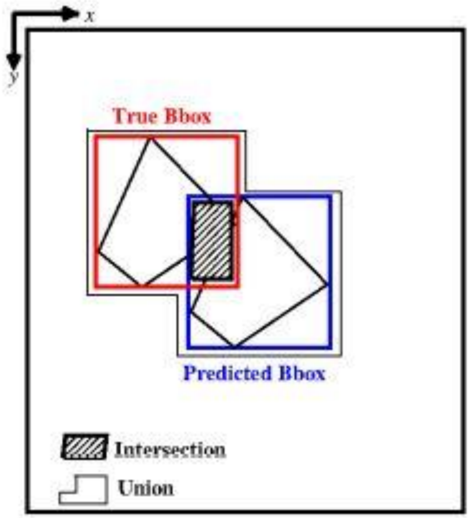
Loss for the model - MSE



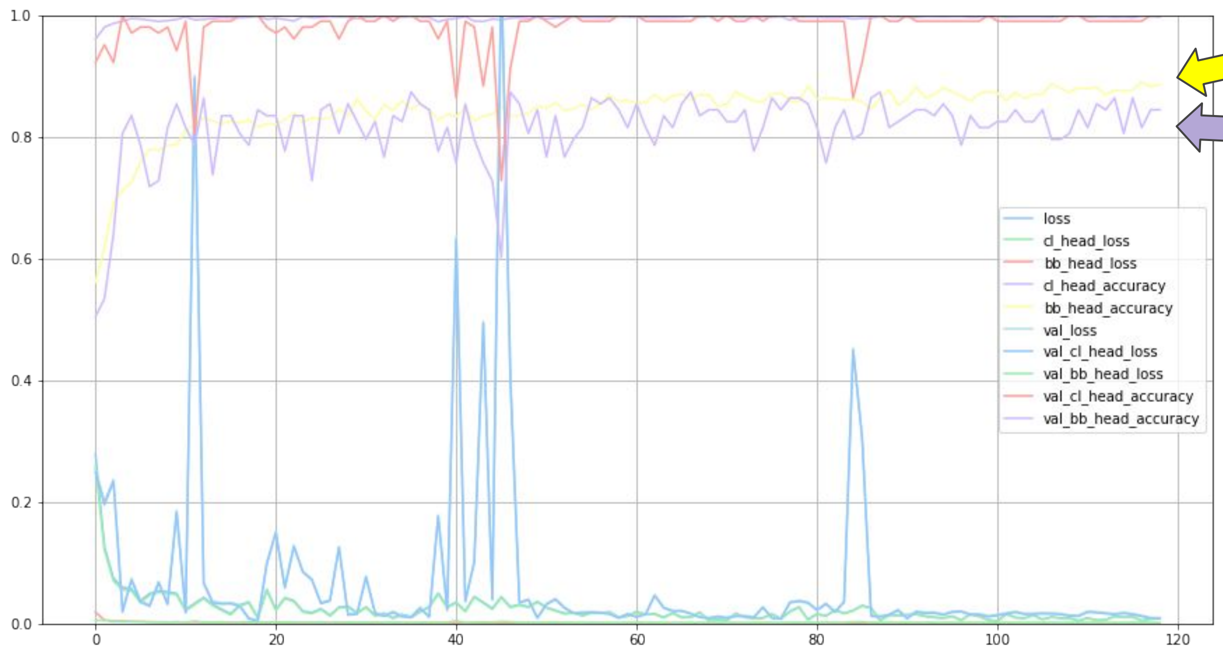
MSE Loss

Localization Metric - IoU

IoU - Intersection over Union



Training History
Classification & Bounding Box Regression



Bounding Box Accuracy

Classification Accuracy

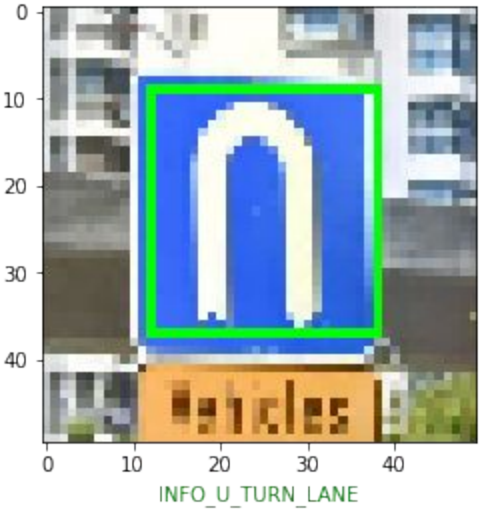
Sharing base layers
Training for 2 branches

200 Epochs with Early Stop callbacks.

Classify & Localize

Blue - Groundtruth
Green - Predicted

Classify & Localize - Blind test



Blind test results

Image was taken randomly from Google Street View.

Model managed to classify and localize the street sign.

Metric - Fast R-CNN Architecture

Classification Accuracy score for 106 photos achieved:

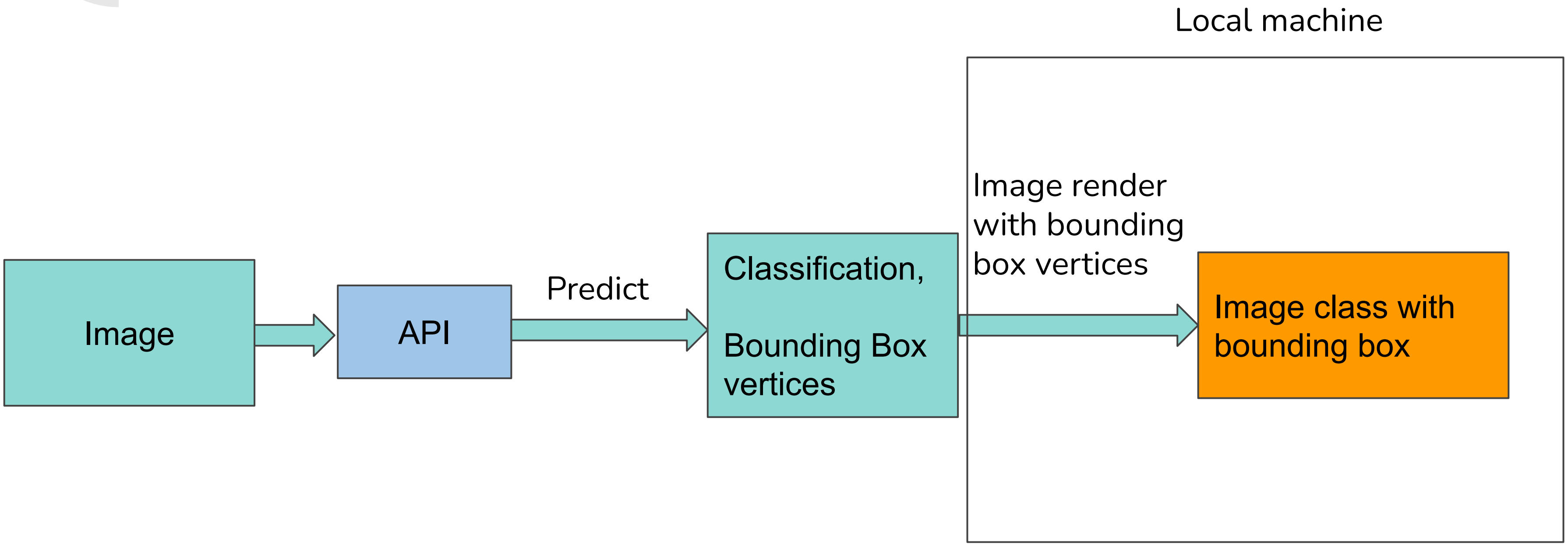
0.97

IoU Accuracy score for 106 photos achieved:

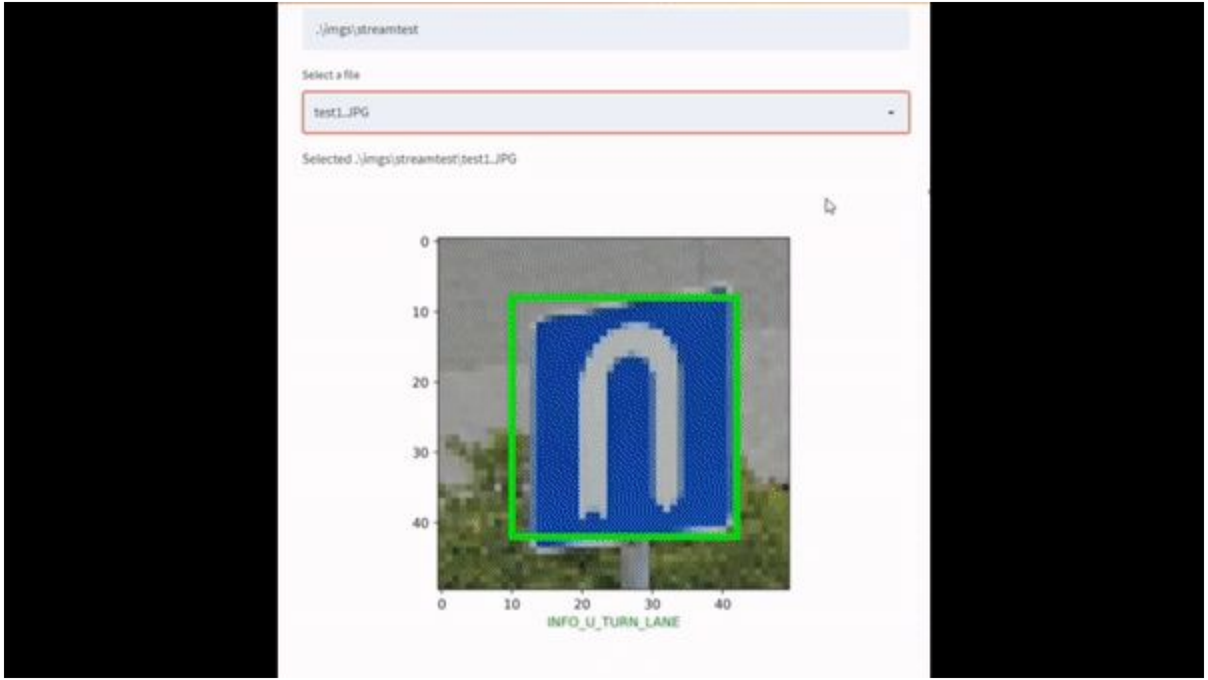
0.88

Building API

Structure of API



Deployment to Streamlit



Basic Object Tracking - OpenCV with YOLOv4 weights

Trajectory Tracking

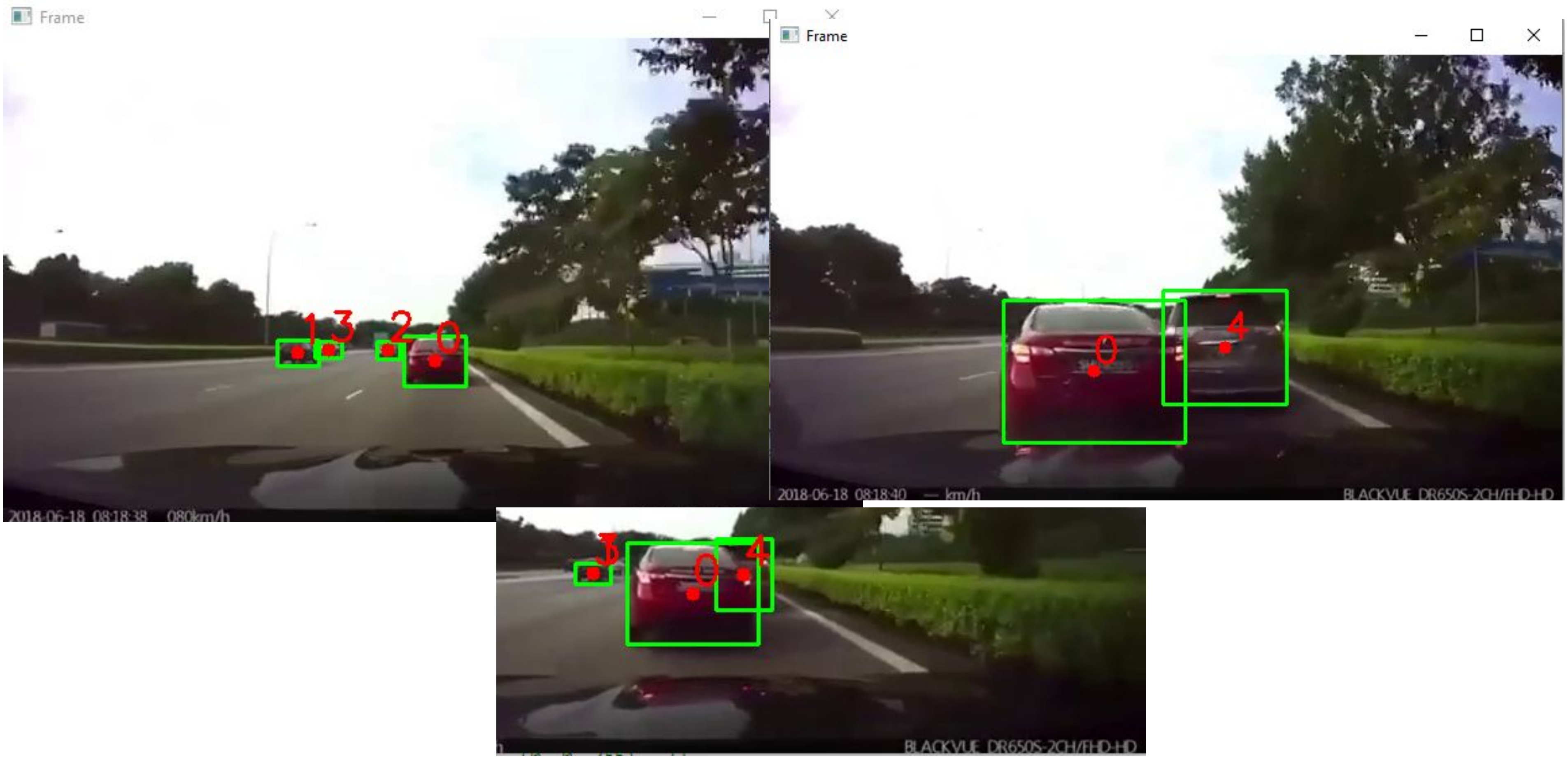
Object Trajectory Tracking using Rule Based CV

By using the bounding box generated by YOLOv4:

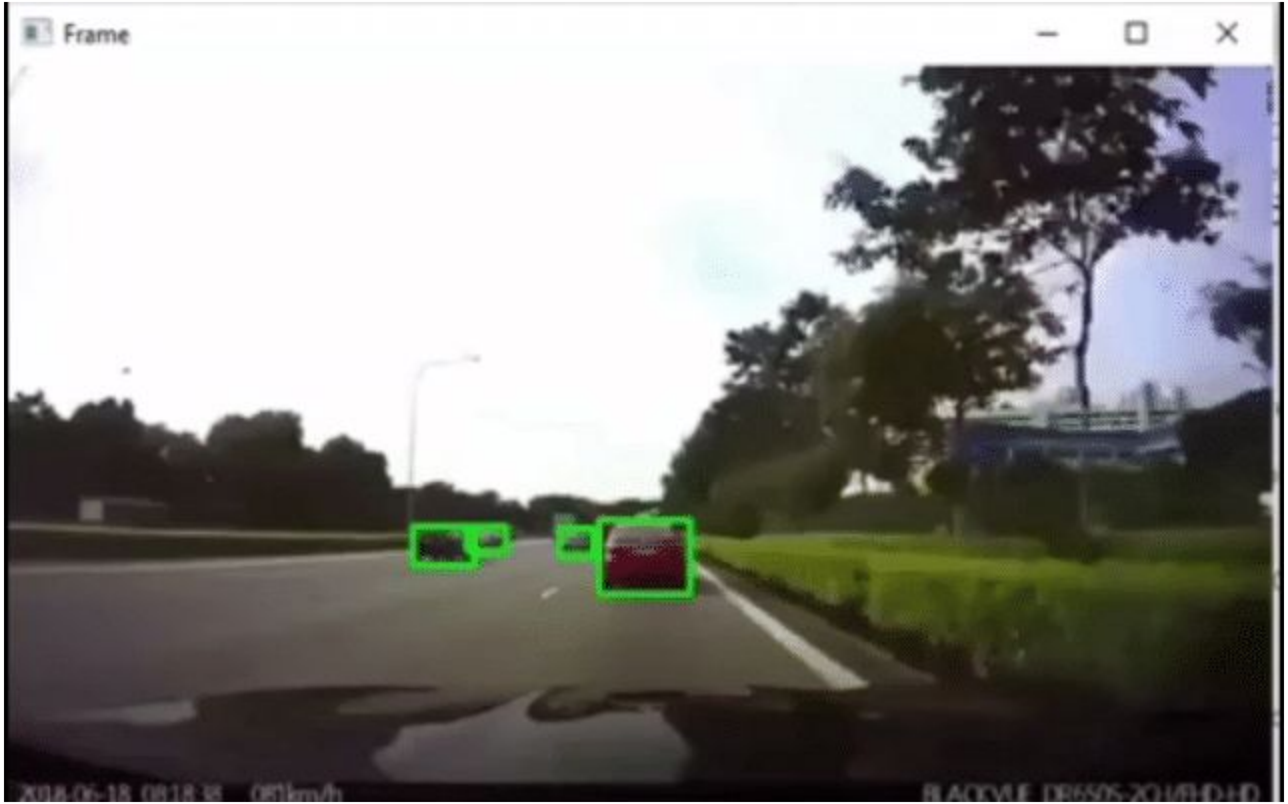
1. By tracking the center of the bounding box, the trajectory of the bounding box can be tracked frame by frame.

2. Trajectory tracking was used to track the object if previous frame and current frame does not exceed 30 pixels.

OpenCV with DNN module - YoloV4 weights



OpenCV with DNN module - YoloV4 weights



Conclusion

In conclusion, from this project, it can be seen that the objectives intended were achieved.

The model was built without using transfer learning from pretrained models. And this solidified the foundation of understanding the architecture of Fast R-CNN models.

Object tracking was successful and managed to track objects even as it moves to the next lane.

Future Works

1. Lane keeping
2. Image Segmentation with Mask R-CNN
3. Stereo Camera Depth Detection
4. LiDAR point clouds using DBSCAN combined with camera vision system for accurate object detection & tracking

Potential developments for this project includes, cameras and LiDAR that can be merged with precise alignment to become what is known as a fusion sensor so that the rate of false positives & false negatives can be minimized.

Combined with LiDAR point clouds and ML methods like DBSCAN, the accuracy of object detection and tracking has been better than ever, even in bad weather conditions.

Deployment would be useful if it can be deployed on embedded systems eg. FPGA in the automotive industry and can be researched further. [d]

Thank you!

Sources

Sources:

[1]
https://traveltips.usatoday.com/air-travel-safer-car-travel-1581.html#:~:text=In%20absolute%20numbers%2C%20driving%20is,air%20travel%20to%20be%20safer.

[2] https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

[3] https://towardsdatascience.com/region-of-interest-pooling-f7c637f409af

[4] https://towardsdatascience.com/with-keras-functional-api-your-imagination-is-the-limit-4f4fae58d90b

[5] https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c

[6] https://medium.com/analytics-vidhya/train-a-custom-yolov4-object-detector-using-google-colab-61a659d4868

[7] https://traveltips.usatoday.com/air-travel-safer-car-travel-1581.html

[8] https://medium.com/analytics-vidhya/iou-intersection-over-union-705a39e7acef

[9] https://medium.datadriveninvestor.com/2-layers-to-greatly-improve-keras-cnn-1d4d1c3e8ea5

Sources:

[10] https://towardsdatascience.com/the-vanishing-exploding-gradient-problem-in-deep-neural-networks-191358470c11?gi=27e999e4010f

[11] https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax

[12] https://towardsdatascience.com/understand-your-algorithm-with-grad-cam-d3b62fce353#:~:text=Gradient%2Dweighted%20Class%20Activation%20Mapping,regions%20in%20the%20image%20for

[13] https://arxiv.org/pdf/1610.02391.pdf

[14] https://towardsdatascience.com/fast-r-cnn-for-object-detection-a-technical-summary-a0ff94faa022

[15] https://github.com/tzutalin/labelImg

[16] https://towardsdatascience.com/what-is-the-difference-between-object-detection-and-image-segmentation-ee746a935cc1

Sources:

[17]
https://www.researchgate.net/figure/Object-tracking-based-on-image-segmentation-and-similar-object-feature-matching_fig1_4222458

[18]
https://m.futurecar.com/4632/Computer-Vision-Developer-StradVision-to-Showcase-its-Most-Advanced-Perception-Camera-for-Autonomous-Driving-&-ADAS-at-Auto-Tech-2021

[a] Research papers and materials: With great thanks to the following authors for sharing their research papers and materials on the topic of object classification and localization.

[b] Object Detection and Localization with Deep Networks, Avi Kak and Charles Bouman, Purdue University

[c] Universal Bounding Box Regression and Its Applications, Seungkwan Lee, Suha Kwak and Minsu Cho, Dept. of Computer Science and Engineering, POSTECH Korea

[d] Intelligent Vision Systems & Embedded Deep Learning Technology for ADAS, Jiun-In Guo, National Yang Ming Chiao Tung University

Sources:

[e] Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization, Ramprasaath R. Selvaraju · Michael Cogswell · Abhishek Das · Ramakrishna Vedantam · Devi Parikh · Dhruv Batra

[f] Stereo RCNN based 3D Object Detection for Autonomous Driving, https://github.com/srinu6/Stereo-3D-Object-Detection-for-Autonomous-Driving

Sequential Model

Keras - Custom Convolution Layers

2 Important layers were used:

1. Batch normalization - Reduce overfitting
2. Dropout - Weight independent, due to randomly dropping weights
3. Softmax Activation for multi-class classification (32 classes)

There were a total of 3,707,136 trainable params

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 48, 48, 16)        448

 conv2d_1 (Conv2D)           (None, 46, 46, 32)        4640

 max_pooling2d (MaxPooling2D  (None, 23, 23, 32)       0
 )

 batch_normalization (BatchN  (None, 23, 23, 32)       128
 ormalization)

 conv2d_2 (Conv2D)           (None, 21, 21, 64)        18496

 conv2d_3 (Conv2D)           (None, 19, 19, 128)       73856

 max_pooling2d_1 (MaxPooling  (None, 9, 9, 128)        0
 2D)

 batch_normalization_1 (Batc  (None, 9, 9, 128)        512
 hNormalization)

 conv2d_4 (Conv2D)           (None, 7, 7, 256)         295168

 conv2d_5 (Conv2D)           (None, 5, 5, 512)         1180160

 max_pooling2d_2 (MaxPooling  (None, 2, 2, 512)        0
 2D)

 batch_normalization_2 (Batc  (None, 2, 2, 512)        2048
 hNormalization)

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 1024)              2098176

 batch_normalization_3 (Batc  (None, 1024)             4096
 hNormalization)

 dropout (Dropout)           (None, 1024)              0

 dense_1 (Dense)             (None, 32)                32800

=================================================================
Total params: 3,710,528
Trainable params: 3,707,136
Non-trainable params: 3,392
_________________________________________________________________
```

https://medium.datadriveninvestor.com/2-layers-to-greatly-improve-keras-cnn-1d4d1c3e8ea5
https://machinelearningmastery.com/how-to-accelerate-learning-of-deep-neural-networks-with-batch-normalization/

Modeling

https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping

https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1

Using the following parameters with Adam:

Learning rate = 0.001

Epochs = 50

Callbacks = EarlyStopping, patience = 8, restore_best_weights

AMSGrad = True

Methods tested but got poor results:

1. Convolution layer was increased to 3 as 2 convolution layers caused the validation accuracy to diverge mid training
2. Learning rate was optimized to 0.001 as any higher learning rate couldn't get the model to converge
3. Exponential decay and exponential learning rate was experimented but it was too unpredictable & not repeatable
4. Adding dropout in between layers to combat overfitting turned out to be too difficult for the model to improve validation accuracy as epochs increased)