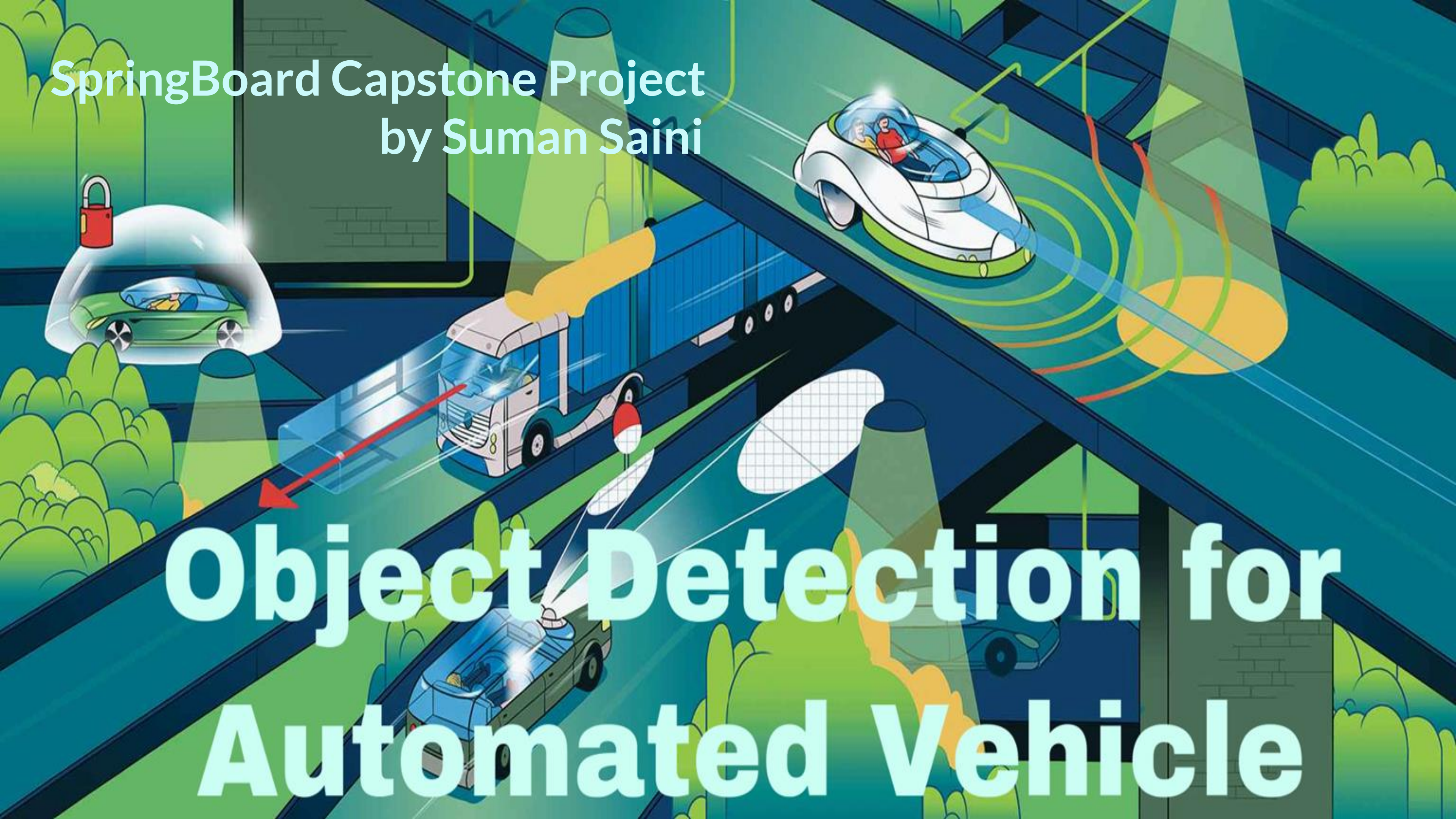


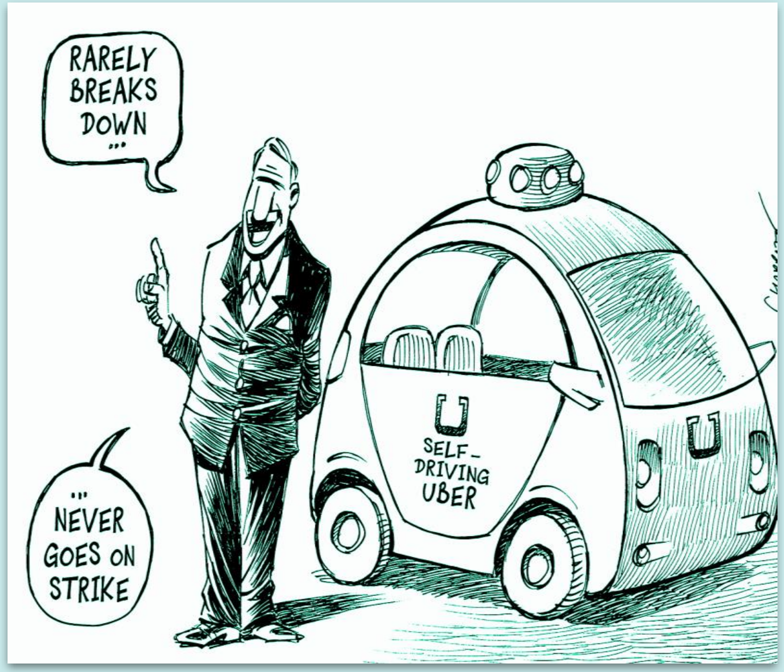
SpringBoard Capstone Project

by Suman Saini

Object Detection for Automated Vehicle

Project Aim

- ❑ Many automobile companies are currently working on manufacturing fully automated versions of cars. Automated vehicles offer exciting opportunities for transit providers and transit users and are expected to redefine transportation.
- ❑ This project aims to build and optimize algorithms based on a large-scale dataset and focus on hard problems in this perception system i.e. object detection. Object detection is a crucial task for autonomous driving.



Data Loading

The data for this project is fetched from Audi driving dataset which can be retrieved from this link.

- ❑ **Image**

  The overall image data containing 11,754 frames is split into 8227 frames as train set, 1176 frames for test set and 2351 frames for validation set.

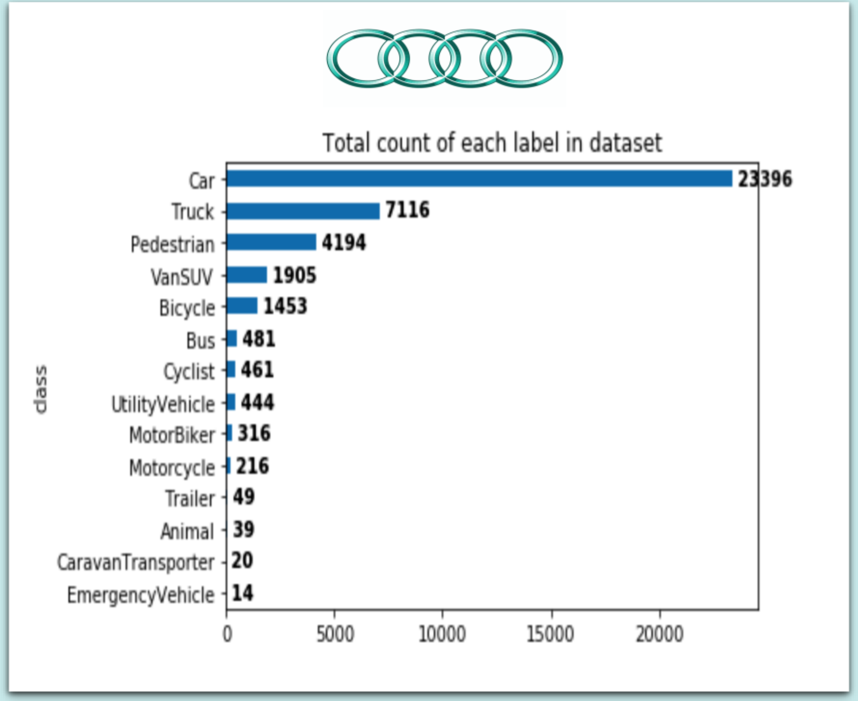
- ❑ **Bounding boxes**

  The json file in the dataset has the '2d_bbox' details which gives us the object annotations i.e. bounding boxes.

  It also has corresponding 'class' labels for each image.

How to handle imbalance in class count?

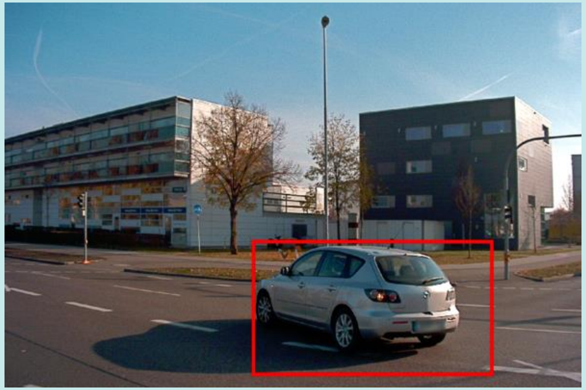
Data Augmentation

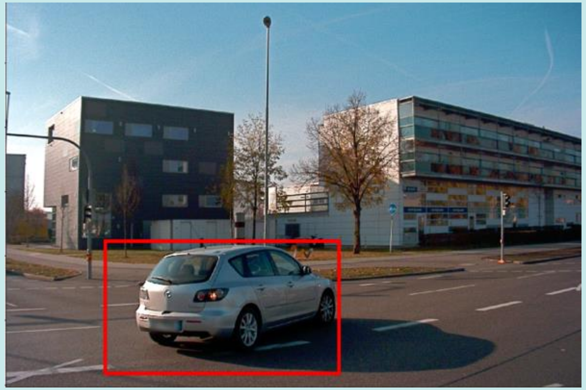


Data Augmentation

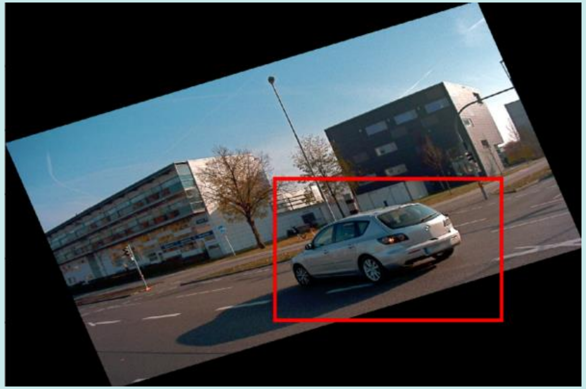
- ❑ Data augmentation is a way to artificially expand the dataset and thinking of it as added noise to the dataset.
- ❑ The difference in the images can be seen as noise being added to the data sample each time, and this noise forces the neural network to learn generalised features instead of overfitting on the dataset.

Original Image

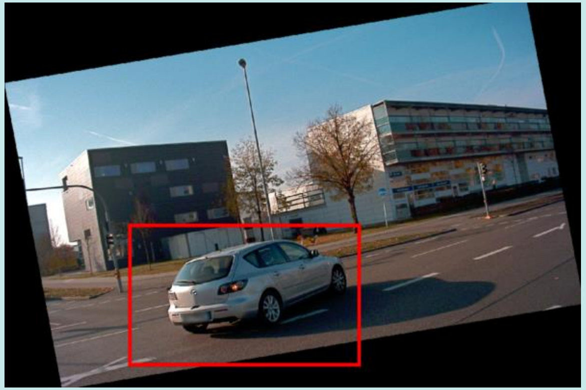




Horizontal Flip



Rotation (20°)



Horizontal Flip + Rotation (10°)

TensorFlow

2.0.0

TFRecord

Data Conversion

- ❑ Tensorflow is an open-source deep learning framework created by Google Brain.

- ❑ In this project we will be using TensorFlow as the Machine Learning framework, hence we need to create TFRecords that can be served as input data for training of the object detector.

- ❑ TFRecord file format is a simple record-oriented binary format that TensorFlow applications use for training data. It is the default file format for TensorFlow. The TFRecord format consists of a set of shared files where each image is a serialized tf.Example message.

continued...

TensorFlow

2.0.0

TFRecord

Data Conversion

To create a tf.Example message from the dataset, we follow the below procedure:

- ❑ For each image, each value needs to be converted to a tf.train.Feature. This tf.train.Feature message type can accept one of the following three types:
    - tf.train.BytesList
    - tf.train.FloatList
    - tf.train.Int64List
- ❑ Then create a map (dictionary) from the feature name string to the encoded feature value produced in the step 1.
- ❑ The map produced in step 2 is converted to a Features message.

Once, the tf.Example is created for the image dataset, then all proto messages are serialized to a binary-string using the .SerializeToString method.

You Only Look Once

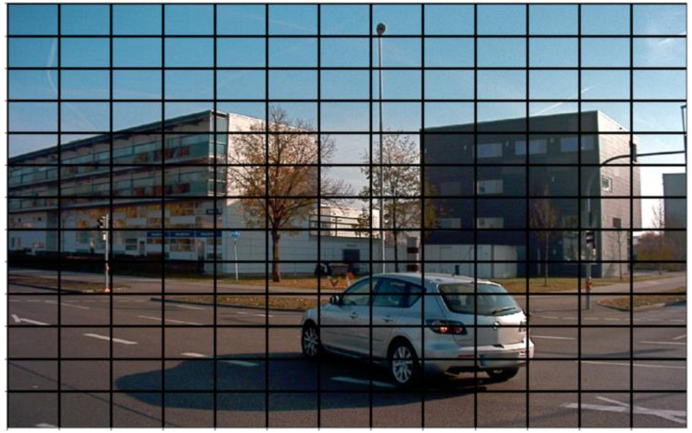
You only look once (YOLO) is a state-of-the-art, real-time object detection system which takes the entire image in a single instance and predicts the bounding box coordinates and class probabilities for these boxes.

- ❑ Why
  YOLO runs significantly faster than other detection methods with comparable performance.

- ❑ Variations
  The first version (YOLOv1) proposed the general architecture, whereas the second version (YOLOv2) refined the design and made use of predefined anchor boxes to improve bounding box proposal, and version three (YOLOv3) further refined the model architecture and training process.

YOLOv3

- ❑ YOLOv3 network divides an input image into [S x S] grid of cells and predicts bounding boxes as well as class probabilities for each grid.
- ❑ Each grid cell is responsible for predicting B bounding boxes and C class probabilities of objects whose centers fall inside the grid cell.

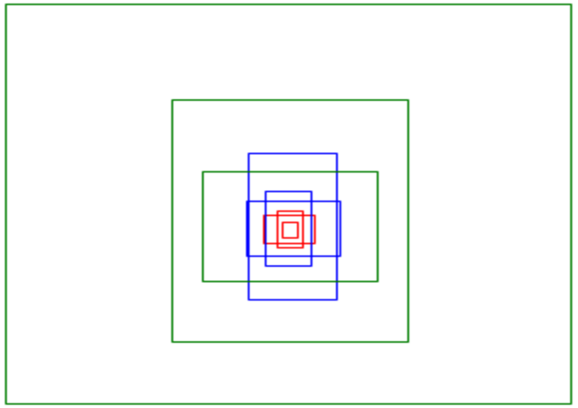


- ❑ If we have S x S grid of cells, after running one single forward pass of convolutional neural network to the whole image, YOLOv3 produces a 3D tensor with the shape of [S, S, B * (5 + C)].

continued...

YOLOv3

- YOLOv3 uses 3 different anchor boxes for every detection scale. The anchor boxes are a set of predefined bounding boxes of a certain height and width that are used to capture the scale and different aspect ratio of specific object classes that we want to detect.
- There are 3 predictions across scale, so the total anchor boxes are 9, they are: (10×13), (16×30), (33×23) for the first scale, (30×61), (62×45), (59×119) for the second scale, and (116×90), (156×198), (373×326) for the third scale.



continued...

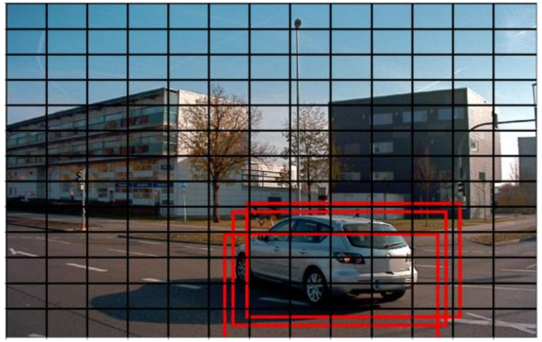
YOLOv3

- The YOLOv3's algorithm returns bounding boxes in the form of (b_x, b_y, b_w, b_h). The b_x and b_y are the center coordinates of the boxes and b_w and b_h are the box shape (width and height). To draw boxes, we need the top-left coordinate (x1, y1) and the box shape (width and height). To get this we can convert them using this simple relation:

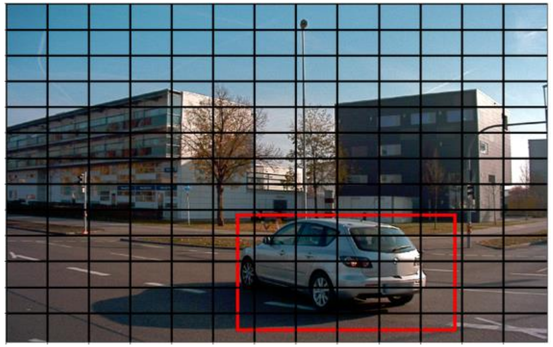
$$x_1 = b_x - (b_w/2)$$
$$y_1 = b_y - (b_h/2)$$

- After a single forward pass, YOLOv3 network suggests multiple bounding boxes. Here, it is required to decide which one of these bounding boxes is the right one. Fortunately, to overcome this problem, a method called non-maximum suppression (NMS) is applied.

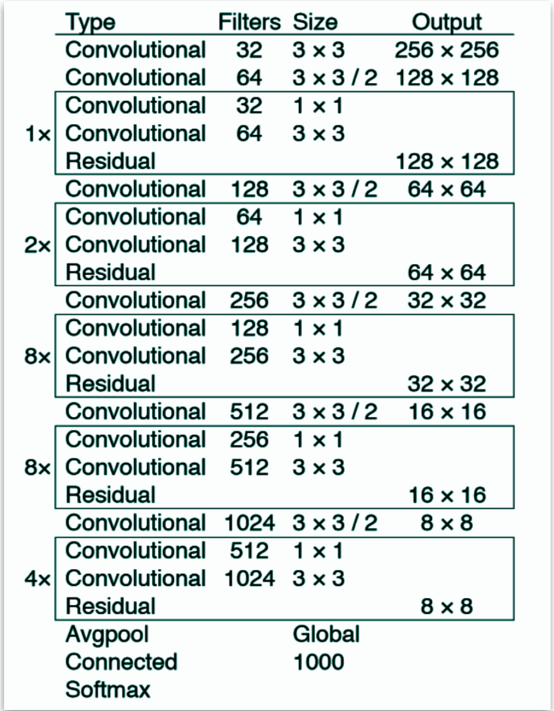
Before Non-max suppression



After Non-max suppression



YOLOv3 Architecture

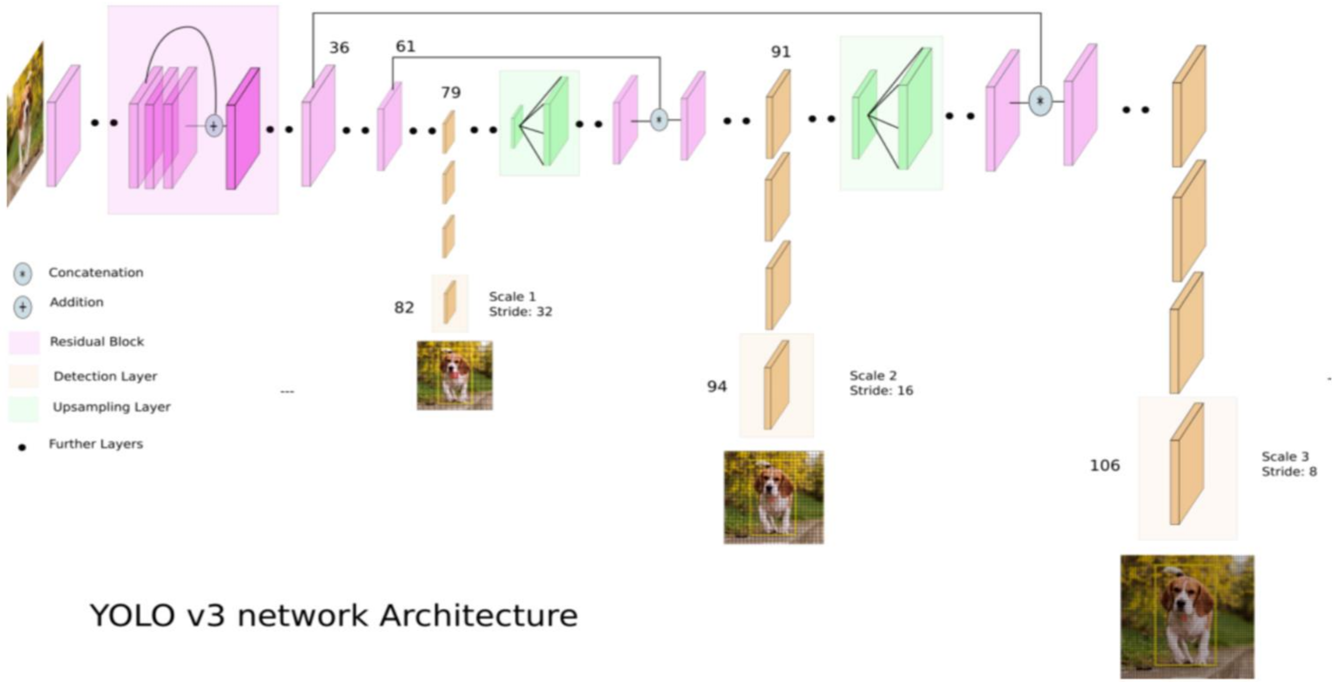


	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
	Convolutional	32	1 × 1	
1×	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
	Convolutional	64	1 × 1	
2×	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
	Convolutional	128	1 × 1	
8×	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
	Convolutional	256	1 × 1	
8×	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
	Convolutional	512	1 × 1	
4×	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

- ❑ YOLOv3 has 53 convolutional layers called Darknet-53 as shown below.
- ❑ The whole network is a chain of multiple blocks with some strides 2 Convolutional layers in between to reduce dimension.
- ❑ Inside the block, there is a structure of (1x1 followed by 3x3) plus a skip connection.
- ❑ Almost every convolutional layer in YOLOv3 has batch normalization after it.

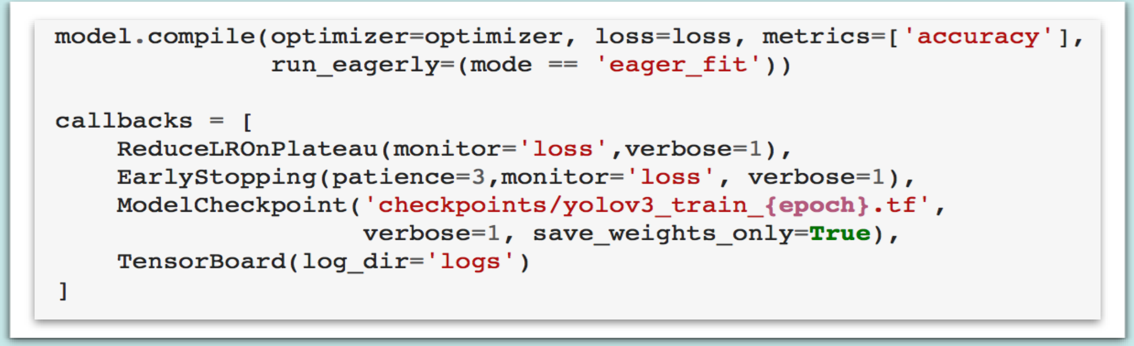
continued...

YOLOv3 Architecture



YOLO v3 network Architecture

- ❑ We will be using Darknet weights which are pre-trained on the COCO dataset as mentioned in the 'Weight Conversion' section.
- ❑ We will use the checkpoints from these trained models and then apply them to our custom object detection task.
- ❑ Optimizers methods are used to change the attributes of the neural network such as weights and learning rate in order to reduce the losses.
- ❑ Then, we initialize the loss object which is our metrics to be monitored while training the model. For compiling the model, we pass parameters such as optimizer object, loss object and also set run_eagerly field to 'True'.



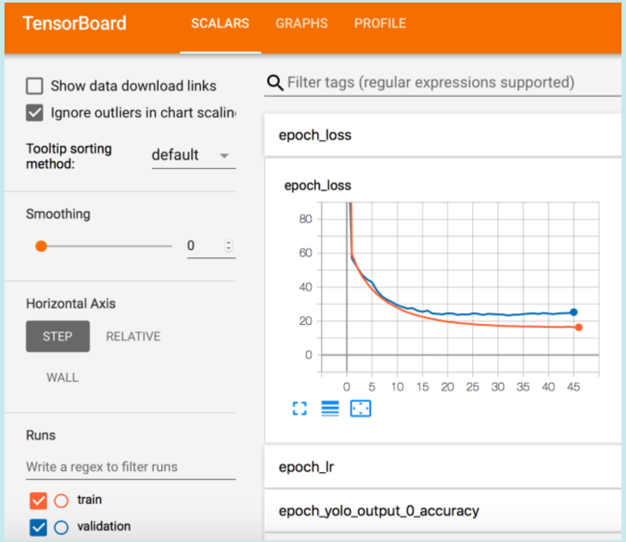
```
model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'],
              run_eagerly=(mode == 'eager_fit'))

callbacks = [
    ReduceLROnPlateau(monitor='loss',verbose=1),
    EarlyStopping(patience=3,monitor='loss', verbose=1),
    ModelCheckpoint('checkpoints/yolov3_train_{epoch}.tf',
                    verbose=1, save_weights_only=True),
    TensorBoard(log_dir='logs')
]
```

- ❑ TensorFlow's eager execution is an imperative programming environment that evaluates operations immediately, without building graphs.

continued...

- Callbacks provide a way to execute code and interact with the training model process automatically. We can pass a list of callbacks to the callback argument when fitting the model.
  - ReduceLROnPlateau callback: It is set to monitor loss of the model.
  - EarlyStopping callback: This callback allows us to specify the performance measure to monitor, the trigger, and once triggered,
    it will stop the training process.
  - ModelCheckpoint callback: This callback to save weights after each epoch completion.
  - TensorBoard callback: TensorBoard is a visualization tool provided with TensorFlow.


TensorBoard
SCALARS
GRAPHS
PROFILE
Filter tags (regular expressions supported)
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
train
validation
epoch_loss
epoch_lr
epoch_yolo_output_0_accuracy

Show data download links

Ignore outliers in chart scalin

Tooltip sorting method: default

Smoothing 0

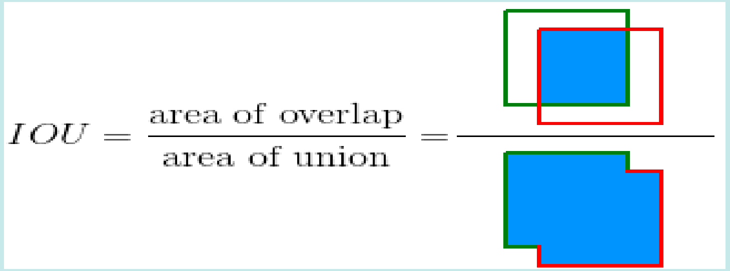
Testing of model

- ❑ Intersection Over Union (IOU) is a measure that evaluates the overlap between two bounding boxes. It requires a ground truth bounding box B_{gt} and a predicted bounding box B_p.
- ❑ Precision is the ability of a model to identify only the relevant objects. It is the percentage of correct positive predictions.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{\text{all detections}}$$

- ❑ Recall is the ability of a model to find all the relevant cases (all ground truth bounding boxes). It is the percentage of true positive detected among all relevant ground truths.

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{\text{all ground truths}}$$



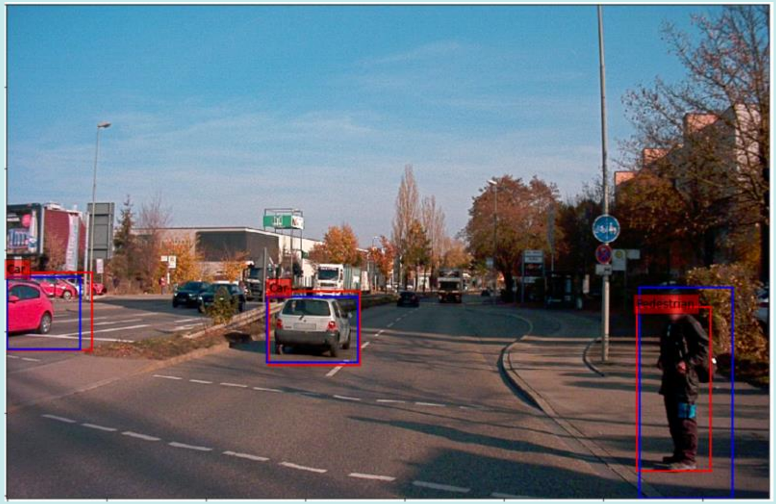
$$IOU = \frac{\text{area of overlap}}{\text{area of union}} =$$

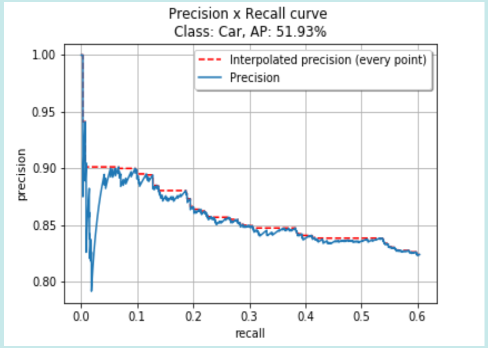
- ❑ threshold: depending on the metric, it is usually pre-set to 50%, 75% or 95%.
- ❑ True Positive (TP): A correct detection. Detection with IOU ≥ threshold
- ❑ False Positive (FP): A wrong detection. Detection with IOU < threshold
- ❑ False Negative (FN): A ground truth not detected.
- ❑ True Negative (TN): It would represent a corrected misdetection.

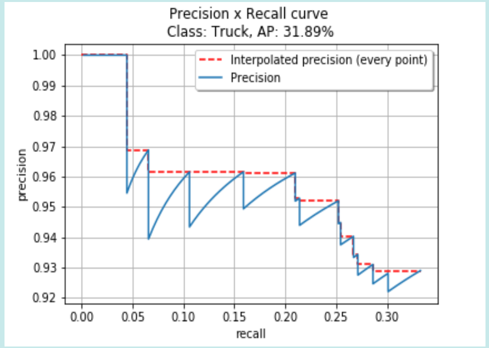
continued...

Testing of model

- ❑ For the detection of objects we need to load the YOLOv3 model with the best checkpoint weight which we got from the training stage.
- ❑ Then this model is executed to detect the objects in the test set images.
- ❑ Once we have both ground truth and detection files, we can pass the data to calculate the accuracy precision and plot the Precision X Recall plot for each class detected.
- ❑ The mean average precision (mAP) of the model developed is around 35.95%.







Future Work

- ❑ After looking at the accuracy of the model, we concluded above, we can further improve this model by training with a larger dataset. Also, we can train enough to have lesser training and validation loss.
- ❑ We can perform various other data augmentation processes such as scaling, shearing, translation, etc.
- ❑ In the Audi dataset, we also have 3D annotations and LiDAR (Light Detection and Ranging) sensor details. This project can further be enhanced by incorporating these details from the dataset for autonomous vehicles to have a better sense of surrounding. We can enhance the current project to perform 3D object detection using these additional details.

Thank you!

GitHub link: https://github.com/Suman-Saini/CapstoneProject_2