Object Detection and Classification in Images using Deep Learning for Autonomous Vehicle Vision

Sushma Suresh Kalkunte, Varun Jagadeesh, Pranshu Diwan

Kalkunte.s@husky.neu.edu, Jagadeesh.v@husky.neu.edu, Diwan.p@husky.neu.edu

Khoury College of Computer Science, Northeastern University
Boston, MA

Abstract

Object Detection is a classical problem in the field of Computer Vision. It includes finding objects of interest in the image, computing where they lie and classifying them to a known set of categories. In this project we use three Deep Learning algorithms namely YOLO, Mask-RCNN and Inception V3 using transfer learning for object detection and classification, to assist autonomous vehicle vision. We highlight the inherit difference between each technique and how they can be used to achieve multiple goals while performing detection and classification. The objective is to accurately detect the presence of commonly found objects on the road, such as, type of vehicle, pedestrians, and traffic signal; thereby classifying them to their respective categories by predicting the probability of it belonging to a specific class. A comparative analysis of each algorithm in terms of its accuracy, loss and output results help understand the advantages and drawbacks of using the techniques in a variety of real-world situations.

Introduction

Computer vision is the field of computer science that enabling computers to "see" by identifying and processing objects in images and videos in the same way that humans do. The power of computer vision in Artificial Intelligence (AI) is beyond our imagination. With neural network and deep learning, AI has empowered like never before. Deep learning refers to a subdivision of Machine Learning where in a system of neural networks is used for learning patterns and features of the data. It allows computational models of multiple processing layers to learn and represent data with various levels of abstraction mimicking how the brain perceives and understands multimodal information, thus implicitly capturing intricate structures of large-scale data. One of the most popular deep learning techniques is Convolutional Neural Networks (CNN) which is a class of deep feed forward artificial neural network primarily used on images. The CNN architecture combines the benefit of a standard neural network training with the convolution operation to efficiently classify the images.

The advantage that CNN has over traditional machine learning algorithms is the concept of transfer learning. Transfer Learning enables reusing models trained for a specific task as a starting point for modelling a second and similar task. Learning a model for a complex application like ours requires lot of computational resources and time for training and testing. It is a tedious process and may lead us to achieve very low accuracy rates and incorrect classification. Transfer learning on the other hand overcomes this drawback by saving time, enabling better performance of the models, and not needing a huge amount of data.

This project involves using transfer learning for detecting and localizing objects on road images and classifying the detected objects to their respective category. We are using Microsoft COCO dataset with specific set of classes for training and evaluation.

Our Problem:

In recent years, there has been a significant increase in research interest supporting the development of the autonomous vehicles.

In many autonomous driving systems, the object detection subtask is itself one of the most important prerequisites to autonomous navigation, as this task is what allows the car controller to account for obstacles when considering possible future trajectories; it therefore follows that we desire object detection algorithms that are as accurate as possible. To be economical and widely deployable, object detector must operate on embedded processors that dissipate far less power than powerful GPUs.

While recent research has been primarily focused on improving accuracy, for actual deployment in an autonomous vehicle, there are other issues of object detection that are equally critical. For autonomous driving some basic requirements for image object detectors include the following:

- Accuracy: The detector ideally should achieve 100% recall with high precision on objects of interest.
- Speed: The detector should have real-time or faster inference speed to reduce the latency of the vehicle control loop.
- Small model size: It brings benefits of more efficient distributed training, less energy consumption and more feasible embedded system deployment.
- Computational requirement: The solution should be easily deployable, scalable and not have very demanding computational requirements (should run smoothly on CPU machines).

Solution Methods:

To solve the problem of Object Detection and Classification, we employ several deep learning algorithms. The pipeline of our solution is as follows: We first collect labelled images of traffic data from the Microsoft COCO dataset. We then perform some basic pre-processing on our data so that every image is uniform. Afterwards, we split our data into training and testing sets. The training set will be used to train our algorithms while the testing set will be used for validation purposes.

To select which models will perform better, we conducted a literature survey in which we explored various models which could give us good accuracies. We then selected and build three models, namely YOLO, Mask-RCNN and Inception V3. After every model was trained, we evaluated the accuracy on our test data. We also looked at sample results by providing test images and compared which model is the best use case for our project.

Background

In this section we will provide a brief description of the methods used in our project. We start by summarizing information about autonomous vehicles, and move on to elaborate object detection, segmentation, and classification methods, which is the crux here. We then explain Convolutional Neural Networks, a base model on top of which our algorithms are built. We end the section by providing a brief description of our dataset.

Image Classification: It is a supervised learning problem that defines a set of classes or the objects to identify in images, trains a model to recognize them using labeled example. For Example: An image of a car can be classified as a class label "car"

Object Localization: This algorithm locates the presence of an object in the image and represents it with a bounding box. It takes an image as input and outputs the location of the bounding box.

Object Detection: It is a combination of image classification and object localization. It takes an image as input and produces one or more bounding boxes with the class label attached to each bounding box.

Image segmentation: It is the process of partitioning an image into multiple segments, here we change the representation of an image into something that is more meaningful and easier to analyze. Segments represents parts of objects and comprise sets of pixels. Image segmentation sorts pixels into larger components. There are three level of image analysis:

- Classification
- Object Detection
- Segmentation

Segmentation identifies parts of image and understand what object they belong and can be further classified into two parts namely:

- **Semantic Segmentation:** It classifies all the pixels of an image into meaningful classes of objects and they resemble real world objects. For example, we can separate all pixels associated to a particular class and color them with a same color.
- **Instance Segmentation:** It identifies each instance of each object featured in the image instead of categorizing each pixel like in semantic segmentation. For example, instead of classifying 5 cars as one instance, it will identify each individual car.

Convolutional Neural Network:

A Convolutional Neural Network (CNN) is a Deep Learning algorithm which can take in an input image, assign weights and biases to various aspects in the image and be able to differentiate one from the other. As mentioned earlier, CNNs are extensively used in the field of Computer Vision for Object Detection and classification tasks. The basic architecture of CNN is shown in figure -1

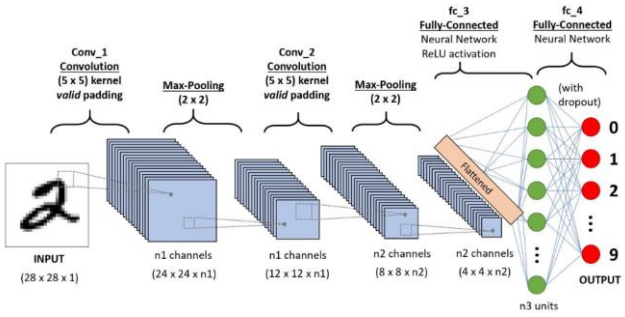


Figure -1: Architecture of a Convolutional Neural Network

Convolution Layer: It is one of the core building blocks of CNN that does most of the computational task. The objective of the Convolution Operation is to extract the high-level features from the input image. It need not be limited to only one Convolutional Layer. With added layers, the architecture adapts to the High-Level features as well, giving us a network, which has an overall understanding of images in the dataset, similar to how we would.

Pooling Layer: It is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model. Max Pooling returns the maximum value from the portion of the image covered by the Kernel.

By adding the fully connected network, we learn the outputs of the convolutional layers and use them to predict the actual output class with the help of activation functions.

Some of the widely used architectures of CNN's built on top of the base CNN model are: AlexNet, VGGNet, ResNet. In our project, we picked three models – YOLO, Mask R-CNN and Inception V3. They are discussed in detail in the project description section.

Related Work

CNN's for object detection:
In 2013, Girshick et al. proposed Region-based Convolutional Neural Networks (R-CNN), which led to substantial gains in object detection accuracy. The RCNN approach begins by identifying region proposals (i.e. regions of interest that are likely to contain objects) and then classifying these regions using a CNN. One disadvantage of R-CNN is that it computes the CNN independently on each region proposal, leading to time-consuming ($\leq$ 1 fps) and energy-inefficient ($\geq$ 200 J/frame) computation. To remedy this, Girshick et al. experimented with a number of strategies to amortize computation across the region proposal, resulting in Faster R-CNN.

Fully convolutional networks (FCN) were popularized by Long et al., who applied them to the semantic segmentation domain. FCN defines a broad class of CNNs, where the output of the final parameterized layer is a grid rather than a vector. This is useful in semantic segmentation, where each location in the grid corresponds to the predicted class of a pixel.

OverFeat: OverFeat is a Convolutional Neural Network model released in 2013 that jointly performs object recognition, detection, and localization. OverFeat is one of the most successful detection models to date, winning the localization task in the ImageNet Large Scale Visual Recognition Challenge 2013. OverFeat is eight layers deep and depends heavily on an overlapping scheme that produces detection boxes at multiple scales and iteratively aggregates them together into high-confidence predictions. However, we did not use it because today's newer algorithms provide way better results. To give an idea of its efficiency, it has an error rate of about 29.9% on localization and 14.2% for classification.

VGG16: VGG16 was an attempt to usurp OverFeat's dominance in object classification and detection by exploring the effects of extreme layer depth. A 16-layer and 19-layer model was produced, setting new benchmarks on localization and classification in the ImageNet ILSVRC 2014 Challenge. However, Fast RCNN attempts to capture the accuracy of deeper models while improving their speed, so we chose that instead.

Project Description

The algorithms that we have considered can be split into two broad groups:

1) Based on Regression: Instead of selecting interesting parts of the image, these algorithms predict classes and bounding boxes with probabilities for the entire image in one single run of the algorithm.
   Eg: YOLO
2) Based on Classification: They are implemented by first selecting regions of interest in the image and classifying the regions using CNN
   Eg: Mask R-CNN and Inception V3

YOLO

You Only Look Once (YOLO) framework is a popular object detector and classifier algorithm. It works by taking the entire image as an input instance and learns to predict bounding box coordinates and class probabilities for each of the identified boxes. YOLO consists of only convolution layers thus making it a Fully Convolutional Network. We use the latest variant: YOLO v3 for our analysis. Its architecture is an advanced feature detector called Darknet-53 [] and it is much stronger and faster than its predecessors (YOLO v2 and just YOLO).

Layer Type	# of Filters	Size	Output	
Convolutional	32	3x3	256x256	
Convolutional	64	3x3/2	128x128	
Convolutional	32	1x1		
Convolutional	64	3x3		1x
Residual			128x128	
Convolutional	128	3x3/2	64x64	
Convolutional	64	1x1		
Convolutional	128	3x3		2x
Residual			64x64	
Convolutional	256	3x3/2	32x32	
Convolutional	128	1x1		
Convolutional	256	3x3		8x
Residual			32x32	
Convolutional	512	3x3/2	16x16	
Convolutional	256	1x1		
Convolutional	512	3x3		8x
Residual			16x16	
Convolutional	1024	3x3/2	8x8	
Convolutional	512	1x1		
Convolutional	1024	3x3		4x
Residual			8x8	
Connected		1000		
Softmax				

Fig 2: YOLO Architecture

Figure - (2) shows the consolidated architecture of Darknet-53. Counting from the image above, there are 53 convolution layers purely for the purpose of detection and classification, each of them followed by a batch normalization

layer and a Leaky ReLu activation. A convolution layer with stride 2 is used to down sample the feature map with no separate pooling layer. This technique helps to prevent the loss of low-level features that is at most times attributed to pooling.

The input size to the first layer of the network is 416 x 416 x 3. Predictions are made at three scales by down sampling the input image dimensions by 32, 16 and 8 respectively. The detection is performed by applying kernels of size 1x1 on different sized feature maps at three different places in the network. The detection which happens at the 13^{th} layer is primarily for identifying large and prominent objects in the image. Layer 52 is responsible to detecting smaller objects and areas in the image with minute distinguishing details. Probabilities are assigned to each of the detections and the SoftMax layer in the end is responsible for the final classification based on these probabilities. This complex architecture makes YOLO v3 very strong, robust, and capable of accurately detecting objects of various sizes in the image.

The task at hand is to predict classes of objects commonly found on the road and draw a bounding box specifying the location of the object. Each bounding box can be described by important features like its center, width, height, and a value corresponding to the object class (truck, car, traffic light, people etc). Additionally, the probability that an object exists in the box also must be computed.
Implementation of this technique in the project can be broken down into the following major steps:

1) Resizing images
2) Feeding images into network
3) Performing object detection
    a. Performing Non-Maximal Suppression
    b. Implementing Intersection Over Union (IOU) to obtain final bounding boxes
4) Performing object classification
5) Returning bounding boxes with their respective classes and probability

Since the input images used for training and testing are of various sizes, the first step is to resize them to 416 x 416 so uniformity is maintained. The next step is to feed them into the network for object detection and set the stride. Stride of the layers in the network is the factor by which the output of the layer is smaller than the input image to the network. Since the stride in our case has been set to 32, the size of the output image will be 13 x 13. While training, each image is split into cells using a 13 x 13 feature mapping grid. For each cell, 5 bounding boxes are generated with the probability of containing the required objects. This results in 160 bounding boxes being created for each image. Most of these boxes do not contain the required objects and hence are redundant. Using NMS, boxes with low object probability (below a set threshold) are removed. Intersection Over Union metric/Jaccard index is computed to quantify the area of overlap between bounding boxes. By setting a IOU threshold, only those bounding boxes having the highest shared area is retained.

Each of the bounding boxes have seven distinct parameters namely: centroid, width, height, confidence detection level, probability of object class and id of object class. Classification of the detected objects in the bounding box is then performed by computing the probability of it belonging to a specific class. SoftMax classifier is used in the last layer as classes are not mutually exclusive. For example, SUVs can belong either to car or truck and its classification to either of the classes is considered correct.

The final vector containing bounding boxes of detected objects with their respective classes and probabilities are returned and stored for displaying on the input image.

MASK R-CNN

Mask R-CNN is a deep learning architecture explored by Facebook AI. It stands for Mask-Region Convolutional Neural Network and aim at solving instance segmentation problem in machine learning and computer vision. Instance segmentation is challenging because it not only requires correct detection of all objects but also segment each instance. Hence there is combination of elements where it detects, classifies individual objects and localize each of them using a bounding box and mask. Mask R-CNN evolves through first three versions of famous object detection algorithm namely R-CNN, Fast R-CNN and Faster R-CNN. Figure-1 shows the architecture of Mask R-CNN.

Faster R-CNN uses a CNN feature extractor to extract feature maps from the images. Once the feature maps are extracted these are passed through a Region Proposal Network that returns the bounding boxes. RoI pooling is applied to warp them into fixed dimension and then fed into fully connected layers to make classification and output the bounding boxes for objects.

Mask R-CNN is an extension of Faster R-CNN object detection architecture. The backbone of the model is similar to Faster R-CNN where it uses CNN to extract feature maps but in case of Mask R-CNN, ResNet is used to do the same. These features act as an input for the next layer. The ResNet is on of the early layer of the network and will detect low level features where later layers will detect higher level features. The image is converted from $1024 \times 1024 \times 3$ to a feature map of shape $32 \times 32 \times 2048$. It also has an identical stage RPN and along with the output of Faster R-CNN it adds a branch which outputs the object mask along with the outputs. The mask branch is a small fully connected network applied to each RoI predicting a segmentation mask in a pixel-to-pixel manner. Pixel level segmentation requires a lot of fine-grained alignment than bounding boxes. Hence Mask R-CNN has RoIAlign layer which can be better and map the regions of the image precisely.

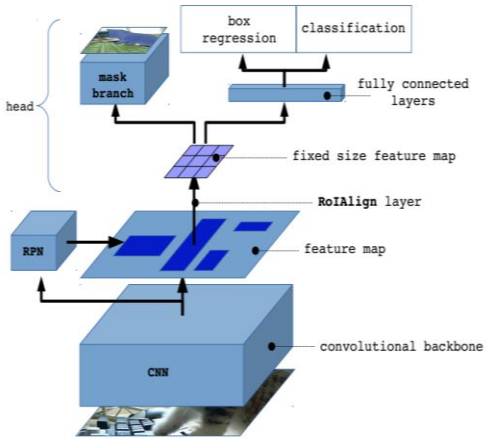


Fig 3: Mask R-CNN Architecture

RoIAlign layer is designed to fix the location misalignment caused by quantization of the RoI pooling. RoIAlign removes the hash quantization. For example, it uses x/16 instead of [x/16], so that extracted features can be aligned with the input pixels. RoI Align in which feature map is sampled at different points and then. a bilinear interpolation is applied to get a precise idea of what would be at pixel 2.93. RoIAlign leads to large improvements. A CNN is used which takes regions selected by ROI classifier and generates masks for them. The generated masks are low resolution 28 × 28 pixels. During training the masks are scaled down to 28×28 to compute loss and during inference the prediction masks are scaled up to size of ROI bounding box which gives the final masks for each object.

Loss function: The multitask loss function of Mask R-CNN combines the loss of segmentation mask, localization and classification.

$$L = L_{mask} + L_{box} + L_{cls}$$

L_{box} and L_{cls} are same in Faster R-CNN. The mask branch generates a mask of dimension m×m for each RoI and each class, if there are k classes in total then the total output is of size K. m^2. This is because model is trying to mask for each class.

L_{mask} is defined as the average binary cross entropy loss, only including k-th mask if the region associated with the ground truth class k.

We are training with 8 images per batch with three classes namely car, person and traffic light. The image dimension is set to 128×128 and we have set a 10 epoch to train over the dataset to get decent results. We are using pre-trained coco weights which has a wide variety of the classes. We are not changing the whole architecture of Mask R-CNN but just tuning the head layers, we are freezing all the backbone layers and training only the randomly initialized layers of the heads. We are using accuracy as a metric to evaluate the model which each image is loaded, and it is given a detection with a bounding box enclosed in a mask with a probability percentage attached to it.

Inception V3

Inception is a deep Convolution Neural network architecture developed by GoogleNet assisting in classification of objects in computer vision. It is a widely and most popularly used image recognition model on ImageNet dataset.

The architecture of Inception v3 is shown in Figure 2. It is a convolution neural network that is 48 layers deep. The original network was trained on more than a million images from the ImageNet database and has an ability to classify more than 1000 object categories. Therefor the network has rich feature representations for a wide range of images. The network has a different image input size compared to any other deep CNN. It has an input size of 299×299. We are using the ImageNet weight in our model because of its wide range of classification of objects. We are not changing the whole architecture. The whole of input layers remains same which is the original architecture of Inception V3. But we just tuning the final layer of an existing model and this will allow us to retrain the final layer of an existing model and by doing this we are decreasing the training time. This process is called transfer learning and is immensely popular. We add a dropout later followed by a Dense layer which generates softmax class score for each class and compile the final model using Adam optimizer with a low learning rate. We are training the model using a different dataset so that it has the ability to classify new data along with the existing knowledge of the model which was originally trained on ImageNet. This will result in high accuracy with respect to classification without the need of high computational power.

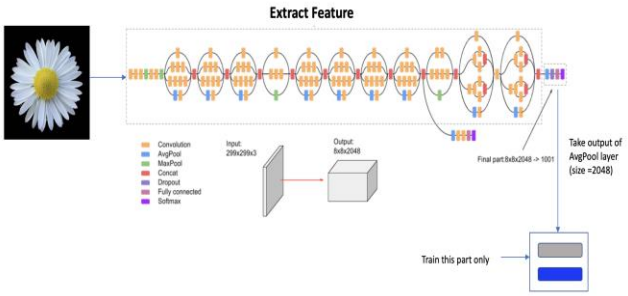


Fig 4- Inception V3 Architecture

The size of this dataset used for training has around 3000 images and the around 1000 images for validation. We tried tuning the batch size as CPU used to run out of memory and finally tuned to 28. We ran the model over 12 epochs. We

are using Accuracy as a evaluation metric to evaluate the models. Figure 3 shows the summary of how accuracy increased over 12 epochs for training and validation data. We tested the images over multiple images with the one's that they were trained originally, and our dataset and also the one's which they hadn't seen, and we got decent results in the first two. But the accuracy score of the images which the model had no knowledge were less and this was expected. Also, Inceptionv3 will look into the details of various things present in an image. For example: If we pass a image of a car to the model it will see the various components of car such as wheel, windshield, etc.

Experiments

YOLO, Mask RCNN, and Inception were trained on approximately 1000 image samples from Microsoft COCO dataset. The hardware specifications for running the experimentation are as follows:
Machine type- CPU
Operating System- Windows/iOS
Processing Power- 10th Gen Intel Quad Core Processors
Memory: 16GB
Graphics: NVIDIA MX250

Shown in Figure (5), (6), and (7) are the results of accuracy and loss during the training phase for each algorithm. As it can be observed, YOLO, Mask RCNN and Inceptionv3 have very similar accuracy and loss values after training over 10 epochs. The accuracy was observed to be high at approximately 0.93 and the loss was close to 0.20. Thus, comparing the three algorithms based on this parameter does not add much value to our analysis.

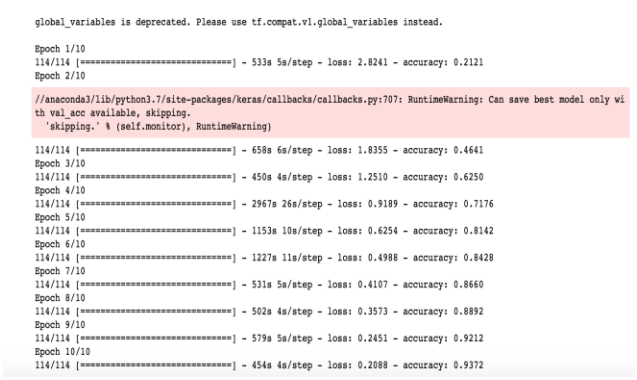


Fig 5: YOLO Training

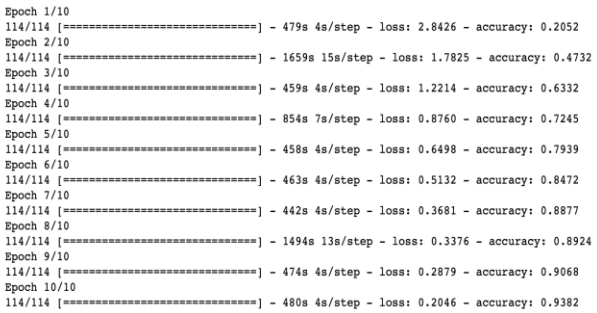


Fig 6- Mask R-CNN model training results

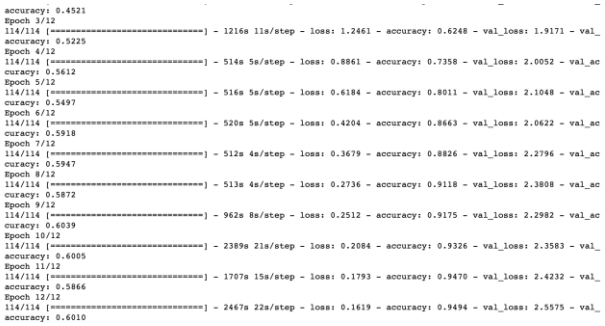


Fig 7- Inception V3 model training results

We proceed to understand how these algorithms performed during testing. The graphs shown in Figure (8), (9) and (10) indicate the validation loss and accuracy for YOLO, Mask R-CNN, and Inception v3 respectively.

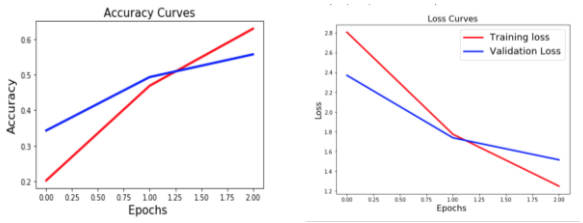


Fig 8- Yolo accuracy and loss curves

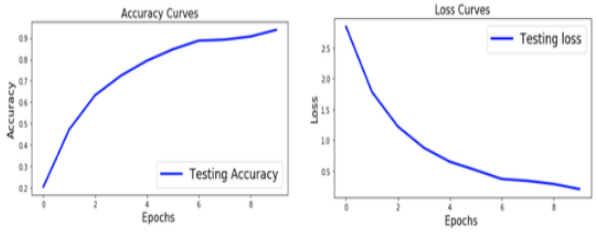


Fig 9- Mask R-CNN accuracy and loss curves

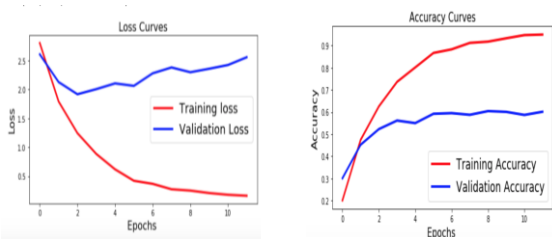


Fig 10- Inception V3 accuracy and loss curves

The number of epochs while testing YOLO was limited to 3. We noticed that the time taken during testing was high and the validation loss began to increase after reaching a value of 1.6. The accuracy at the end of epoch 3 was close to 0.55. The increase after this point was extremely gradual and less significant.

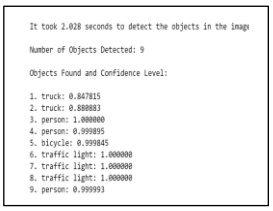
In case of Mask R-CNN, the testing accuracy reached 0.91 and the loss went as low as 0.35 after 10 epochs. This is an indicative that the algorithm was able to detect and classify objects correctly.

The performance of Inception v3 on testing data was comparatively poor. The loss reached a minimum of 1.9 at 3 epochs after which it began to increase and decrease. We suppose that these are the local minimums and the algorithm was not able to converge correctly within the specified number of iterations. This is the reason why the gap between the training and validation curves is very high in Figure (10).

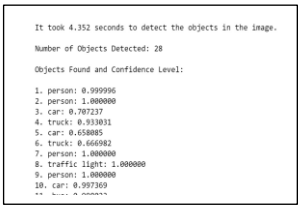
We now proceed to compare the detection and classification performance of the above-mentioned algorithms on images by analyzing the results with the bounding boxes and probability of assignment.

Few results of the YOLO algorithm have been displayed before and after classification in Figure (11). The first set of images is of a street with relatively fewer vehicles and people. As it can be observed, the algorithm has currently managed to detect all of them. Results also show that people, bicycle and traffic signals have been accurately classified with full probability scores whereas the two SUVs have been mis-classified as a truck(with slightly lower probability score) as it might seem too big for it to be classified as a car. It took a total of 2.028 seconds for YOLO to make these predictions.

In the second set depicted of images, notice that the input images were more complex with a lot of vehicles and people on a busy street. Again, the YOLO is able to correctly detect the objects and classify them with good probability, even if they are far, occluded and small. The time taken for predictions to be made on this image was 4.35 seconds which is reasonable due to the large number of objects.



(Set 1)



(Set 2)

Fig 11- Results of YOLO

Next shown are the output images after performing Mask R-CNN.

Fig 12- Results of Mask RCNN

The first set of images in Fig. 12 are the that of the same busy street with a lot of objects. Mask RCNN has been performed on it to compare its results with YOLO. As observed, object segmentation has been performed in addition to drawing the bounding box. This enabled us to understand the actual shape and form of the object which is more intuitive and useful to the autonomous vehicle than just the outer bounding box. Although good classification has been performed, we notice that the algorithm fails to detect objects that are further away and small. This is a slight drawback of Mask RCNN when compared to YOLO.

In the second set of images, we consider a scenario of a busy highway road with a lot of vehicles. We observe that the white car at the bottom of the image has failed to be detected as half of the car is not present in the image. When significantly large portions of the complete object are present, Mask RCNN detects, segments, and classifies them accurately.

Figure 13- Results of Inception V3

Moving on to analyze the output results of the third technique: Inceptionv3. Figure (13) depicts Inception v3 technique applied on the same image as that of YOLO for comparative purposes. Since Inception is purely an image classification technique, it fails to detect multiple objects such as bicycle, cars, and people in this image. As bicycle appears closer and larger, Inception v3 considers it to be the only object in the image and assigns relevant probabilities for sub-categories.
In the second image, there is only one object in the image, that of the blue car. Inception identifies it accurately and attempts to categorize it based on sub-classes. Thus, we can safely conclude that inception has tremendous classification capability, but it fails to identify when multiple objects are present in the image along with its localization.

We to prove that our implementation of YOLO was scalable, we created a real time version of the same algorithm that can detect the vehicle types, pedestrians, and traffic signals as and when the video is being taken. We notice that there is no lag during the computation and accuracy of detection and classification is also very high.

Conclusion

Implementing the deep learning algorithms such as Yolo, Mask R-CNN, and Inception V3, we were able to detect the objects on the road accurately and classify them.
YOLO is an extremely robust algorithm for the purpose of detection and classification. The resultant image has bounding boxes where the probability of an object of a particular class is high. It was easiest to implement as compared to the other algorithms owing to the single stage architecture. The computation time was also slightly lesser than that of Mask RCNN and Inception V3. The algorithm performed well even on dense, cluttered images with multiple occlusions. Mask RCNN has the added advantage of performing image segmentation as well, apart from detection and classification. Although it had a slightly higher training time as compared to YOLO, it failed to detect objects which are small and occluded. Inception V3 works extremely well as a single object classifier with moderate training time. The major disadvantage of this technique is its inability to perform object detection. In future we will try to combine the results on Inception with another detection algorithm so that we can classify different objects in an image and include the details of each of the object accurately.

References

[1] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Region-Based Convolutional Networks for Accurate Object Detection and Segmentation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 142-158, 1 Jan. 2016.
[2] A. Eskandarian, C. Wu and C. Sun, "Research Advances and Challenges of Autonomous and Connected Ground Vehicles," in *IEEE Transactions on Intelligent Transportation Systems*.
[3] B. Wu, A. Wan, F. Iandola, P. H. Jin and K. Keutzer, "SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving," *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Honolulu, HI, 2017, pp. 446-454.
[4] Y. Tang, J. Wang, B. Gao, E. Dellandréa, R. Gaizauskas and L. Chen, "Large Scale Semi-Supervised Object Detection Using Visual and Semantic Knowledge Transfer," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, 2016, pp. 2119-2128.
[5] X. Zhou, W. Gong, W. Fu and F. Du, "Application of deep learning in object detection," *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, Wuhan, 2017, pp. 631-634.
[6] https://machinelearningmastery.com/object-recognition-with-deep-learning/
[7] https://arxiv.org/abs/1703.06870.pdf
[8] https://keras.io/applications/#inceptionv3
[9] https://arxiv.org/abs/1512.00567.pdf
[10] Real-time Object detection and Classification for Autonomous Driving, Seyyed Hamed Naghavi, Hamidreza Pourreza.
[11] Object Detection for Autonomous Vehicles. Gene Lewis, Stanford University