Pedestrian Detection: Performance Comparison Using Multiple Convolutional Neural Networks

Meenu Ajith[1] and Aswathy Rajendra Kurup[1]

University of New Mexico, Albuquerque NM, USA

Abstract. Pedestrian Detection in real world crowded areas is still one of the challenging categories in object detection problems. Various modern detection architectures such as Faster R-CNN, R-FCN and SSD has been analyzed based on speed and accuracy measurements. These models can detect multiple objects with overlaps and localize them using a bounding box framing it. Evaluation of performance parameters provides high speed models which can work on live stream applications in mobile devices or high accurate models which provide state-of-the-art performance for various detection problems.These convolutional neural network models are tested on the Penn-Fudan Dataset as well as Google images with occlusions, which achieves high detection accuracies on each of the detectors.

1 Introduction

Pedestrian detection has made immense progress over the last few years with the arrival of convolutional neural networks. It persists as one of the challenging problems because of the large variability of pedestrians in clothing, so that only a few areas can be included as the real feature for distinguishing this category. In addition, the lighting conditions, background overlaps, articulation, and occluding accessories such as umbrellas and backpacks may cause changes to the silhouette of the pedestrian. Conventional pedestrian detection methods require complex feature extraction manually and have a limitation of slow processing time. However, modern convolutional neural network models such as Faster R-CNN [1], R-FCN[2], Mobilenets[3] and SSD[4] has been found to have more accurate and fast performances. But the appropriate tradeoff between speed and accuracy for each of the models has not been evaluated for the application of pedestrian detection. The detection speed is calculated in terms of seconds per frame(SPF) and the accuracy metric used is mean average precision(mAP).

The State-of-the-art pedestrian detection frameworks depend on region proposal algorithms to estimate the location of the pedestrians. After the advancement of networks like SPPnet [5] and Fast R-CNN [6] the running time was reduced thereby making the region proposal computation to impede. Th Region Proposal Network (RPN) introduced cost-free region proposals that share convolutional features with the detection network. Thus, Faster R-CNN is composed of a deep fully convolutional network that proposes regions, and these

regions are used by the Fast R-CNN detector to predicts object bounds and objectness scores at each position. In case of PASCAL VOC 2007, 2012, and MS COCO datasets this model achieved high detection accuracy with only 300 proposals per image. The region-based, fully convolutional networks(R-FCN), in contrast to the region-based detectors such as Fast/Faster R-CNN have all computations shared on the entire image. Meanwhile, the Faster R-CNN applies a costly per region network on the image large number of times thereby increasing the computational burden. The R-FCN model can also use the Residual Networks(ResNets) [7] as a fully convolutional image classifier backbone for various detection applications. To manage real-world applications such as self-driving car and augmented reality, the recognition tasks must be performed on a computationally limited platform. MobileNets are such small, low latency models that can be easily matched to the design requirements for mobile and embedded vision applications. They were primarily constructed by performing depth wise separable convolutions. It was subsequently used in Inception models [8] to reduce the computation in the first few layers.While accurate, the current approaches have been too computationally intensive for embedded systems and, too slow for real-time applications, even with high-end hardware. A single deep neural network named SSD was introduced which achieved 77.2% mAP for 300300 input and 79.8% mAP for 512512 input on VOC2007, outperforming a comparable state-of-the-art Faster R-CNN model. Several attempts were made to build faster detectors, but so far, significantly increased speed comes only at the cost of significantly decreased detection accuracy.

In this paper, the speed/accuracy trade-off of modern detection systems are explored in the application of pedestrian detection. The test time performances, duration of training, learning rate and loss functions for each model are compared to find the optimal detector for this application. Here lesser training time denotes faster convergence to a more accurate model using fewer parameters. This reduces the complexity of the system as well as prevents overfitting. The Faster R-CNN, R-FCN, and SSD at a high level consist of a single convolutional network and are trained with a mixed regression and classification objective thereby making it easier to compare and analyze these systems.The implementations of these architectures were done in TensorFlow which finally provided the tradeoff results for various detection systems.

2 Model Architecture

The detection framework consists of the convolutional object detectors with different meta architectures and feature extractors. The meta architectures include proposal based methods such as R-FCN and Faster R-CNN and proposal free method with SSDs. In Faster R-CNN the computational burden is shared and the region proposals are generated using neural networks. Hence this architecture has improved efficiency while in R-FCN the removal of fully-connected layers improves speed as well as accuracy. SSD uses different bounding boxes and small convolutional filters for prediction. It achieves high detection speed

even while using relatively low-resolution input. In particular, several pre-trained models such ResNet, MobileNet, and Inception are used for feature extraction. Thus the proposed network uses a combination of these architectures and trained them with the multi-task loss function for object detection and localization.

2.1 Modern Convolutional Detectors

Single Shot Detector SSD uses a single deep Neural Network to detect the objects. SSD makes the output space of bounding boxes discrete to form a set of default boxes. These default boxes are formed over different aspect ratios and scales for each of the feature map location. During the prediction, scores are generated for each of the object category in each default box and the box dimensions are adjusted to fit the object shape better[4] .

Single Shot Multibox Detector is a feed forward Convolutional network producing a collection of bounding box which have fixed-size. The scores are generated for the presence of an object class instance in those boxes, following which a non-maximum suppression step is done which produce the final detection. The base network layers are based on standard architecture used for higher quality image classification. Feature Layers are added to the truncated base network which decrease in size progressively as shown in Fig. 1 . Hence, the detections at multiple scales is possible. Using a set of Convolutional filters each of the added feature layers formulates a fixed set of detection predictions. For a feature Layer, the parameters used for prediction is mainly a kernel of size 3 x 3 x p (the feature layer having size m x n with p channels) which produces either a score for a category or the shape offset with respect to the default box position relative to feature map location [4]. Each feature map cell has a default bounding box assigned to it. For each of these feature map cell the offsets relative to the default box shapes in the cell are calculated. Also, the scores that indicate the presence of a class instance in these boxes are predicted for each class. At the training time, the default boxes are first compared with the ground truth boxes following the calculation of model loss. The model loss calculated is hence a weighted sum between Localization loss [6] which corresponds to the shape offset and confidence loss calculated from the confidences for all object categories.

SSD Training Objective The training objective is obtained from the Multibox Objective [9][10]. The Objective Loss function is the weighted sum of Localization Loss (loc) and Confidence Loss (conf) [1] given by equation (1).

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \tag{1}$$

N is the number of matched default boxes. Localization Loss is computed in the form of a smooth L1 loss [6] between the parameters corresponding to the predicted box (l) and the ground truth box (g). [4]. Confidence loss is mainly the loss over different class confidences (c). For N=0, loss is set to zero.

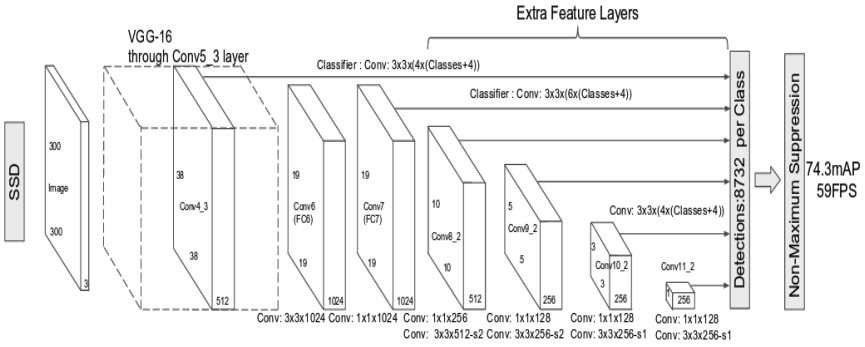


Fig. 1: Architecture of SSD

Faster R-CNN Faster R-CNN (Faster region-based convolutional neural networks) is a single, unified network mainly used for object detection [1]. Fast R-CNN developed before the Faster R-CNNs were able to achieve near real-time rates using very deep networks [11], ignoring the time spent on region proposals. [1] The Region proposal step was computationally expensive with a huge running time.

Faster R-CNN has two modules. The first module being the deep fully convolutional network proposing regions. The second module is the detector module from Fast R-CNN [6]. These detectors use the proposed regions. The recently popular term attention [12] mechanisms, the Region Proposal Network tells the Fast R-CNN detector module where to look at [1]. Fig. 2 shows the description of the layers. A Region Proposal Network (RPN) takes an image as input. The output is a set of rectangular proposals for the objects with a score depicting the objects presence. RPN is modelled using a fully connected convolutional network. The region proposals are generated by sliding a small network over the convolutional feature map. This feature map is obtained from the previous convolutional layer. The small network takes in an n x n spatial window of the feature map as the input. These windows are then mapped to a lower dimensional feature. The Lowdimensional features are then fed to two fully connected layers namely box regression layer (reg) and box-classification layer (cls). [1] In Fig. 3 , for k locations corresponding to maximum possible proposals reg layers have 4k outputs showing coordinates of k boxes whereas cls layer has 2k scores which help in estimating the probability of object occurrence for each proposal. These k proposals are parameterized with respect to k reference box called as anchor. Anchor based approach helps in addressing the multiple scale and aspect ratio issues. This design hence avoids extra cost for addressing scales.

The loss function for an image while training RPN is defined as:

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{2}$$

i is the index for an anchor in mini batch and p_i is the predicted probability of anchor I being an object. The ground-truth label p_i^* is 1 if the anchor is positive and 0 if negative. t_i represents the coordinated and t_i^* is the ground-truth box counterpart. L_{cls} is the log loss over two classes (whether object or not) and L_{reg} is the regression loss. The two terms obtained from the cls and reg are normalized by mini-batch size. During training either the RPN is trained first and the proposals are used to train the Fast R-CNN or the alternative technique involves both the models to be merged into one network during training which is shown in Fig. 2 .

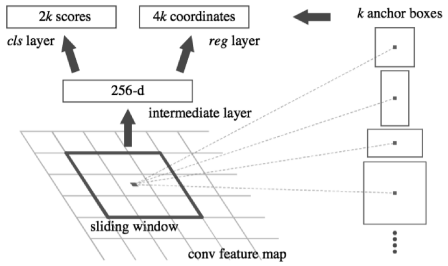


Fig. 2: Region Proposal Network (RPN)

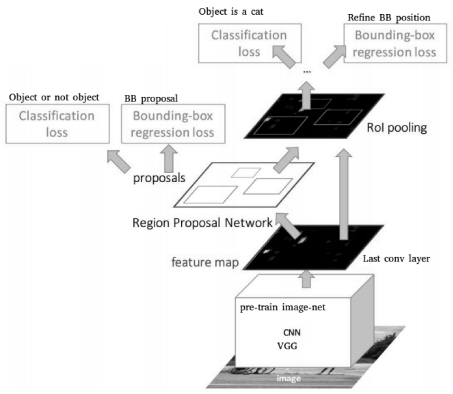


Fig. 3: Architecture of Faster R-CNN

R-FCN (Region based Fully Convolutional Network) R-FCN (Region based Fully Convolutional Network) consists of shared, fully convolutional architectures. The translation variance is incorporated into FCN. To do this a set

of position-sensitive score maps are constructed by using a bank of specialized convolutional layers as the FCN output. [2] The architecture mainly contains Region Proposal Network which extracts candidate regions. The Region Proposal Network used here are fully Convolutional. These features are then shared between the RPN and R-FCN (similar to [1]). The proposal regions constitute the RoIs, The R-FCN architecture classifies these objects into categories and background.

In the R-FCN all the weight layers are convolutional and takes in the entire image as shown in Fig. 4 . R-FCN ends with a position-sensitive RoI pooling Layer [2]. The last convolution layer outputs are aggregated in this layer. This layer generates scores for each region proposal, hence making this layer to learn specialized position-sensitive score maps. The main difference of R-FCN from R-CNN is that all the layers are fully convolutional. The learning mechanism is similar.

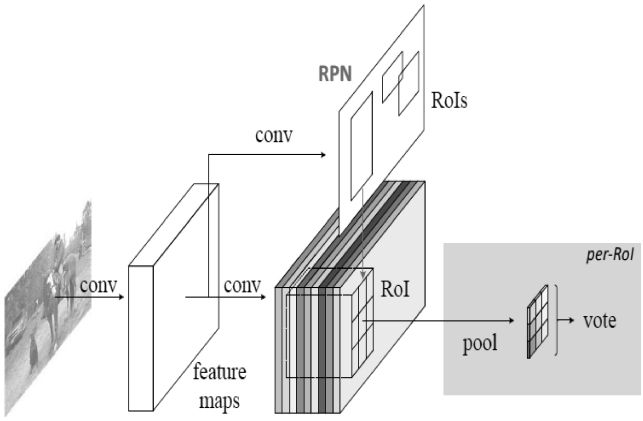


Fig. 4: Overall Architecture of R-FCN

2.2 Feature Extractors

Mobilenet MobileNet model uses depthwise separable convolutions. Depthwise separable convolutions factorizes a standard convolution into Depthwise Convolution and 1 x 1 convolution. The 1 x 1 convolutions are known as pointwise convolution [3]. The Depthwise convolution applies a single filter to each input channel. Pointwise convolution on the other hand applies 1x1 convolution where the outputs of depthwise convolution are combined. Pointwise convolution linearly combines the output from the depthwise Convolutional layer. Unlike basic convolution where the filters and inputs combine into a set of outputs in one step; in MobileNets the depthwise separable convolution splits this convolution

process into two different layers. The first layer does filtering, and the second layer performs the combination. This mainly helps in reducing the computational cost and reduces the Model size.Both batchnorm and ReLU nonlinearities are used for both layers [3].

In the MobileNet structure, all the layers are built based on depthwise separable convolutional Layers except the first layer which is full convolution [3]. All Layers are followed by batchnormal and ReLU nonlinearity. The final Layer is the exception which has no nonlinearity but is followed by a softmax Layer instead which helps in classification [13]. MobileNet models were trained in TensorFlow [14] using RMSprop [15] with asynchronous gradient descent similar to Inception V3 [3][16].

Inception v2 This model mainly aims at utilizing added computation as efficiently as possible to perform factorized convolutions and aggressive regularization. The convolutional networks can be scaled up in an efficient way by using Inception concepts. Th convolutions involving a kernel greater than size 3x 3 can be easily and efficiently performed using a series of smaller convolutions. [16] Further, factorization of convolutions and improved normalizations can be other tricks that are adopted to improve its efficiency.

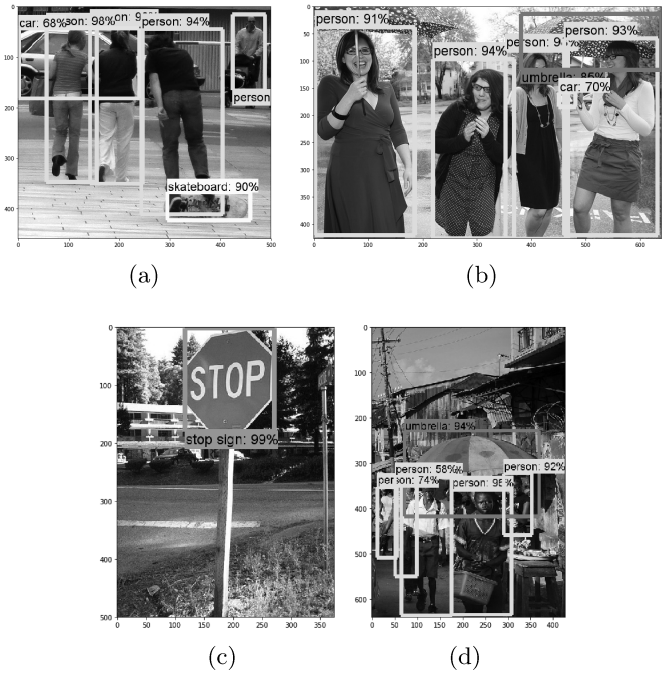
Inception networks are fully convolutional, and each weight corresponds to multiplication per activation. If the factorization is done properly the parameters can be more disentangled and hence can lead to faster training. Inception-v2 network is 42 layers deep. The computational cost is only about 2.5 higher than that of GoogLeNet and is more efficient than VGGNet [16].

3 Experimental Setup

The pedestrian detector is trained on the Penn-Fudan dataset [17]. This database contains images of pedestrians are taken from scenes around campus and urban street. Each image will have at least one pedestrian in it. There is a total of 170 images in which 80% are used for training while the rest are used for testing. The convolutional neural network models are pre-trained on the COCO dataset, the Kitti dataset, and the Open Images dataset and hence lesser data is required for re-training the models. The detections on COCO dataset based on these models are shown in Fig. 5. In case of the COCO dataset, the different models provide a trade off between speed of execution and the accuracy in placing bounding boxes as shown in table .

Further, in order to train the model, for each image, the width, height and each class with their bounding box parameters are required.Hence the input data is labeled manually using the graphical image annotation tool named LabelImg. It uses Qt for its graphical interface and the annotations for each image are saved as XML files in PASCAL VOC format. A labeled map associated with each of the datasets and this label map defines a mapping from string class name (person) to integer class ids which always start from id 1. During training, TensorFlow

uses the TFRecord format in order to optimize the data feed.Initially, the XML files are converted to CSV files which are further converted to TFRecord files.



(a) (b)

(c) (d)

Fig. 5: Sample detection on COCO 2017 test images

Table 1: Performance comparison for COCO dataset

Model name	Speed	COCO mAP	Outputs
ssd_mobilenet_v1_coco	fast	21	boxes
ssd_inception_v2_coco	fast	24	boxes
faster_rcnn_inception_v2_coco	medium	28	boxes
rfcn_resnet101_coco	medium	30	boxes
faster_rcnn_resnet101_coco	medium	32	boxes

The pre-trained models such as SSD MobileNet, SSD Inception, Faster RCNN Inception, R-FCN Resnet101 and Faster RCNN Resnet101 are chosen for analyzing the dataset.Each model has corresponding checkpoint files as well as configuration files for the training process. The pre-trained model using the COCO dataset was designed to work in 90 categories. In the configuration files, transfer

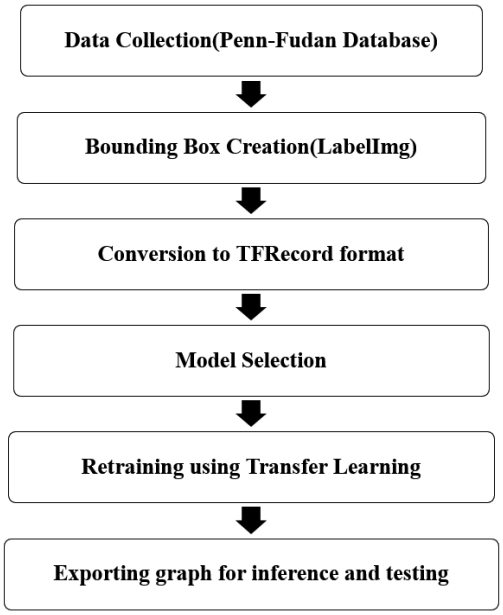


Fig. 6: Experimental flow chart

learning is done by removing the last 90 neuron classification layer of the network and replacing it with a new layer. Here this is implemented so that training will be quicker and the data required will be less.Thus the five pre-trained models are taken and the last layer is clipped off and replaced with the class of the current dataset.The SSD MobileNet and SSD Inception used a batch size of 10 while the rest of the models used a much lesser batch size of 3. Here each of the five models is trained for 5000 number of steps and these steps mainly depends on the size of the dataset. The initial learning rates are in the range of 0.002-0.004 for each of the models.

The fine-tuning of an existing model is highly accurate and easy since most of the features that are learned by CNNs are often object agnostic.Thus all the feature detectors trained in the previous model are used to detect the new class.When the loss function for each of the models is around 1 or starts rising the training is stopped.Finally, the graph for inference is exported and the newly trained model is validated using the remaining 20% test data as well raw Google images. These images are in the RGB format with varying sizes and are in .png image format. Many of these images contain multiple pedestrians and those holding many occluding object artifacts such as bags and umbrellas.Each of the trained models is tested at multiple checkpoints to see which one performs the best.The entire experimentation is shown in the flowchart in Fig. 6.

4 Results

4.1 Detection time on test images on CPU

The detection time per image was calculated for each model as shown in table 2 .It was seen that faster RCNN inception and SSD models are faster requiring around 2.5 s on average detection time per image. However faster RCNN resnet models has a higher computation burden and thereby require around 18 s.

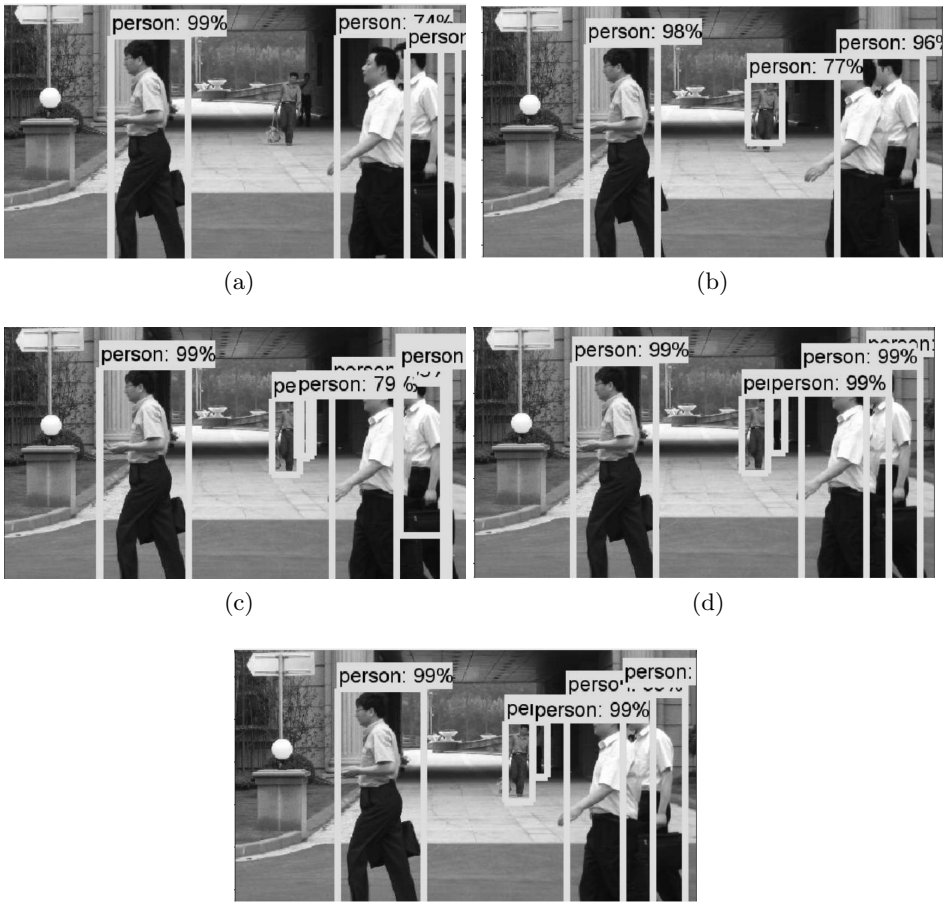
The overall mAP calculation of different models is shown in table 2 .It is observed that faster rcnn inception outperformed other models with a higher accuracy value.Based on the ranking of detection scores for each class the performance of the detectors are evaluated using mean average precision over entire test data.

Table 2: Speed accuracy trade-off

Model name	Time(s)	mAP
ssd_mobilenet	1	0.78
ssd_inception_v2	1.2	0.92
faster_rcnn_inception	4.7	0.96
rfcn_resnet101	7.31	0.95
faster_rcnn_resnet101	18.23	0.95

4.2 Sample Detections

The visualization of detections in the test images of the Penn-Fudan dataset can be seen in Fig. 7.Thus a comparative analysis is done between the 5 models to find the optimum detector for the pedestrian detection application. The Penn-Fudan dataset consist of different categories of pedestrians such people carrying umbrellas, suitcases and other occluding objects. There also exist variable cases such as multiple people on the same screen and overlapping scenarios. Thus sample detections done on this dataset can be considered to be highly comparable to real life situations. From the figures it is clear that all the models perform consistently well except that SSD is unable to detect pedestrians which are farther away and it also shows low detection rate in case of overlaps. Further, sample detection using Faster R-CNN Inception on a Google image is shown in Fig. 10.



(a) (b)

(c) (d)

(e)

Fig. 7: Example detection for (a) SSD Mobilenets (b) SSD Inception (c) R-FCN Resnet (d) Fast RCNN Resnet (e) Fast RCNN Inception

4.3 Total Loss Function

The loss function for the models is the total loss in doing detection and generating bounding box. Each of the models was trained for 5000 steps since the size of the dataset was comparatively small.The loss for SSD (both with ResNet 101 and Inception-v2) models started off with a value around 13 and converged around the value 2. In case of Faster R-CNN (both with ResNet 101 and Inception-v2) and R-FCN models, the loss function converged to a relatively smaller value around 0.2.

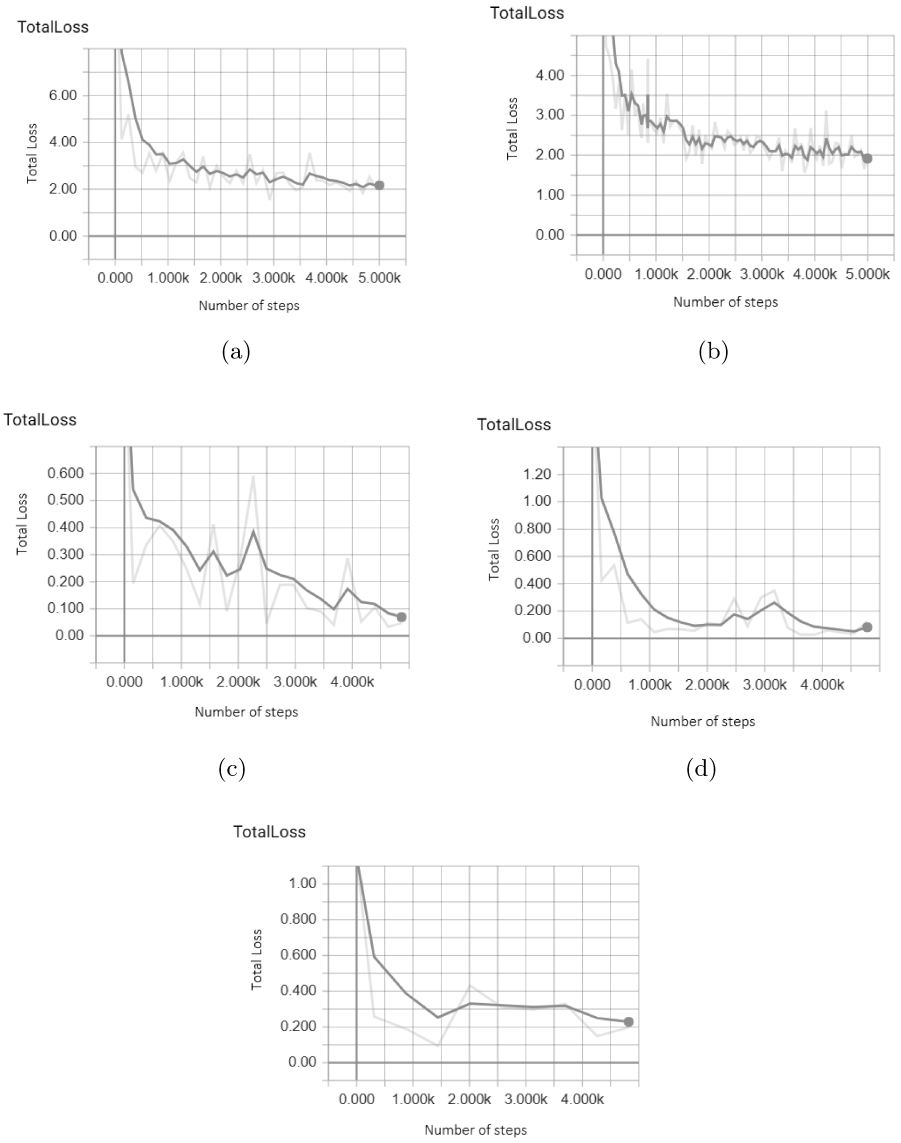


Fig. 8: Total loss for (a) SSD Mobilenets (b) SSD Inception (c) R-FCN Resnet (d) Fast RCNN Resnet (e) Fast RCNN Inception

Therefore, it can be seen that appropriate training was done for the latter models, till the loss function converged to a very less value, particularly below 1. The training loss corresponding to each of the models is shown in Fig. 8

4.4 Learning rate

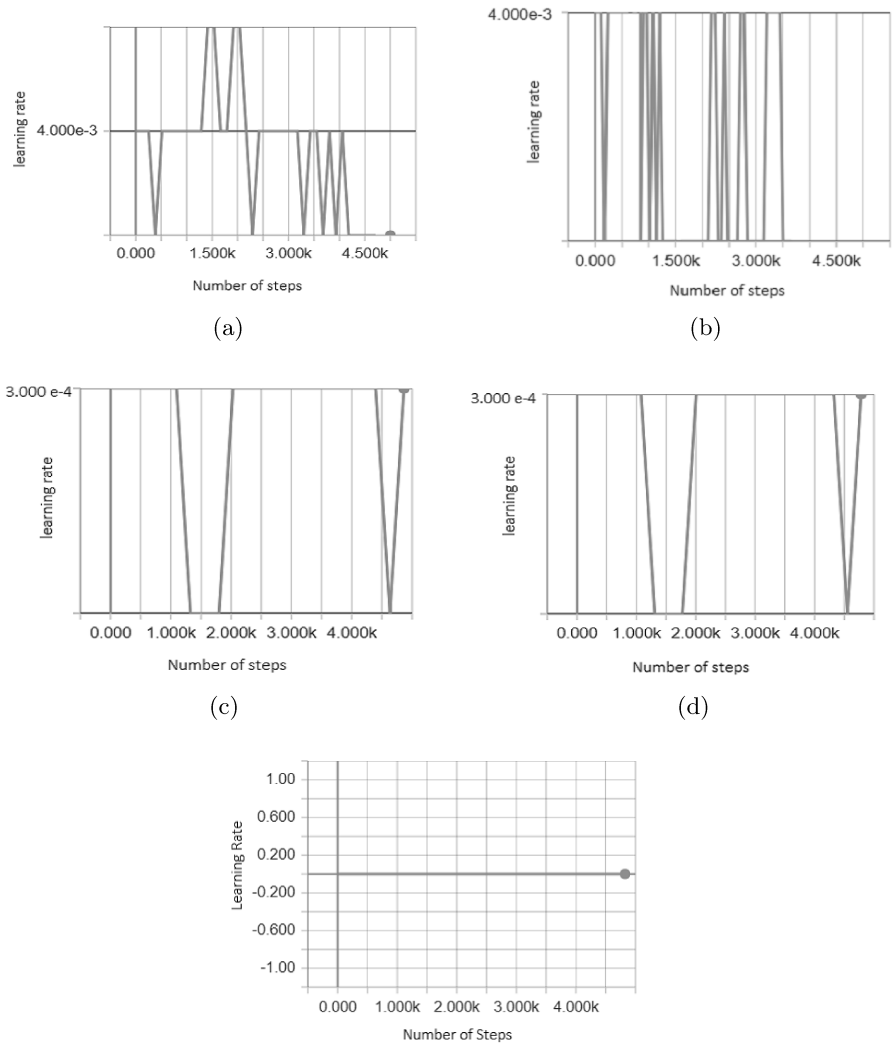


Fig. 9: Learning rates for (a) SSD Mobilenets (b) SSD Inception (c) R-FCN Resnet (d) Fast RCNN Resnet (e) Fast RCNN Inception

Learning rate is a hyper-parameter used for adjustment of the weights in the network with respect to the loss gradient. Learning rate is a measure of travel down the slope to reach the local minima and it decides how quickly a model can converge to a local minima. From the above figures it is clear that better accuracy of Faster R-CNN Inception is due to its high learning rate compared to rest of the models. Lower value for learning rate means, the descent is slower along the slope. With the perfect learning rate a lesser training time can be achieved. The learning rates for different models are shown in figure 9.

Fig. 10: Detection using faster R-CNN on a Google Image

5 Conclusion

An experimental analysis was done on the modern object detection model to find the appropriate trade off in speed and accuracy. It was observed that Faster RCNN Inception model gave the most optimal values in speed and accuracy for the application of pedestrian detection. It was able to detect even far away people with overlaps with an accuracy of 96%. In addition,the rfcn resnet 101 and faster rcnn resnet 101 gave comparable accuracies but the test time was found to be very high.Thus the pedestrian can move out of the frame by the time the model tries to detect it.The SSD models were much faster to train but the detection rate was lesser compared to other models and occluded people were much difficult to detect using this model.

References

1. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. (2015) 91–99
2. Dai, J., Li, Y., He, K., Sun, J.: R-fcn: Object detection via region-based fully convolutional networks. In: Advances in neural information processing systems. (2016) 379–387
3. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
4. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European conference on computer vision, Springer (2016) 21–37
5. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. In: european conference on computer vision, Springer (2014) 346–361
6. Girshick, R.: Fast r-cnn. arXiv preprint arXiv:1504.08083 (2015)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 770–778
8. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
9. Erhan, D., Szegedy, C., Toshev, A., Anguelov, D.: Scalable object detection using deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2014) 2147–2154
10. Szegedy, C., Reed, S., Erhan, D., Anguelov, D., Ioffe, S.: Scalable, high-quality object detection. arXiv preprint arXiv:1412.1441 (2014)
11. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
12. Chorowski, J.K., Bahdanau, D., Serdyuk, D., Cho, K., Bengio, Y.: Attention-based models for speech recognition. In: Advances in neural information processing systems. (2015) 577–585
13. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
14. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al.: Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467 (2016)
15. Tieleman, T., Hinton, G.: Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning **4**(2) (2012) 26–31
16. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 2818–2826
17. Wang, L., Shi, J., Song, G., Shen, I.f.: Object detection combining recognition and segmentation. In: Asian conference on computer vision, Springer (2007) 189–199