REPORT "CAN YOU STILL SEE ME?"
DATA SCIENCE PRACTICAL
WINTER SEMESTER 2019/20

CAN YOU STILL SEE ME?

Diana Davletshina: d.davletshina@campus.lmu.de
Hitansh Singla: hitansh.singla@campus.lmu.de

In cooperation with HARMAN International
Supervisor: Prof. Dr. Matthias Schubert
Mentor: Dr. Tobias Emrich

26th February 2020

Contents

1 Introduction

Nowadays autonomous car driving is one of the technologies that is developing rapidly. The research field that made this development possible is Computer Vision. Now it is the core technology in autonomous navigation, since it gives a sense of perception and navigation to computer programs, bringing along several challenges. In the article [1], Martinez-Diaz and Soriguera describe various problems that are faced by researchers in the field of Computer Vision. They notably highlight object detection as a crucial process. Since it is desirable to carefully interact and avoid hitting objects on the street and particularly moving vehicles, object detection task, as the video stream analysis, should have high accuracy, preventing the car from disorientation.

Artificial neural networks (NN) are considered as the most powerful technique to perform image analysis [2]. According to Deng et al., training dataset size has a significant impact on the final quality of a neural network performance [3]. Therefore, large datasets are needed to achieve better results. To reduce disk space occupation and network traffic, engineers might convert images to different file formats that allow compression, resulting in smaller data file size. Unfortunately, high compression implies image quality degradation, which can affect the accuracy of object detection and recognition.

However, with the development of the hardware installed on the autonomous driving cars, the utilisation of resources such as storage size is increased. Having front-facing cameras installed on 10 cars, the company Harman [4] end up with about one petabyte of high-quality image data per month, which is expensive to store. Motivated by this problem, in this work, the effect of image compression on the state-of-the-art object detection neural networks is investigated. The main objective of the work is to find out how much data can be compressed such that accuracy of an object detection task is preserved, i.e. whether is it necessary to train neural networks with high-quality image data. Among the minor goals are the impact of low resolution and finding possible improvements to cope better with image compression.

In the frame of the "Can you still see me?" project, we explore the real-world dataset provided by the company Harman and investigate the mentioned earlier goals on the given data. In the report, we introduce the data, explore related works in the area, describe an applied methodology and a developed solution pipeline. Finally, we summarise the results and highlight various findings.

2 DATA

Figure 2.1: Examples of image frames with ground truth labels

In the frame of this project, 58 thousand labelled images were provided by the company Harman. Images were obtained from front-facing wide-angle (120°) cameras installed on cars driving on city streets with the frequency of frames is three per minute. The images are in a raster graphics image file format *.bmp*. It is worth to notice that the lighting conditions on the images are different: we can observe sunny/cloudy daylight or evening twilight. A couple of examples can be seen on figure 2.1.

The labels distributed between 7 classes: Passenger Car, Truck, Van, Bus, Pedestrian, Bicycle and Motorbike. Looking at the label's distribution on figure 2.2, highly unbalanced situation is observed: about 80% of all the labels belong to Passenger Car class, while Motorbike class has only 121 labels, which is less than 0,05%.

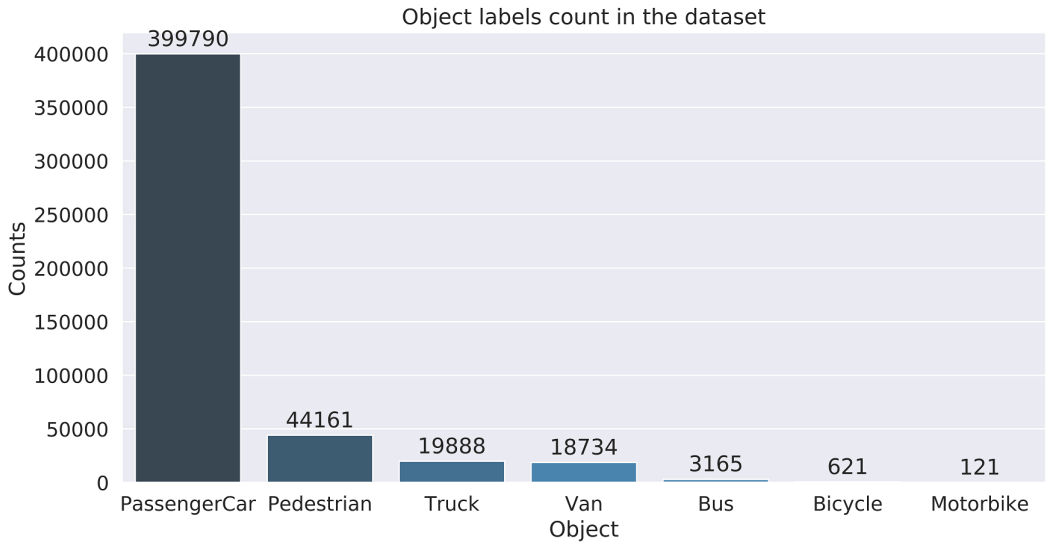


Figure 2.2: Labels distribution among objects

Labels are provided along with bounding boxes around the objects. The average object size can be seen in figure 2.3. The mean area trend corresponds to the median of the area as

well. We can see that the largest objects are from Bus and Truck classes, while the smallest are from Pedestrian and Bicycle classes.

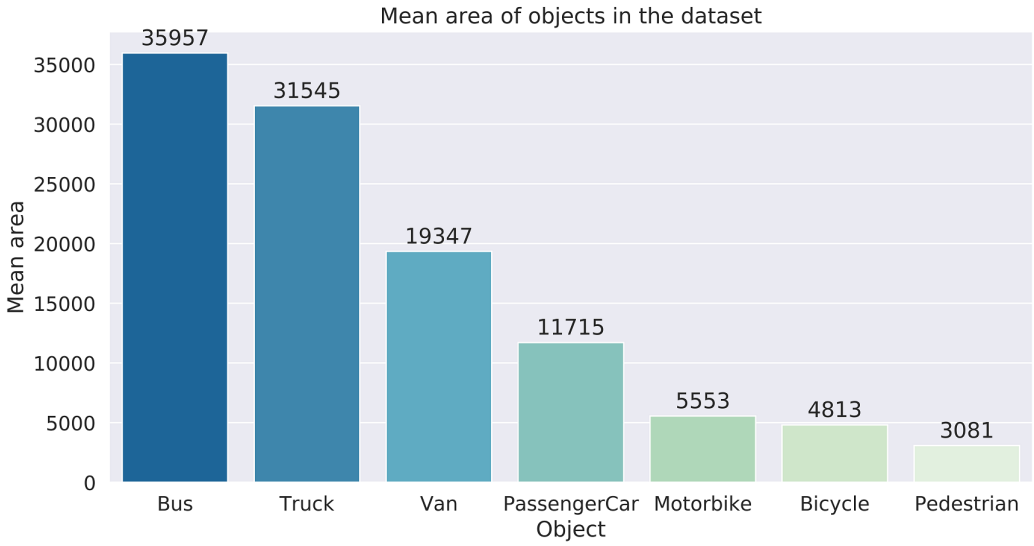


Figure 2.3: Average object area on frame

3 Literature overview

This section gives an overview of the impact of image compression, low resolution and other various image transformation techniques on image processing quality. A comparison of the influence of commonly used image formats, such as different variations of lossy JPEG, as well as an effect of different types of image distortion are discussed. Additionally, several approaches to improve a compressed image processing pipeline to restore the accuracy of a neural network model are reviewed.

3.1 Low resolution performance

The most basic approach to reduce picture file size is to decrease its resolution. Koziarski and Boguslaw in their work have researched the influence of low resolution on the accuracy of object recognition using the artificial neural network [5] models. They considered three popular network architectures for image processing: AlexNet, VGGNet, and ResNet. The authors trained the mentioned models on original images and calculated the performance metrics on them. Then, they converted dataset to different levels of lower resolution and repeated the training and evaluation processes. After comparison of results, Koziarski and Boguslaw found that despite network architecture, the final accuracy decreases with picture size. However, in some scenarios, when they account for the low-resolution problem, they can prevent a decrease in accuracy. As an example, the authors mentioned that

applying super-resolution to an image with low resolution allows achieving classification accuracy close to that with original size image [5].

3.2 Image compression

Another common approach to reducing disk space occupation is to convert pictures to compressed image formats. One of the most popular formats is JPEG, which allows to significantly decrease image size by decreasing image quality [6]. Zanjani et al. analysed how image compression affects performance in neural network classification [7]. They trained different variations of VGGNet architecture on uncompressed data. Then, the F1 score and AUC (Area Under the Curve) values of images with different levels of compression were evaluated. Based on their experiment, pictures with similar-to-original quality (up to a compression factor of 24) have close values of target metrics. However, when factor exceeded 24, both metrics the F1 score, as well as AUC, declined. This result is confirmed by Dodge and Karam. According to their paper, for JPEG images with a similar quality parameter as in the above experiment, a neural network performance remained the same [8]. Likewise, similar results were also obtained by Dejean-Servieres et al., where authors conducted an analogous experiment with AlexNet model [9].

Moreover, Zanjani et al. conducted another experiment, where they trained an artificial neural network on a compressed dataset. In contrast to the results of the previous experiment, target performance metrics were much closer to the original values [7]. This can be explained by the adaptation of the neural network to the specific features of changes caused by compression transformation.

3.3 Image distortion

Besides the research about the influence of compression on classification, Dodge and Karam investigated the correlation between different picture distortions and neural network performance. They considered two types of image transformation: blur and noise. In the first experiment, the authors blurred the pictures in a test dataset by applying Gaussian kernels of different sizes, which made pictures smoother. With such modification of pictures, neural network accuracy dropped dramatically [8]. The potential explanation given by the authors is based on the assumption that neural networks rely heavily on textures in object classification, which get erased after blurring. Another experiment conducted by the Dodge and Karam examined the influence of noise in the context of an object detection task using neural networks. After applying small changes at the pixel level, the authors evaluated the target performance metrics in the same pipeline. The experiment demonstrated that the noise has a smaller impact on the final results, as it preserves the textures of objects, which is consistent with the explanation for blurred images.

4 Data Compression

As it was mentioned in the earlier section, the original format of images is *.bmp*, which is capable to preserve many aspects of image data in terms of colour, various colour depths, data compression, alpha channels, and colour profiles. For this work, we apply commonly used JPEG compression [10]. Although this compression technique is lossy, meaning that some original image information is lost and cannot be restored, affecting image quality, it can significantly reduce disk occupation size, preserving visual indistinguishability for low and average compression rates [6].

In this work, we investigate the following JPEG compression qualities: 100, 40, 30, 20, 10, 5, and 1 per cent. On the fig. 4.1 we can see a significant drop in data size already going to 100% JPEG and even larger drop (98.54%) for 40% quality data.

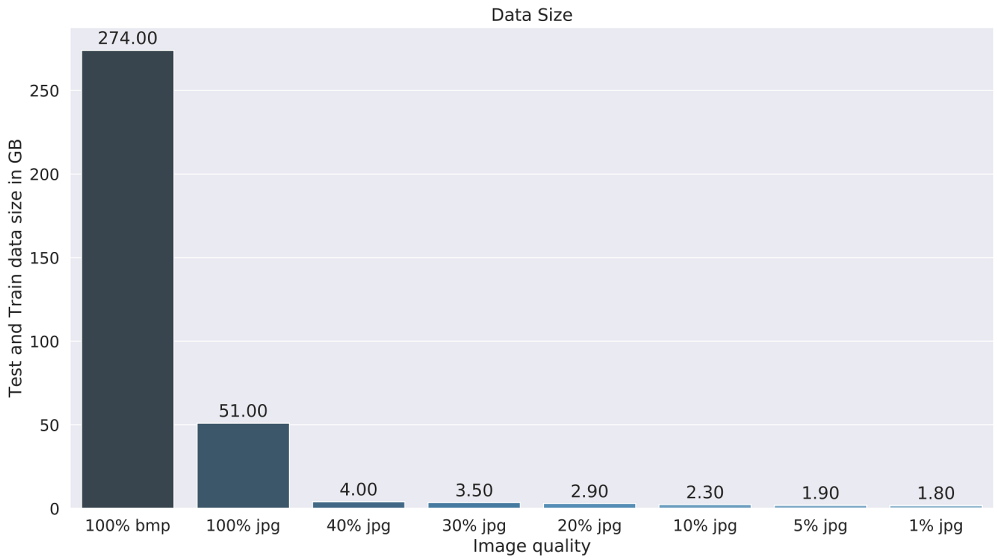


Figure 4.1: Data size reduction with different JPEG compression levels

5 Transfer Learning

Instead of learning a deep neural network from scratch which could take days, we can harness the power of an already-trained neural network and modify it to work for our own data set. These neural networks have the layers which are used to identify the curves, lines, outlines and other features. Since these layers require a lot of training data, we can save time by feeding our dataset into further training those modified neural networks. This works by first removing the “loss output” layer, which is the final layer to make predictions, and replacing it with a new loss output layer for our dataset’s classes. Then we take our dataset and training on that new neural network by applying these transfer learning strategies [11].

5.1 Pre-trained Neural Networks

In this project, we have used 3 different types of pre-trained neural networks:

- SSD Inception V2 COCO,
- Faster R-CNN Inception V2 COCO,
- Faster R-CNN Resnet-101 Kitti.

5.1.1 SSD

Single Shot MultiBox Detector (SSD) [12] refers to an architecture that uses a single feed-forward Convolutional Neural Net (CNN) to directly predict classes and anchor offsets without requiring a second stage per-proposal classification operation. SSD generates default bounding boxes with different aspect ratios and scales per feature map location. For prediction, the network generates scores for the presence of each object class in each default box and produces adjusted boxes to better match the object shape. Moreover, the SDD network collects predictions from multiple feature maps with different resolutions to detect objects of various sizes [12].

The "SSD Inception V2 COCO" model was pre-trained on the COCO dataset [13]. COCO is a large-scale object detection, segmentation, and captioning dataset. It has over 200 thousand labelled images, over 1.5 million object instances and 80 object categories.

5.1.2 Faster R-CNN

Faster R-CNN [14] (FRCNN) is a combination of R-CNN and Fast R-CNN. An R-CNN is a region-based convolutional neural network consisting of 3 steps. It first scans the input image for possible objects using an algorithm called Selective Search, generating 2000 region proposals. Then it runs a CNN on top of each of these region proposals. At last, the output of each CNN is taken and fed into a Support Vector Machine (SVM) to classify the region and into a linear regressor to tighten the bounding box of the object, if such an object exists.

To improve the detection speed, Fast-R-CNN model is used, which alters the R-CNN model by performing feature extraction over the image before proposing regions, thus only running one CNN over the entire image instead of creating many separate CNNs. Then the SVM is replaced with a softmax layer, thus extending the neural network for predictions instead of creating a new model [14].

To create a Faster R-CNN model, in the last layer of an initial CNN a 3x3 sliding window moves across the feature map and maps it to a lower dimension. For each sliding-window location, it generates multiple possible regions based on the certain number fixed-ratio bounding boxes. Each region proposal consists of an object score for that region and four coordinates representing the bounding box of the region. Once region proposals are obtained, they are fed straight into what is essentially a Fast R-CNN. A pooling layer, some fully-connected layers, finally a softmax classification layer and bounding box regressor are added [14].

The inner architecture of Faster R-CNN can be built by using multiple methods. We used two of the methods, namely Inception V2 [15] and Resnet-101 [16].

The "Faster R-CNN Resnet-101 Kitti" model was pre-trained on the KITTI dataset [17], which consists of images of cars and pedestrians, collected in the city of Karlsruhe, Germany. Up to 15 cars and 30 pedestrians are visible per image. The object detection and object orientation estimation benchmark consists of 7481 training images and 7518 test images, comprising a total of 80.256 labelled objects.

6 Solution pipeline

In order to perform all of the experiments, a sophisticated solution pipeline was developed (fig. 6.1), starting from data preprocessing and finishing with evaluation process. Our main technological stack was built on Python 3.7 and TensorFlow 1.12. The developed project can be found in a Github repository [18].

Data preprocessing mainly includes filtering out broken images (about 800 broken images were found) and conversion into grey-scale on demand. Then the full dataset is divided into training and testing sets by 90% and 10% respectively. Those steps are only done once for all experiments. For each compression level, we can run a compression script to get compressed training and testing datasets.

To create TFRecords (TensorFlow .record files to feed into NN), besides data, we need to have annotations of classes and bounding boxes. Annotations parser was developed to parse provided labels in .xml format. After that, we need to convert annotations into .csv format to be able to create TFRecords. Parsing the annotations and conversion into .csv format are done only once for all the experiments. TFRecords are created each time for newly compressed images.

When the above steps are done, the training process can be started, after which evaluation on test data needs to be performed to get final performance metrics.

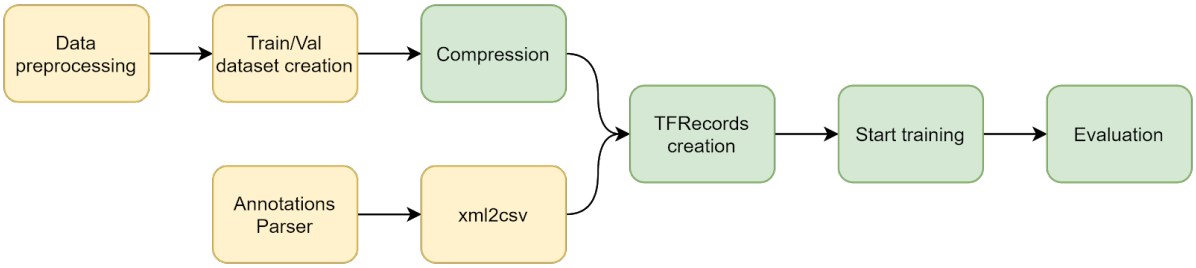


Figure 6.1: Solution pipeline

7 Experiments and Results

In this section, we see how the state-of-the-art object detection NN models perform on lower quality data. First, we start by describing the setup of the experiments and then proceed with their results. We describe evaluation metrics and obtained scores for each model. Moreover, we will investigate how the transformation to grey-scale affects image compression and, as a result, the accuracy of a model. Some performance insights are going to be explored as well.

7.1 Setup of Experiments

We performed the experiments on an instance of the server, provided by the Leibniz Supercomputing Centre, located in Garching, Germany. On table 7.1 we provide the characteristics of the machine the experiments were run on.

VCPUs	20
CPU Model name	Intel Xeon Skylake Processor
CPU Frequency	2.4 GHz
RAM	368 GB
Disk Space	2TB
GPU	Nvidia Tesla v100.1
GPU Memory	16GB
OS	Ubuntu 18.04

Table 7.1: Machine characteristics

The following NN model settings were used for running the training experiments:

- SSD Inception v2 (COCO Dataset)
  - Resolution: 300x300, 600x600
  - Batch size: 128 (for 300x300 configuration), 32 (for 600x600 configuration)
  - Steps: 15000
  - Training time: ~12 hours
- Faster R-CNN Inception v2 (COCO Dataset)
  - Resolution: 1820x940
  - Batch size: 16
  - Steps: 20000
  - Training time: ~6 hours
- Faster R-CNN ResNet101 (KITTI Dataset)
  - Resolution: 1820x940

- Batch size: 4
- Steps: 20000
- Training time: ~3 hours

For Faster R-CNN models, batch sizes are 16 and 4, which are much smaller than the SSD model's batch size. The reason is by using original resolution, we increase usage of resources. Therefore, batch sizes are chosen to optimally utilise the machine resources.

The above model settings were run on original .bmp image files and .jpg qualities of 100%, 40%, 30%, 20%, 10%, 5% and 1% image files.

7.2 Evaluation of performance

The section is devoted to the evaluation of the NN models' performance. We will describe evaluation metrics, performance on different quality data, performance on grey-scale images and different factors, which might affect performance. On figure 7.1 we can see a sample image over different compression rates, first visualising annotated bounding boxes (top left corner), followed up by predicted bounding boxes by Faster R-CNN Inception V2 model for 100%, 40%, 30%, 20%, 10%, 5% and 1% quality JPEG images. As for labels, it is important to mention that only object larger than 20x20 pixels on a frame were annotated, while the networks were able to detect them.

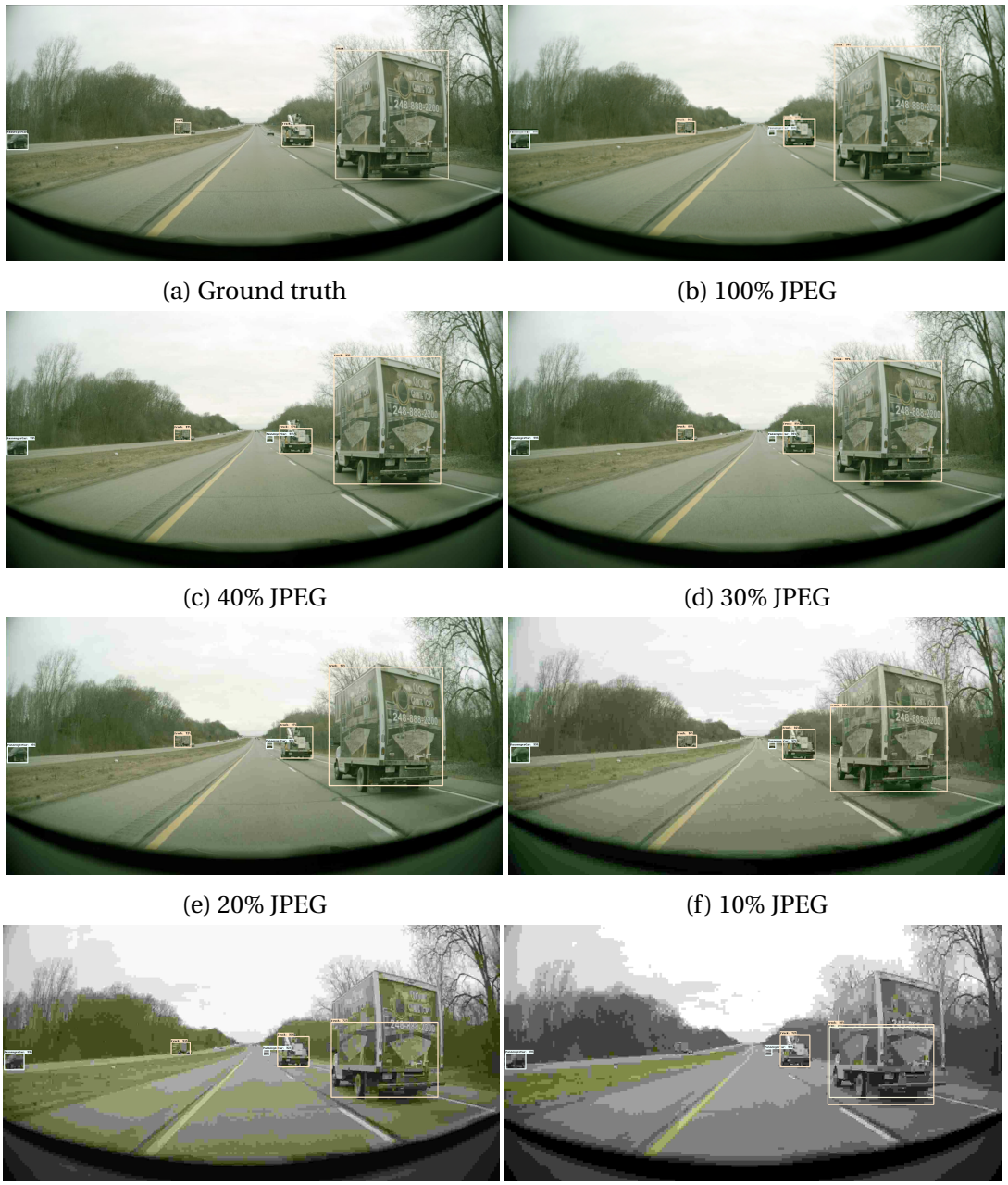
7.2.1 Evaluation metrics

An object detection task is composed of detection an object by localising it (bounding boxes) and classifying it into a predefined class (labels). In the classification part, the common metrics are *precision* and *recall*, which are able to tell about how many items were correctly classified among all and how many correct items were correctly classified, respectively. For localising an object on image, often bounding boxes are used. Having a ground truth bounding box and a box predicted by a NN, we can calculate an area of overlap.

The idea above gives rise to a metric called *Intersection over Union*, or IoU, which is a ratio of the overlapping area over the union of them. According to practical experience, if IoU > 0.5, then the prediction is considered to be correct. Thus, a metric *mean Average Precision*, or mAP, is a mean over different classes' average precision values. This metric is utilised in many related works [19] [20] as a final evaluation score. Similarly in this work, following the best practices in the field, we use *mAP* to assess object detection NNs performance. Due to the unbalanced class labels in our scenario, a weighted version of the mAP metric is used.

7.2.2 Effect of Image compression

Figure 7.2 describes the effect of image compression (quality of the image) on the overall weighted mAP results for four different types of model settings. Overall, we can observe the following trend: lower quality results in smaller mAP values. However, for .jpg format qualities until 20% there is no significant drop in accuracy. While for 40% we have similar mAP values, for 30% we have only a slight decrease.



(a) Ground truth (b) 100% JPEG

(c) 40% JPEG (d) 30% JPEG

(e) 20% JPEG (f) 10% JPEG

(g) 5% JPEG (h) 1% JPEG

Figure 7.1: Top left: original bounding boxes, followed row by row decreasing quality from 100% to 1% JPEG quality, where bounding boxes predicted by FRCNN Inception V2 model

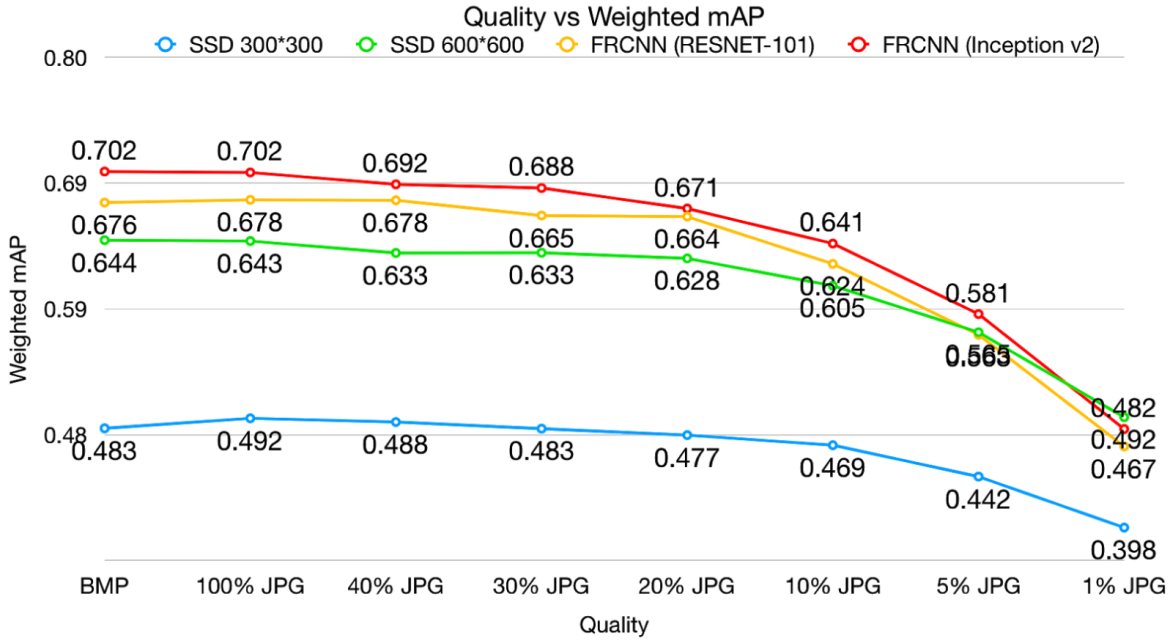


Figure 7.2: Weighted Mean Average Precision over models a) SSD with 300x300 configuration b) SSD with 600x600 configuration c) Faster R-CNN ResNet101 d) Faster R-CNN Inception v2

7.2.3 Effect of Image resolution

Among others, on figure 7.2 we can see particularly SSD Inception v2 COCO with a 300x300 input layer and SSD Inception v2 COCO with a 600x600 input layer. We can clearly observe an increase in accuracy when we increase the size of the input layer by 4 times. However, this resulted in an increased number of parameters to learn, hence a reduction in batch size from 128 for 300*300 to 32 for 600*600 was made, in order to fit the batch in GPU memory.

7.2.4 Effect of Grey-scale images

The idea behind applying grey-scale transformation before compression is that it might help to preserve edge and corner features better since there is no need to maintain colour information. Figure 7.3 shows examples of grey-scale and colour images for 5% and 40% compression qualities. However, visual investigation of several images shows no evidence of the above hypothesis.

Conversion to grey-scale was done using a default image library method, namely ITU-R 601-2 luma transform [21].

(a) Grey-scale 5% JPEG (b) Grey-scale 40% JPEG

(c) Colour 5% JPEG (d) Colour 40% JPEG

Figure 7.3: Examples of grey-scale and colour images for 5% and 40% qualities

Table 7.2 shows the weighted mAP values for 4 different model settings. In 40% .jpg setting, decreasing input channels from 3 (RGB) to 1 (grey-scale) resulted in no change in accuracy, while in 5% .jpg setting, decreasing the input channels led to a slight decrease in accuracy (0.6%).

SSD 600x600	**Color image**	**Grey-scale image**
40%	0.632856	0.635504
5%	0.565084	0.559135

Table 7.2: Effect of training on grey-scale images on accuracy

7.2.5 Effect of Increased training time

In order to see whether NN models still have the potential to learn, we run another experiment with the Faster R-CNN Inception V2 model for a longer time. The table 7.3 shows the summary of the experiment. As we can see, by increasing the number of steps 10 times we gained 8% accuracy.

Faster R-CNN Inception V2	**20.000 steps**	**200.000 steps**
Training time	~6 hours	~60 hours
Weighted mAP	0.6915	0.772963

Table 7.3: Effect of increased training time on accuracy

7.2.6 Comparison of results

Figure 7.4 shows the min-max normalised version for comparing the drop in accuracy with the drop in storage space. The purple-coloured line shows that going from BMP to 100% .jpg, the drop was significant, while all models performed similar (some even better) during the same transition. The same drop was seen when going from 40% .jpg to 30% .jpg, however, reduction in storage space was not as high, as compared to the previous transition. Going further below in quality shows no significant drop in storage space, but major drops in weighted mAP values.

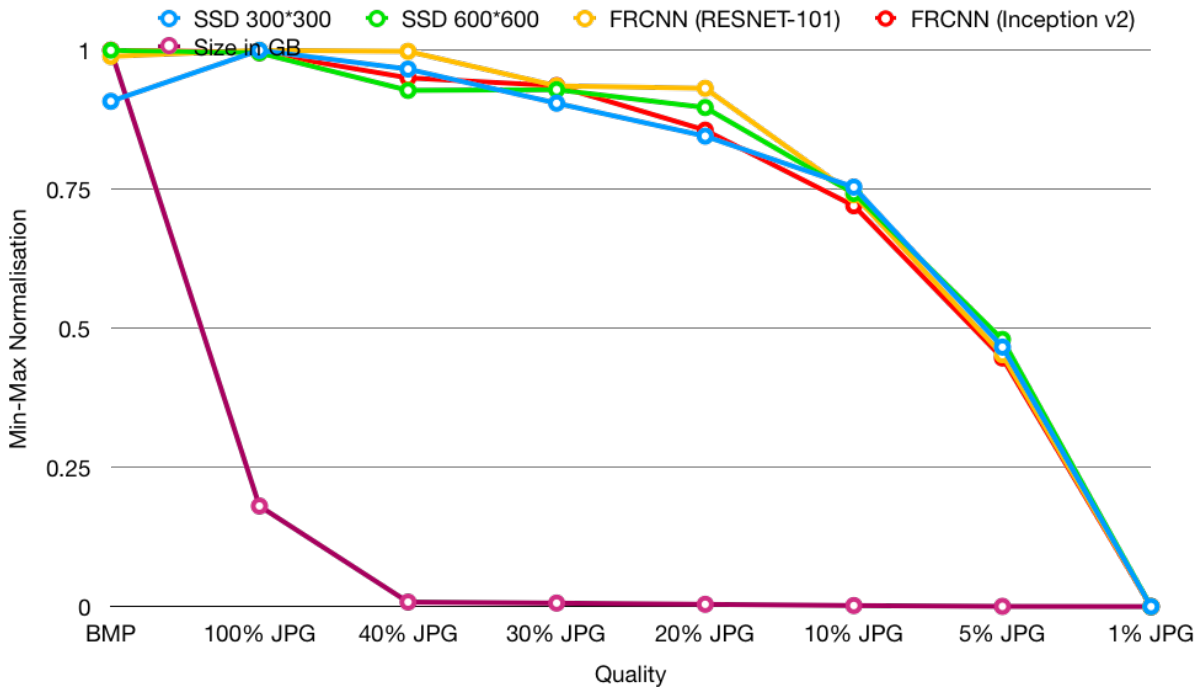


Figure 7.4: Min-max normalised mAP, data size plot

7.3 Performance insights

The section is devoted to the performance insights of the NN models. We will see how different factors, such as particular objects, count of labels, quality of images, mean area and a batch creation method affect the overall precision of a model.

7.3.1 Object-wise precision

On figures 7.5 - 7.8 we can see average precision metrics for each object category. Overall, for all models, we see that the highest precision is achieved for Passenger Car class, which has also the most labels in the annotations. In general, the trend of average precision goes down with the decreasing number of labels (see fig. 2.2 for reference).

However, it is possible to observe the untypical situation of increasing precision for decreasing quality. For example, the described behaviour can be noticed in figure 7.5 for Bus

class. We see that after 40% JPEG compression, average precision is increased for 30% and 20%. The reason behind might hide in the pre-trained dataset, namely COCO dataset for this particular model. Since COCO dataset contains different quality images, it is possible that Bus class was in average lower quality there, so the NN model learned lower quality features representing Bus class.

It is out of the existing trend, that for figure 7.8 we can see a significant drop for Bus class precision. Among the reasons to explain this behaviour, there are the following arguments: the number of labels available and batch creation procedure. Since the particular model was trained on the KITTI dataset, which has only Passenger Car and Pedestrian objects, specific features of other classes are learned from our Harman dataset. Looking at figure 2.2, we see that Bus class is 6 times smaller in the number of labels than Truck and Van classes are. Therefore, having a batch size for this model equal to four, the Bus class has a much smaller chance to be selected randomly into training batch, resulting in that the model was not able to see enough of busses and learn their features.

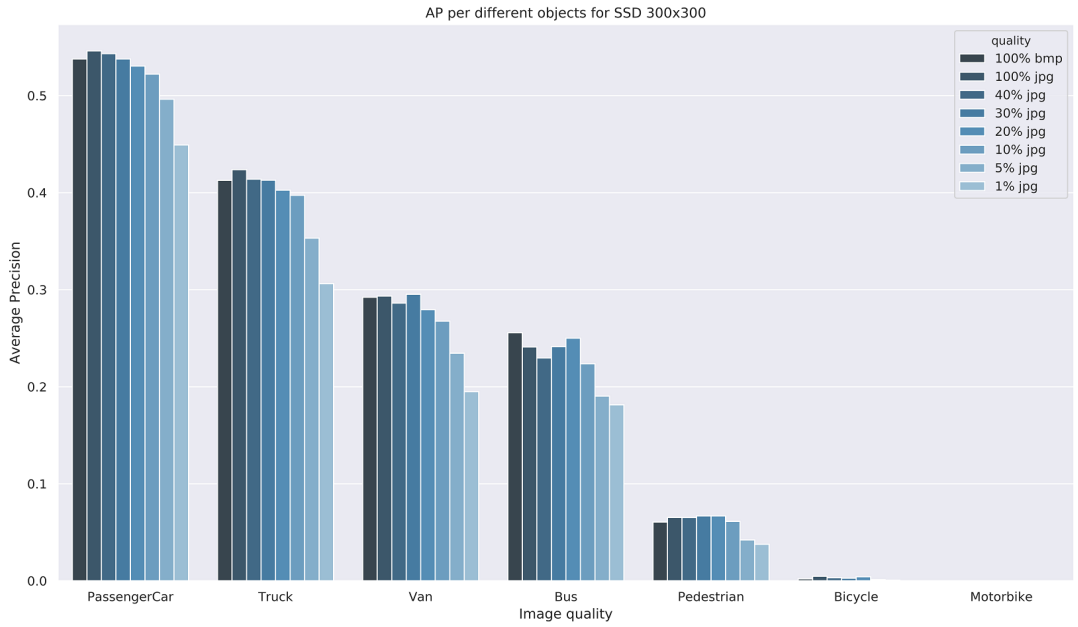


Figure 7.5: SSD 300x300 object-wise average precision

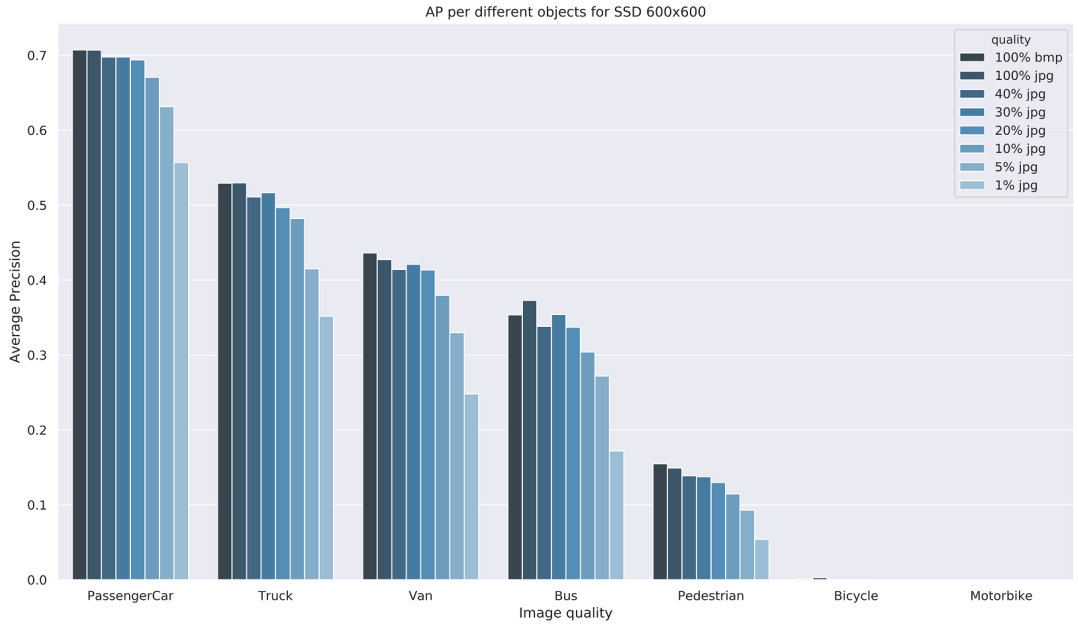


Figure 7.6: SSD 600x600 object-wise average precision

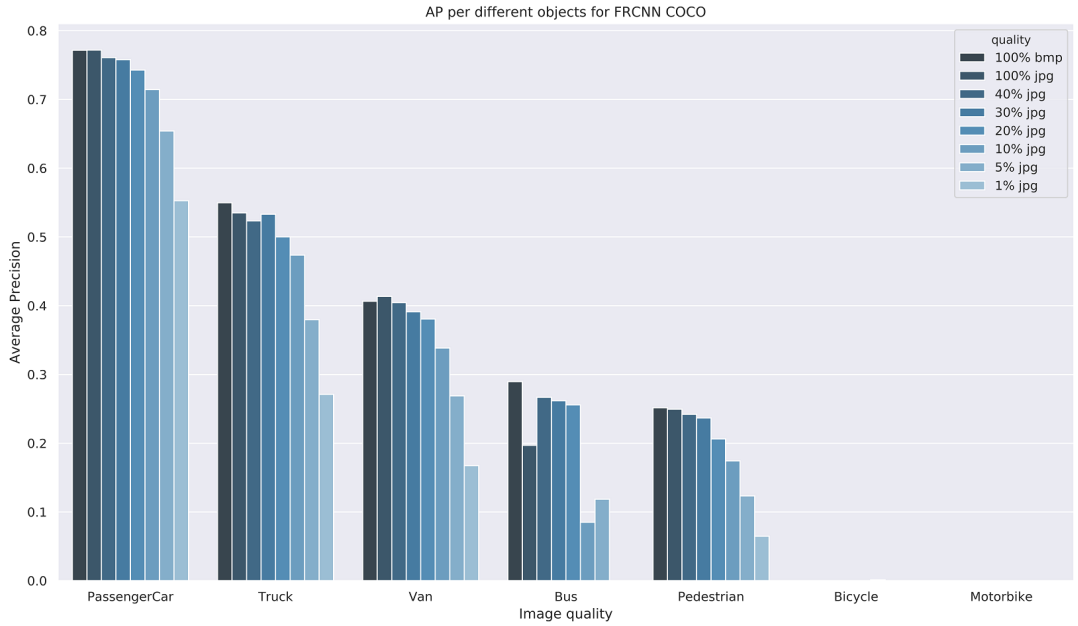


Figure 7.7: Faster R-CNN Inception V2 object-wise average precision

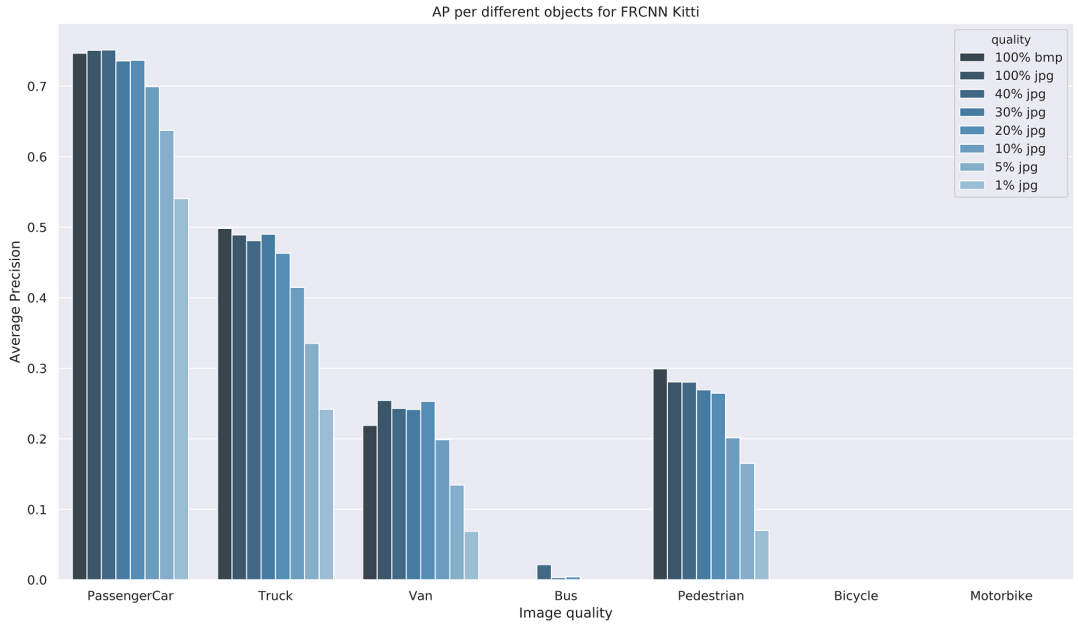


Figure 7.8: Faster R-CNN ResNet101 object-wise average precision

7.3.2 Influence of number of labels and mean area

On figure 7.9 influence of count of labels on the average precision for every object class and different quality is shown. Overall, we observe a growing trend: with more number of labels and higher quality of image we have higher precision. However, in case of number of labels, the above behaviour does not hold for Pedestrian class, where we see a significant drop in precision for all the models. Average object area (2.3) might reveal the reasons for this behaviour. Since the Pedestrian object class is the smallest objects in the frames, the NN models find it difficult to distinguish pedestrians, especially if they are small (e.g. in the background, as they usually appear in our dataset).

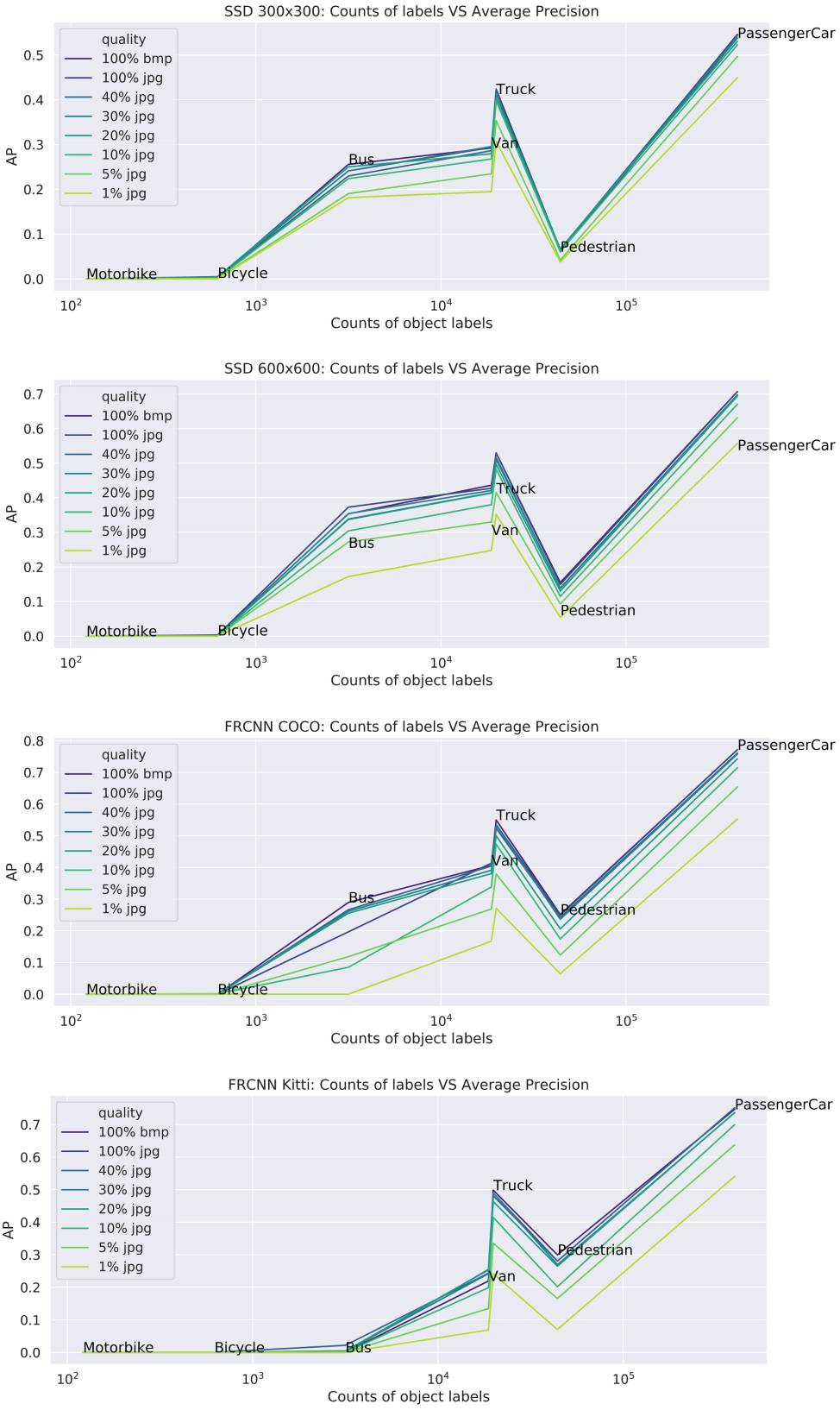


Figure 7.9: Count of labels in log scale VS Average Precision for SSD 300x300, SSD 600x600, Faster R-CNN Inception V2 and Faster R-CNN ResNet10 models

8 FUTURE WORK

To analyse the effect of image compression on object detection accuracy, one needs to consider different image compression techniques as well. Especially, the compression methods implemented using neural networks are very promising nowadays [22]. There might be possibilities to develop such neural network architectures that cope better with reserving image features for object detection. As for JPEG compression, we considered different compression ratios for the fixed type of JPEG image format. It took more than 350 hours of training (35 trainings multiplied by 10 hours per training on an average) for different types of experiments. Considering another compression technique would add even more training time, which was beyond the scope of this project.

There were some fluctuations observed in category-wise average precision score. For example, mAP for a Bus class decreased from 100% .jpg to 40% .jpg, but increased a little from 40% .jpg to 30% .jpg. To answer these problems, a cross-validation approach could be implemented. However, again, due to lack of available machines and limited training time, we did not perform cross-validation, which might help to obtain more stable model performances. However, we fixed the train-test splits for each of the experiments to perform a fair comparison of models. Also due to the same reasons, the training time of each model was limited to a maximum of 20,000 steps. We went ahead in one experiment and trained it for 200,000 steps, which increased accuracy. Hence we can also perform a comparison of different compression ratios with accuracy metrics when the models are trained for a longer time.

Additionally, as the GPU memory was 16GB, it could only fit a limited amount of image tensors in the memory. This restricted us to perform the experiments on a bigger input layer. For example, when the input layer size was 300x300, then we were able to create a batch of size 128. But when the input layer size was increased to 600x600, the batch size was also reduced by 4 times, which is just 32. Hence in future, if the technology allows, then bigger batches could be created. Apart from the size, batches are created randomly. This could affect the training when the classes are imbalanced. For uniformly training the model, we can either create a batch with uniformly taking images from each class or down-sampling/upsampling the data. Instead of altering the training data, we calculated mean average precision score as a weighted score, which considers the frequency of each object in a class.

9 CONCLUSION

After looking at the results, we can conclude that there was a 98.54% decrease in storage space from BMP to 40% .jpg images, but no decrease in detection accuracy at all. This behaviour is typical to all the different architectures we trained our model on. A maximum of 1% decrease in accuracy was observed for FRCNN Inception V2 architecture. Similarly, going from BMP to 30% .jpg compression, we saw a decrease of 98.7% storage space, but the accuracy results were similar to the models trained on 40% .jpg compressed images.

Going further below to the image qualities like 10%, 5% and 1% in .jpg format showed

visible loss of image quality and also a major drop in accuracy. As an example, the accuracy dropped by 10% when going down from 5% .jpg images to 1% .jpg images for all models except SSD (300x300), where the accuracy dropped from 0.45 to 0.40, which was the lowest accuracy value in our set of experiments.

We also observed that grey-scale images took around less than 50% storage space as compared to colour images, but our models showed almost the same accuracy after training on them. This experiment was performed on 40% .jpg and 5% .jpg images. There was also no visible improvement in edges for grey-scale images as compared to the colour images. Another observation we noted for one model is that higher training time led to an increase in test accuracy. We trained the FRCNN Inception V2 model for 10 times more steps on 40% .jpg images, and the accuracy jumped from 0.69 to 0.77. The effect of batch size was also seen in category-wise average precision scores, where the models with bigger batch sizes showed better accuracy as compared to the models with smaller batch sizes, which could be seen in the results table.

To sum up, it is a trade-off between quality and accuracy, which is purely the decision of a reader. The report aimed to provide readers with a scenario where one can make better business choices in case of limited resources.

REFERENCES

[1] Margarita Martínez-Díaz and Francesc Soriguera. Autonomous vehicles: theoretical and practical challenges. *Transportation Research Procedia*, 33:275–282, 2018.

[2] M. Egmont-Petersen, D. de Ridder, and H. Handels. Image processing with neural networks—a review. *Pattern Recognition*, 35(10):2279–2301, October 2002.

[3] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009.

[4] Harman international company website. https://www.harman.com. Accessed: 2020-02-17.

[5] Michał Koziarski and Boguslaw Cyganek. Impact of low resolution on image recognition with deep neural networks: An experimental study. *International Journal of Applied Mathematics and Computer Science*, 28:735–744, 12 2018.

[6] G. K. Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, Feb 1992.

[7] Farhad Ghazvinian Zanjani, Svitlana Zinger, Bastian Piepers, Saeed Mahmoudpour, and Peter Schelkens. Impact of JPEG 2000 compression on deep convolutional neural networks for metastatic cancer detection in histopathological images. *Journal of Medical Imaging*, 6(02):1, April 2019.

[8] Samuel F. Dodge and Lina J. Karam. Understanding how image quality affects deep neural networks. *CoRR*, abs/1604.04004, 2016.

[9] Mathieu Dejean-Servières, Karol Desnos, Kamel ABDELOUAHAB, Wassim Hamidouche, Luce Morin, and Maxime Pelcat. Study of the Impact of Standard Image Compression Techniques on Performance of Image Classification with a Convolutional Neural Network. Research report, INSA Rennes ; Univ Rennes ; IETR ; Institut Pascal, December 2017.

[10] Jpeg compression. https://en.wikipedia.org/wiki/JPEG. Accessed: 2020-02-17.

[11] What is transfer learning? https://blogs.nvidia.com/blog/2019/02/07/what-is-transfer-learning/. Accessed: 2020-02-17.

[12] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[13] Coco dataset. http://cocodataset.org/. Accessed: 2020-02-17.

[14] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[15] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Kitti dataset. http://www.cvlibs.net/datasets/kitti/. Accessed: 2020-02-17.

[18] The project repository on github. https://github.com/DianaDI/compressedCV. Accessed: 2020-02-17.

[19] Paul Henderson and Vittorio Ferrari. End-to-end training of object class detectors for mean average precision. In *Asian Conference on Computer Vision*, pages 198–213. Springer, 2016.

[20] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[21] Grey-scale conversion. https://pillow.readthedocs.io/en/3.1.x/reference/Image.html. Accessed: 2020-02-17.

[22] Yuhui Xu, Yongzhuang Wang, Aojun Zhou, Weiyao Lin, and Hongkai Xiong. Deep neural network compression with single and multiple level quantization. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.