ARTICLE

# Automatic Number Plate Recognition using Transfer Learning

Y. Chokhany

**Abstract**

Through the use of transfer learning and Optical Character Recognition (OCR), I present a method of extracting text from number plates using a single model pipeline. Through the use of Single Shot Detector (Liu et al. 2016) and more than 500 data samples used in training, the model predicts bounding boxes with a high accuracy, scoring each one based on the probability of the presence of a number plate. The area of the bounding box is then extracted from the image, and passed through the OCR software, EasyOCR, where the text on the number plate is identified and presented as output. The aim of the research is to explore the hyper parameters and optimize the running time and accuracy of this pipeline, with the objective of minimising the loss - Softmax and Smooth L1 - of the object detection model. The final model achieves a loss of 0.576. Code is available here.

**Keywords:** Single Shot Multibox Detector, SSD, Transfer Learning, Optical Character Recognition, Convolutional Neural Network

Transfer learning is a machine learning technique that focuses on using knowledge gained while solving one problem to solve a second different but related problem. (Bozinovski and Fulgosi 1976)

## 1. Introduction

The data used in this research project was found on GitHub after intense research for several days, this data set was found to be varied and detailed, albeit relatively small in size.

However, images in the data set were of various sizes, therefore, they all had to be reshaped to 320px by 320px and the bounding box data had to be entered into xml files for the model to access during training.

To make up for the limited size of data set, the data was augmented using horizontal flips. The larger data set enabled the SSD model to predict locations of the number plate. However, horizontal flips cannot be used after the model has been implemented as the car number will not be legible. Moreover, vertical flips were not used as a form of augmentation because this would lead to large changes in the background of images: the car and other parts of scenery would be upside down. As a result, vertical flips would result in the model being unable to find a constant pattern between flipped and original images.

While random cropping is another potential method of data augmentation, it was quickly dismissed as an option as it can lead to the cropping into areas of the image that do not contain the number plate, increasing redundancy in the data set.

## 2. The Single Shot Detector

### *2.1 The model*

The Single Shot Detector (SSD) a state-of-the-art technology developed by Liu et al. in 2016 with the prospect of challenging the top technology at the time including YOLO and Faster R-CNN models.

SSD achieves its high detection accuracy through the utilisation of feature maps of different scales and explicitly different aspect ratios. The use of small convolutional filters enables the model to achieve end to end training while simultaneously improving the accuracy – speed trade-off.

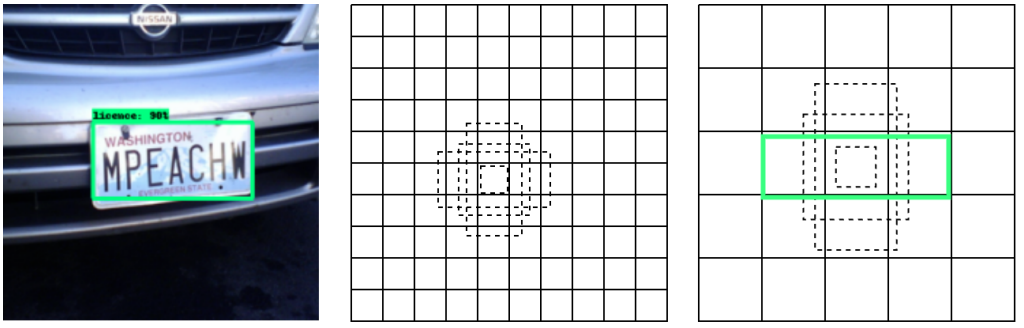


**Figure 1.** Bounding boxes in action

As illustrated in Fig. 1(a), SSD uses the Ground Truth boxes in training. A fixed number of boxes of default aspect ratios are found in each location of each feature map. This can be seen in (b) and (c), which both have different scales of 10 × 10 and 5 × 5 respectively. For each box, the confidence of being a number plate and the edges of the bounding box are predicted. The boxes are matched with the ground truth boxes, as seen in (c) where 1 box is matched with the ground truth box. The model loss is then calculated, which is the weighted sum of softmax loss (confidence) and smooth L1 loss (localization), both are weighed equally in my model.

### *2.2 The Structure*

SSD is a convolutional neural network, that produces a set number of bounding boxes and confidences of different classes in each of the boxes. Non-maximum suppression follows this process to provide the final detections.

**Non-maximum suppression** or NMS is an indispensable component of all state-of-the-art object detectors that are made in today's age. NMS is used as a post-processing tool to remove all redundant detections made by a model. The goal of object detection models is to output *exactly one* bounding box for each object present in the image. However, SSD and other popular models output numerous boxes (SSD outputs 8732 boxes per class), therefore, boxes that are close to the ground truth will all have a high confidence and overlap each other. NMS assumes that high confidence detections that overlap more than a chosen threshold $\vartheta$, are assumed to belong to the same object, thus, it chooses the detection with the highest confidence, overcoming the problem. (Hosang, Benenson, and Schiele 2017)

**SSD**'s early layers are based on the architecture of VGG-16 (truncated before any classification layers), which is called the base network. (Liu et al. 2016) The SSD model adds convolutional filters to the base network that creates feature layers of reducing size as we go deeper in the network, this attribute of feature layers allows predictions to occur at varying scales.

There are a set of bounding boxes and aspect ratios that are pre-defined before starting the training of the model. A set of boxes are taken from each feature layer as can be seen in Fig. 2. This leads to the boxes being detected at different scales, enabling them to detect objects of different sizes.

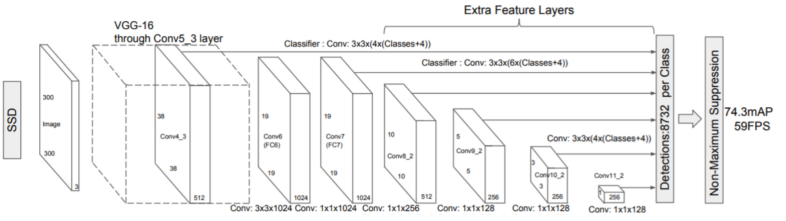


**Figure 2.** SSD Model Structure

## 3. Hyperparameters

To begin with, the model was found at the TensorFlow Model Garden (TensorFlow 2016), where the *'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8'* model was used. This model accepts 320px × 320px images as input, and has been pre-trained on the COCO-17 data set.

The following hyper parameters were optimised during the training process:

**Freezing the base network** In transfer learning, the pre-trained model is reconfigured on the images from the newer data set. When retraining a model, a certain number of layers can be frozen, meaning that the weights of the model aren't changed during back-propagation, they remain constant. During the first model, the base network (VGG-16) was frozen, this means that only the layers after the base were tweaked and configured for the number plate data set. Since VGG-16 is a classification model, it conducts a bulk of the classification of the SSD model, therefore, because the base network has been frozen, the classification of the model is improving at the optimal rate. This results in a higher classification loss compared with when all layers are unfrozen.

**Activation Function** In deep learning, the output of a neuron is fed into an activation function, similar to the stimulation of a biological neuron. The output from the activation function is fed into the next layer. The Rectified Linear Unit, or ReLU, is a piece-wise linear function that returns the input if it is positive, or returns 0 if it is negative. ReLU performed well with the model, however, it was dwarfed by ReLU-6, which capped the output of the neuron at 6. In my opinion, this performed better as it prevented arbitrarily large numbers from being carried forward in the network.

**Optimizer** Optimizers are algorithms used to minimize the loss function. They achieve this goal by manipulating the model's learnable parameters i.e. weights and biases. The momentum optimizer is similar to the concept in physics, essentially the algorithm retains the previous update to the parameters of the model and takes it into account when calculating the next update. This method minimises the loss at a faster rate, improving the model's accuracy in a short period of time. The momentum optimizer performed the best on my model, when compared with Adam or RMSProp. (Ruder 2016)

**Initializer** Optimizer algorithms require a starting point in the space of possible weight values from where to begin the process of optimization. Random initialization is a process where the value of weights are set to small random numbers which are manipulated by the optimizer during the training process. However, this causes a major issue: vanishing and exploding gradients. They occur when the parameters of the layers of the neural network either do not change at all, or rise exponentially. Neither are desirable as it can lead to an unstable network, with highly flexible parameters. Therefore, to solve this problem, Glorot et al. made the following arguments:

1. The variance of outputs of each layer should be equal to the variance of the inputs.
2. The gradients should have equal variance before and after flowing through a layer in reverse direction.

This remarkable solution was then modified to be used with ReLU activation functions by He et al. (He et al. 2015), leading to He initialisation outperforming random initialisation in my model.

**Feature maps** As stated above, different feature maps are taken from different layers in Single Shot Detectors and potential objects in the feature maps are evaluated. Different feature maps represent different resolutions of output images, enabling the model to detect objects of varying scale. However, if too many feature maps are taken, the number of potential boxes evaluated increases, slowing down the speed of the model. This brings back the topic on the accuracy – speed trade-off. The final model uses feature maps from 5 layers of the model, encouraging accuracy without hurting speed.

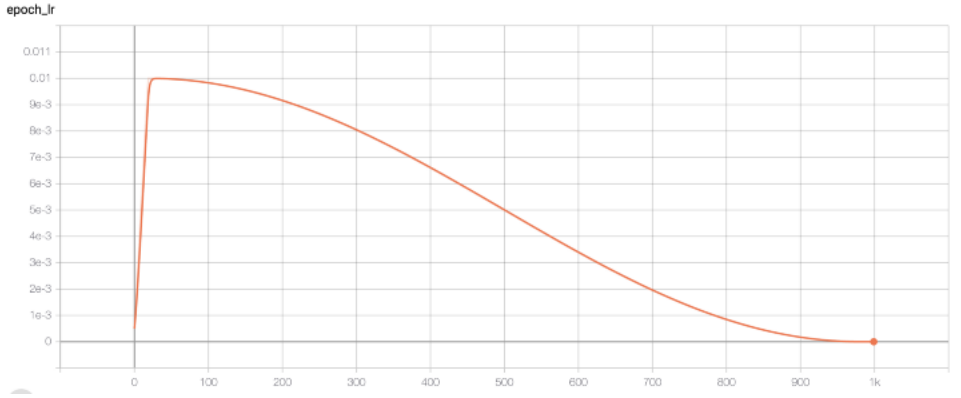


**Figure 3.** Source

**Learning Rate** The optimizer suggests the direction in which the weights must move to minimise the loss. Learning rates are used to define how large a step must be taken in that direction. A low learning rate means that it takes incredibly long for the parameters to converge towards the minima. However, an incredibly large learning rate will lead to convergence in a noisy manner. Therefore, for the optimum results, the learning rate must be decreased during the training process, so that while the model is far from the minima it can move quicker, however, once it gets closer, increasingly smaller steps will lead to a smoother convergence. My model uses a cosine decay learning rate, which uses the cosine function to reduce the learning rate over time. This has been portrayed by the graph in Fig. 3 above.

## 4. Training

The table below shows the results of the training optimization. As different hyper parameters were optimised, their total loss and run time was evaluated and recorded in the table. (The run time was evaluated per 100 epochs of training).

**Table 1.** Results

| | Layers frozen | Optimizer | Activation Function | Initializer | Feature maps | Total Loss | Run time (s) |
|---|---|---|---|---|---|---|---|
| 1 | Base model | Momentum | ReLU | Random | 4 | 1.147 | 1.75 |
| 2 | None | Momentum | ReLU | Random | 4 | 0.759 | 2.43 |
| 3 | None | Adam | ReLU | Random | 4 | 0.841 | 2.28 |
| 4 | None | RMSProp | ReLU | Random | 4 | 0.863 | 2.14 |
| 5 | None | Momentum | $ReLU_6$ | Random | 4 | 0.562 | 2.39 |
| 6 | None | Momentum | $ReLU_6$ | Xavier | 4 | 0.598 | 1.44 |
| 7 | None | Momentum | $ReLU_6$ | Xavier | 5 | 0.520 | 2.16 |

As can be observed in the table, during the training process, the hyper parameters explained in the previous section were optimised. The final model presented had the lowest loss of 0.520 and one of the lower run times.

When comparing the 4th and 5th model, although when using Xavier initializer, the total loss increased from 0.562 to 0.598, the run time was incredibly faster. As a result, Xavier was used for the final model over Random initialization. Similarly, when using 5 feature maps in the predictor, the total loss was nearly a tenth lower than 4 feature maps, therefore it was chosen as the final model despite the higher run time.


(a) Input (b) Output

**Figure 4.** Input and Output of final model

## 5. Optical Character Recognition (OCR)

As can be seen above in Fig. 4(b), the SSD model locates the presence of a licence in the image with a confidence rate. This green box is extracted, as seen in Fig. 4(c), and sent to the OCR software. The OCR software used in this model was EasyOCR (JaidedAI 2020). It returns the position and content of all text in the given image. Below are examples of input and output of the OCR software.


(a) Number plate sent to OCR (b) Output from OCR

**Figure 5.** Input and Output of SSD Model

The OCR software outputs all text present on the licence. For instance, in Fig. 5(b) we can see the text "WASHINGTON" has been outlined, which, although is present on the number plate, is not the car number. Therefore, the output from the OCR software, goes into a post processing module. Here each region of text is iterated through, and if the area of the input occupied by the text is greater than a threshold, 0.6, it is deemed to be part of the car number. As a result, "WASHINGTON", which occupies a small fraction of area of the image, is not displayed as output, and the sole output of the model is "MPEACHW".

## 6. Conclusion

This implementation of The Single Shot Detector object detection model followed by EasyOCR is an effective implementation. However, the implementation could be improved through the use of other sophisticated models like YOLO and Faster R-CNN which may perform better in this situation.

While SSD can be desirable as it has the potential to be extremely quick without conceding on accuracy, YOLO tends to perform better when handling smaller objects.

Faster R-CNN is broken up into two distinct phases. The first is *region proposal* where an RPN (region proposal network) gives region proposals. The proposals are then sent to the second component of the model, *a feature extractor*, which outputs confidences for each object class for each region proposed. Faster R-CNN is usually slower than SSD, because it's speed is based on the number of regions proposed by the RPN, while SSD just has a single model with a fixed number of computations. Nevertheless, two-stage detectors like Faster R-CNN have always out-performed one shot detectors with a higher accuracy. This is a result of a high class imbalance present in single shot detectors (Lin et al. 2018).

Moreover, the lack of access to high computational power could have compromised on the performance and speed of the model. This model may perform even better in the hands of improved computational power.

## Acknowledgement

Mentors:

- Mr Aditya Modi
- Mr Mrugank Parikh

## References

Bozinovski, Stevo, and A Fulgosi. 1976. The influence of pattern similarity and transfer of learning upon training of a base perceptron b2. (original in croatian: utjecaj slicnosti likova i transfera ucenja na obucavanje baznog perceptrona b2). *Informatica* 44 (September). https://doi.org/10.31449/inf.v44i3.2828.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: surpassing human-level performance on imagenet classification (May). https://doi.org/10.48550/arXiv.1502.01852.

Hosang, Jan, Rodrigo Benenson, and Bernt Schiele. 2017. Learning non-maximum suppression (May). https://doi.org/10.48550/arXiv.1705.02950.

JaidedAI. 2020. *Easyocr,* March. https://github.com/JaidedAI/EasyOCR.

Lin, Tsung-Yi, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2018. Focal loss for dense object detection (August). https://arxiv.org/abs/1708.02002v2.

Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. 2016. Ssd: single shot multibox detector. 9905, no. 1 (September): 21–37. https://doi.org/10.1007/978-3-319-46448-0_2.

Ruder, Sebastian. 2016. An overview of gradient descent optimization algorithms. (September). https://doi.org/10.48550/arXiv.1609.04747.

TensorFlow. 2016. *Model garden for tensorflow [source code],* January. https://github.com/tensorflow/models.