Drivable area segmentation

Ritu Bajpai, Sundeep Kolliboyina
Stanford University
rbajpai2@stanford.edu, ksundeep@stanford.edu

Abstract

The goal of this project is to develop a deep learning model for drivable area detection using images from berkley driving dataset. These images are taken under different weather conditions, time of the day e.t.c. We present two different models trained using transfer learning on popular architectures, Mask R-CNN and FCN. Qualitative and quantitative results are presented.

1 Introduction

In recent years, the self-driving car has rapidly been developing around the world. Reliable detection or segmentation of drivable area is an important part of the workflow of an autonomous vehicle. In this project we aim to implement a deep learning model for drivable area detection. Our input dataset is Berekely Driving Dataset (BDD) (1) which has driving images and corresponding driving area labels for the training and the validation set. In addition the dataset also provides unlabeled test images. The model trains to generate a mask for drivable area given a driving image as an input.

2 Related work

Navigating through an image in sliding window fashion to predict class label of each pixel was among the early approaches to train a network for image segmentation (2). One of the drawbacks of this approach is that it is quite slow. More recently, fully connected networks (FCN) have been a popular choice for instance segmentation as they can use state of the art contemporary classification networks and transfer their learned representations to segmentation tasks (3). Another popular architecture is U-Net (4). It is also an FCN which has been modified and extended such that it can be trained end to end from very few images. Authors also claim that it yields more precise segmentations. One of the more recent architectures for image segmentation is Mask R-CNN (5). It extends Faster R-CNN (6) by adding a branch to predict mask for an object in parallel with the existing branch for bounding box recognition. Faster R-CNN enables object detection at 5-17 fps. Mask R-CNN adds only a small overhead to Faster R-CNN. Given the above points, we chose to implement transfer learning training for drivable area segmentation using i) Mask R-CNN due to the speed advantage, and ii) FCN because of it's versatility to use state of the art contemporary classification networks

3 Dataset and Features

The BDD data set is one of the biggest open data sets for driving related images that are available today. The data set has 70k training images, 10k validation and 20k test images. The original resolution of the images is 720x1280. Segmentation labels and annotations are also available for download. Images are captured during different driving conditions related to weather, time of the day and different cities. More details about the dataset and download options are available at `http://bdd-data.berkeley.edu/wad-2018.html`

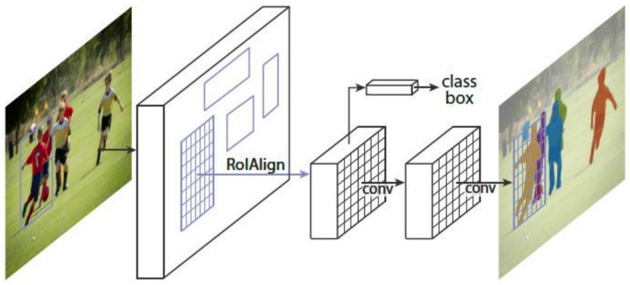


Figure 1: The Mask R-CNN framework for instance segmentation.

4 Methods

Instance segmentation is the task of identifying object in the class and generating it's unique pixel level outline. The two instance segmentation architectures that we used for transfer learning of drivable area images are Mask R-CNN and FCN. These are briefly described below.

4.1 Mask R-CNN

The basic framework of Mask R-CNN (regional convolutional neural network) (1) is shown in figure 1 and is composed of following legs i) **Backbone** is the initial part of the framework which consists of standard convolutional neural network (typically, ResNet50 or ResNet101) that extracts low level features from the images. ii) **Region Proposal Network (RPN)** stage takes the low-level feature maps from the backbone stage and applies a light neural network to scan for areas that contain objects. The regions that RPN scans over are called anchors and generates two outputs from each anchor a) anchor class which is either background or foreground b) bounding box refinement for the foreground object. Non max supression is used to eliminate highly overlapping anchors. iii) **ROI Classifier and Bounding Box Regressor** stage runs on the regions of interest (ROI) proposed by RPN and generates 2 outputs from each ROI a) class (which is more specific than RPN) b) bounding box refinement which is a further refinement from RPN stage. iv) **Segmentation Masks** branch is a convolutional network which takes the bounding boxes as input and generates segmentation masks as output.

4.2 FCN Model

There are variants of the FCN architecture, which mainly differ in the spatial precision of their output. For example, there are FCN-32, FCN-16 and FCN-8 variants. The three different architectures differ in the stride of the last convolution, and the skip connections used to obtain the output segmentation maps.It is worth noting that the three FCN architectures share the same downsampling path, but differ in their respective upsampling paths. FCN-8, sums the 2x upsampled conv7 (with a stride 2 transposed convolution) with pool4, upsamples them with a stride 2 transposed convolution and sums them with pool3, and applies a transposed convolution layer with stride 8 on the resulting feature maps to obtain the segmentation map. From the experiments it was observed that FCN8 produced more precision segmentation masks. The current FCN architecture is based on a VGG-16 image classifier to identify drivable road area from the given set of input images. The BDD data set is used for training, validation and testing. A pre trained neural network (VGG-16) is converted to a fully convolutional neural network. The final fully connected layer is dropped. A 1x1 convolution with depth equal to number of classes is added and skip connections are introduced.

5 Experiments/Results/Discussion

Mask R-CNN model transfer learning was done on AWS deep learning instance p2.xlarge with TensorFlow, Keras2 and Python3. The instance has 1 GPU. Mask R-CNN code is available at `https://github.com/beeRitu/drivable-area-segmentation`. FCN code is available at `https://github.com/sundeepkolliboyina/cs230`.

In this section we discuss experiments and results from 2 architectures independently followed by a comparison between the results from the two architectures.

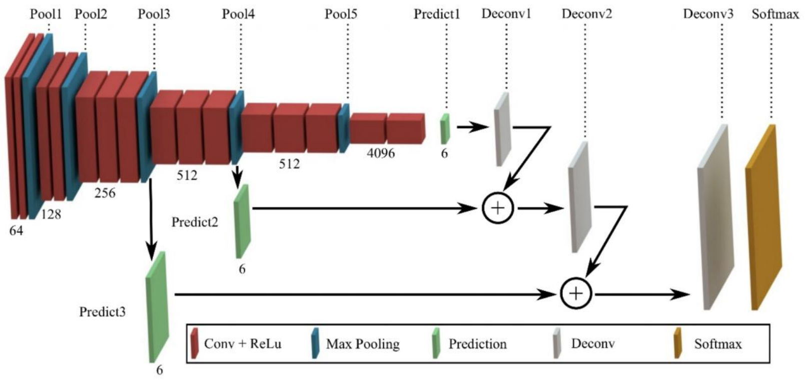


Figure 2: FCN architecture

Experiment	Image Size	Training Images	Validation Images	Epochs	Batch Size	Loss	Validation Loss	IoU
1	200x256	1k	20	60	2	0.94	1.1	0.55
2	200x256	10k	20	60	2	1.18	1.15	0.59
3	200x256	1k	20	60	1	1.19	1.29	0.54
4	200x256	1k	20	60	4	0.84	1.09	0.57
5	100x128	1k	20	60	4	0.83	1.24	0.46
7	200x256	1k	300	100	4	0.65	1.55	0.58
8	200x256	70k	10k	100	4	0.93	1.1	0.63

Table 1: Tabular representation of hyperparameter training

Evaluation metrics Loss and intersection over union (IoU) were used as performance metrics. Where loss was primarily used for hyperparameter tuning, IoU metric has been used for performance comparison of the two architectures. Standard Jaccard Index, commonly known as the PASCAL VOC intersection-over-union metric is defined as $IoU = TP/(TP + FP + FN)$ (7), where TP, FP, and FN are the numbers of true positive, false positive, and false negative pixels, respectively. We have determined IoU metric for each image in the validation set and then averaged it to get a IoU metric for each architecture.

Hyperparameter Tuning

Input image size and batch size greatly affected the model run time and memory . The original images in BDD are 720x1280 in size. Images were down sized to obtain increased speed. There are 70K labeled images for drivable area detection. We started out with 1K training images to experiment with the image size, number of epochs, and batch size. Once these hyperparameters was optimized, the training dataset was increased to 10k and 70k. Given number of validation images were used during training to calculate the validation loss after each epoch. Number of training epochs certainly impacted the loss. However an optimal number was chosen so as to optimize between run time and performance. Batch size was calculated as the number of images per GPU. This was again optimized for run time speed vs. performance.

5.1 Experiments/Results/Discussion for Mask R-CNN

Loss function is defined as $L = L_{cls} + L_{box} + L_{mask}$ where, L_{mask} is the binary cross-entropy loss where per pixel sigmoid applied to mask of each class. L_{cls} is log loss for a true class, L_{box} is the loss on the bounding box [8].

5.1.1 Results

Hyperparameter training results and IoU calculations from various training runs are summarized in table 1.

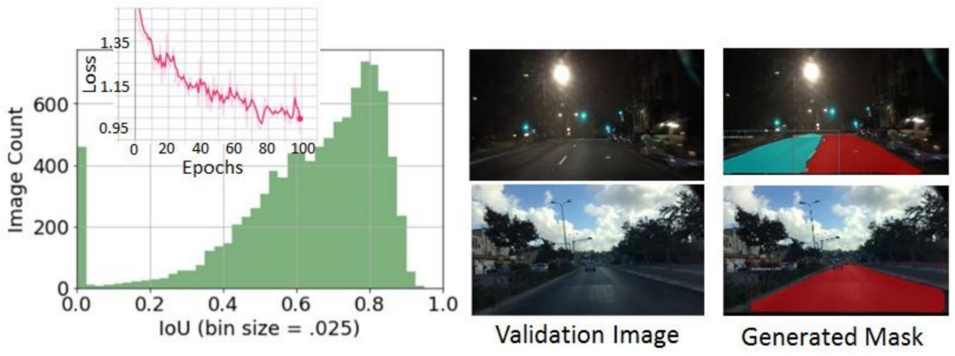


Figure 3: For the final training run (experiment 8 in table 1)- Left: Histogram of IoU for the images in validation dataset. Left (inset): Loss function plotted against number of epochs. Right: Two example images and corresponding generated masks where IoU>0.9.

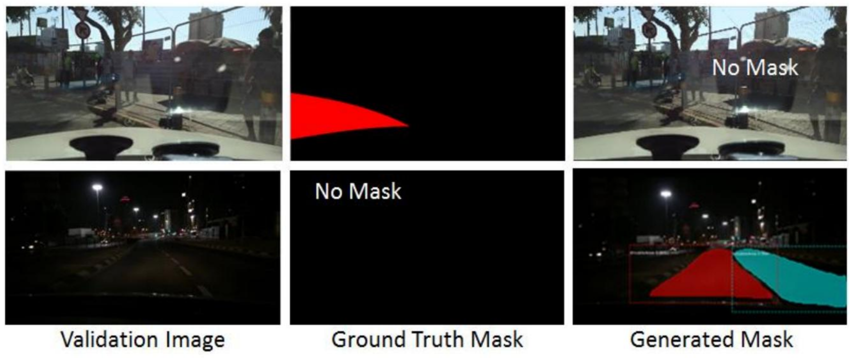


Figure 4: For the final training run (experiment 8 in table 1)- Two example images, corresponding ground truth mask and generated mask. For both the cases estimated IoU=0. Bigger size images and more result visualization can be found in the github repo.

Model run time, loss and validation loss were chosen as metrics for hyperparameter tuning. Input image size had a big impact on the model run time.Images were shrunk to 200x256 for increased computation speed. However on further reduction in size to 100x128, we saw increased validation loss. This was also reflected in IoU metric which drastically reduced from 0.57 to 0.46. Using bigger batch size helped reduce the loss but increased the runtime. Therefore maximum 4 images per GPU were selected. Increasing the number of epochs helped achiever lower loss. Again due to time and speed constraints, maximum 100 epochs were trained. Increasing the size of training data had the biggest impact on performance. Notably by increasing the size of training data from 10k to 70k images, the IoU metric increased from 0.58 to 0.63 as seen in table 1.

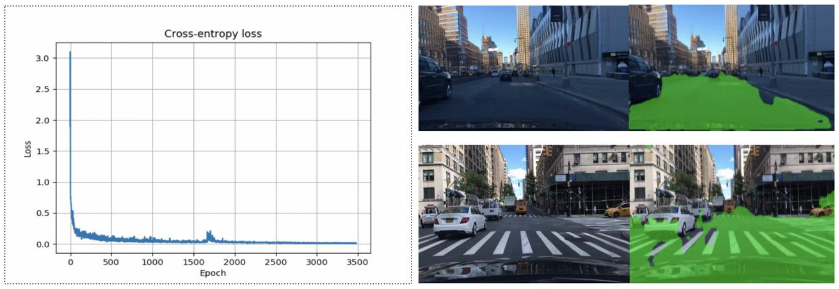
Final training is run corresponds to hyperparameters of experiment 8 in table 1. IoU is calculated for each image in validation set of 10k images. Overall IoU is averaged over 10k images. Figure 3 shows histogram of IoU for 10k images in validation dataset. Inset within the histogram shows the loss function against the number of epochs. It is interesting to note from the histogram that about 450 images have $IoU = 0$. This indicates images where either the ground truth mask or the generated mask does not exit. Some images had IoU>0.9. Two of these example images from the validation dataset are shown in figure 3 (right).

5.1.2 Discussion

Images with $IoU = 0$ were visually inspected. It is observed that for some images as shown in figure 4, top row, labels in the ground truth annotation marked very small portions of the road (usually in congested images) as drivable area where as the model failed to recognize it. On the other hand in some images as shown in figure 3, bottom row, ground truth does not label visible portion of the driving area (probably because the image is a night time image) whereas the model labels it.

Model	Image Size	Training Images	Validation Images	Epochs	Batch Size	Loss	IoU
FCN-32	160x576	10K	1K	30	4	1.29	0.39
FCN-8	160x576	10K	1K	30	4	0.82	0.44
FCN-8	160x576	70K	10K	40	20	0.98	0.52

Table 2: FCN hyper parameter tuning



Loss vs Epoch

Test images & generated mask

Figure 5: FCN Results

Images with IoU>0.9 were also closely exammined. These are generally images with big drivable area and little to no clutter in the the image.

5.2 Experiments/Results/Discussion for FCN

Architectural model search was also done on FCN in addition to hyperparameter tuning. As described previously in the FCN model section, FCN model has three variants FCN-32/16/8. Analysis was done on FCN-32/8 architectures and it was found that FCN-8 provided more precise segmentation maps because of additional predictions from pool3, at stride 8 unlike FCN-32 where it upsample stride 32 predictions back to pixes in a single step

The size of image and batch size significantly contributed to performance and run-time of the model. After performing multiple experiments the hyperparameters on the final 70K data set were tuned to have the following learning rate = 0.0005, keep prob = 0.5, epochs = 40, batch size = 20. It was noticed that the increasing the size of training data contributed significantly in improving the IoU.

5.3 Comparison of results

We use IoU metric to compare the performance of Mask R-CNN vs. FCN model. The best tuned hyperparameters from both the models were used to run the training on 70k images. Average IoU on the validation dataset of 10k images was calculated. Mask R-CNN gave an average IoU of 0.63 whereas FCN produced an average IoU of 0.52. Unfortunately we did not have enough time to evaluate the distribution of IoU metric for individual images for FCN based model. We don't know if the model performs poorly on specific kind of images such as night time images or gives a poor overall performance on the entire dataset.

6 Conclusion/Future Work

We see that better performance was achieved from Mask R-CNN in comparison to FCN. For the Mask R-CNN results we also see that the model performs better on images with bigger and clearer driving area as compared to images with small and cluttered driving area. In fact, inconsistencies in the ground truth mask labels were also found for images where IoU of zero was detected. In order to improve the performance, we could include more images with cluttered driving conditions and pay attention to correct ground truth labeling. Similar detailed analysis of the results from FCN model are needed to understand why the model performs poorly and to suggest possible improvements.

7 Code

`https://github.com/beeRitu/drivable-area-segmentation`

`https://github.com/sundeepkolliboyina/cs230`

8 Contributions

Ritu Bajpai worked on transfer learning using Mask R-CNN architecture which included training the model, tuning the hyperparameters, result analysis and writing related sections of the report. She also did literature review and study of other state of the art architectures for drivable area segmentation.

Sundeep Kolliboyina worked on FCN model exploring various FCN architectures, tuning hyperparameters, model analysis and writing related sections of the report. Did literature review and analysis on various FCN models.

References

[1] `http://bdd-data.berkeley.edu/`

[2] Ciresan, Dan, et al. "Deep neural networks segment neuronal membranes in electron microscopy images." Advances in neural information processing systems. 2012.

[3] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[4] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015.

[5] He, Kaiming, et al. "Mask r-cnn." Computer Vision (ICCV), 2017 IEEE International Conference on. IEEE, 2017.

[6] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems. 2015.

[7] https://www.cityscapes-dataset.com/benchmarks/

[8] Girshick, Ross. "Fast r-cnn." Proceedings of the IEEE international conference on computer vision. 2015.