Road Segmentation using Fully Convolutional Network and VGG16 Transfer Learning

Than Hui Ru
School of Engineering and Technology
Sunway University
Subang Jaya, Malaysia
20006441@imail.suway.edu.my

***Abstract*—Autonomous driving is becoming increasingly relevant in today's world. Road segmentation plays an important role is distinguishing drivable and non-drivable areas for autonomous vehicles to drive. This paper experiments with a Fully Convolutional Network (FCN) with transfer learning using the VGG16-network for road segmentation. Equipped with a dataset that consist of only 31 images, this model was able to perform road segmentation that reached a Mean Intersection over Union (MIoU) of 96.34% with processing speed of 2.99 seconds on marked images.**

In this paper, data augmentation and video recorder marking labels were removed to improve performance of the model. This paper managed to achieve good results despite a small dataset using machine learning methods.

Keywords—road segmentation, computer vision, machine learning, autonomous driving, road recognition

I. INTRODUCTION

With developments in autonomous driving, road segmentation remains a critical task in enabling vehicles to understand their environment to navigate traffic safely and efficiently. Road regions can be identified through information captured by vehicles' sensors, such as monocular cameras, as well as that incorporate additional sensors like LiDAR or radar for more accurate classification of road regions across varying environments and conditions.

The main objective of road segmentation is to differentiate drivable and non-drivable areas of the road. This task can be accomplished using traditional methods or deep learning models. Traditional methods often perform faster due to their simple computation; however, they are less robust against changes in image quality, leading to less accurate segmentation.

With the advent of deep learning, significant strides have been made in the accuracy and efficiency of road segmentation. Deep learning algorithms are better at dealing with uncertainties such as illumination issues, dynamic backgrounds, and variations in appearance[1]. They provide superior performance through the approach they are able to learn hierarchical features from raw data. Deep learning increases the quality of semantic segmentation; however, it is often slower than traditional methods due to its computational complexity.

In this paper, we examine a Fully Convolutional Network proposed in [2] which implements transfer learning using the VGG16 network. The mean IOU of this experiment was able to reach 94.92% similarity with its mask.

II. METHODOLOGY AND PROPOSED METHOD

A. Architecture Implementation

The proposed method by [2] aims to perform semantic segmentation using Fully Convolutional Network(FCN). This network implements transfer learning using the VGG16 model. The VGG16 model is used as a feature extractor. Three layers from the model are extracted to build the FCN model. By extracting these specific layers, the model can leverage the learned features from the VGG-16 network hence, improving the performance on the new task.

The input images are initially passed through the pretrained VGG-16 layers to extract feature maps. These feature maps are then used as inputs to the additional layers which builds the FCN. This FCN is created by combining the feature maps using up-sampling and skip connections to produce the final segmentation output. Refer to Figure 1 for a visualisation of the model.

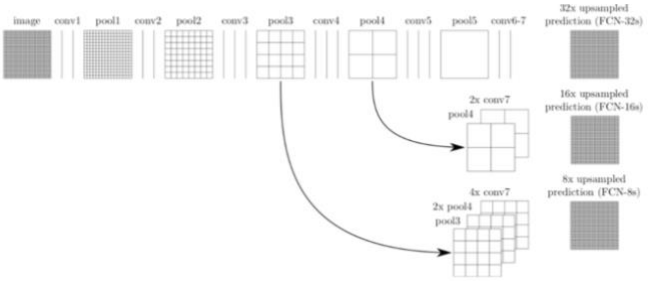

Fig. 1. Architecture of FCN Model [2]

First Up-sampling and Skip Connection: Tensor for the output of VGG-16 layer 3, layer 4 and layer 7 are extracted. To create the FCN, a 1x1 convolution is applied to the output of layer 7. This convolution reduces the depth of the layer to number of classes we have. The next step is to up-sample the output of 1x1 convolution by a factor of 2 using a transposed convolution that increases the spatial dimensions of the feature maps. After that, another 1x1 convolution is applied to layer 4 so that its depth matches our number of classes. To from the skip connections, the unsampled output of layer 7 and the 1x1 convolved layer 4 is added together.

Second Up-sampling and Skip Connection: Using the skipped connection result, it is up-sampled by a factor of 2. A 1x1 convolution to layer 3 is applied to make its dept match the number of classes. The next step is skip connections, by adding the up-sampled skip connection 1 and the 1x1 convolved layer 3.

Final Up-Sampling: The final up-sampling step increases the spatial dimensions of the feature maps to match the original input image size. It uses transposed convolution with a stride 8 to achieve this. This is the last layer of the neural network.

Training Details: The optimization process uses Cross Entropy Loss as the loss function and Adam Optimisation as the learning rate optimization algorithm. The batch size for this training was 5 for 60 epochs in the original paper. The Learning Rate (LR) was 0.0009. The dataset used was the KITTI dataset which has approximately 200-400 images (number of images were not mentioned in the proposed method). The input size used was 576(W)x160(H).

B. Methodology

Hardware Requirements: The model used for this project is the Apple MacBook Pro 14-inch. The processor is the Apple M3 Pro with an 11-core CPU (5 performance cores and 6 efficiency cores), a 16-core Neural Engine and a 14-core GPU. The RAM is 18GB Unified Memory and the Operating System used is macOS Sonama 14.5.

Training Tools: The training tools used were TensorFlow, Keras, Keras OCR, Pillow, Matplotlib and NumPy. Visual Studio Code (VSCode) was the Integrated Development Environment used.

Dataset Details: The data set used in the project consist of 31 images. Each image is paired with a mask. The size of the images is 1280(W)x720(H).

Fig. 2. Image of road scene

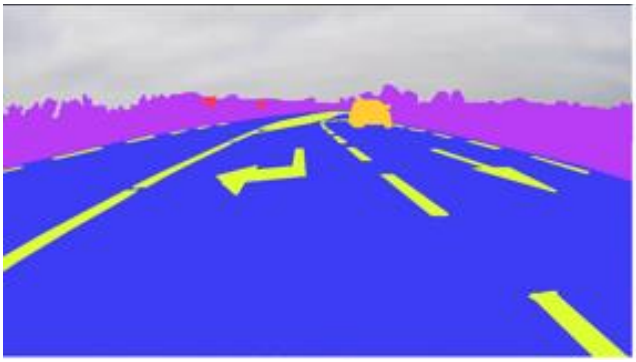

Fig. 3. Mask image of road scene

Data Augmentation: In the originally proposed work, data augmentation techniques were provided to increase robustness of the model. These include scaling images, flipping, adding salt and pepper noise, and adding white noise. The total images at the end were 248 image files for training. The dataset of 31 images was separated into 70% for training and 30% for testing.

The original source code was implemented using TensorFlow1. However, since TensorFlow 1 is no longer supported, the code was upgraded to TensorFlow 2.

Fig. 4. White noise added to the original image (darken)

Fig. 5. Flip added to the original image

Fig. 6. Salt and pepper noise added to the original image

Fig. 7. Scaling applied to the original image

Loading Data: Both images and their masks are first loaded into batches for training. The image is first resized to 576(W)x320(H). The original dimensions of the dataset could not be utilized directly due to incompatibility with the current model architecture. Utilizing the original dimensions would require difficult restructuring of the architecture to accommodate the different input size. The skipped connection in the neural networks is used to connect feature maps from different layers, if the input size changes the spatial dimensions also change causing a mismatch in the dimensions of the feature maps being combined.

The images are changed to RBG only. From the experimentation it was noticed that the mask consists of the `A` layer (structure is RGBA), this is converted to RGB only so that it can be passed into the model.

The provided training masks and images were first loaded and prepared into batches for training. The color of the road in the mask was identified to create a mask (Refer to Fig. 12 for the image of the mask) for the road only. The original source code training masks consisted of labels which allowed for masks generated from the labels. However, this dataset did not contain labels in the mask; hence the color of the mask was used instead.

Markings Removal: The original images have characters at the bottom right corner. To reduce the noise for during training and testing, it is removed using Keras OCR and CV2.inpaint function. This method was seen in [3]

Fig. 8. Original image with words at bottom right corner

Fig. 9. Images that words have been extracted

Training Details: The model is built following the FCN architecture mentioned above. The loss function is to measure the difference between the predicted probabilities and the actual labels. The loss function used for our classification is the Cross-Entropy Loss function and it is calculated for every batch. The Adam optimizer is used for updating the network weights to minimize the loss function during training. The LR is 0.0009, the batch size was set to 4 for 40 epochs. The dropout rate remains 0.5.

Testing: The original paper did not implement any performance metrics in measuring the performance of the model. I have chosen Mean Intersection over Union (MIoU) to measure the overlap between the predicted segmentation and the ground truth segmentation. The runtime was recorded based on how much time was needed for the inference of each image. Testing was done on 8 images.

III. Experimentation and Results

In this section we review the results from training the FCN based on the performance metrics of MIoU and runtime. In part B we will review the limitations that was discovered during the development of this project. Two models were observed. One that was trained with the bottom right corner still visible and another with the bottom right corner removed.

A. Results and Discussion

TABLE I. Results of FCN model in Road Segmentation

Models	**Performance Metrics**	
	MIOU	***Run Time (seconds)***
Model 1 (with markings)	95.95%	3.04
Model 2 (without markings)	96.34%	2.99

By removing the words during training, it was able to lead to an increase of 0.34% increase in MIoU. Without removing the words, the model will sometimes misidentify areas with white pixels (mistook them as markings). Refer to Fig. 10 and Fig. 11 to see the difference of segmentation and how the road sign that has white boxes are recognized as roads.

Fig. 10. Predicted mask by model 2

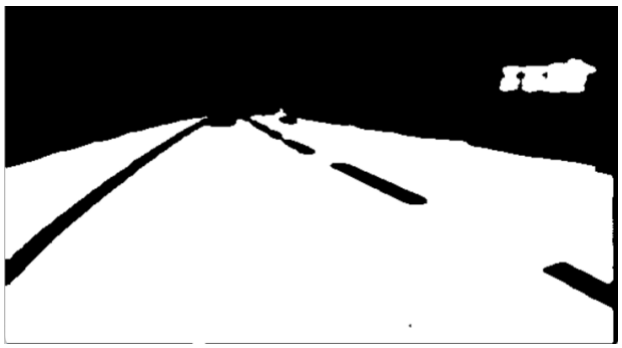

Fig. 11. Predicted mask by model 1

The final model managed to demonstrate a 96.35% MIoU in correctly recognizing road areas. While it has demonstrated high MIoU, there are some instances where the model is less effective in identifying regions that are far in distance. Road lanes and cars are not accurately distinguished. This could be due to the farther away an object in the image that becomes smaller and less distinct. Resizing images to a different aspect ratio can also lead to negative effects.

Some mistakes were made in properly distinguishing road areas and grass land at the side of the road. This is due to a few reasons. The grass land at the side and road surfaces sometimes has similar visual texture and color which can confuse the model.

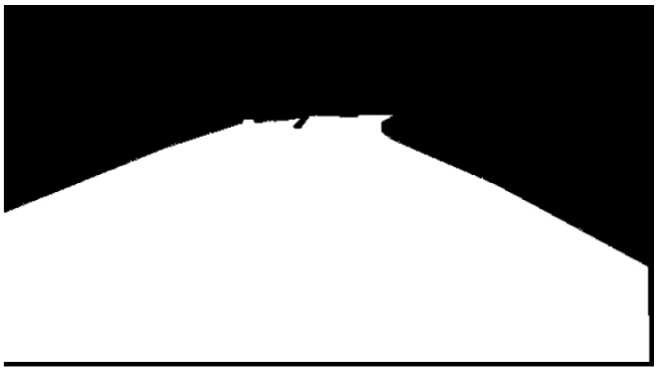

Fig. 12. Mask correctly segmenting the road from the grass

Fig. 13. Predicted mask which is less accuracte n segmenting the road from grass land that resembles road area.

The total time used per image is 2.99s per image which is not ideal for real time processing. However, it is not the FCN that took so much time but rather the words extractor. Without the markings extractor the model could process testing images with an average time of 60.70ms per image which is satisfactory for real-time road segmentation usage.

B. Limitations

The size of the image is resized from 1280x780 to 576x320. This is due to 2 reasons:

- The model architecture was unable to handle the dimensions of the original image due to architecture mismatch.
- The hardware architecture used in this experiment was not able to handle such big image dimensions. Training larger images requires more computational resources and higher memory usage. The hardware specifications crashed multiple times attempting to train larger images.

Resizing images to a smaller dimension can lead to loss of fine details required for better performance in object detection and segmentation. For example, objects that are far in distance such as cars and lane markings in images are often loss/minimised when images are resized.

The image was also resized according to the aspect ratio acceptable by the architecture. Without maintaining the aspect ratio of the original image, it can introduce distortion to the object in the images that might negatively affect training.

Even though we had performed data augmentation, the number of images collected for training is only 248 images and 8 images for testing. This is in comparison to 200 images(before data augmentation) that was used in the initial implementation for training.

The original implementation implemented multiclass segmentation, however in this paper we experimented with binary classification, classifying road and non-road areas. Dataset for cars and road signs are relatively small, hence it was not used in this experiment.

IV. Conclusion and Future Works

This model has demonstrated a satisfactory performance by reaching a 96.34% MiOU and a processing speed of 2.99ms per image. Though the time is over real-time segmentation requirements, the processing speed can easily be reduced to 60.70ms per image if non-marked images are used.

In this project, we demonstrated that good performance can be achieved using this FCN and transfer learning VGG16 for road segmentation even when only 31 images were used (23 images for training and 8 images for testing). In future projects, segmentation would ideally be extended to segment other objects such as car, road signs, people and etc. Efforts would be directed to further enhance the model structure so that resizing images to a predefined aspect ratio can be avoided.

[1] S. Beucher and M. Bilodeau, Road Segmentation and Obstacle Detection by a fast watershed algorithm. 1994.

[2] Bo, lb5160482/Road-Semantic-Segmentation. (Jun. 07, 2024). Python. Accessed: Aug. 03, 2024. [Online]. Available: https://github.com/lb5160482/Road-Semantic-Segmentation

[3] C. Borella, “Remove Text from Images using CV2 and Keras-OCR,” Medium. Accessed: Aug. 03, 2024. [Online]. Available: https://towardsdatascience.com/remove-text-from-images-using-cv2-and-keras-ocr-24e7612ae4f4