Highway Lane Line and Objects Segmentation

1. Models Overview

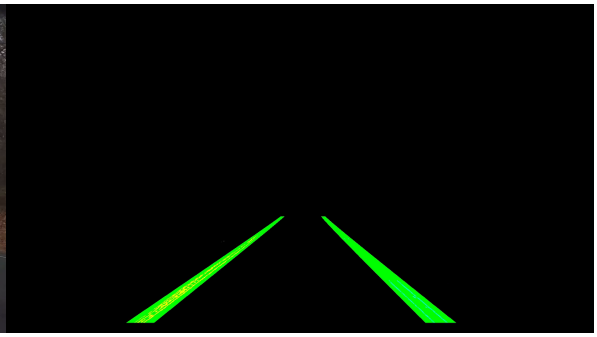
Model Purpose	Training Data Set	Neural Network	Input	Output
Lane line Segmentation	158 examples of camera images and lane lines annotation	Customized LeNet	540x960x3 RGB	540x960x2 logits
Objects Segmentation	2738 examples of camera images and objects annotation	VGG-16	270x480x3 RGB	270x480x7 logits

2. Lane Line Segmentation Model Details

2.1 Training Data

The training dataset for lane line segmentation contains 158 frames picked from original video

Camera Image



Annotation

2.2 Result Comparison

Segmentation Result

Training Annotation

2.3 Neural Network Architecture

Layer (type)	Output Shape	Param #	Connected to
input_2 (InputLayer)	(None, 540, 960, 3)	0	
lambda_2 (Lambda)	(None, 540, 960, 3)	0	input_2[0][0]
Conv1 (Conv2D)	(None, 135, 240, 24)	1824	lambda_2[0][0]
pool1 (MaxPooling2D)	(None, 68, 120, 24)	0	Conv1[0][0]
conv2 (Conv2D)	(None, 68, 120, 48)	28848	pool1[0][0]
pool2 (MaxPooling2D)	(None, 34, 60, 48)	0	conv2[0][0]
conv3 (Conv2D)	(None, 34, 60, 96)	41568	pool2[0][0]
conv4 (Conv2D)	(None, 34, 60, 484)	5622628	conv3[0][0]
Drop4 (Dropout)	(None, 34, 60, 484)	0	conv4[0][0]
Conv5 (Conv2D)	(None, 34, 60, 484)	234740	Drop4[0][0]
Drop5 (Dropout)	(None, 34, 60, 484)	0	Conv5[0][0]
Conv6 (Conv2D)	(None, 34, 60, 2)	970	Drop5[0][0]
scale_pool1 (Lambda)	(None, 68, 120, 24)	0	pool1[0][0]
Deconv1 (Conv2DTranspose)	(None, 68, 120, 2)	66	Conv6[0][0]
Pool1_feat (Conv2D)	(None, 68, 120, 2)	50	scale_pool1[0][0]
Add2 (Add)	(None, 68, 120, 2)	0	Deconv1[0][0] Pool1_feat[0][0]
Deconv3 (Conv2DTranspose)	(None, 544, 960, 2)	1026	Add2[0][0]
Deconv2_crop (Cropping2D)	(None, 540, 960, 2)	0	Deconv3[0][0]

Total params: 5,931,720
Trainable params: 5,931,720
Non-trainable params: 0

2.4 Code Documents and Overview

Section	Code Document	Overview
Generate Training Data	DataGenerate.ipynb	Implement CV tools to mark the lane lines and generate video as training data
Train Neural Network	Train.ipynb	Customized a LeNet architecture to train for 10 epochs with Adam optimizer and MSE loss function

Produce Segmentation Result	SegmentationOutput.ipynb	Add a SoftMax layer to the model output and crop the lower half as lane line graph

2.5 Further Improvements

In the video, sometimes the lane lines are not completed marked, it can be improved by implementing the segmentation on a recurrent neural network such as LSTM, to utilize the lane lines information from previous frames. In addition, apply a window search on the segmentation result and fit each line in a second order polynomial (same as in the DataGenerate.ipynb) to describe the entire lane line position. And I think the polynomial will be easier to follow in the path planning process than raw pixels. Lastly, transforming the perspective to bird eye view will enhance the accuracy substantially, but this requires a dynamic camera calibration algorithm to keep track of small camera movement during driving.

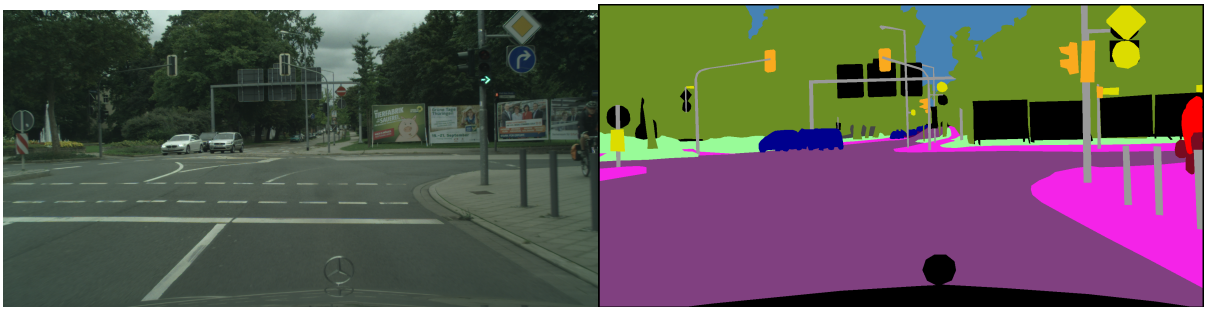
3. Objects Segmentation Model Structure and Details

3.1 Training Dataset

The training data for objects segmentation is download from https://www.cityscapes-dataset.com/, which contains 2,738 training examples and corresponding annotations. The project picked 7 labels for segmentation as following:

Label	Color Value	Color
Unlabeled	(0,0,0)	
Car	(0,0,142)	
Road	(128,64,128)	
Trees	(107,142,35)	
Bridge	(150,100,100)	
Sky	(70,130,180)	
Traffic Sign	(220,220,0)	

Example of the image and ground truth annotation:



3.2 Neural Network Architecture

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 270, 480, 3)	0	
Block1_Conv1 (Conv2D)	(None, 268, 478, 64)	1792	input_1[0][0]
Block1_Conv2 (Conv2D)	(None, 266, 476, 64)	36928	Block1_Conv1[0][0]
Block1_Pool1 (MaxPooling2D)	(None, 133, 238, 64)	0	Block1_Conv2[0][0]
Block2_Conv1 (Conv2D)	(None, 133, 238, 128	73856	Block1_Pool1[0][0]
Block2_Conv2 (Conv2D)	(None, 133, 238, 128	147584	Block2_Conv1[0][0]
Block2_Pool2 (MaxPooling2D)	(None, 67, 119, 128)	0	Block2_Conv2[0][0]
Block3_Conv1 (Conv2D)	(None, 67, 119, 256)	295168	Block2_Pool2[0][0]
Block3_Conv2 (Conv2D)	(None, 67, 119, 256)	590080	Block3_Conv1[0][0]
Block3_Conv3 (Conv2D)	(None, 67, 119, 256)	590080	Block3_Conv2[0][0]
Block3_Pool3 (MaxPooling2D)	(None, 34, 60, 256)	0	Block3_Conv3[0][0]
Block4_Conv1 (Conv2D)	(None, 34, 60, 512)	1180160	Block3_Pool3[0][0]
Block4_Conv2 (Conv2D)	(None, 34, 60, 512)	2359808	Block4_Conv1[0][0]
Block4_Conv3 (Conv2D)	(None, 34, 60, 512)	2359808	Block4_Conv2[0][0]
Block4_Pool4 (MaxPooling2D)	(None, 17, 30, 512)	0	Block4_Conv3[0][0]
Block5_Conv1 (Conv2D)	(None, 17, 30, 512)	2359808	Block4_Pool4[0][0]
Block5_Conv2 (Conv2D)	(None, 17, 30, 512)	2359808	Block5_Conv1[0][0]
Block5_Conv3 (Conv2D)	(None, 17, 30, 512)	2359808	Block5_Conv2[0][0]
Block5_Pool5 (MaxPooling2D)	(None, 9, 15, 512)	0	Block5_Conv3[0][0]
Block6_Conv1 (Conv2D)	(None, 9, 15, 4096)	52432896	Block5_Pool5[0][0]
Drop1 (Dropout)	(None, 9, 15, 4096)	0	Block6_Conv1[0][0]
Block6_Conv2 (Conv2D)	(None, 9, 15, 4096)	16781312	Drop1[0][0]
Drop2 (Dropout)	(None, 9, 15, 4096)	0	Block6_Conv2[0][0]
Block7_Conv1 (Conv2D)	(None, 9, 15, 7)	28679	Drop2[0][0]
Block7_Deconv1 (Conv2DTranspose	(None, 18, 30, 7)	791	Block7_Conv1[0][0]
scale_pool4 (Lambda)	(None, 17, 30, 512)	0	Block4_Pool4[0][0]
cropping2d_1 (Cropping2D)	(None, 17, 30, 7)	0	Block7_Deconv1[0][0]
Pool4_feat (Conv2D)	(None, 17, 30, 7)	3591	scale_pool4[0][0]
Add1 (Add)	(None, 17, 30, 7)	0	cropping2d_1[0][0] Pool4_feat[0][0]
scale_pool3 (Lambda)	(None, 34, 60, 256)	0	Block3_Pool3[0][0]
Block8_Deconv2 (Conv2DTranspose	(None, 34, 60, 7)	791	Add1[0][0]
Pool3_feat (Conv2D)	(None, 34, 60, 7)	1799	scale_pool3[0][0]
Add2 (Add)	(None, 34, 60, 7)	0	Block8_Deconv2[0][0] Pool3_feat[0][0]
Block10_Deconv3 (Conv2DTranspos	(None, 272, 480, 7)	3143	Add2[0][0]
cropping2d_2 (Cropping2D)	(None, 270, 480, 7)	0	Block10_Deconv3[0][0]

Total params: 83,967,690
Trainable params: 83,967,690
Non-trainable params: 0

3.3 Code Documents and Overview

Section	Code Document	Overview
Generate 7 classes of Annotated Ground Truth Labels	Train.ipynb	Import image and annotation images from dataset, and label the 7 objects
Train Neural Network	Train.ipynb	Load pre-trained VGG16 weights, adopt the model that trained only 4 epochs, as additional training appear overfitting due to a small size of dataset
Produce Segmentation Result	SegmentationOutput.ipynb	Add a SoftMax layer to the model output and mask object color accordingly, combine with lane line mask model result, then cast back on original video image

3.4 Further Improvements

- Train or convert to tensor flow environment and freeze the graph, so the output will be faster and can be utilized in the C++ environment
- Add an SSD or YOLO object detection model to draw boxes around vehicles
- Feed more training data and build a generator to prepare batch data on CPU while training on GPU
- Randomize the contrast or brightness of the training image, so the model will perform better at shadow
- Build a recurrent neural network to memorize time series information to enhance the confidence of objects pixels