Semantic Segmentation for Self Driving Cars Using U-Net

Name: Abid Hossain Haque ID: 20101453 Section:18

May 13, 2023

1 Abstract

For self-driving cars to navigate safely and effectively, they must be able to precisely see their surroundings and comprehend the situation. Each pixel in an image receives a class label during the pixel-level categorization process known as semantic segmentation. It is essential for developing strong perceptual abilities in autonomous vehicles. In order to follow autonomous driving tasks including object detection, path planning, and motion control, semantic segmentation has been widely used in scene understanding. But in computer vision, precise semantic segmentation is a difficult task. Popular semantic segmentation network U-Net is used to do segmentation tasks. We examine the changes made to the original U-Net architecture to enhance its functionality and make it suitable for the particular difficulties of autonomous driving. We also demonstrate the utility of U-Net in precisely classifying and detecting diverse objects in urban driving scenarios through comprehensive experimentation and review.

2 Introduction

The transportation sector is being transformed by self-driving cars, and for them to operate safely and effectively, precise scene understanding is essential. And this effective perception, interpretation of the surroundings by autonomous vehicles are made possible by semantic segmentation. Semantic segmentation is the process of classifying each pixel in an aerial image into different semantic categories

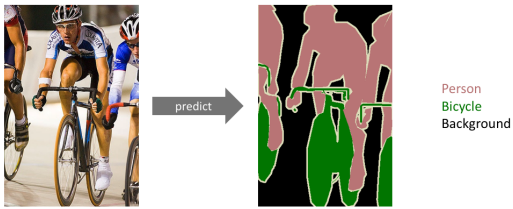

Figure 1: An overview of Semantic Image Segmentation

such as buildings, roads, trees, water bodies, and so on. More specifically the goal of semantic image segmentation is to label each pixel of an image with a corresponding class of what is represented. This task is known as dense prediction because we are predicting for every pixel in the image.

Deep learning techniques are typically used to analyze the imagery and identify distinct objects and regions within it. A naive approach towards constructing a neural network architecture for this task is to simply stack a number of convolutional layers and output a final segmentation map. This directly learns a mapping from the input image to its corresponding segmentation through the successive transformation of feature mappings; however, it's quite computationally expensive to preserve the full resolution throughout the network. One popular approach for image segmentation models is to use an encoder/decoder structure, where we downsample the spatial resolution of the input then develop it into a lower-resolution feature mappings that are learned to be highly efficient at class discrimination, and

finally upsampling the feature representations into a full-resolution segmentation map.
This is where we introduce the U-Net which is famous and widely used for Semantic Segmentation. It is named for its U-Shaped architecture and it consists of an encoder and decoder path. The encoder path extracts features from the input image, which are then upsampled by the decoder path to generate a segmentation map. The U-Net architecture is well suited to semantic segmentation of aerial imagery because it can capture both local and global image features, which is a must for accurate segmentation. Later on we will discuss more about it.

The key contribution of this paper is:
* By leveraging deep learning technique and the U-Net architecture, we aim to enhance semantic segmentation performance in autonomous driving applications.
* Addressing the challenge of limited labeled data by incorporating transfer learning and domain adaptation techniques. This enables the model to leverage pre-trained weights and adapt to different driving environments, further improving its performance and generalization capabilities.
* Overall, this paper contributes to the advancement of semantic segmentation in self-driving cars by presenting a U-Net architecture tailored to the specific requirements of autonomous driving scenarios. The proposed modifications improve perception capabilities, enabling more reliable decision-making and enhancing the safety and efficiency of self-driving vehicles.

3 Problem Statement

Self-driving cars are a promising technology that has the potential to revolutionize transportation. However, there are a number of challenges that need to be overcome before they can be deployed on a large scale. One of the biggest challenges is accurately perceiving and understanding the environment. Self-driving cars need to be able to see and understand everything around them, including other cars, pedestrians, and traffic signs. This is a challenging task, as the environment can be complex and ever-changing. Another challenge is dealing with the limitations of sensors. Self-driving cars rely on a variety of sensors, such as cameras, radar, and lidar, to perceive their surroundings. However, these sensors have limitations, such as occlusions, adverse weather conditions, and sensor failures. Self-driving cars also need to be safe and reliable. This is a challenging task, as they need to be able to handle a variety of situations, including unexpected events. For example, what should a self-driving car do if it encounters a pedestrian who suddenly steps out in front of it? Self-driving cars also need to be able to make complex decisions in real time. This is a challenging task, as they need to be able to weigh all of the available information and make the best decision possible in a short amount of time. For example, what should a self-driving car do if it is faced with a choice between hitting a pedestrian or swerving off the road and crashing into a tree? In addition to these technical challenges, there are also a number of legal and ethical challenges that need to be addressed. For example, who is liable in the event of an accident involving a self-driving car? And what ethical considerations should be taken into account when developing self-driving cars?

We can overcome these problems with the help of Semantic Segmentation. By precisely classifying objects at the pixel level, semantic segmentation enhances the car's perception and scene understanding. This means the car can accurately identify lanes, pedestrians, vehicles, and traffic signs, leading to improved object detection, tracking, and overall interpretation of the environment. Additionally, semantic segmentation allows for better sensor fusion, compensating for limitations and occlusions in individual sensors. This fusion of data from cameras, lidar, and radar provides a more comprehensive understanding of the surroundings. With more accurate perception, self-driving cars can make reliable decisions and plan appropriate maneuvers. Semantic segmentation also aids in enhancing human-machine interaction by allowing the car to communicate its intentions through visual interfaces, thereby building trust with passengers and pedestrians. Although semantic segmentation alone may not directly address cybersecurity concerns, it contributes to the robustness of the perception system, aiding in the early detection of potential cyber threats. Overall, semantic segmentation using U-Net is an integral part of a larger system that combines various algorithms to address the challenges faced by self-driving cars.

4 Model Architecture

We will use the U-Net architecture to train our semantic segmentation model. The U-Net model, named after its U-shaped architecture, was originally developed in 2015 for biomedical image segmentation tasks. However, this model has become a very popular choice

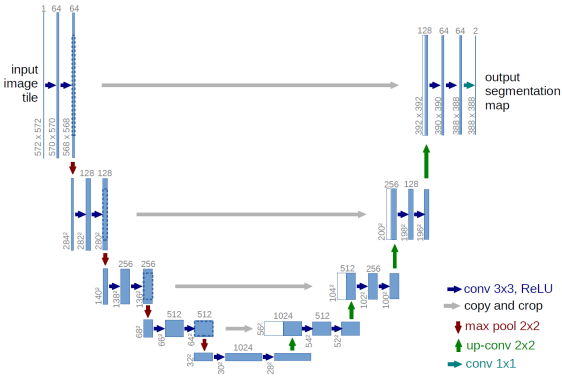

Figure 2: U-Net Architecture

for other semantic segmentation tasks.
U-Net builds on an earlier architecture called Fully Convolutional Networks (FCNs), which uses dense layers of common CNNs to store feature maps of the size of the original input image while preserving the size of the image. Replace with a transposed convolutional layer that upsamples to the original input image and retains spatial information. This is necessary because dense layers destroy the spatial information (image "location") that is an integral part of the image segmentation task. An additional benefit of using transposed convolutions is that we no longer need to have a fixed input size, as we do with dense layers. The U-Net architecture consists of:
1. Contract path (encoder with downsampling step):

The convolutional layer, its activation layer, and a pooling layer are

used in the reduction pass to downcompute the image and retrieve its features. The image is first passed through several convolutional layers, which increase the number of channels while decreasing the height and width. Applying two 3 3 unpadded convolution iterations, a modified linear unit (ReLU) for each, and a 2 2 max pooling operation with step 2 for downsampling make up this procedure in detail. increase. The number of feature channels doubles with each downsampling step. Before size reduction (combining features), the convolutional output is saved in a different variable for the reduction phase. During the decoding phase, this is sent as a feature map to the augmentation block.

2. Enhanced pass (decoder with upsampling step):

The expansion pass reverses the reduction pass's procedure, enlarging the image to its original size while progressively constricting the channels. In more detail, the feature map is upsampled at each stage of the expansion pass, then a 2 x 2 convolution (also known as a transposed convolution or upsampling) is applied. The number of feature channels is cut in half while the image's height and breadth are increased by this transposed convolution. Next, we concatenate two 3 x 3 convolutions, each followed by a ReLU, to the correctly pruned feature map from the reduction pass.

3. Final feature mapping block:

Each 64-component feature vector is given the desired number of classes at the top level using a 1x1 convolution. The number of filters employed corresponds to the channel dimension of the previous level. In order to modify this dimension using 1x1 convolution, select the required number of 1x1 filters. By using this concept to the highest level, we can minimize the channel's dimension to one level per

class.
However, there are a total of 23 convolutional layers in the U-Net network.

5 Dataset and Data preparation

The CARLA self-driving car simulator was used to capture the data images and annotated semantic segmentations in this dataset. In the 2018 Lyft Udacity Perception Challenge, 5000 photographs and their semantic segmentations were collected as data which is around 5GB size in memory. The objectives will be:
1. Train a model to predict semantic segmentations of the dataset images
2. Evaluate model performance using various metrics such as Model Accuracy, Mean Precision, Mean Recall, Mean Specificity, Mean Intersection over Union (mIoU), Mean True Detection Rate (TDR), Mean F1 Score/Dice Co-efficient (TDR)
3. Predict masks/segmentations of the dataset using the model and compare predictions with ground-truth masks.

But for now we need to prepare our data to perform all those action, For this, the Semantic Segmentation for Self-Driving Cars in Lyft Udacity contains five directories (dataA, dataB, dataC, dataD, and dataE) hold the challenge data (pictures and masks). We will load photos and masks from each of the five directories as part of the data preparation step and do a few preprocessing operations to ensure that we give our model a high-quality dataset.

5.1 Load the images and masks from their directories

In this data preparation step, we will:
1. Create 2 lists containing the paths of images and masks.
2. Split the lists into training, validation and test sets.

5.2 Create lists containing the paths of images and masks

In this step, we will:
1. Create a list that contains all the paths to all directories in the main directory (a list that contains the path to dataA, dataB, dataC, dataD, and dataE)
2. Create a function to iterate over all the direcory paths where our data are located (list in 1.) and return the list of the image paths in those directories.
3. Create lists of image and mask paths by initializing the function above.
4. Preview some masked and unmasked images by reading them from their paths.

5.3 Create a data pipeline to read and preprocess our data

We will be using the tf.data.Dataset API to load our images and masks for our model to process. The Dataset API allows us to build an asynchronous, highly optimized data pipeline to prevent our GPU from data starvation. It loads data from the disk (images or text),

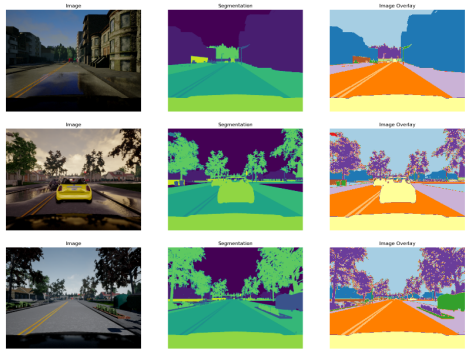

Figure 3: Random masked and unmasked images by reading them from their paths

applies optimized transformations, creates batches and sends it to the GPU. Unlike former data pipelines made the GPU, the Dataset API wait for the CPU to load the data, leading to performance issues.
To do this, we will:
1. Create a function to read image and mask paths and return equivalent arrays.
2. Create a data generator function to read and load images and masks in batches.
3. Create data pipelines for the training, validation, and test sets using both functions.
4. Preview sample images and their segmentations from the three dataset categories.

Again to read images and mask paths and return equivalent arrays, we will be using a read-Image function which will read an image and its mask from their paths and convert the digital image and its mask to image arrays. Also, it will normalize the datasets and resize the image and its masks to a desired dimension.

6 Model Training and Evaluation

Following are the steps we will take to design our model:

1. Create a function for a block of encoding. For each appropriate block in the model, the function will return the next layer output and the skip connection output.
2. Create a decoding block's function. The skip-connection input and the prior layer will be combined, processed, and output by this function.
3. Create a model using the output from the encoding and decoding phases.

However, there are total params: 8,652,247, Trainable params: 8,646,359 and Non-trainable params: 5,888. After training, we have gotten something like this:

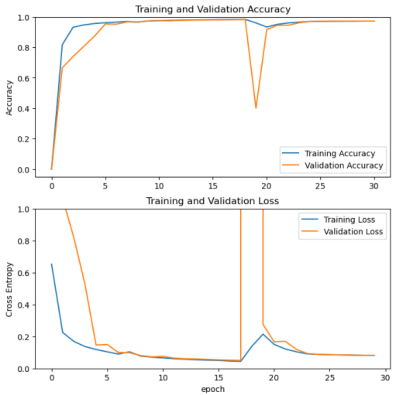

Figure 4: Training and Validation Accuracy, Loss chart

```
Model Accuracy on the Training Dataset: 97.34%
Model Accuracy on the Validation Dataset: 97.28%
Model Accuracy on the Test Dataset: 97.3%
```

Figure 5: Model Accuracy Rate

6.1 Model Evaluation

The model development process includes a step called model evaluation. Finding the model that best depicts our data and predicts how well the model will perform in the future is helpful. Precision and recall are frequently employed in addition to model accuracy in classification tasks to assess model performance because model accuracy isn't always enough to determine whether a model is optimal or

not (especially if our dataset is biased). The same principle holds for the majority of dense prediction tasks, such as image segmentation, whose objective is to streamline and/or transform an image's representation into classes that are more meaningful and understandable. Since the purpose of our model is to divide an input image into different classes, it is frequently challenging to tell if our model is having trouble accurately dividing one or more classes because it isn't always reflected in the model accuracy or readily visible to the eye. As a result, additional indicators are required to assess model performance.

Recall, precision, specificity, true detection rate (TDR), intersection over union (IoU), and F1-score will be used in this study as supplemental measures to assess the performance of our models. These metrics were determined by computing the confusion matrix between the predicted segmentations and the ground truth segmentations, which allowed for the identification of the variables true positive (TP), true negative (TN), false positive (FP), and false negative (FN). The definitions of these measures' expressions are as follows:

Precision = TP/(TP + FP)

Recall/Sensitivity = TP/(TP + FN)

Specificity = TN/(TN + FP)

True Detection Rate (TDR) = 1 - (FN/(TP + FN))

Intersection over Union (IoU)/Jaccard Similarity = TP/(TP + FP + FN)

F1-score(JS)/Dice coefficient = 2 ((Precision Recall)/(Precision + Recall))

After evaluating our model, we have received pretty decent results for predicted segmentations for both training and validation. The chart is given below.

	Class	Recall	Precision	Specificity	IoU	TDR	F1-Score
0	All Classes	0.78	0.83	1.0	0.73	0.78	0.8
1	Class 1	0.99	0.99	1.0	0.98	0.99	0.99
2	Class 2	0.97	0.94	0.99	0.91	0.97	0.95
3	Class 3	0.75	0.81	1.0	0.63	0.75	0.78
4	Class 4	0.68	0.79	1.0	0.58	0.68	0.73
5	Class 5	0.0	0.0	1.0	0.0	0.0	0.0
6	Class 6	0.54	0.79	1.0	0.47	0.54	0.64
7	Class 7	0.96	0.98	1.0	0.94	0.96	0.97
8	Class 8	0.99	0.99	1.0	0.99	0.99	0.99
9	Class 9	0.97	0.96	1.0	0.93	0.97	0.96
10	Class 10	0.96	0.95	0.99	0.91	0.96	0.95
11	Class 11	0.99	0.99	1.0	0.98	0.99	0.99
12	Class 12	0.84	0.84	1.0	0.72	0.84	0.84
13	Class 13	0.52	0.81	1.0	0.46	0.52	0.63

Figure 6: Predicted segmentations of the training images

	Class	Recall	Precision	Specificity	IoU	TDR	F1-Score
0	All Classes	0.77	0.83	1.0	0.72	0.77	0.79
1	Class 1	0.99	0.99	1.0	0.98	0.99	0.99
2	Class 2	0.97	0.94	0.99	0.92	0.97	0.95
3	Class 3	0.73	0.79	1.0	0.61	0.73	0.76
4	Class 4	0.65	0.79	1.0	0.55	0.65	0.71
5	Class 5	0.0	0.0	1.0	0.0	0.0	0.0
6	Class 6	0.5	0.77	1.0	0.43	0.5	0.61
7	Class 7	0.95	0.98	1.0	0.93	0.95	0.96
8	Class 8	0.99	0.99	1.0	0.99	0.99	0.99
9	Class 9	0.97	0.96	1.0	0.93	0.97	0.96
10	Class 10	0.95	0.94	0.99	0.9	0.95	0.94
11	Class 11	0.99	0.99	1.0	0.98	0.99	0.99
12	Class 12	0.83	0.84	1.0	0.72	0.83	0.83
13	Class 13	0.53	0.8	1.0	0.47	0.53	0.64

Figure 7: Predicted segmentations of the validation images

7 Predict segmentations using the trained Model

Despite the fact that our model has quite good accuracies and IoUs on our training, validation, and test datasets, seeing how it performs on these datasets should help us make even more progress. Thus, now we will perform the following steps:
1. Create a function that preprocesses a set of photos and shows the true state, true mask, and predicted mask for each one.
2. Masks of the training set's images can be predicted and compared.
3. Predict and evaluate picture masks from the validation set.
4. Predict and contrast the masks of the test set's photos.
Now, lets Predict and compare masks of images in the training set:

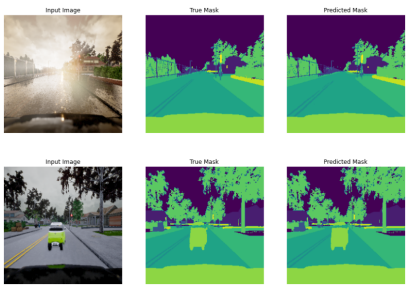

Figure 8: Predict and compare masks of images in the training set

For validation set:

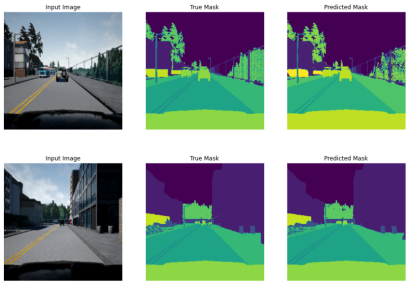

Figure 9: Predict and compare masks of images in the validation set

and finally for test set:

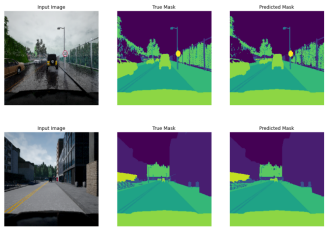

Figure 10: Predict and compare masks of images in the testing set

8 Conclusion

Semantic segmentation is a very important process for recognizing self-driving cars. This plays an important role in understanding private vehicles in traffic. As deep learning techniques have improved over the past decade, an increasing amount of research has focused on using deep learning to achieve better outcomes in all aspects of autonomous processes, including cognition and decision making. In this work, we proposed a U-Net model that builds and trains on the design principles of deep learning models and uses augmentation techniques to improve the training quality. Our goal was to make the designed architecture as accurate as possible for real-time observation. The model was trained and tested on the CARLA dataset and achieved high accuracy with a relatively higher number of parameters. A comparison was done, which proved the efficiency of our model compared with state-of-the-art models. In the future, more research can be conducted to increase the performance of the proposed model and overcome the current drawbacks.