Drivable Area Detection

Chau Huynh, Sandesh Banskota
Instructor: Mari Ostendorf
TA: Sitong Zhou
University of Washington
Seattle, WA 98195 USA

ABSTRACT

Image segmentation is a critical part of autonomous vehicle perception algorithms. In this paper, we present a Convolution Network architecture to detect drivable surface area on images taken from the dashboard of vehicles. The convolution network developed in this project is heavily influenced by U-Net[1], the state of the art in medical image segmentation, which relies mainly on augmented data rather than a large dataset to train the network. We show that this medical image segmentation technique is quite practical for drivable area detection for autonomous vehicles. We trained and tested the network on images and labels from the Berkeley Deep Drive (BDD) dataset[2] and achieved outstanding results on clear, daytime images of roads and satisfactory results on nighttime and crowded images. The network is fast and can keep up with the bandwidth of on-demand drivable area detection for autonomous driving applications.

1. INTRODUCTION

In recent years, self-driving cars have been rapidly developed around the world. Autonomous driving softwares relies heavily on object recognition techniques to identify traffic indicators and avoid obstacles. To allow autonomous vehicles to recognize its surroundings, object detection and segmentation algorithms are an area of much current research. Recognition of surroundings often includes detecting drivable roads, detecting lanes, identifying curbs, traffic signs, and obstacles used for navigation. Reliable drivable area segmentation or detection provides a critical foundation for avoiding obstacles during autonomous driving.

Previous researches have shown that convolutional neural networks (CNN) perform well for image classification tasks and have been widely used for this application. However, since CNN typically assigns only a single class label for an image, it is insufficient for image segmentation, where each pixel needs its own label. More advanced convolutional network architectures have been developed to perform image segmentation tasks. U-Net is a model that does not have any fully connected layer and only uses the relevant convolutional layer to retain the context from the input image. This convolution network architecture implements upsampling operations after traditional convolutional layers with downsampling using maxpool. This upsampling allows the network to produce an output with the same resolution as the input. The downsampling part and the upsampling part are symmetrical, resulting in a u-shape architecture. This U-Net also propagates context information from the downsampling part to the corresponding upsampling part of the same level, allowing the network to reinforce information learned from previous layers.

This paper applied a simplified U-Net architecture to perform drivable areas detection. This architecture provides a solid way to classify each pixel as belonging to a drivable surface or not. After implementing, training, and testing with the Berkeley Deep Drive dataset, the CNN model was able to predict drivable area by an accuracy of 81%. For comparison, a baseline classification algorithm was implemented using logistic regression. The accuracy of the baseline algorithm is 56%. The model was able to increase the accuracy by around 30%.

2. APPROACH

A. <u>DATASET</u>

The BDD dataset contains 90k RGB images from dashboard of vehicles looking towards the road and correspondingly labeled 90k colored label images that either have a [0,0,0] (black) RGB value, indicating the pixel is not part of the drivable area class on the image, or a non-black RGB value, indicating that the pixel is part of the drivable area class on the image. Figure 1 is an example of a pair of an input image and its corresponding label. With these 90k samples, the dataset was split into 50k for training, 20k for validation, and 20k for testing. Since the dataset is large, the cross validation method is using a held-out set instead of k-fold validation.

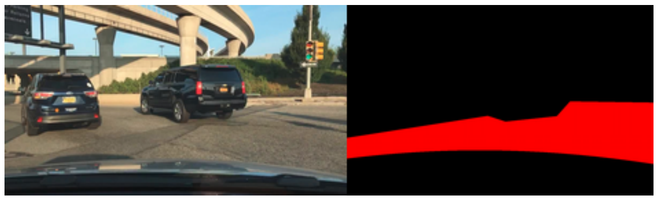

Figure 1. *Example input image and corresponding label in the BDD dataset*

B. IMAGE PREPROCESSING

Due to limited computing resources, both the input and the label images' resolutions were reduced from the original 1280x720 to 128x72 using nearest neighbor filtering. While this reduced the dimension of the images, the filtering did not reduce much information since many features in the images are generally large and are distinctly visible in the compressed images.

In addition to being compressed, the label images were modified to grayscale. Finally, while being converted to numpy arrays, any non-black pixel was replaced with 1, indicating the pixel is of the drivable class, and the rest of the pixels were assigned a zero. The input images were also converted to numpy arrays, but the original RGB values were maintained.

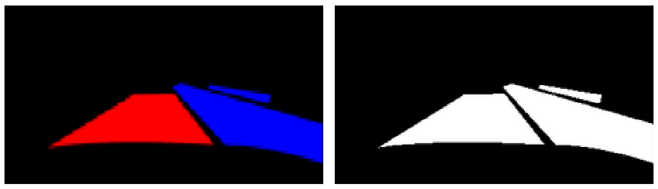

Figure 2. *Initial labelled image and converted labelled image where white is drivable, black is not drivable*

C. COMPUTATIONAL SOFTWARE AND HARDWARE

The CNN in this project was built using PyTorch and primarily its torchvision libraries. The training, validating, and testing happened on a CUDA 11.0 enabled machine with a NVIDIA GeForce GTX GPU with a 4 GB memory size.

D. CONVOLUTION ARCHITECTURE

We incrementally built the CNN in order to improve performance without having to build an unnecessarily large neural network. We also referenced the results from the U-Net paper to improve our model. We chose BCEWithLogitsLoss as it is typically best suited for image reconstruction error and due to its numerical stability.

$$\ell(x,y) = L = \{l_1,\dots,l_N\}^\top, \quad l_n = -w_n\left[y_n \cdot \log\sigma(x_n) + (1-y_n)\cdot\log(1-\sigma(x_n))\right]$$

Figure 3. *BCEwithLogitsLoss function*

The initial design was using a CNN with two convolution layers. This did not work well and the loss value saturated at 0.4. Adding too many convolution layers in series was also not working since much of the feature was being lost by the end of the network. After many iterations, we built the CNN depicted in Figure 4.

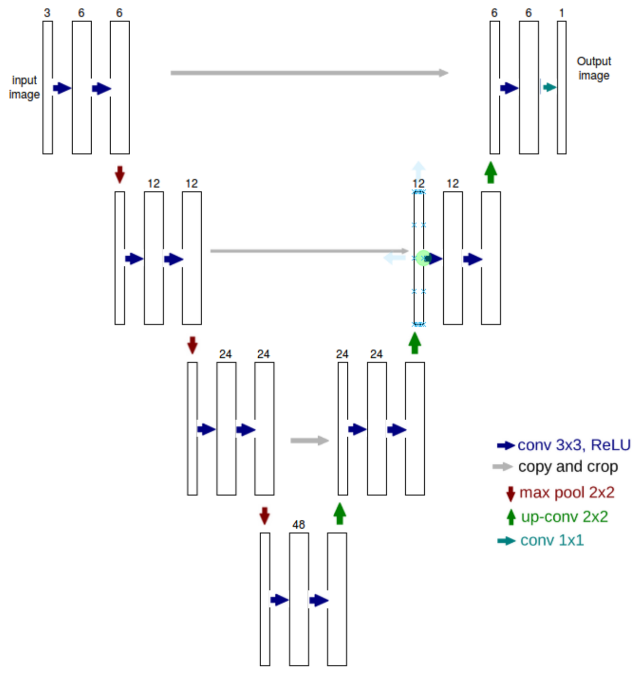

Figure 4. *Final Convolution Network Architecture*

The architecture was heavily influenced by U-Net. Though U-Net was built for medical application, it has characteristics that are transferable to this application. The drivable area detection model has to classify the pixels but also has to reconstruct the position mapping of the pixels. The initial designs had many convolution layers but much of the positional and edge information was getting lost as images were convolved many times. The bypass allows this type of positional information to be saved from the initial convolution layers and concatenated with the output of a later convolution layer. The convolution levels and bypassing allows the model to learn classification and positioning.

3. EXPERIMENTAL RESULTS

The performance of the model was evaluated based on accuracy. For each input image, the model outputs a scale of how likely each pixel belongs to a class. The pixels are then assigned to the class that they are more likely to be in. The accuracy is calculated by taking the ratio of the total number of pixels in the predicted images that match the true label images, over the total number of pixels to classify.

A. CNN RESULTS

The model was trained with 1000 batches, each batch has 50 image and label pairs, to cover all of 50k training images. The learning rate was set at 0.005 and the model made 5 passes over the entire training set. Figure 5 below shows the evolution of the loss values throughout the training process.

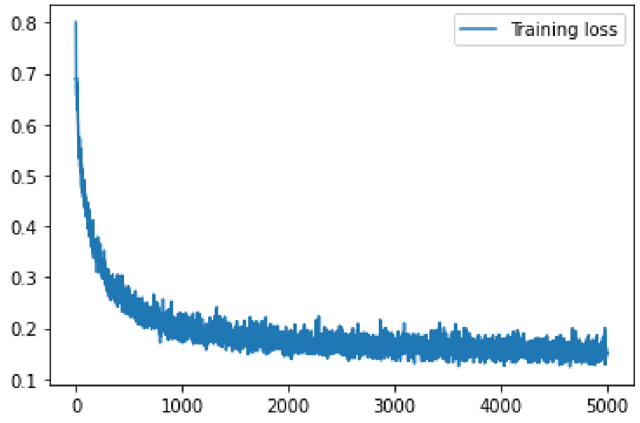

Figure 5. *Evolution of training loss over the epochs*

When tested with the test data set, the CNN model produced predictions with the average accuracy of 81.38%. The average testing loss over the batches was 0.144, reflecting the value at the end of the curve in Figure 5 above. This loss value shows that with the test set, the model performs as well as with the training set, so the model did not overfit the training set. The model also performed well on both daytime and nighttime images. To visualize the performance of the model, refer to Figure 6 and 7 below for examples predicted output during the daytime and nighttime.

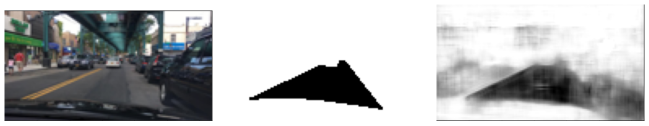

Figure 6. *Daytime example of input image, label image, and predicted output image*

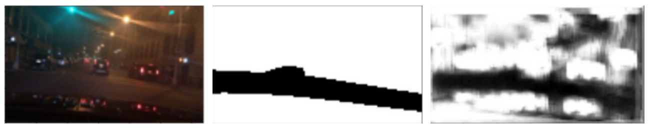

Figure 7. *Nighttime example of input image, label image, and predicted output image*

B. BASELINE RESULTS

To better evaluate the results of the CNN model, a simple binary classification method was created using logistic regression. Since logistic regression can only assign a single label for one input, some additional preprocessing steps were necessary. First, for simplicity, input images were converted into grayscale. Second, input images were padded and broken down into a list of 11x11 square patches, where each pixel in an input image will have one 11x11 square patch associated with it. These square patches will then be used as the input for the logistic regression model. The label images are unrolled into a one-column vector and then used as the true label for the model.

To prevent overfitting, the logistic regression model was trained with several regularization approaches and hyperparameter alpha for the amount of regularization. The regularization schemes attempted were L1 regularization, L2 regularization, and no regularization applied. For alpha, the model was trained with values 0.10, 0.25, 0.5, 0.75, 1. The model was also set to balance out the class weights for each class to avoid predicting the most frequent class every time. Using the validation data set, the parameters with the highest accuracy were L1 regularization with 0.25 for the alpha. With these parameters, the logistic regression model achieved an accuracy of 56.141% on the test data set. Figure 8 below is an example of how well the logistic regression model could predict for a single input image.

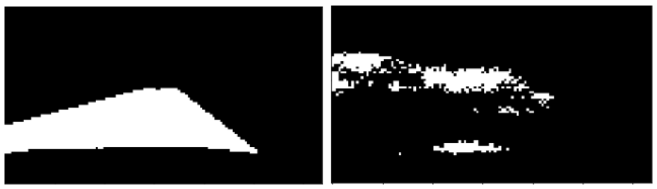

Figure 8. *Example of label image and logistic regression predicted output image*

4. ANALYSIS

Table 1 below shows that the CNN performed better than the baseline logistic regression.

	Logistic regression	CNN
Accuracy	56.14%	81.38%

Table 1. *Accuracy results*

An accuracy of 56.14% for a binary classification problem indicates that the logistic regression model only does a little bit better than randomly guessing the label for each pixel. Meanwhile, the CNN model could detect the drivable area with a much higher accuracy. However, there are some edge cases in the data that affects the CNN performance. For example, there might be a patch of road in the input image that is too small to fit a car. This area is not labeled as drivable by the true label image. However, our model will classify this area as drivable. This labeling from the model is technically correct, even though it does not match the label. These edge cases require additional processing on the model's output images. For this example, a post-processing method can be calculating the area of the predicted drivable area and rejecting the areas that are smaller than the dimensions of the vehicle. Without these edge cases, the model would have performed better, producing a higher pixel label accuracy.

5. SUMMARY

While the developed CNN architecture is functional, it could be further improved with better training hardware and data augmentation to better account for edge conditions. The CNN performs better than the baseline. It also does a great job in clear and simple conditions but struggles with rainy, shadowy, and nighttime conditions. given the hardware resources it was trained on.

The promise in using U-Net for drivable area image segmentation is that fewer images might be needed to train the model. If further improved, this model could be better at handling rarely seen edge cases than other models.

There were many limitations in this project. Our model could not train on data with large dimensions, thus some of the feature information was lost in the preprocessing image reduction step. Additionally, there was a limitation to the number of layers and channels that could be used because of GPU memory constraints. The data was also sometimes flawed with significant glare and inconsistent dashboard sizes.

While the model could be further improved, it is currently fast and accurate enough for general driver verification in daytime, clear conditions. The result of the limited CNN is that although the result isn't as accurate as it could be, the processing time is quick. This model could be used for non-safety critical drivable detection applications as it is.

REFERENCES

[1] 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.

[2] BerkeleyDeepDrive, https://bdd-data.berkeley.edu/, 2020.

[3] BCEWithLogitsLoss, https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html, 2020.