Robust Cityscape Segmentation

Sebastian Joseph
Eindhoven University of Technology
Eindhoven, Netherlands
Email: s.joseph1@student.tue.nl

Xingyi Li
Eindhoven University of Technology
Eindhoven, Netherlands
Email: x.li@student.tue.nl

Abstract—**Most semantic segmentation models perform well under standard weather conditions and sufficient illumination but struggle with adverse weather conditions and poor image quality. Collecting and annotating training data under these conditions is expensive, time-consuming, error-prone, and not always practical. Usually, synthetic data is the most feasible way to overcome this situation. The model architecture can play an important role in accuracy too. We propose a training procedure that can perform accurate semantic segmentation in all weather conditions. We used the Cityscapes Dataset as our base dataset and checked the performance of three different model architectures. Our code is available at https://github.com/5LSM0/team7.**

1. Introduction

Autonomous driving is expected to be an achievable dream within this decade. It can help mankind to reduce traffic congestion, reduce carbon emissions and improve safety. Developing object detection models with good performance is an important obstacle to achieving this. The model architecture and training process plays a significant role in the performance. A baseline model with U-net architecture was developed for assignment 2 that can perform semantic segmentation on the Cityscape dataset with a mean dice score of 0.313. The Cityscape dataset contains high-resolution images with pixel-wise annotations of various objects such as vegetation, vehicles and buildings from various cities in Germany. Most images in the dataset are taken during daytime and clear weather conditions. Due to this the baseline model doesn't perform well for images of cities outside Germany, of poor image quality and taken during adverse weather conditions. This issue is attempted to overcome here. This issue is attempted to overcome here. The ideal solution to overcome this issue will be to collect new images in these conditions and train the model on these images. But collection and annotation of these images is a tedious process. The practical solutions are data augmentation and model upgrades. To improve the segmentation, the UNet architecture was upgraded to a more complex model such as ResNet50 and DeepLabv3+. The data augmentation and performance of different models are explained in the following sections.

2. Data Augmentation

The CityScape dataset is the dataset used for training. To improve the performance of segmentation in weather conditions such as rain, fog and snow, new images are created by augmentation on the cityscapes dataset. Augmented images for rain and fog were obtained from the cityscape website [1] itself. The rain image dataset contains augmented images with different rain patterns and raindrop size on the images of cityscape dataset. Similarly, the fog dataset images contain fog augmentation of different parameters for every image of cityscape dataset. The training image for snow weather conditions was created by us itself. A percentage of pixels associated with vegetation and pavement was updated to make it look like snow. Then a random number of white circles that looked like snow was added to the image. Finally, random pixels are selected and increased brightness to give a snowy atmosphere. All these transformations are done probabilistically to prevent a fixed transformation. The augmented dataset only contained the images, the labels were created from the cityscape dataset based on their filenames. The dataset contains images of size 2048 x 1024. It was resized to 256 x 128 for fast training. Data augmentation such as adding random noise and changing brightness is applied randomly to the image when it is loaded to the data loader. These transformations will help to increase performance for low-quality images. The images and masks are also rotated and flipped randomly to familiarise the model with images from both left and right side driving countries.

3. Model

3.1. UNet

A U-Net architecture was used to implement the baseline model. It is a widely used architecture for the segmentation task. The architecture can be divided into two parts, encoder and decoder. The encoder is a series of convolutional and max pooling layers that will perform the feature extraction. It will extract high-level features of the image while reducing the spatial information and increasing the feature channels. Decoder consists of a series of upsampling and convolutional layers that will produce the segmentation mask of the image from the feature map. It will increase spatial information and decrease the feature channels during its upsampling process. U-Net has skip connections between the encoder and the decoder. It is able to regain the spatial information lost during downsampling due to these skip connections.

A 10-layer U-Net was used for the baseline implementation. Each layer had two convolutional-BatchNorm-ReLu transformations. The model was trained with cross-entropy loss function and Adam optimizer. The learning rate was 1e-4. It was trained for 40 epochs and the loss is plotted in figure 1. It was trained only on the normal dataset.

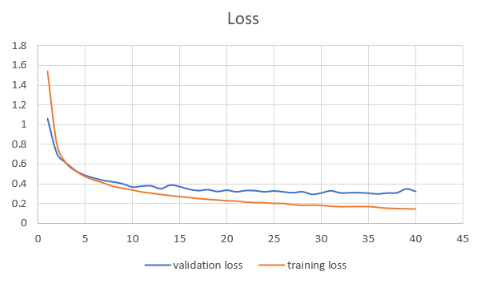

Figure 1. UNet Fixed Learning rateTraining

We were able to obtain a mean dice score of 0.313 for this model.

3.2. ResNet 50

ResNet50 is a deep network that was introduced in 2015 by Kaiming He et al. at Microsoft Research [2]. ResNet 50 is a variant of the ResNet architecture that has 50 layers. It performs very well with tasks related to image processing. The main part of the model is residual connections which connect the output of the previous layer to the input of the next layer. This residual connection helps to develop deeper models by solving the issue of vanishing gradients. The model consists of backbone, classifier and auxiliary classifier. The backbone contains convolutional layers and residual blocks that can perform feature extraction. The classifier is a fully connected layer with softmax activation that produces the output classification. The auxiliary classifier works in parallel with the classifier. It is only used during training and discarded during prediction. It is added to reduce the issue of vanishing gradient.

A model with just one layer was created. The pretrained FcnResNet50 [3] model for segmentation of the pytorch library was assigned as its only layer. The initial weights provided by the library were used to initialise the model. Then the model was fine tuned for our dataset. The classifier, auxiliary classifier and layer 4 of the backbone were unfrozen while keeping the other layers frozen. The model was trained for 20 epochs with a fixed learning rate and as shown in figure 2 it was observed the validation loss gets higher than the training loss after 4 epochs. Gradual unfreezing is a very common method used to fine-tune models and is expected to provide better results. The fixed learning rate was replaced by gradual unfreezing with varying learning rates. The inner layers were unfrozen and the learning rate was reduced periodically during training. But the performance was similar to the fixed learning rate as shown in figure 3.

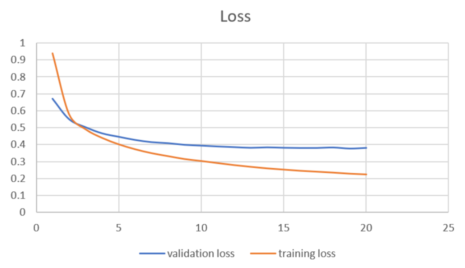

Figure 2. ResNet50 Fixed Learning Rate Training

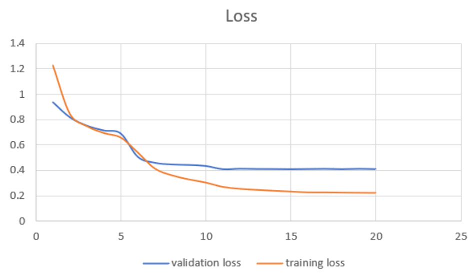

Figure 3. ResNet50 Gradual Unfreezing Training

This model had many advantages over U-net. ResNet50 is a deep network and has a high computational cost. But since we were able to use the pre-trained model, the computational cost was very low. Due to the depth of the model, it is able to segment more accurately than the U-net model.

There are several factors that should be taken into account before utilizing this model. The model gets overfitted very easily. It can be observed in the figures related to loss. The model was getting overfitted easily without learning key features. Its performance was improving on the trained data significantly with training but the performance of the test image was remaining bad. So Deeplabv3+ was implemented.

3.3. Deeplabv3+

Therefore, after considering the limitation of ResNet 50 model, we will enhance semantic segmentation with improved ASPP and DeepLabV3+ architecture.

DeepLabv3+ is an advanced and efficient semantic segmentation model that builds upon the previous versions of DeepLab. It employs an encoder-decoder architecture with atrous spatial pyramid pooling (ASPP) to capture multi-scale contextual information [4]. DeepLabv3+ improves the segmentation performance by refining object boundaries and capturing fine-grained details. We can see the original model structure in Figure 4.

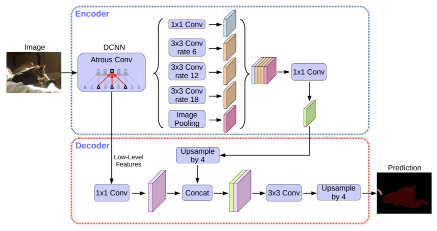

Figure 4. DeeplabV3+ Model Net Structure

3.3.1. Model Structure . Below is the working structure of this improved DeeplabV3+ Model:

1.Input: The model takes the RGB graphs with 256"*"128 size as input.

2. Backbone (ResNet-152): The backbone is responsible for extracting hierarchical features from the input image. It is composed of a series of convolutional, batch normalization, ReLU activation, and pooling layers. In our model, we use a pre-trained ResNet-152 as the backbone. The backbone network's output, specifically the features from layer4, is passed to the ASPP module. Low-level features from layer1 are later combined with the decoder's output to refine the segmentation.

3. ASPP (Atrous Spatial Pyramid Pooling) module: The ASPP module captures multi-scale contextual information by employing parallel atrous convolutions with different dilation rates. The module incorporates separable convolutions as an optional choice in the ASPPConv module. Separable convolutions factorize a standard convolution into a depthwise convolution followed by a pointwise convolution, resulting in fewer parameters and less computation compared to standard convolutions. This modification can lead to a more efficient model without significantly sacrificing performance. Meanwhile, the ImprovedASPP consists of two sets of atrous rates: (6, 12, 18) and (12, 18, 24) which enables the model to better segment objects with varying sizes and shapes, making it more robust to scale variations in the input data. Each set is combined with a 1x1 convolution and a global average pooling branch, creating a rich feature representation at different scales.

4. Projection layer: After processing the input through the ASPP module, the features are concatenated and passed through a 1x1 convolutional layer to reduce dimensionality. This is followed by batch normalization, ReLU activation and dropout layers.

5. Decoder: The DeepLabPlusDecoder is responsible for upsampling the output from the ASPP module and combining it with low-level features from the backbone's layer1. This fusion of low-level and high-level features refines the segmentation map, preserving fine-grained details and accurate object boundaries. The decoder consists of several convolutional, batch normalization, ReLU activation and dropout layers to process the combined feature maps, which is different from that in the original DeepLabv3+ decoder. These changes can provide more capacity for refining the segmentation map and potentially improve performance.

6. Output: The final output of the model is an upsampled segmentation map with the same height and width as the input image. Each pixel in the segmentation map corresponds to a class label. However, it doesn't include a softmax activation layer, unlike the original DeepLabv3+ model.

All in all, this DeepLabv3+ model leverages a ResNet-152 backbone for hierarchical feature extraction, an ImprovedASPP module to capture multi-scale contextual information, and a decoder to refine the segmentation map by fusing low-level and high-level features. By running this model, it may provide more diverse feature representations, increased efficiency, and better segmentation refinement.

3.3.2. Model Adjustment. Compared to the previous model, the following improvements have been made

Data Argumentation We added the brightness adjustment section, which can randomly adjusting the image brightness helps the model to be more robust against variations in lighting conditions. Meanwhile, we added the color dithering section, which can randomly perturbing the brightness, contrast, saturation, and hue of the image can help the model become more robust to changes in color and lighting conditions. By incorporating these data augmentation techniques, we can improve the model's ability to generalize to new and unseen data, potentially leading to better performance and increased robustness against various input variations.

Learning Rate We added ReduceLROnPlateau learning rate scheduler to this model. This scheduler adjusts the learning rate based on a monitored metric. If the metric does not improve for a specified number of consecutive epochs, the learning rate is reduced by a factor. The benefits of using this learning rate scheduler include the ability to automatically find an optimal learning rate during training and the potential to escape local minima or plateaus in the loss landscape by reducing the learning rate when progress stalls. For our model, after several training experiments, we finally set patience=7 and factor=0.1 to improve the efficiency and stability of model's training process.

Atrous rates The atrous rates determine the dilation factors used in the atrous spatial pyramid pooling (ASPP) module, which is a key component in the DeepLabV3+ architecture. The larger the atrous rate, the more spaced-out the resulting features will be, which can help capture more global context information. Meanwhile, the combination of different atrous rates can also help the model better delineate object boundaries and capture fine-grained details. After several training experiments, we finally use two sets of atrous rates: (6, 12, 18) and (12, 18, 24).

Epochs and Batch Size By several training experiments, we set our epochs=40 to lead to better generalization and performance on the validation set. We will use smaller batch sizes(10), which can lead to more frequent weight updates and help the model converge faster and escape local minima. Additionally, smaller batch sizes tend to provide better generalization due to the inherent noise in the gradient estimates.

3.3.3. Results. The improved DeepLabV3+ model capitalizes on the strengths of the original DeepLabV3+ architecture while incorporating several enhancements, and these modifications have significantly contributed to the model's superior performance in capturing fine-grained details and accurately delineating object boundaries.

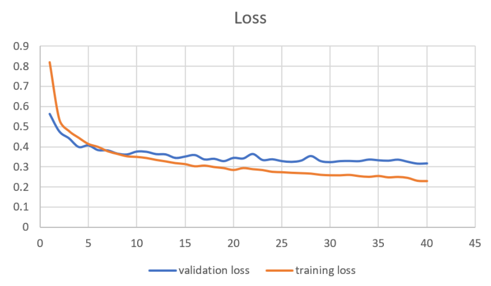

Figure 5. DeepLabV3 Loss Function During Training

3.3.4. Model Limitations and Improvement. There are several limitations and improvements.

Model Architecture

1. The model does not include 1x1 convolution layers, which are often used to reduce the number of feature channels and improve computational efficiency. Including 1x1 convolution layers in the model might help to reduce the number of parameters and computational cost without significantly affecting performance. Therefore, we can incorporate 1x1 convolution layers to reduce the number of feature channels and improve computational efficiency.
2. The model does not include a softmax activation function in the output layer. Adding softmax activation can help normalize the output probabilities and provide more interpretable probability distributions for each class. Therefore, we can add softmax activation in the output layer to provide normalized class probability distributions.
3. The model uses a ResNet-152 backbone, which is quite deep and computationally expensive. For the improvement, we can use other alternative backbones, such as EfficientNets or MobileNets, could result in a more efficient model with faster inference times while maintaining comparable performance.
4. The model employs two sets of atrous rates in the ASPP module, it may be beneficial to explore different combinations of rates or even additional sets to capture more diverse context information.

Training Procedure

1. Implement an early stopping mechanism to halt training when the validation performance does not improve for a specified number of epochs. This can save computational resources and prevent overfitting.
2. Use K-fold cross-validation to obtain a more reliable estimate of your model's performance on unseen data, as well as to reduce overfitting.

Data Argumentation

1. Implement advanced data augmentation techniques like MixUp and CutMix to create new samples by combining existing images and labels. This can help the model generalize better to unseen data.
2. As the cityscpes dataset has imbalanced classes, consider using techniques like oversampling, undersampling, or weighting the loss function may be help balance the class distribution during training.

Learning Rate

1. Utilize a learning rate finder technique to systematically find a suitable learning rate for the optimizer by training the model for a few epochs with different learning rates and observing the loss curve.
2. We can try other optimizers like SGD with momentum, RMSprop, or AdamW that can potentially improve training dynamics and convergence.

4. Conclusion

In conclusion, we have extensively analyzed and compared the performance of three semantic segmentation models: Improved DeepLabV3+, U-Net, and a ResNet-50 based model. We used Mapillary Vistas Dataset to create the test data. The segmenatation mask created by the models is shown in the below figures. Through rigorous experimentation and evaluation on test dataset, we have demonstrated that our modified DeepLabV3+ model outperforms the other two models on the test set.

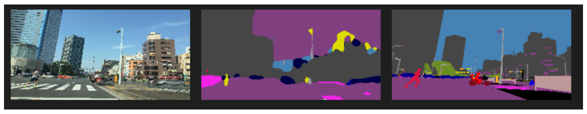

Figure 6. Test Result of DeepLabV3 Model

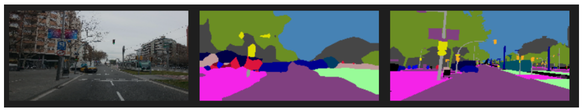

Figure 7. Test Result of ResNet50 Model

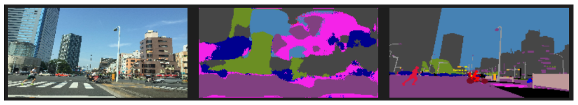

Figure 8. Test Result of UNet Model

References

[1] CityScape Dataset Website, https://www.cityscapes-dataset.com/

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. ”Deep Residual Learning for Image Recognition.” *CoRR*, vol. abs/1512.03385, 2015. http://arxiv.org/abs/1512.03385.

[3] FCN ResNet 50 Pytorch , https://pytorch.org/vision/stable/models.

[4] Chen L C, Zhu Y, Papandreou G, et al. Encoder-decoder with atrous separable convolution for semantic image segmentation[C]//Proceedings of the European conference on computer vision (ECCV). 2018: 801-818.