Object Segmentation (Self Driving Car)

Aufa Biahdillah

8 November 2024

Background & Problem Statement

- Currently, an autonomous vehicle is a trend among society, including a self-driving car.
- One of the issues of a self-driving car is the ability to identify and segment objects within the surroundings.
- Object segmentation is crucial since it allows the vehicle to distinguish between objects (such as pedestrians, other vehicles, traffic signs, and obstacles) and understand their spatial relationships.
- Therefore, training some models to do object segmentation is required in this situation.

Objectives & Scope

- The objectives of this project are:
  - Conducting object segmentation experiments applying two models: UNet and FCN8s.
  - From those experiments, the best algorithm is concluded.

- For study purposes, this project is limited to:
  - The dataset is obtained from Indonesia AI, which comes from the Cityscapper dataset.
  - The dataset contains the surroundings of a car.

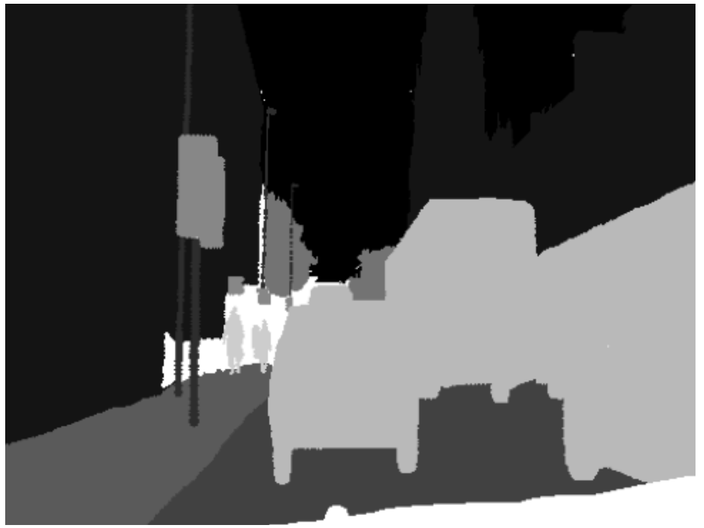
Data Collection

- The dataset has been separated into 367 train datasets and 101 test datasets.
- There are some duplicate images in the dataset, which can be recognized from the file names containing (1) or (2).
- The annotation of the images has been included in the dataset but needs to be grayscaled for visualization.

Sample Image

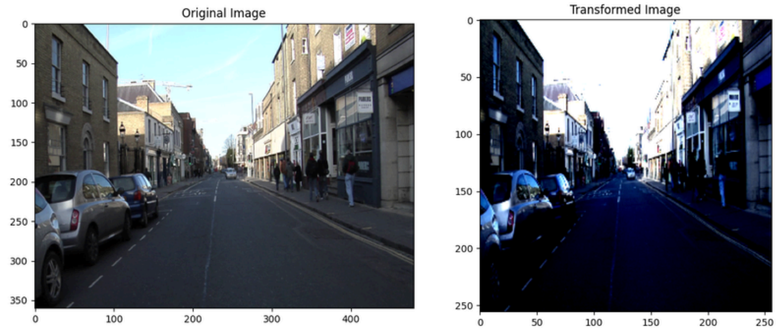
Original Annotation Image

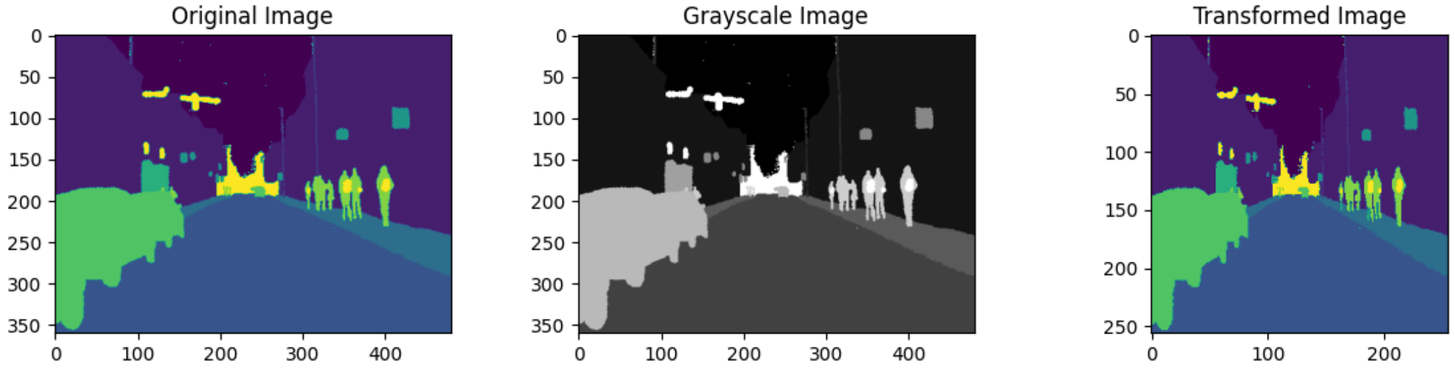
Grayscale Annotation Image



Data Transformation

- We create the transformation function that will be applied to the images and their annotation in the dataset.
- The transformation is simply a resize to 256 x 256 and normalization.





Dataset, Train Val Split, and Dataloader

- We created the Cityscapes dataset class, which consists of image conversion to RGB, annotation conversion to L (grayscale), numpy array conversion, and one hot encode of the mask (there are 12 classes in the annotation image).
- We split the training dataset into the train and validation datasets with proportions of 80% and 20%, respectively.
- Finally, we created the data loader for train, val, and test datasets with 32 batch sizes.

UNet

Model Construction

We set the following for the model and its training process.

UNet
Construct the model class with double convolution.

Epoch
We set the number of epoch to 100.

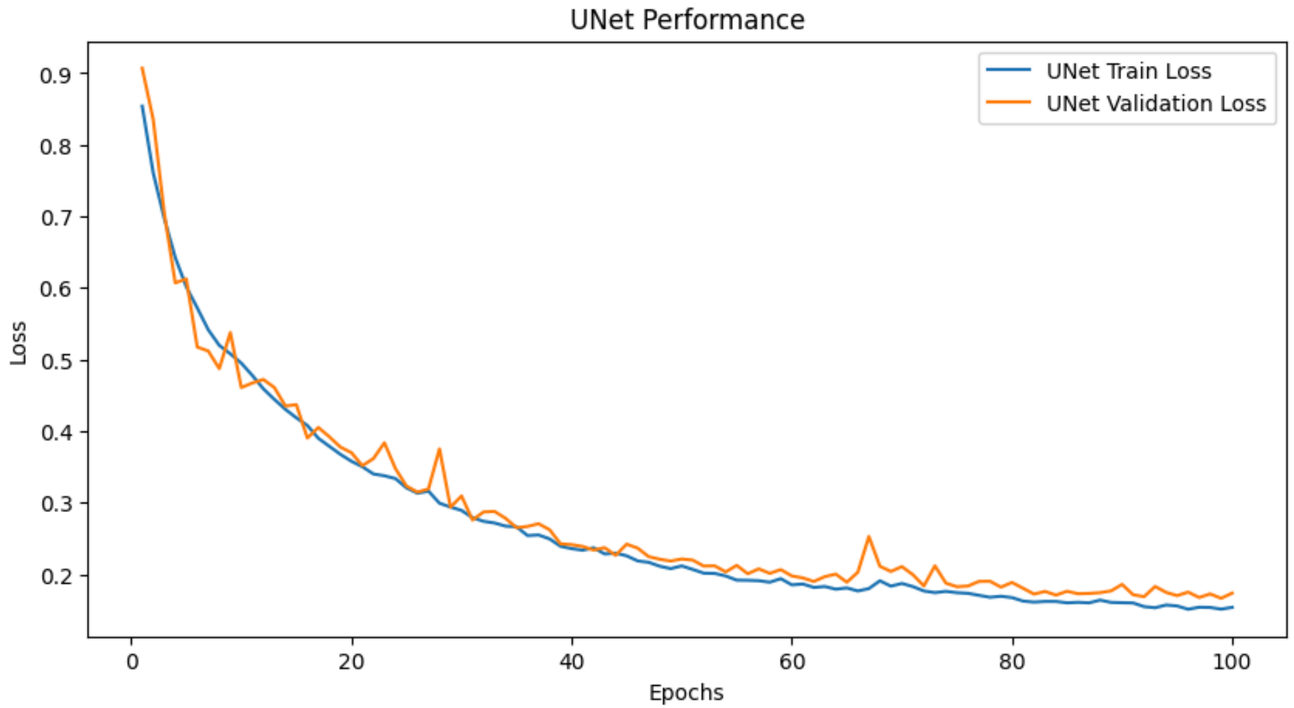
Optimizer
Adam with learning ret is set to 0.0001 or equal to 1e-4.

Channels
Set the in channels to 3 (RGB) and out channels to 12 (class)

Loss
We construct the dice loss and apply it during training.

Gradscaler
Apply the gradscaler during training to speed up the process

The Training Performance



We can see that the model succeed to decrease the loss value from around 0.9 to around 0.1. We also see that the distance between the train loss and validation loss is not high. Hence, we conclude that it is neither overfit nor underfit.

FCN8s

Model Construction

We set the following for the model and its training process.

FCN8s
Construct the model class with VGG 16 as the backbone.

Epoch
We set the number of epoch to 100.

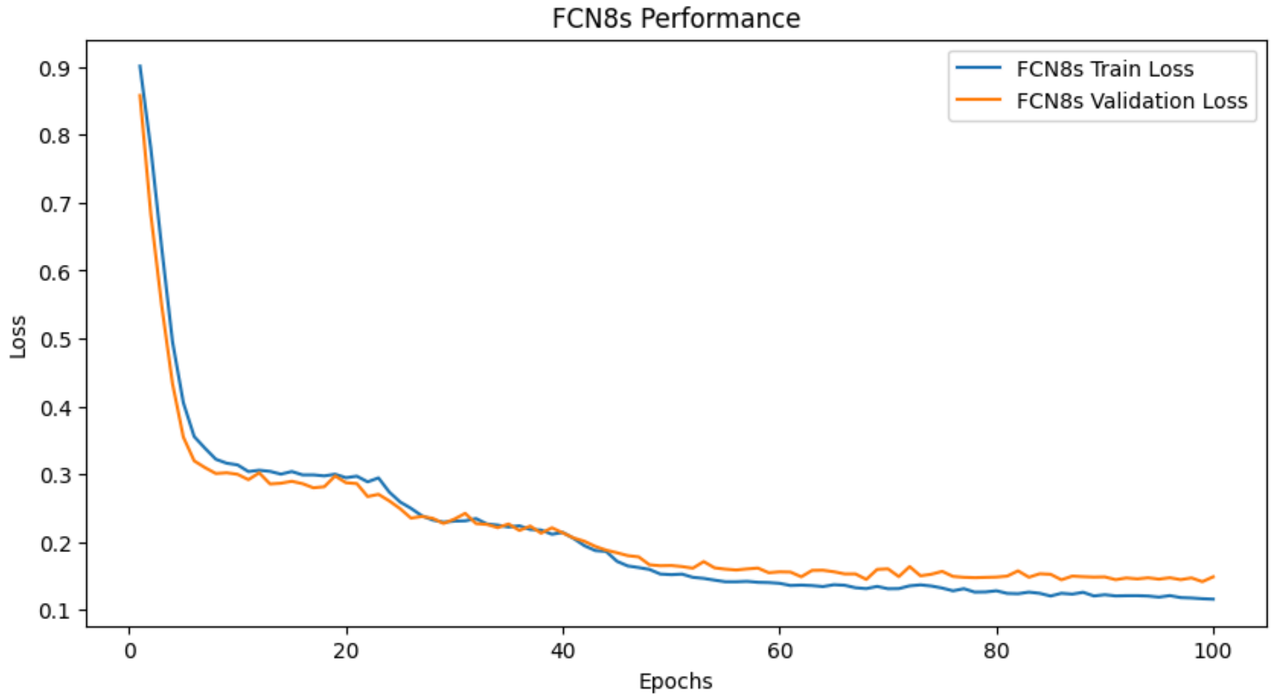
Optimizer
Adam with learning ret is set to 0.0001 or equal to 1e-4.

Num Classes
Set the num classes to 12, including the background.

Loss
We construct the dice loss and apply it during training.

Gradscaler
Apply the gradscaler during training to speed up the process

The Training Performance



Similarly, we can see that the model succeed to decrease the loss value from around 0.9 to around 0.1. We also see that the distance between the train loss and validation loss is not high. Hence, we conclude that it is neither overfit nor underfit.

Evaluation and Comparison

Loss Comparison

Model	Training Loss	Validation Loss	Test Loss
UNet	0.153932	0.173742	0.223557
FCN8s	0.115978	0.148911	0.161057

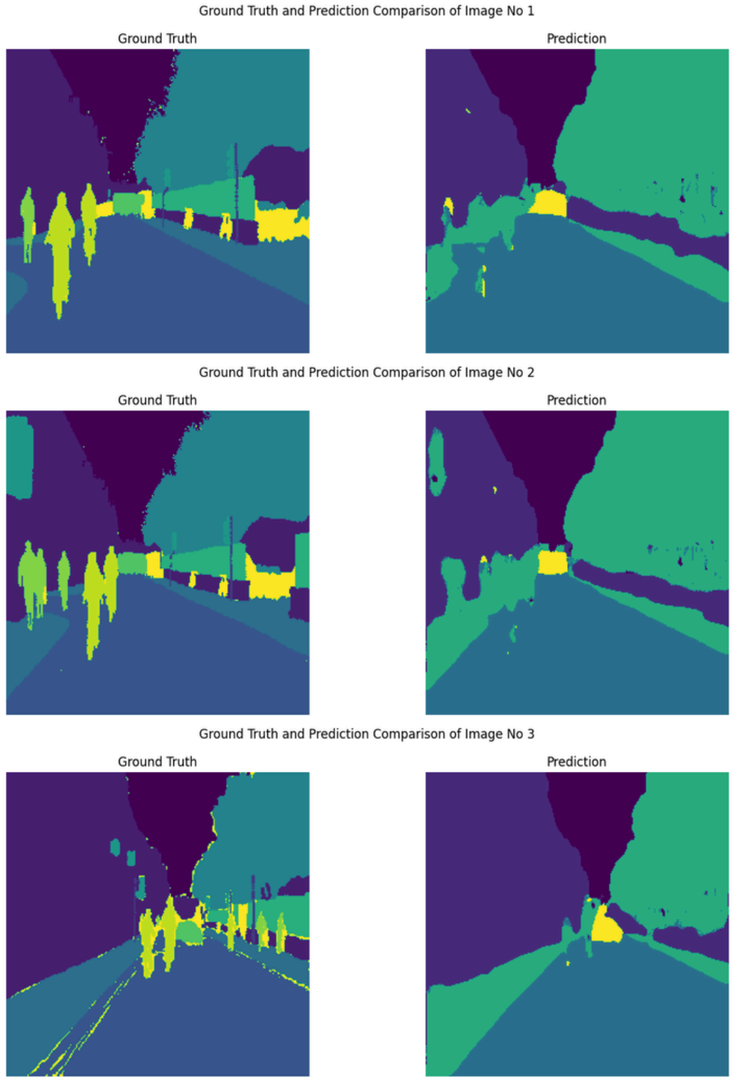
- We attain that in each phase, the loss of FCN8s is smaller than that of UNet. This indicates that FCN8s performs better than UNet.

Dice Coefficient Comparison

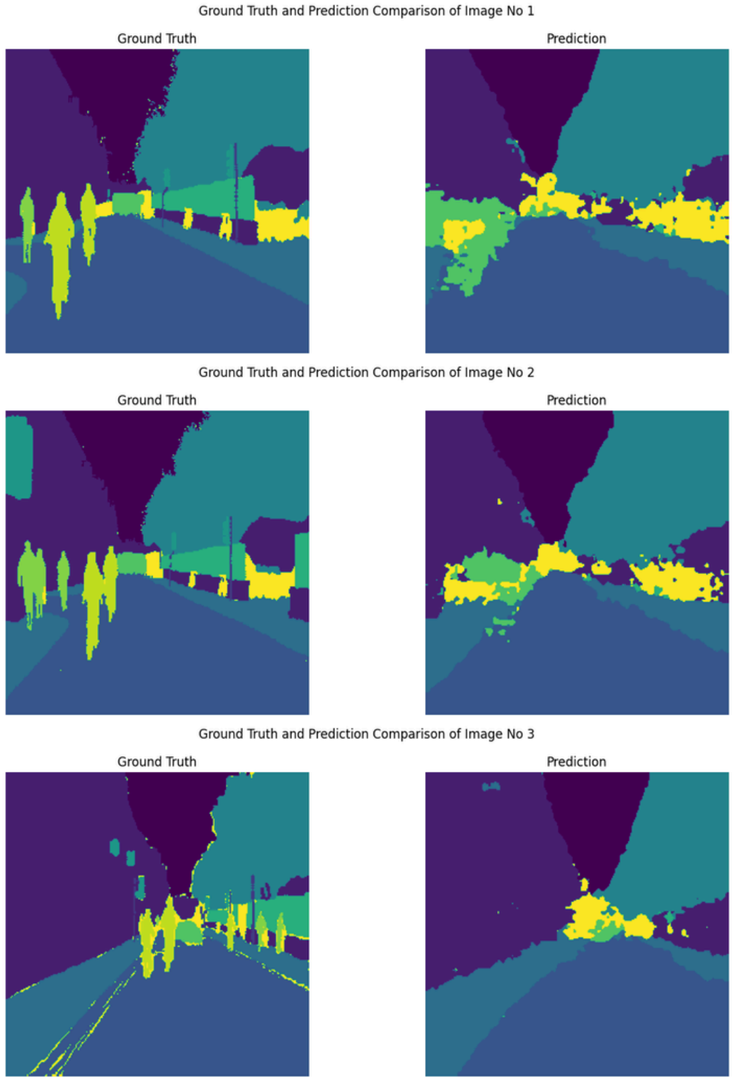
Model	Training Loss	Validation Loss	Test Loss
UNet	0.059195	0.061793	0.049954
FCN8s	0.059946	0.061847	0.049611

- The dice coefficient is obtained from 12 classes, where classes 2 to 12, in this case, have scores of 0. This is probably because the numbers of class 0 and class 1 are significantly higher than those of other classes.
- The score shown above is the mean of 12 classes.
- The dice coefficient score in train and validation datasets of FCN8s are slightly higher than those in the UNet, while in the test dataset, it is otherwise. Nevertheless, the score difference in the test dataset is slight. Thus, this indicates that FCN8s is better than UNet in this study case.

UNet



FCN8s



Visualization

If we compare both visualization, we can see that the UNet has better visualization since the objects are formed well. Meanwhile the object visualization of FCN8s is remains abstract and cannot easily understood.

Future Improvement

The model needs improvements to obtain the best performance based on the dice loss, dice coefficient, and visualization. We need to decrease the loss to less than 0.1 and increase the dice coefficient to above 0.8. The visualization also needs to be enhanced to yield better object segmentation. The future improvement for these models can be:

- Increase the number of images,
- Try more transformation functions,
- Apply patching,
- Experiment with other models.

Real-world Applications

The object segmentation can be applied to real-world situations such as:

1. Autonomous vehicles for detecting lanes, recognizing road signs, and avoiding obstacles.
2. Medical imaging includes tumour detection, cell counting, and organ segmentation.
3. Agriculture, for example, involves segmenting crops from the soil and weeds in aerial or satellite images to monitor health and growth stages and detect diseases.
4. Segmenting and counting people in public places for crowd control and behaviour analysis.
5. Segmenting defects in products for automated inspection and quality assurance.

Conclusion

In conclusion, the FCN8s perform better than UNet based on the loss and dice coefficient values. However, UNet's visualization is better than that of FCN8s. This indicates that FCN8s are better at distinguishing objects but worse at creating masks. If we look at the loss and dice coefficient value again, we will see that the difference between both models is slight, illustrating that UNet is pretty good. Therefore, if our goal focuses on distinguishing the objects, then FCN8s is the best to implement. Still, if our goal focuses on masking the object regardless of what the object is, then UNet is the best to implement.

Thank You !