Exercise 3 – Report

Implementation and training of FCN32 for road segmentation

Eirik Fagerhaug and Novian Habibie

## Code

We improve our custom dataloader class (dataloader.py) utility functions for data augmentation(utility.py) from exercise 2, which now its transformation process become faster.

For network architecture, we use FCN32 implemented in PyTorch. We modified network from <https://github.com/wkentaro/pytorch-fcn> and make it smaller to fit the requirements :

1. Input image with smaller dimension 128px
2. Reduce the amount of kernel into half
3. Weight initialization using Xavier uniform

The result of modified architecture described below:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **width** | **height** | **Kernel-size** | **padding** | **Kernel-amt** | **stride** | **New-width** | **New-height** | **Layer-type** |
| 256 | 128 | 3 | 1 | 32 | 1 | 256 | 128 | conv2d |
| 256 | 128 | 3 | 1 | 32 | 1 | 256 | 128 | conv2d |
| 256 | 128 | 2 | 0 | 0 | 2 | 128 | 64 | maxpool |
| 128 | 64 | 3 | 1 | 64 | 1 | 128 | 64 | conv2d |
| 128 | 64 | 3 | 1 | 64 | 1 | 128 | 64 | conv2d |
| 128 | 64 | 2 | 0 | 0 | 2 | 64 | 32 | maxpool |
| 64 | 32 | 3 | 1 | 128 | 1 | 64 | 32 | conv2d |
| 64 | 32 | 3 | 1 | 128 | 1 | 64 | 32 | conv2d |
| 64 | 32 | 3 | 1 | 128 | 1 | 64 | 32 | conv2d |
| 64 | 32 | 2 | 0 | 0 | 2 | 32 | 16 | maxpool |
| 32 | 16 | 3 | 1 | 256 | 1 | 32 | 16 | conv2d |
| 32 | 16 | 3 | 1 | 256 | 1 | 32 | 16 | conv2d |
| 32 | 16 | 3 | 1 | 256 | 1 | 32 | 16 | conv2d |
| 32 | 16 | 2 | 0 | 0 | 2 | 16 | 8 | maxpool |
| 16 | 8 | 3 | 1 | 256 | 1 | 16 | 8 | conv2d |
| 16 | 8 | 3 | 1 | 256 | 1 | 16 | 8 | conv2d |
| 16 | 8 | 3 | 1 | 256 | 1 | 16 | 8 | conv2d |
| 16 | 8 | 2 | 0 | 0 | 2 | 8 | 4 | maxpool |
| 8 | 4 | 7 | 3 | 2048 | 1 | 8 | 4 | conv2d |
| 8 | 4 | 1 | 0 | 2 | 1 | 8 | 4 | conv2d |
| 8 | 4 | 32 | 0 | 2 | 32 | 256 | 128 | convtrans2d |

Also we are using batch normalization for each layers.

## Results

Experiment run x times. Hyperparameter : learning rate = 0.01, momentum = 0.5, batch size=x)

IoU Accuracy :

Result image :

## Discussion

(Fill here)