Road_Segmentation

May 25, 2025

Semantic Segmentation Competition (30%)

For this competition, we will use a small autonomous driving dataset. The dataset contains 150 training images and 50 testing images.

We provide baseline code that includes the following features:

- Loading the dataset using PyTorch.
- Defining a simple convolutional neural network for semantic segmentation.
- How to use existing loss function for the model learning.
- Train the network on the training data.
- Test the trained network on the testing data.

The following changes could be considered:

1. Data augmentation
2. Change of advanced training parameters: Learning Rate, Optimizer, Batch-size, and Drop-out.
3. Architectural changes: Batch Normalization, Residual layers, etc.
4. Use of a new loss function.

Your code should be modified from the provided baseline. A pdf report of a maximum of two pages is required to explain the changes you made from the baseline, why you chose those changes, and the improvements they achieved.

Marking Rules:

We will mark the competition based on the final test accuracy on testing images and your report.

Final mark (out of 50) = accuracy mark + efficiency mark + report mark

Accuracy Mark 10:

We will rank all the submission results based on their test accuracy. Zero improvement over the baseline yields 0 marks. Maximum improvement over the baseline will yield 10 marks. There will be a sliding scale applied in between.

Efficiency Mark 10:

Efficiency considers not only the accuracy, but the computational cost of running the model (flops: https://en.wikipedia.org/wiki/FLOPS). Efficiency for our purposes is defined to be the ratio of accuracy (in %) to Gflops. Please report the computational cost for your final model and include the efficiency calculation in your report. Maximum improvement over the baseline will yield 10 marks. Zero improvement over the baseline yields zero marks, with a sliding scale in between.

Report mark 30:

Your report should comprise:

1. An introduction showing your understanding of the task and of the baseline model: [10 marks]
2. A description of how you have modified aspects of the system to improve performance. [10 marks]

   A recommended way to present a summary of this is via an "ablation study" table, eg:

Method1	Method2	Method3	Accuracy
N	N	N	60%
Y	N	N	65%
Y	Y	N	77%
Y	Y	Y	82%

3. Explanation of the methods for reducing the computational cost and/or improve the trade-off between accuracy and cost: [5 marks]
4. Limitations/Conclusions: [5 marks]

1 1. Download Data & Set Configurations

1.1 Download the Dataset

Dowanload & Unzip the Dataset.

1.2 Set Configurations

```
Device: cpu
```

1.3 Edit Configurations to Tune Model Performance

These settings should be altered as part of the model's analysis.

```
WARNING! The device is CPU NOT GPU! Please avoid using CPU for training
```

2 Define a Dataloader to Load Data

Add in better more systematic data transformations here. Data augmentation can be used (for example flip, resize)

```
load info for 150 images
```

3 Define a Convolutional Neural Network

A New Network Should be Used.

4 Define a Loss Function & Optimiser

5 The Function Used to Compare the Precision

DO NOT MODIFY THIS CODE!

6 Define Functions to Get & Save Predictions

Multi-scale testing can be used here to reduce the number of training cycles needed to evaluate model parameters' effectiveness.

7 Train the Network

Training on the original base model will take ~1 hour to complete. It is possible to define the false case in order to save on training time. Remember to download the results before closing the notebook.

```
epoch 0 iter 0 loss=2.9444425106048584
epoch 0 iter 1 loss=2.9443254470825195
epoch 0 iter 2 loss=2.9440338611602783
epoch 0 iter 3 loss=2.9425783157348633
epoch 0 iter 4 loss=2.938183546066284
epoch 0 iter 5 loss=2.926652431488037
epoch 0 iter 6 loss=2.8935959339141846
epoch 0 iter 7 loss=2.8362491130828857
epoch 0 iter 8 loss=2.7911245822906494
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[21], line 14
     12 loss = criterion(pred, labels.long().to(device)) # Calculate the Loss
     13 train_loss.append(loss.detach().cpu().numpy().item())
---> 14 loss.backward()
     15 optimizer.step()
     16 print('epoch {} iter {} loss={}'.format(epoch, iter, loss.data.cpu().
 ↪numpy()))

File ~/projects/2025/COMPUTER-VISION/.venv/lib/python3.12/site-packages/torch/
 ↪_tensor.py:648, in Tensor.backward(self, gradient, retain_graph, create_graph, ␣
 ↪inputs)
    638 if has_torch_function_unary(self):
    639     return handle_torch_function(
    640         Tensor.backward,
    641         (self,),
   (…)    646         inputs=inputs,
    647     )
--> 648 torch.autograd.backward(
    649     self, gradient, retain_graph, create_graph, inputs=inputs
```

```
    650 )

File ~/projects/2025/COMPUTER-VISION/.venv/lib/python3.12/site-packages/torch/
↪autograd/__init__.py:353, in backward(tensors, grad_tensors, retain_graph,␣
↪create_graph, grad_variables, inputs)
    348     retain_graph = create_graph
    350 # The reason we repeat the same comment below is that
    351 # some Python versions print out the first line of a multi-line functio
    352 # calls in the traceback and some print out the last line
--> 353 _engine_run_backward(
    354     tensors,
    355     grad_tensors_,
    356     retain_graph,
    357     create_graph,
    358     inputs,
    359     allow_unreachable=True,
    360     accumulate_grad=True,
    361 )

File ~/projects/2025/COMPUTER-VISION/.venv/lib/python3.12/site-packages/torch/
↪autograd/graph.py:824, in _engine_run_backward(t_outputs, *args, **kwargs)
    822     unregister_hooks = _register_logging_hooks_on_whole_graph(t_outputs)
    823 try:
--> 824     return␣
↪Variable._execution_engine.run_backward(  # Calls into the C++ engine to run the backward p
    825         t_outputs, *args, **kwargs
    826     )  # Calls into the C++ engine to run the backward pass
    827 finally:
    828     if attach_logging_hooks:

KeyboardInterrupt:
```


```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[29], line 5
      2 epochs = list(range(101, 180))
      4 # Plot IoU vs. Epoch
----> 5 plt.plot(epochs, IoU_list, label=     )
      7 # X-axis label
      8 plt.xlabel("Epoch")

File ~/projects/2025/COMPUTER-VISION/.venv/lib/python3.12/site-packages/
↪matplotlib/pyplot.py:3838, in plot(scalex, scaley, data, *args, **kwargs)
   3830 @_copy_docstring_and_deprecators(Axes.plot)
   3831 def plot(
   3832     *args: float | ArrayLike | str,
   (…)   3836     **kwargs,
   3837 ) -> list[Line2D]:
-> 3838     return gca().plot(
   3839         *args,
   3840         scalex=scalex,
   3841         scaley=scaley,
```

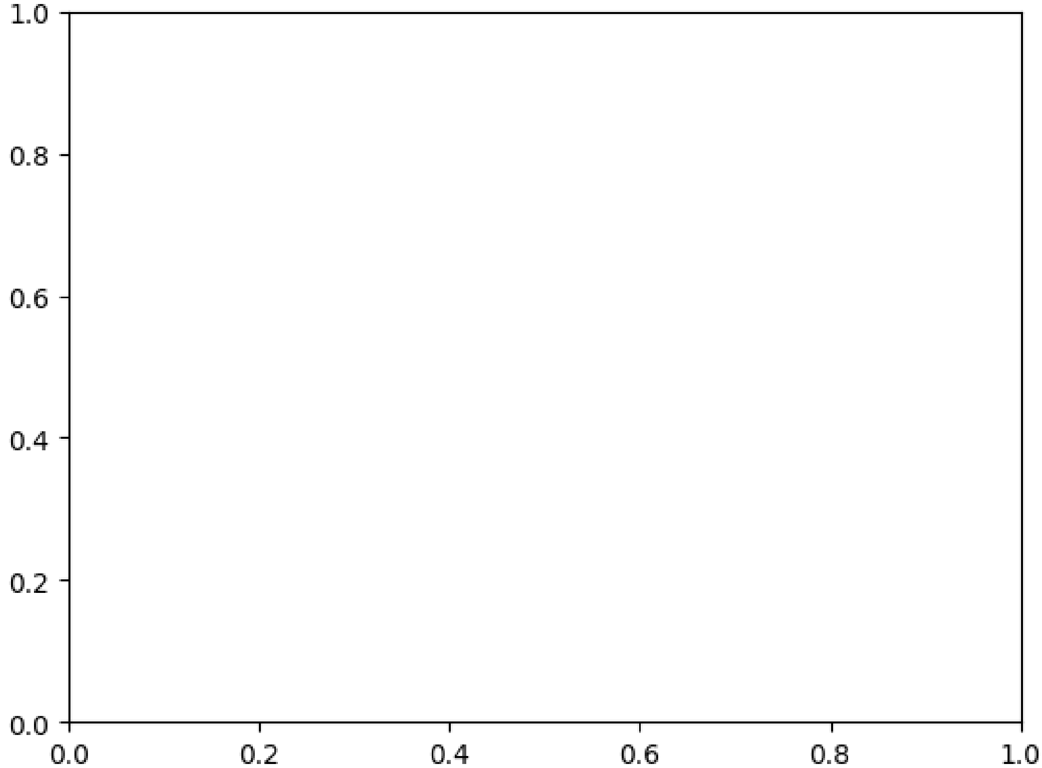
```
   3842             **({       : data} if data is not None else {}),
   3843             **kwargs,
   3844         )

File ~/projects/2025/COMPUTER-VISION/.venv/lib/python3.12/site-packages/
↪matplotlib/axes/_axes.py:1777, in Axes.plot(self, scalex, scaley, data, *args ␣
↪**kwargs)
   1534 """
   1535 Plot y versus x as lines and/or markers.
   1536
   (…)   1774 (``'green'``) or hex strings (``'#008000'``).
   1775 """
   1776 kwargs = cbook.normalize_kwargs(kwargs, mlines.Line2D)
-> 1777 lines = [*self._get_lines(self, *args, data=data, **kwargs)]
   1778 for line in lines:
   1779     self.add_line(line)

File ~/projects/2025/COMPUTER-VISION/.venv/lib/python3.12/site-packages/
↪matplotlib/axes/_base.py:297, in _process_plot_var_args.__call__(self, axes,␣
↪data, return_kwargs, *args, **kwargs)
    295     this += args[0],
    296     args = args[1:]
--> 297 yield from self._plot_args(
    298     axes, this, kwargs, ambiguous_fmt_datakey=ambiguous_fmt_datakey,
    299     return_kwargs=return_kwargs
    300 )

File ~/projects/2025/COMPUTER-VISION/.venv/lib/python3.12/site-packages/
↪matplotlib/axes/_base.py:494, in _process_plot_var_args._plot_args(self, axes ␣
↪tup, kwargs, return_kwargs, ambiguous_fmt_datakey)
    491     axes.yaxis.update_units(y)
    493 if x.shape[0] != y.shape[0]:
--> 494     raise ValueError(f"x and y must have same first dimension, but "
    495                      f"have shapes {x.shape} and {y.shape}")
    496 if x.ndim > 2 or y.ndim > 2:
    497     raise ValueError(f"x and y can be no greater than 2D, but have "
    498                      f"shapes {x.shape} and {y.shape}")

ValueError: x and y must have same first dimension, but have shapes (79,) and␣
↪(0,)
```



A prediction example by using the baseline:

8 FLOPs

In deep learning, FLOPs *(Floating Point Operations)* quantify the total number of arithmetic operations—such as additions, multiplications, and divisions—that a model performs during a single forward pass *(i.e. when making a prediction)*. This metric serves as an indicator of a model's computational complexity. When discussing large-scale models, FLOPs are often expressed in GFLOPs *(Giga Floating Point Operations)*, where 1 GFLOP equals one billion operations. This unit helps in comparing the computational demands of different models.

DO NOT MODIFT THIS CODE!

```
Unsupported operator aten::max_pool2d encountered 4 time(s)
Unsupported operator aten::feature_dropout encountered 2 time(s)
```

```
FLOPs: 66.97 GFLOPs
```