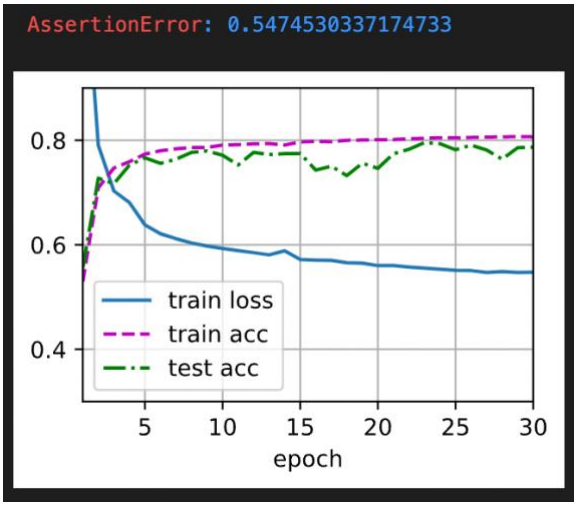
All generations of models tested all using NAdam optimizer and time epoch 30.

Highest test accuracy achieved: **0.8945**

1nd generation (Using 2 Linear layers for Backbone)

The first generation only used two linear layers (one MLP, half a block) for the backbone which resulted in a very low test accuracy, leading to an assertion error as the training loss value (0.5474530337174733) is still bigger than 0.5 and test accuracy could not be recorded.



Hyper-parameters:

- batch size = 256
- num_input = 16
- num_hidden = 128
- num_output = 10
- lr = 0.01
- wd = 0.001

Test Accuracy = ***could not record**

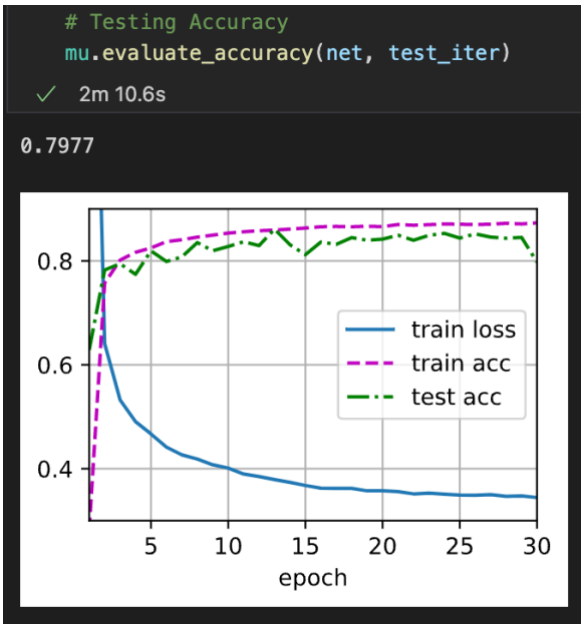
2nd generation (Using 4 Linear layers for Backbone)

For the 2nd generation, four linear layers instead of two are used for the backbone, with all the rest hyper-parameters remain unchanged. Sees dramatic increase in test accuracy and decrease in training loss. However, final output test accuracy does not seem to fit the curve.

Hyper-parameters:

- batch size = 256
- num_input = 16
- num_hidden = 128
- num_output = 10
- lr = 0.01
- wd = 0.001

Test Accuracy = **0.7977**



3rd generation

For the 3rd generation, I decreased my num_hidden to half of before (128), sees a increased test accuracy output with a less fluctuating curve. Very close to 85%.

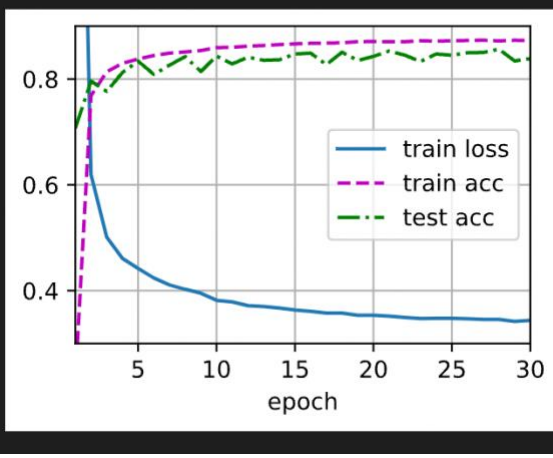
Hyper-parameters:

- batch size = 256
- num_input = 16
- num_hidden = 64
- num_output = 10
- lr = 0.01
- wd = 0.001

Test Accuracy = **0.8387**

```
# Testing Accuracy
mu.evaluate_accuracy(net, test_iter)
```
✓ 1m 39.8s

```
0.8387
```



4th generation

For the 4th generation, I matched my batch_size with my num_hidden, which sees test accuracy above 85%.

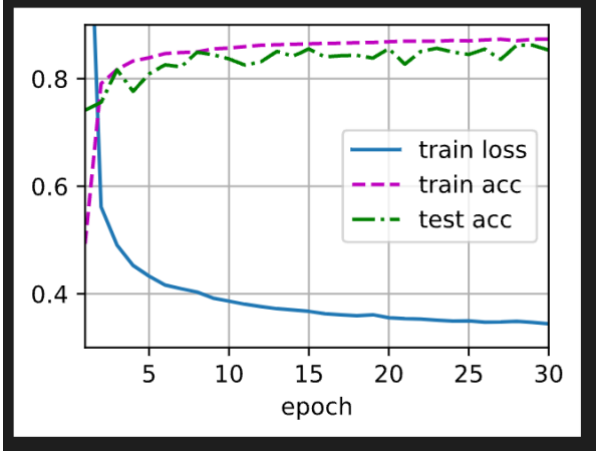
Hyper-parameters:

- batch size = 256
- num_input = 16
- num_hidden = 256
- num_output = 10
- lr = 0.01
- wd = 0.001

Test Accuracy = **0.8531**

```
# Testing Accuracy
mu.evaluate_accuracy(net, test_iter)
```
✓ 2m 40.9s

```
0.8531
```



Final Generation:

For the final generation, I made changes to the my_utils.py file, changing ylim[0.3, 0.9] to ylim[0.3, 1]. I also introduced batch normalisation. Which I used after my linear layer in stem, as well as after each linear layer in backbone except for the final linear layer as it is getting passed to the classifier. I have also decreased lr (learning rate) from 0.01 to 0.0009, wd (weight decay) from 0.001 to 0.00001. Changed both batch_size and num_hidden from 256 to 250. I was able to achieve a test accuracy of **0.8945**. Unfortunately, with further testing, I was not able to get any higher. For future testing, to increase the test accuracy further I could implement dropout function or a learning rate scheduler. I could also increase the number of blocks used by backbone.

Hyper-parameters:

- batch_size = 250
- num_inputs = 16
- num_hidden = 250
- num_outputs = 10
- lr = 0.0009
- wd = 0.00001

Test Accuracy = **0.8945**

```
# Testing Accuracy
mu.evaluate_accuracy(net, test_iter)
```

✓ 3m 43.4s

```
0.8945
```