DEEP LEARNING MC 1

Weiping Zhang

FS 2023
01. Jun. 2023

- Dataset: CIFAR 10
- Model: 2D CNN
- Deep learning framework: PyTorch
- Hyperparameter tuning structure:
  - architecture of 2D-CNN
    - Convolutional layer
      - Number of convolutional layers
      - Number of filters
      - Kernel size
      - Stride
      - Padding
    - Activation
    - Pooling
    - Fully connected layer
      - Number of fully connected layers
      - Number of neurons
  - Learning rate & batch size
  - Optimizer
    - SGD
    - Adam
  - Regularization
    - L1/L2
    - Dropout
  - Batch normalization

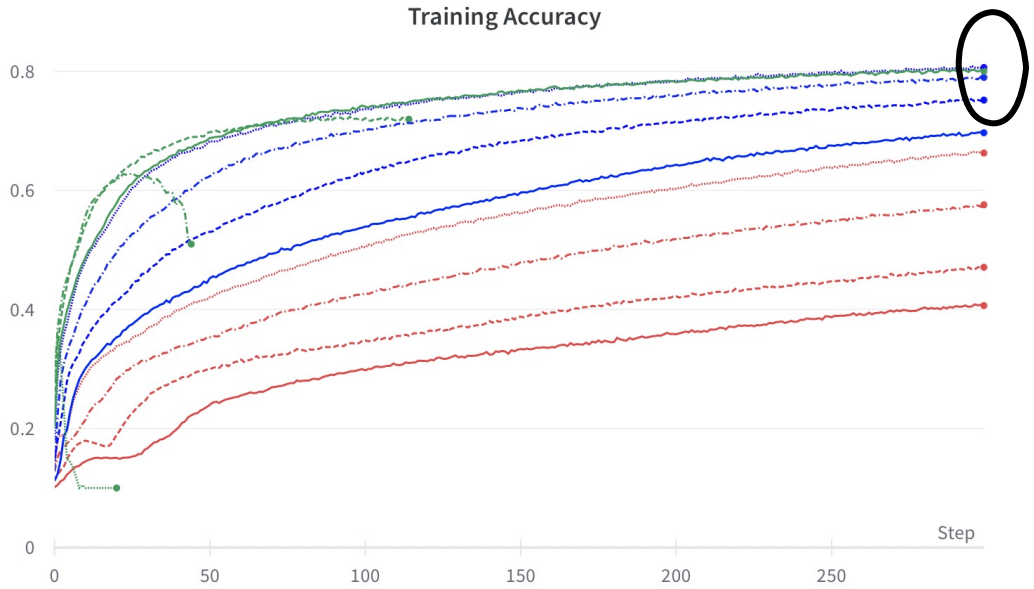
LEARNING STRUCTURE

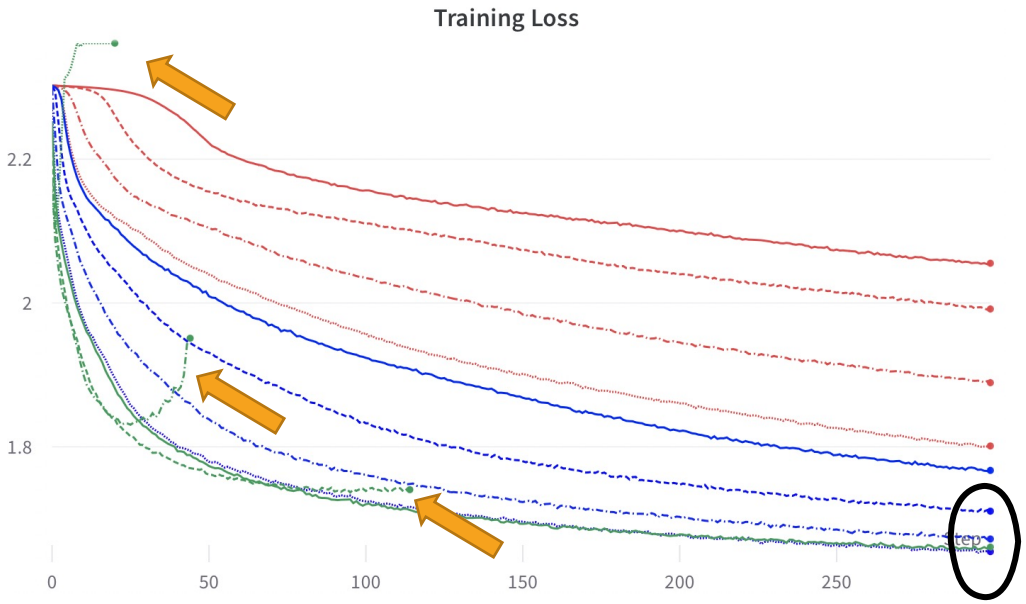
1. HOW MANY CONVOLUTIONAL LAYERS?

- 2 convolutional layers + 2 fully connected layers
- 0 convolutional layers + 3 fully connected layers
- 1 convolutional layer + 2 fully connected layers
- 3 convolutional layer + 2 fully connected layers
- 4 convolutional layer + 2 fully connected layers

2 CONVOLUTIONAL LAYER + 2 FULLY CONNECTED LAYERS

Optimal learning rate



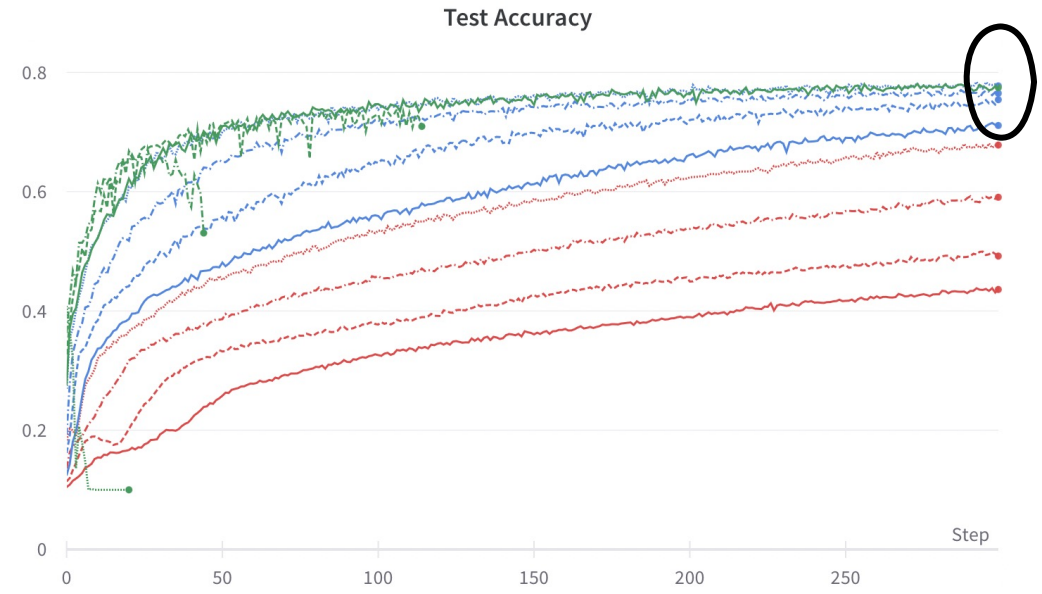


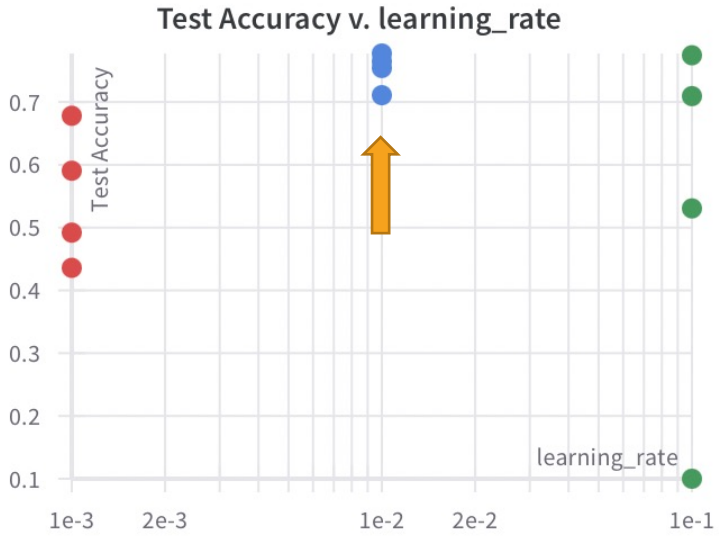
Learning rate:

0.1, 0.01, 0.001

Batch size:

- 16
- 32
- 64
- 128





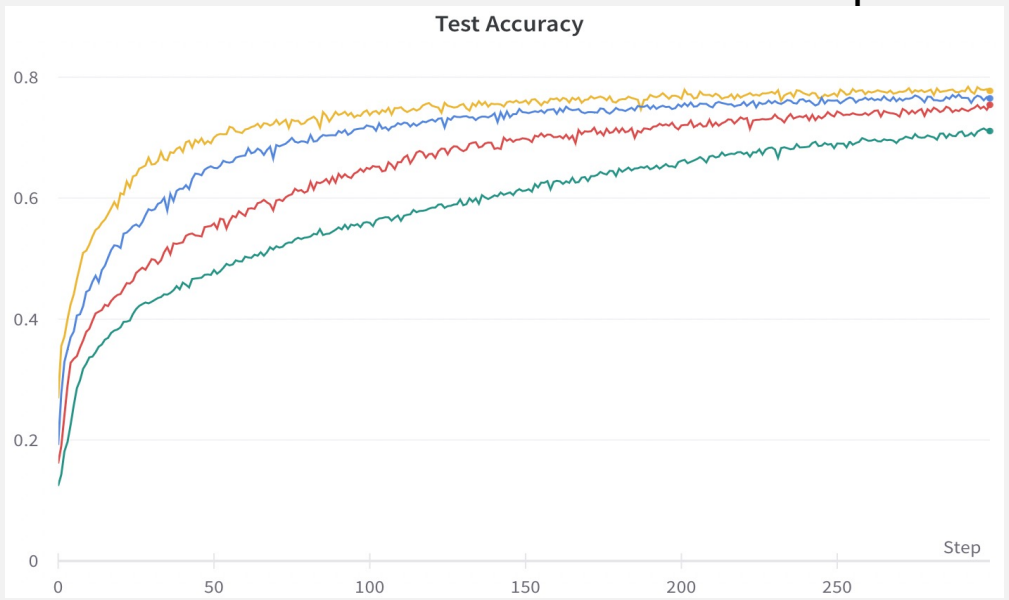
Analysis:

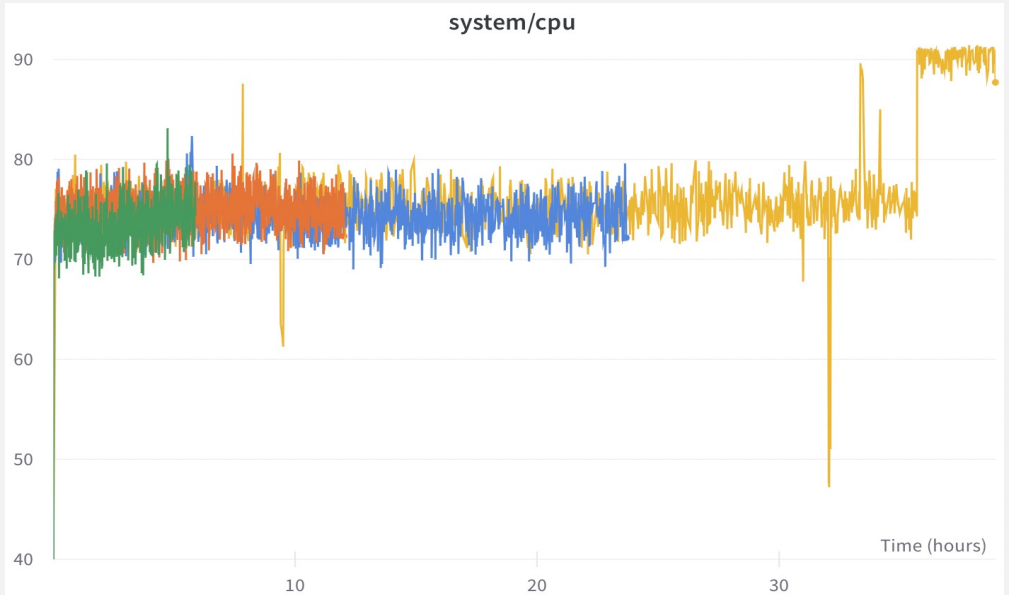
- Learning rate 0.1: three of the four models failed to converge (with 20-epochs early stop)
- Learning rate 0.001: too slow, models need more epochs to converge comparing to learning rate 0.01

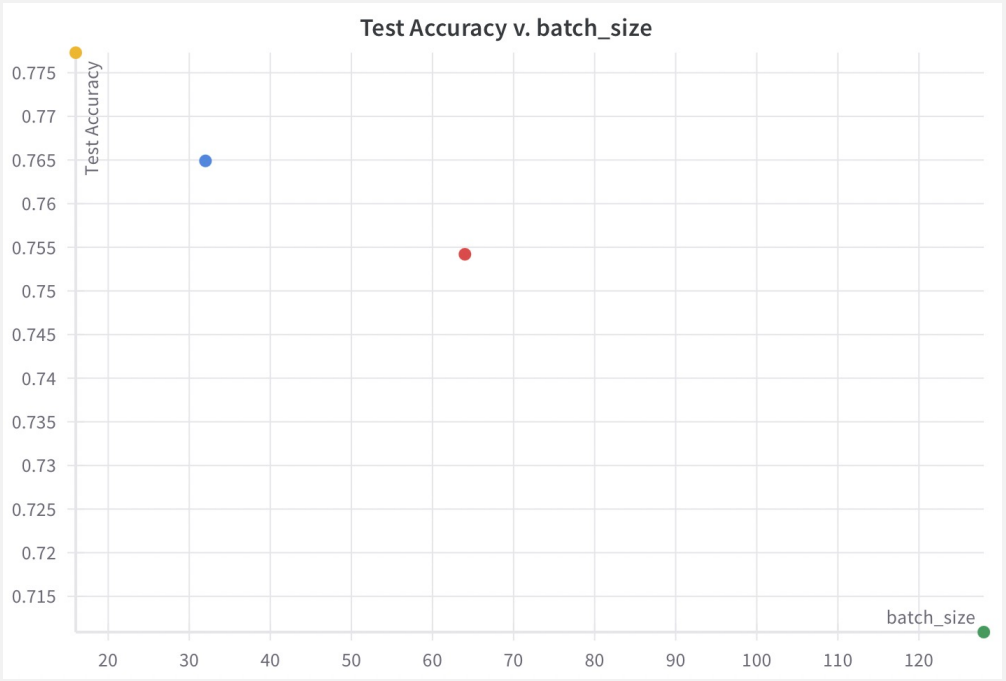
Optimal learning rate 0.01

2 CONVOLUTIONAL LAYER + 2 FULLY CONNECTED LAYERS

Optimal batch size







Learning rate: 0.01

Batch size:

- 16
- 32
- 64
- 128

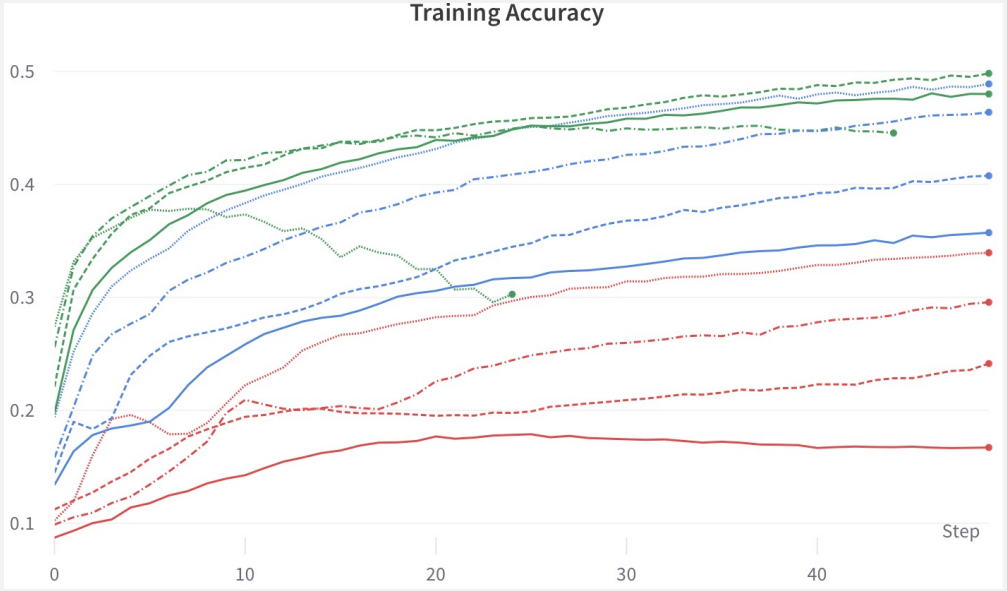
Analysis:

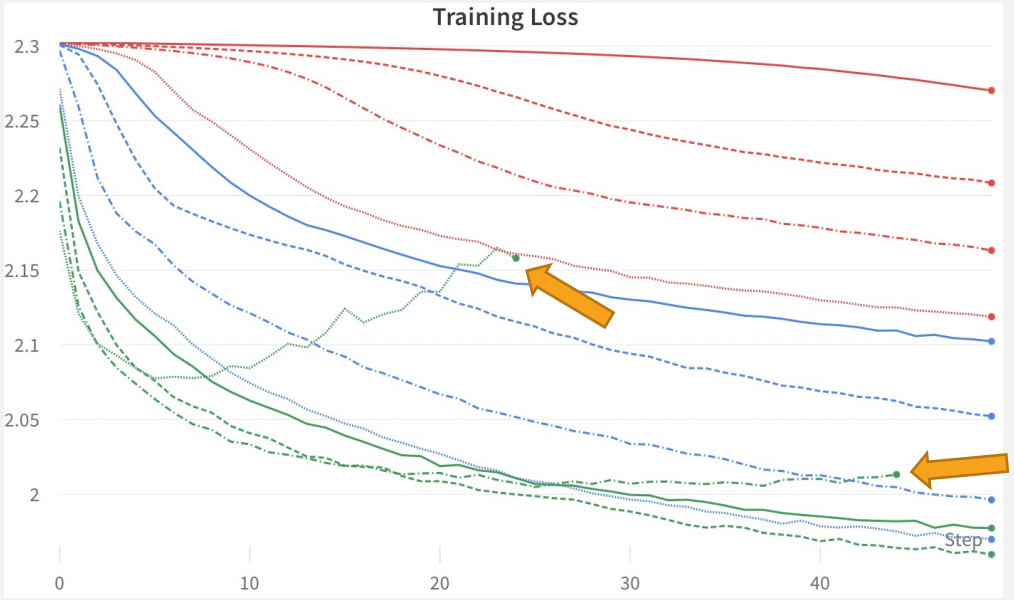
- Accuracy increased when batch size is decreasing
- But computing time also increased when batch size is decreasing
- Similar accuracy with batch size 16, 32, 64, but computing time difference is large

Optimal batch size 64

0 CONVOLUTIONAL LAYER + 3 FULLY CONNECTED LAYERS

Optimal learning rate



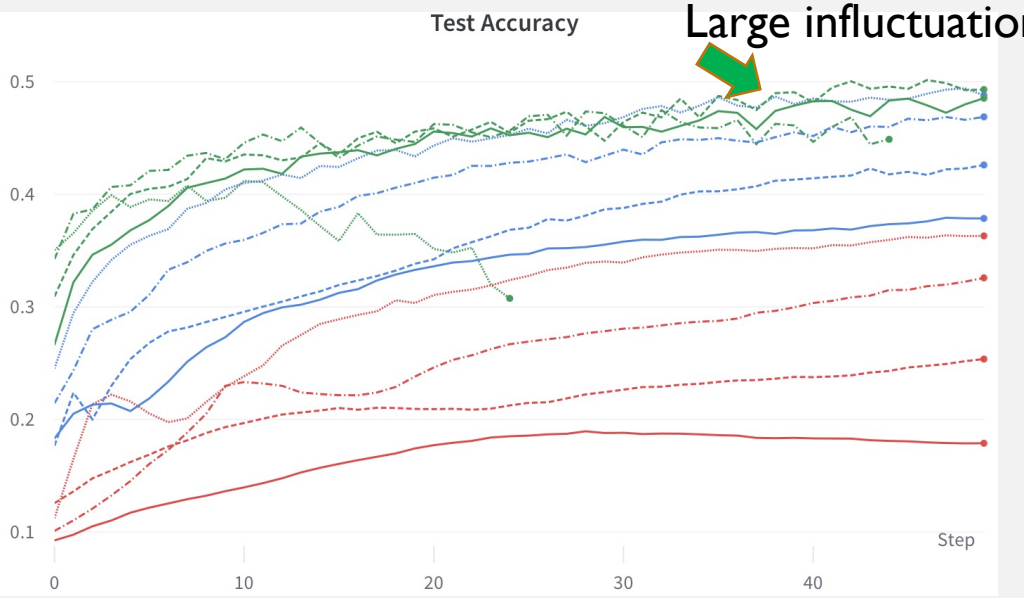


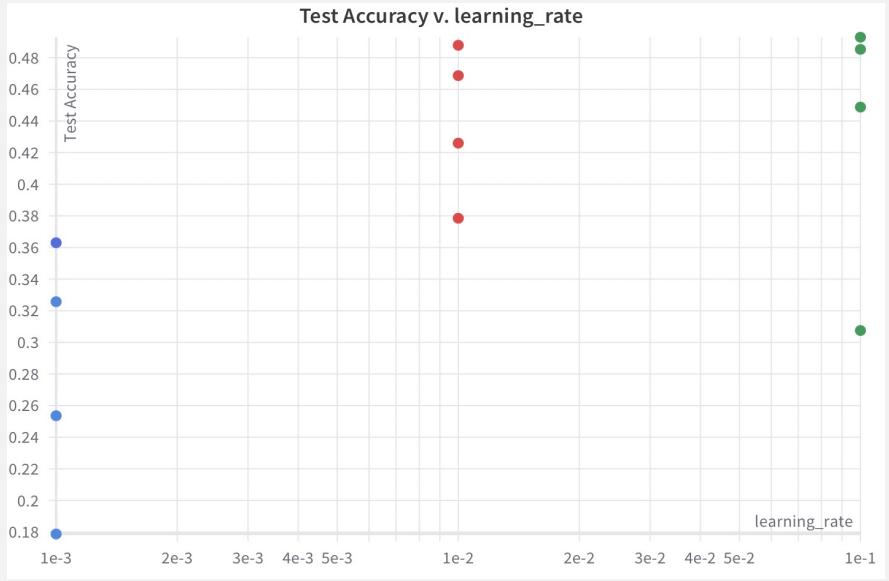
Learning rate:

0.1, 0.01, 0.001

Batch size:

- 16
- —·— 32
- – – 64
- —— 128





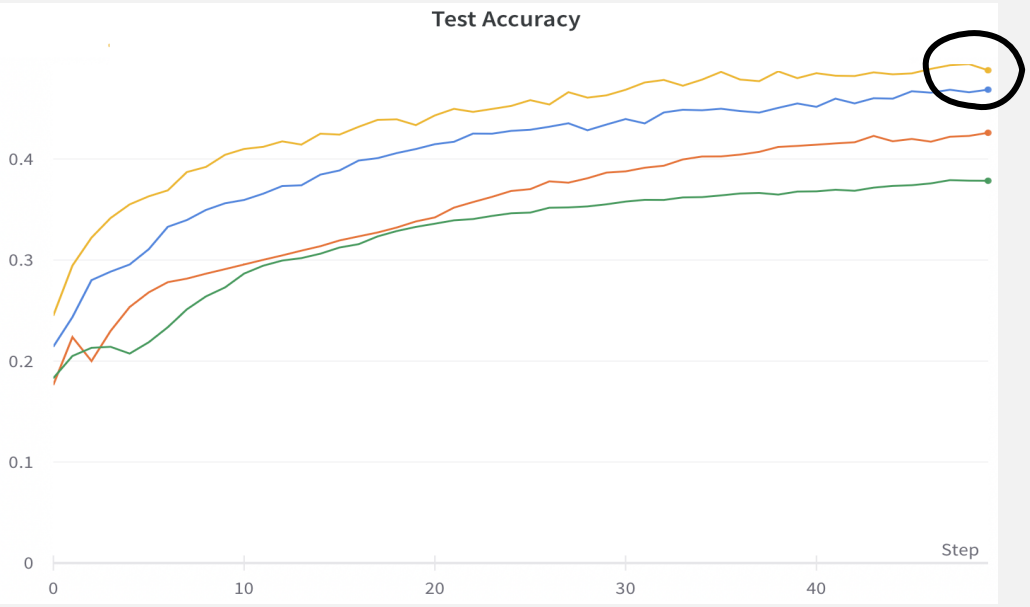
Analysis:

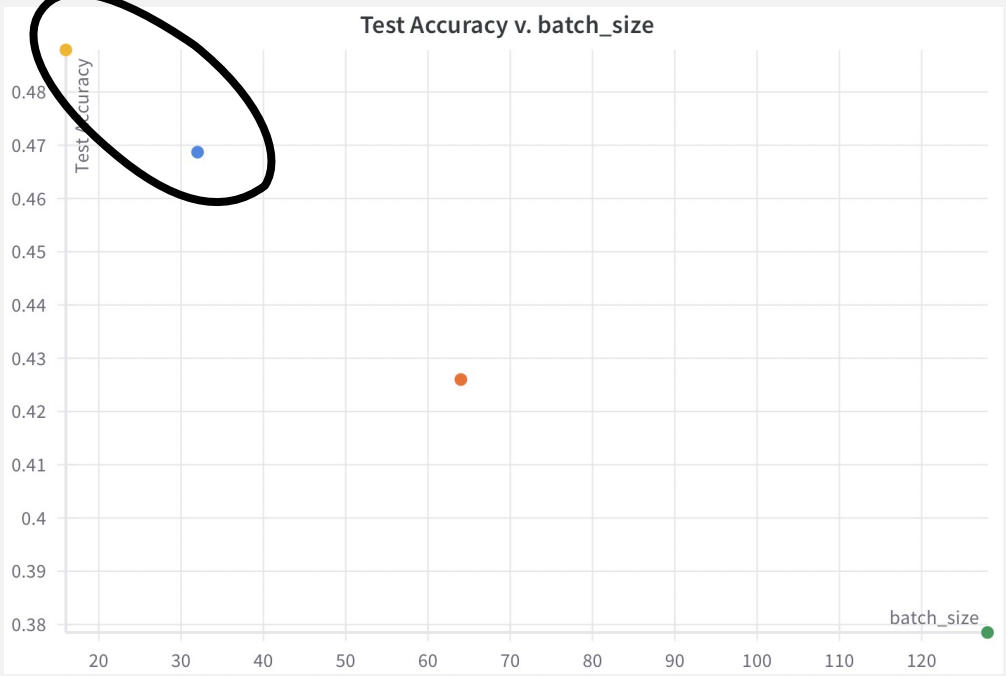
- Learning rate 0.1: large influctuation of test accuracy; two of the four models failed to converge (with 20-epochs early stop)
- Learning rate 0.001: too slow, models need more epochs to converge comparing to learning rate 0.01

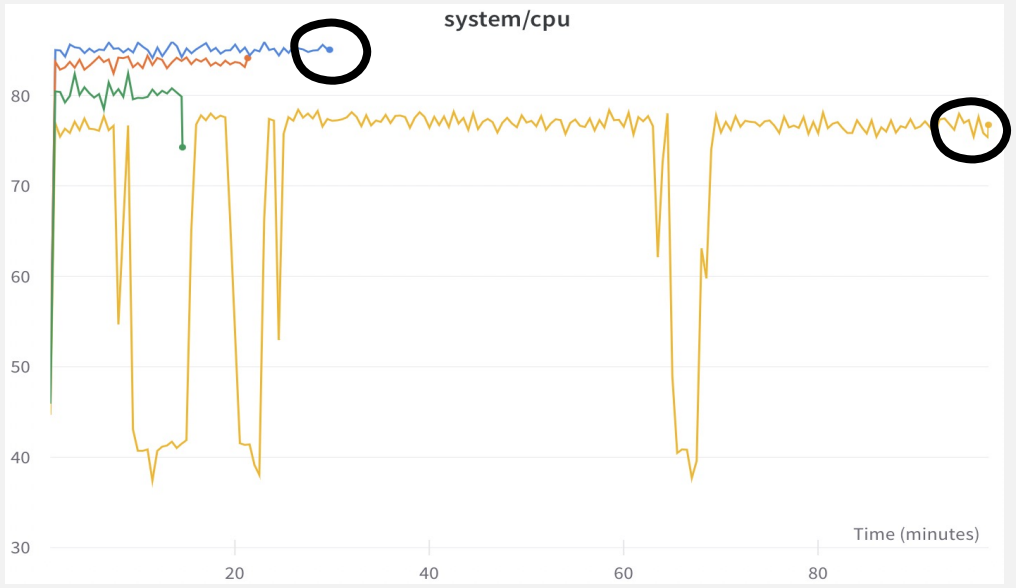
Optimal learning rate 0.01

0 CONVOLUTIONAL LAYER + 3 FULLY CONNECTED LAYERS

Optimal batch size







Learning rate: 0.01

Batch size:

- 16
- 32
- 64
- 128

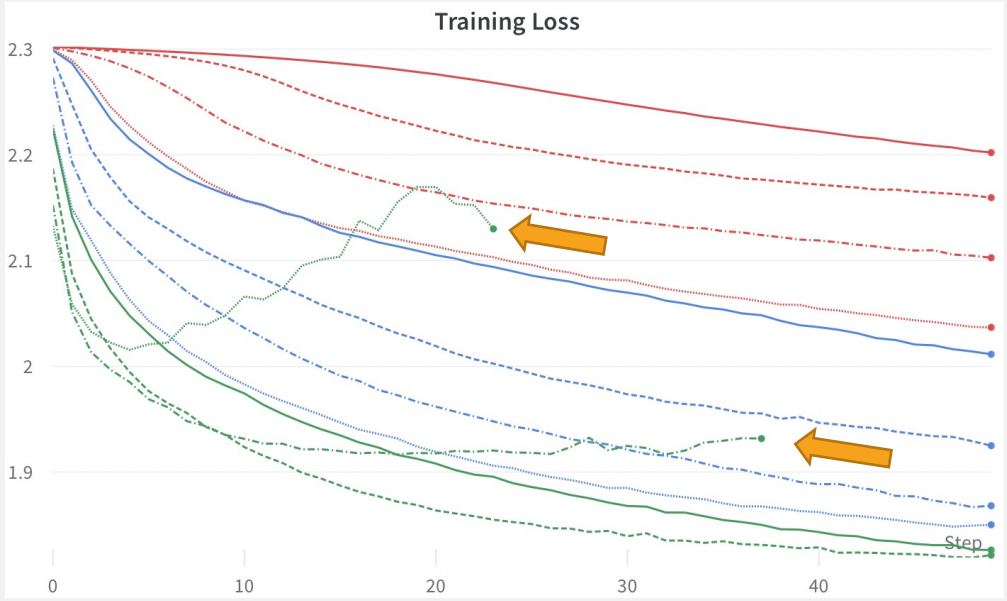
Analysis:

- Accuracy increased when batch size is smaller
- But computing time also increased when batch size is decreasing
- Similar accuracy with batch size 16, 32, but computing time difference is very large

Optimal batch size 32

1 CONVOLUTIONAL LAYER + 2 FULLY CONNECTED LAYERS

Optimal learning rate

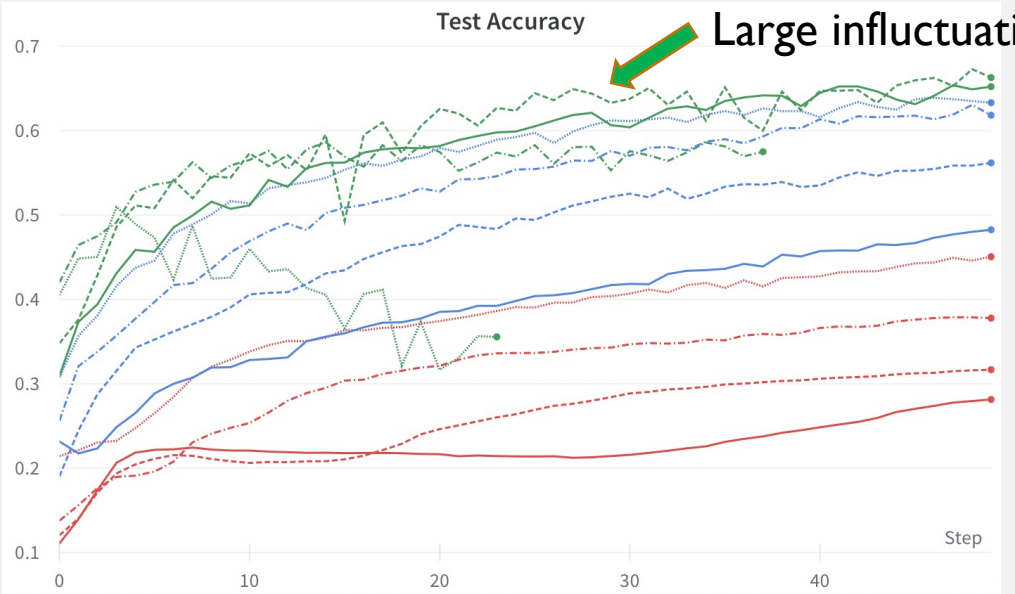


Learning rate:

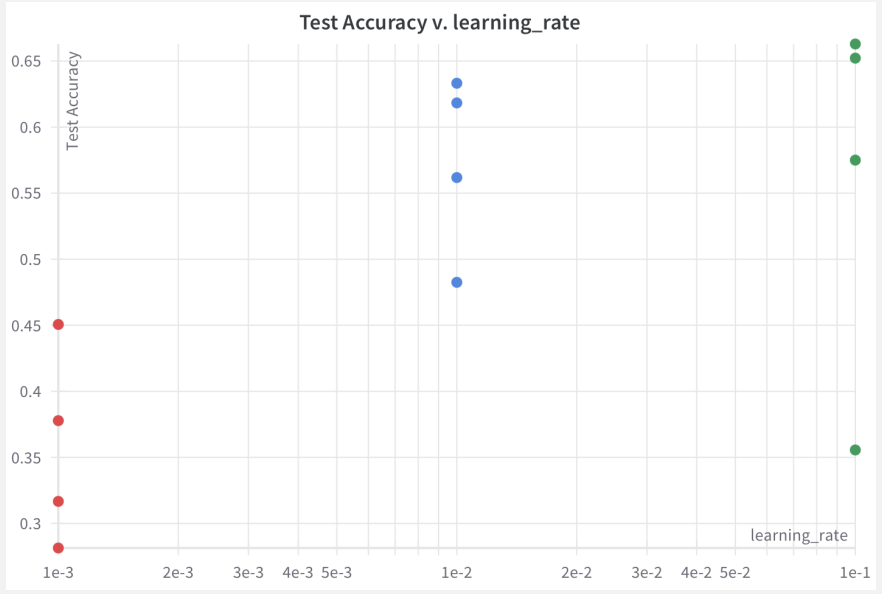
0.1, 0.01, 0.001

Batch size:

- 16
- 32
- 64
- 128



Large influctuation



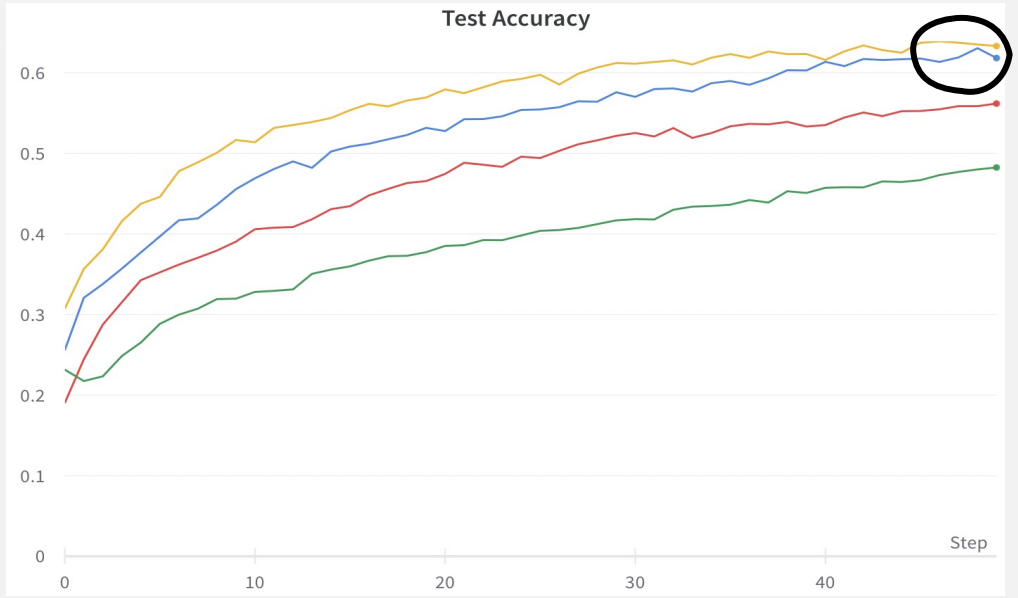
Analysis:

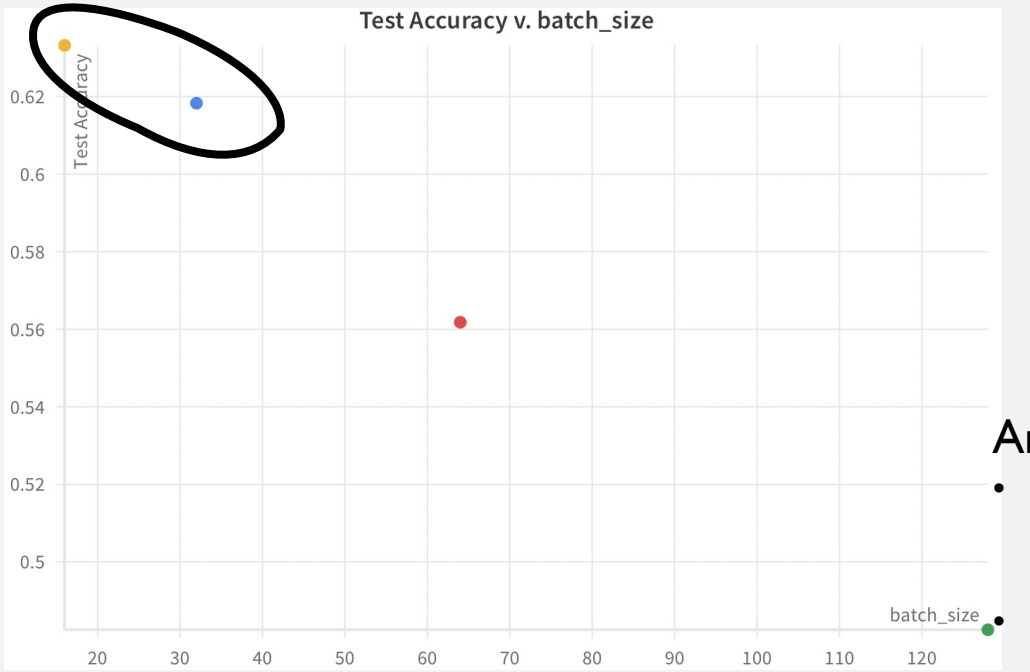
- Learning rate 0.1: large influctuation in test accuracy; two models failed to converge (with 20-epochs early stop)
- Learning rate 0.001: too slow, models will need more epochs to converge comparing to learning rate 0.01

Optimal learning rate 0.01

1 CONVOLUTIONAL LAYER + 2 FULLY CONNECTED LAYERS

Optimal batch size

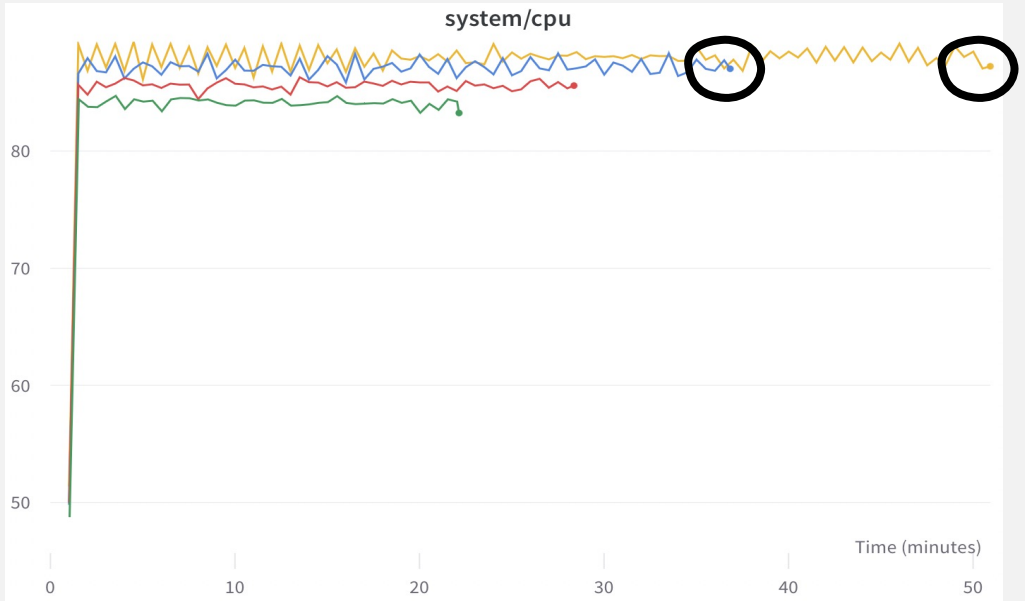




Learning rate: 0.01

Batch size:

- 16
- 32
- 64
- 128



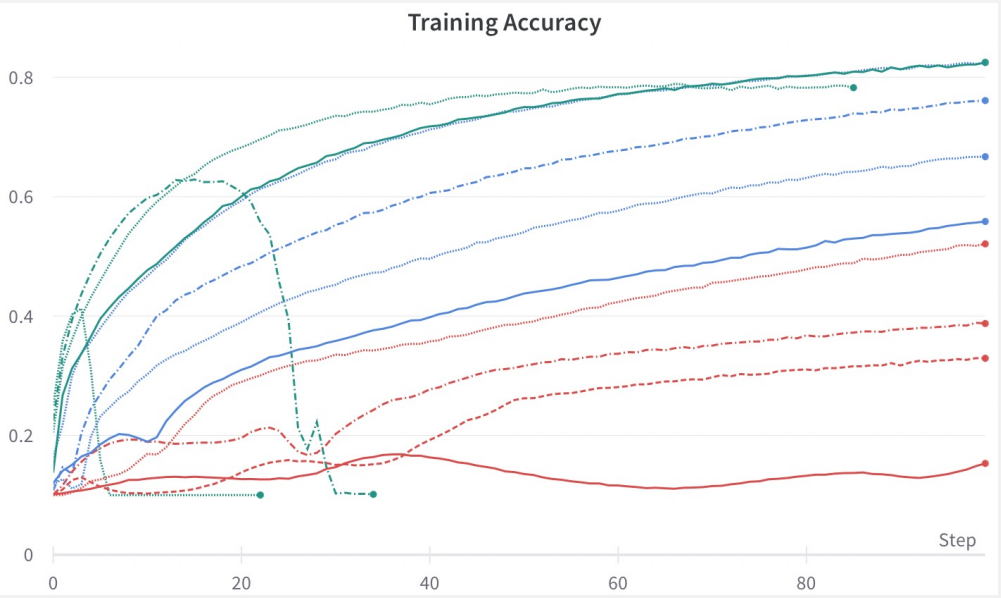
Analysis:

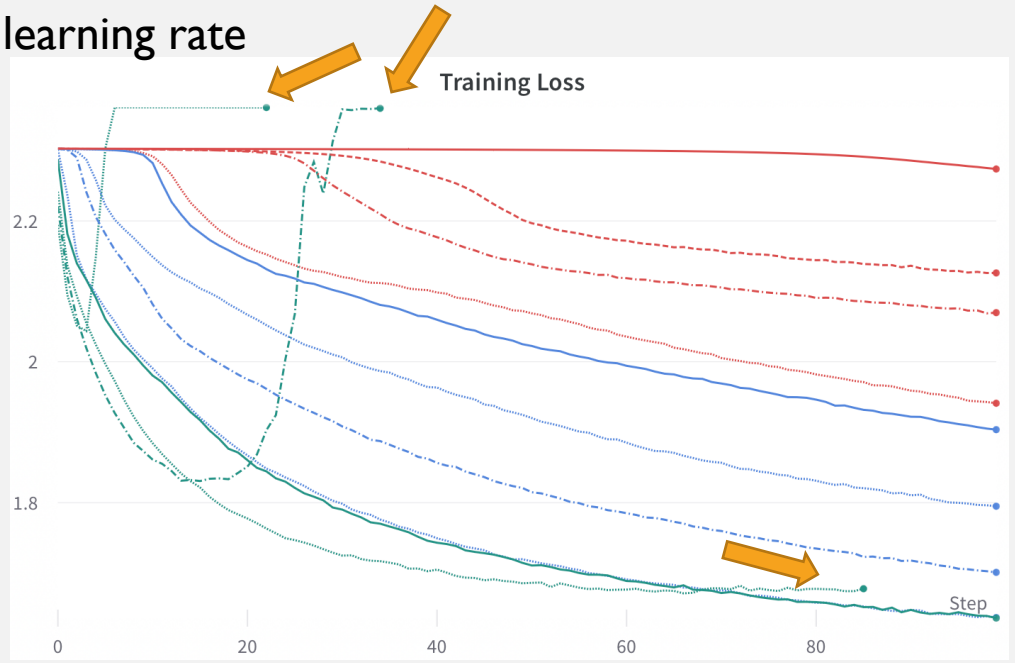
- Accuracy increased when batch size is smaller
- But computing time also increased when batch size is decreasing
- Very similar Accuracy between models with batch size 16 and 32, but computing time difference is not that small

Optimal batch size 32

3 CONVOLUTIONAL LAYERS + 2 FULLY CONNECTED LAYERS

Optimal learning rate



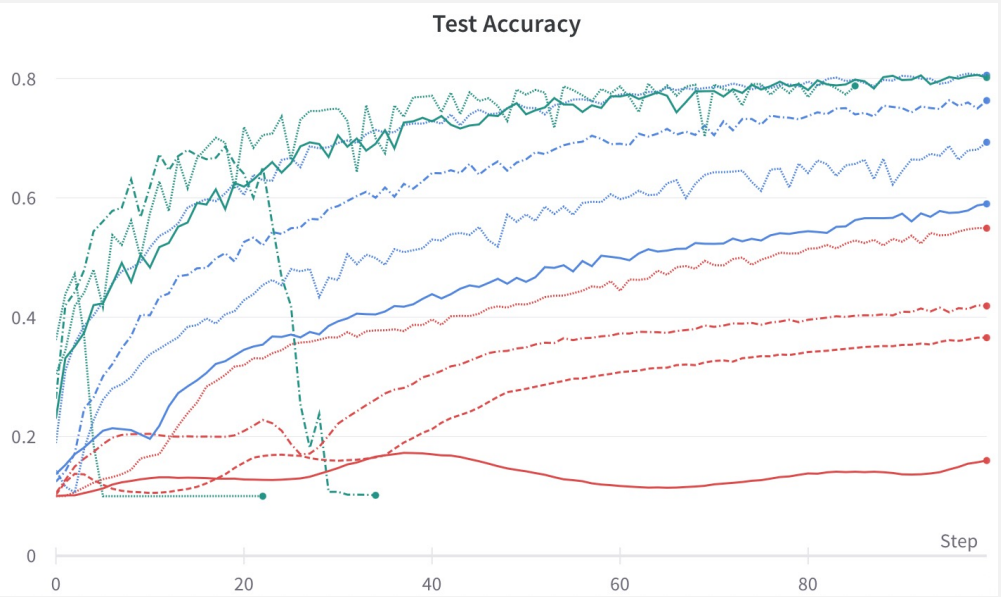


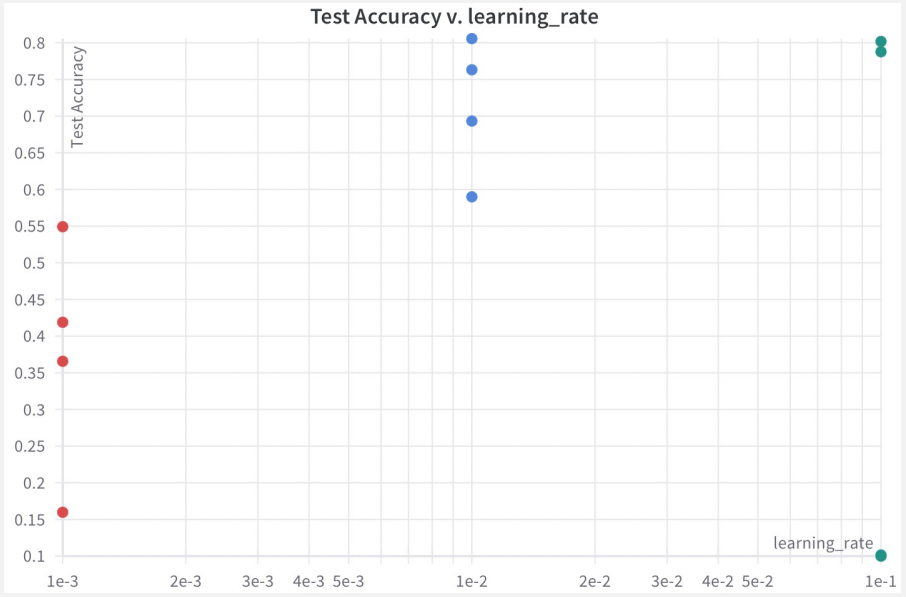
Learning rate:

0.1, 0.01, 0.001

Batch size:

- 16
- —·— 32
- – – 64
- —— 128





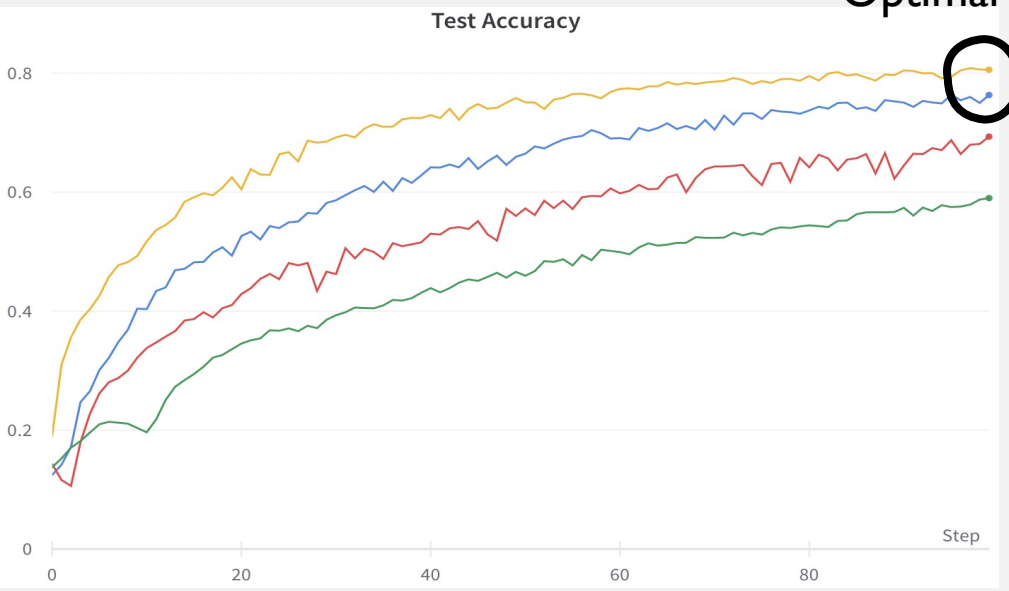
Analysis:

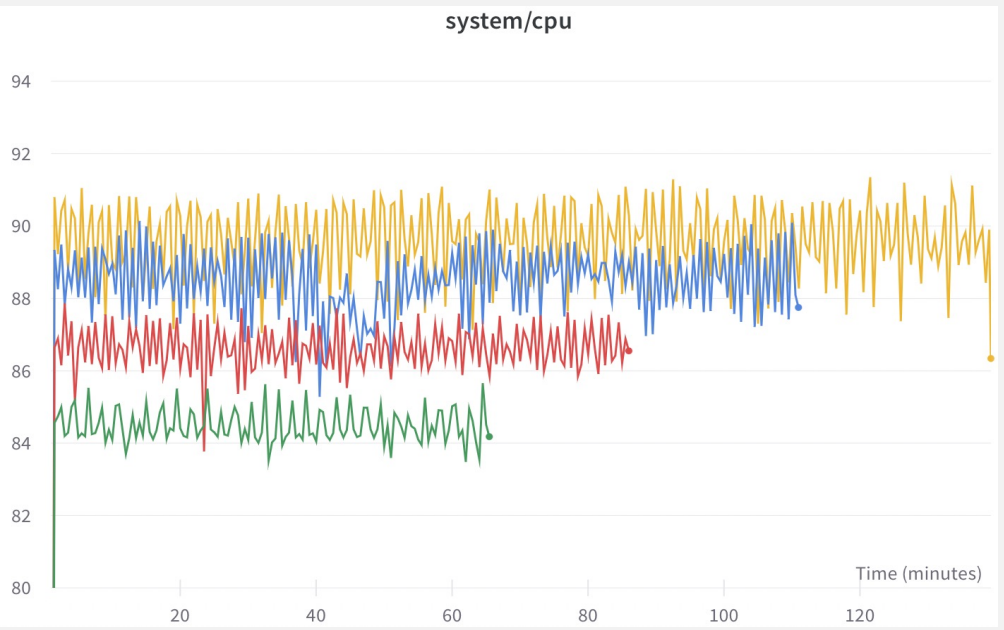
- Learning rate 0.1: large influctuation in test accuracy, three of the four models failed to converge (with 20-epochs early stop)
- Learning rate 0.001: too slow, models need more epochs to converge comparing to learning rate 0.01

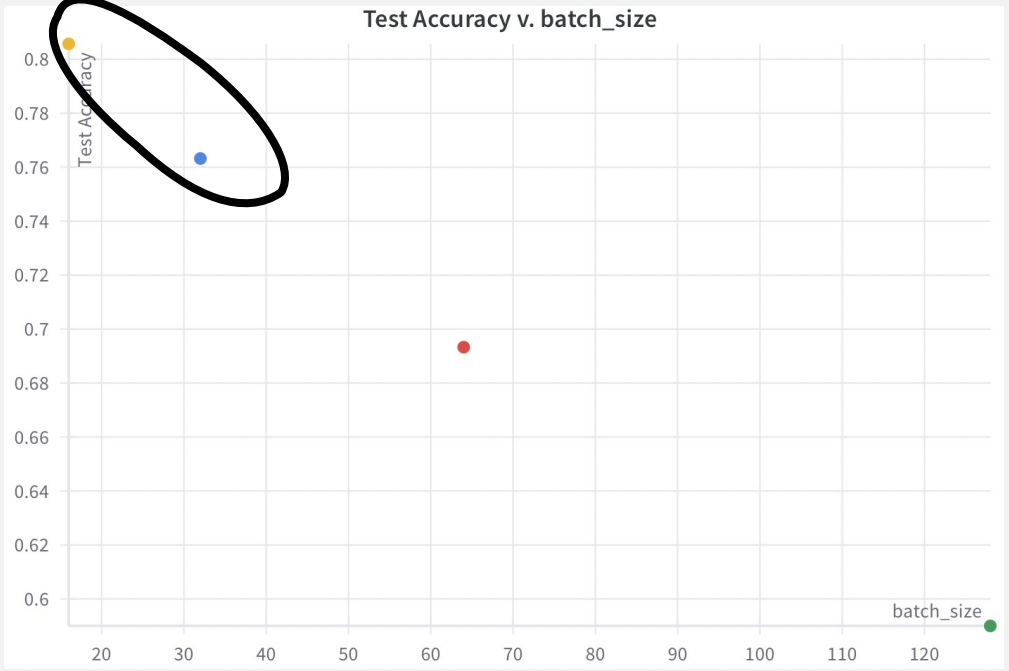
Optimal learning rate 0.01

3 CONVOLUTIONAL LAYERS + 2 FULLY CONNECTED LAYERS

Optimal batch size







Learning rate: 0.01

Batch size:

- 16
- 32
- 64
- 128

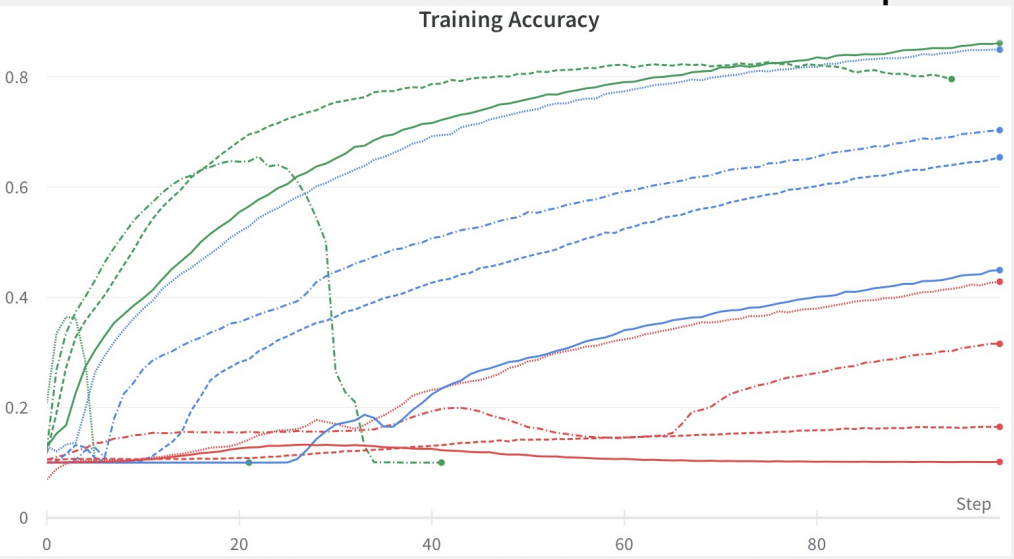
Analysis:

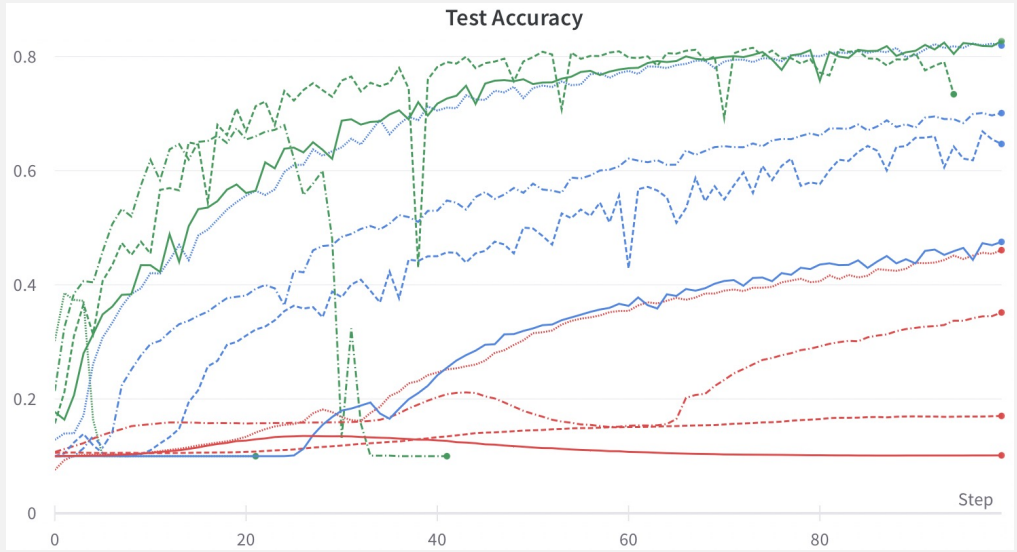
- Accuracy increased when batch size is smaller
- But computing time also increased when batch size is decreasing

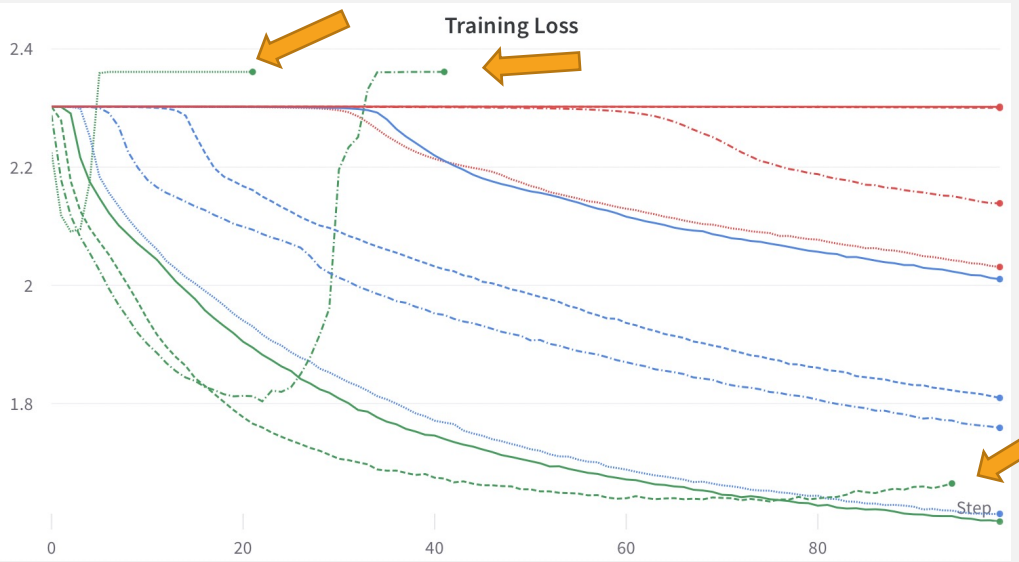
Optimal batch size 32

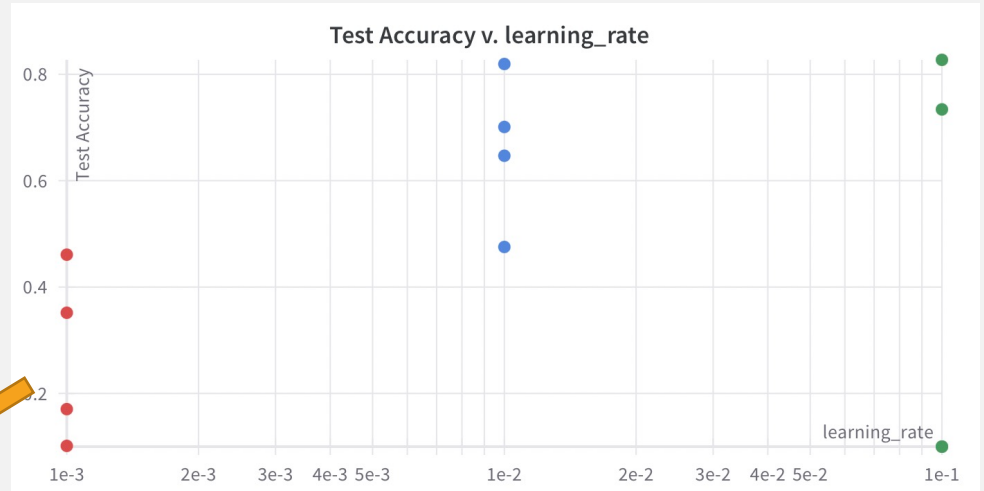
4 CONVOLUTIONAL LAYERS + 2 FULLY CONNECTED LAYERS

Optimal learning rate









Learning rate:

0.1, 0.01, 0.001

Batch size:

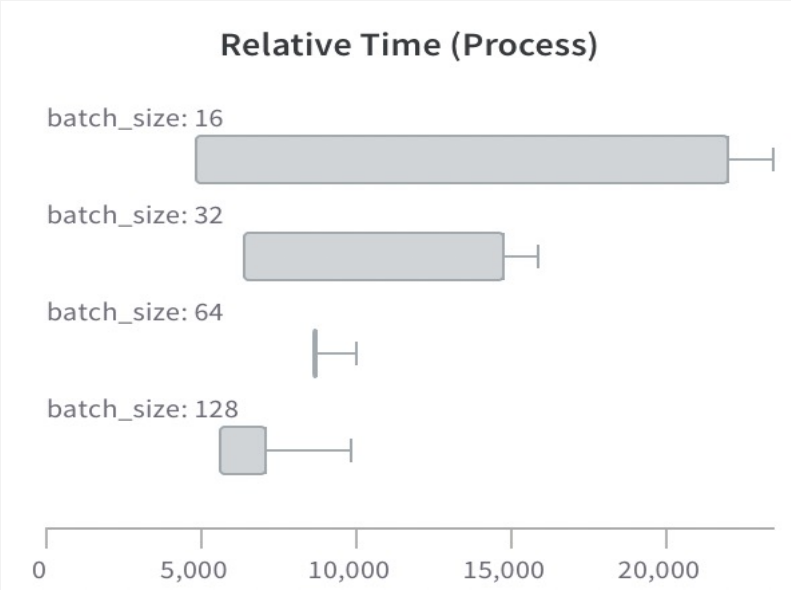
- 16
- —·— 32
- – – 64
- —— 128

Analysis:

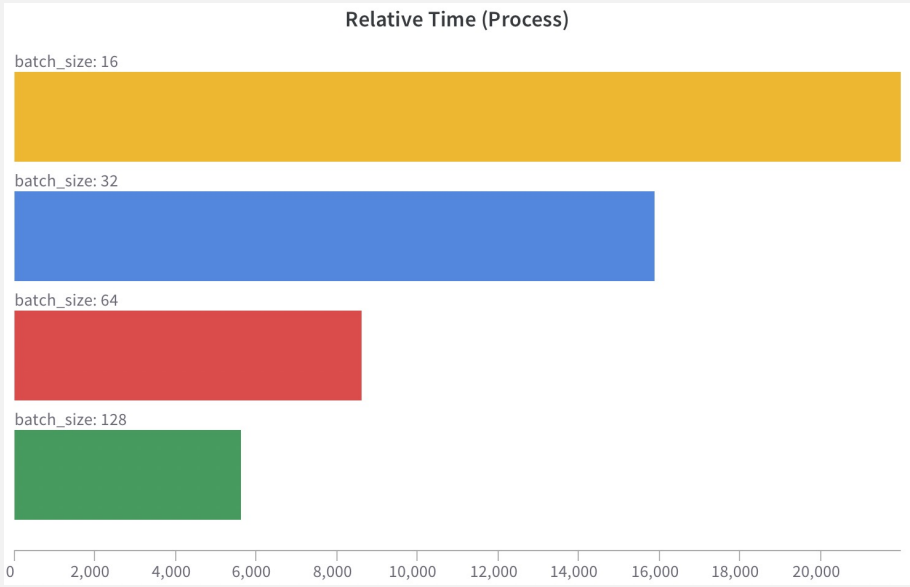
- Learning rate 0.1: large influctuation in test accuracy, three of the four models failed to converge (with 20-epochs early stop)
- Learning rate 0.001: too slow, models need more epochs to converge comparing to learning rate 0.01

Optimal learning rate 0.01

4 CONVOLUTIONAL LAYERS + 2 FULLY CONNECTED LAYERS



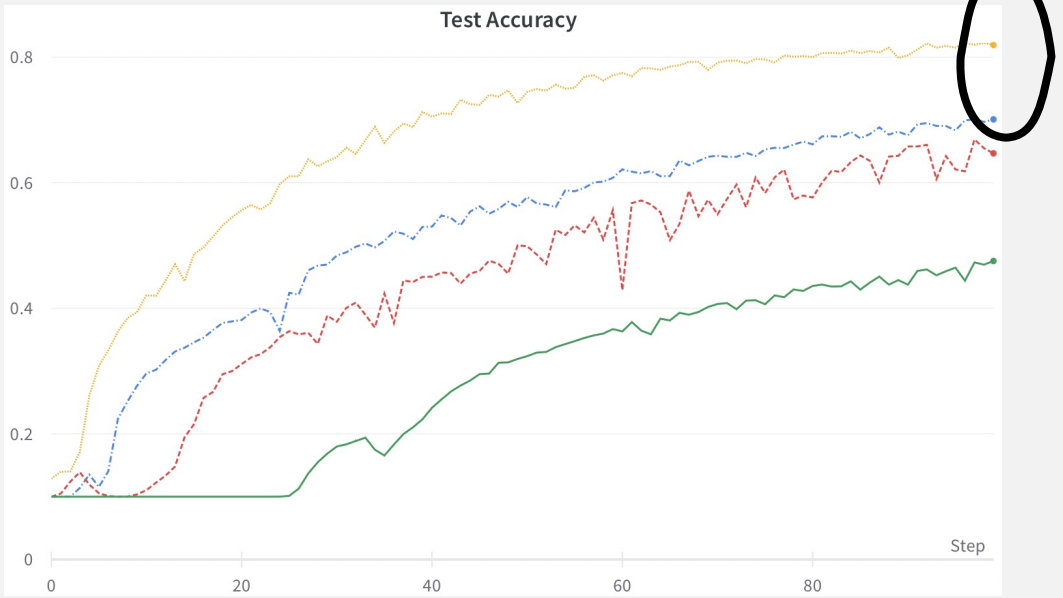
Optimal batch size



Learning rate: 0.01

Batch size:

- 16
- 32
- 64
- 128

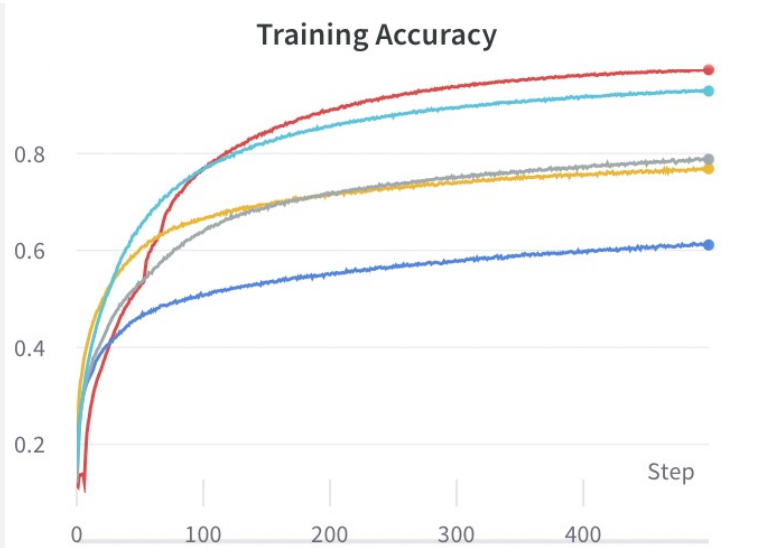


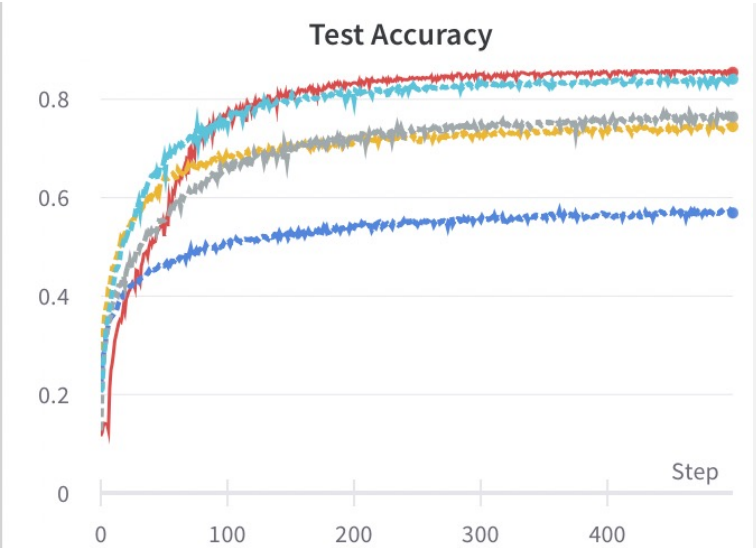
Analysis:

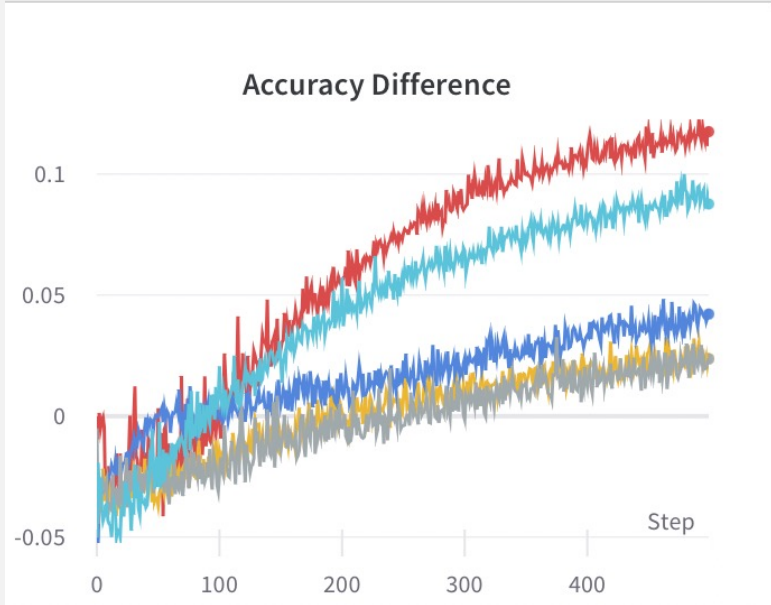
- Accuracy increased when batch size is smaller
- But computing time also increased when batch size is decreasing

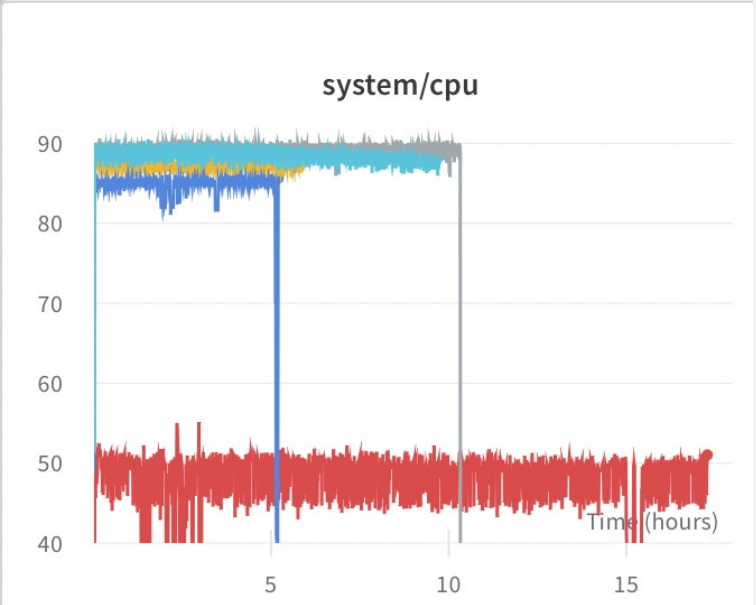
Optimal batch size 16 or 32

COMPARE 5 MODELS









One convolutional layer is more power than one fully connected layer

- Accuracy

4-conv > 3-conv >> 2-conv > 1-conv >> 0-conv

- Accuracy difference

4-conv >> 3-conv >> 2-conv > 1-conv >> 0-conv

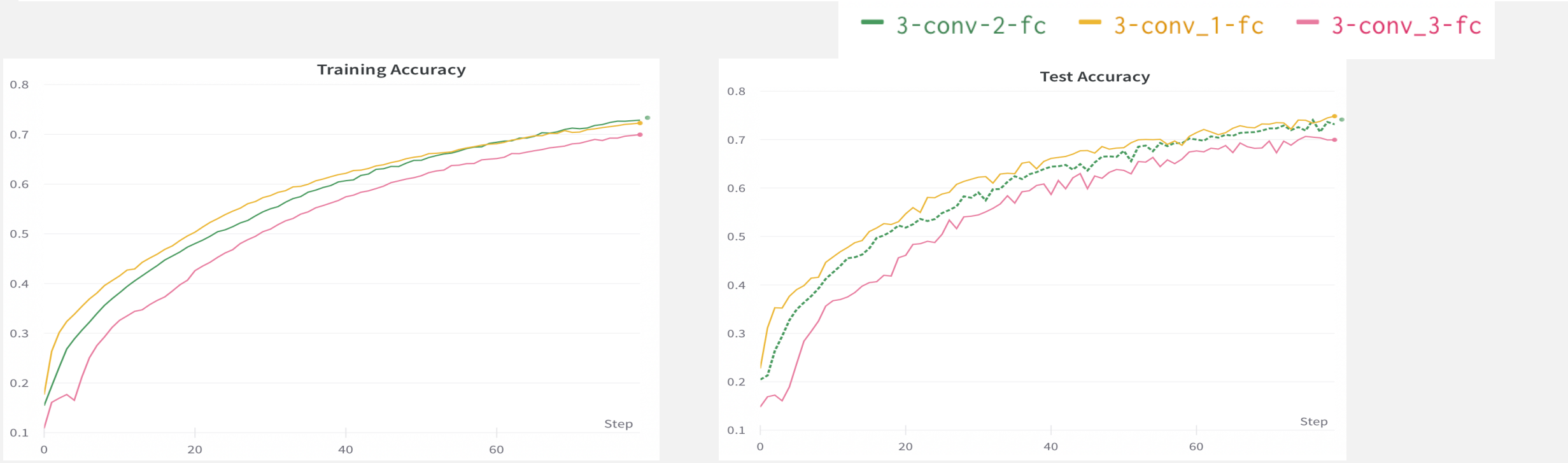
- Computing cost

4-conv > 2-conv > 3-conv >> 1-conv > 0-conv

- Decision

Model with **3 convolutional layer** has a good balance between computing time and accuracy. model is overfitting, but this is quite common in deep learning. It will be reduced later by applying regularization and stopping early at a suitable epoch.

2. HOW MANY FULLY-CONNECTED LAYERS?

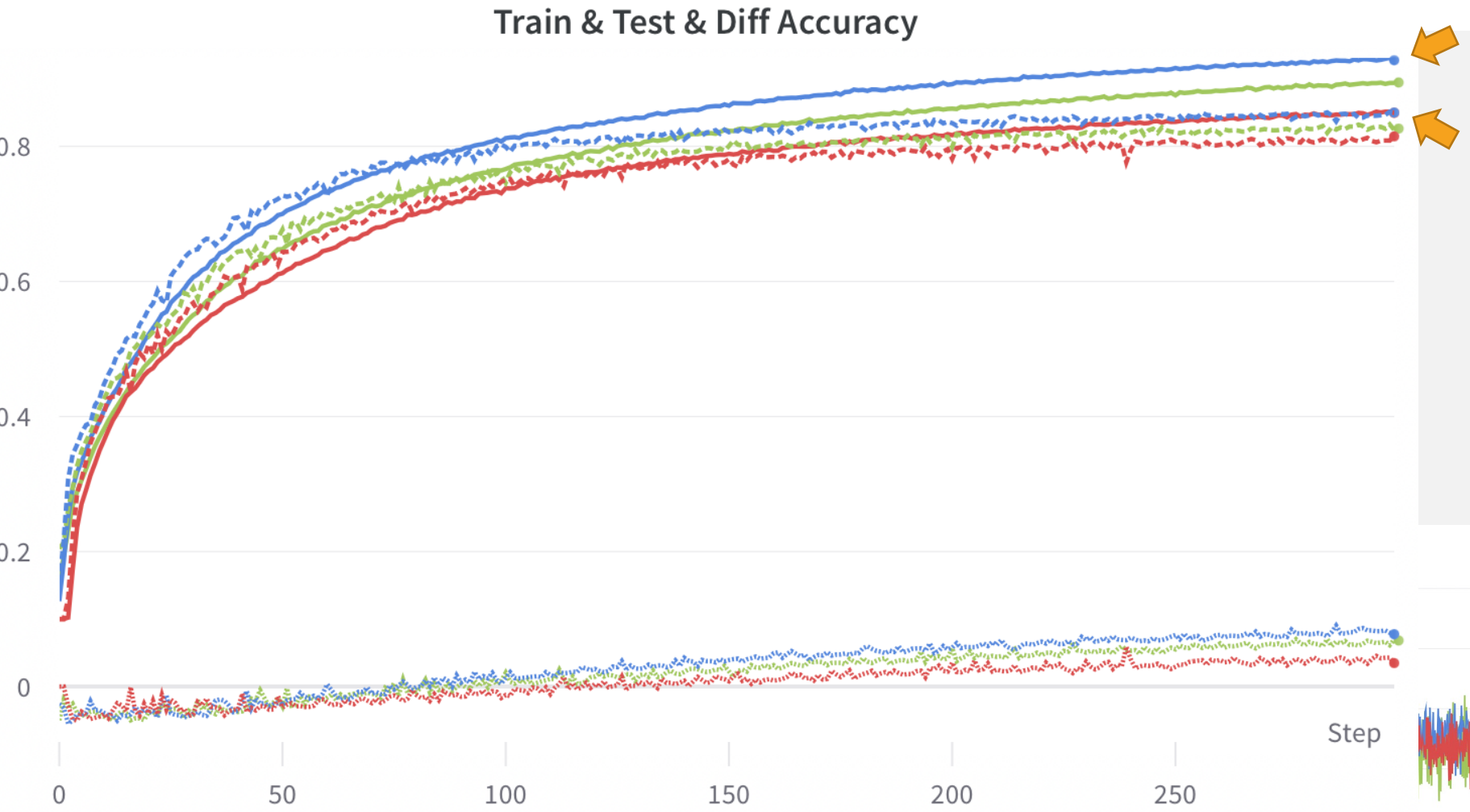


(with **80** epochs)
- performance: 1 fc layer > 2 fc layers >> 3 fc layers
- but the accuracy line with 1 fc layer is much flatter than with 2 fc layers
  -> when use more epochs, the model with 2 fc layers will likely reach much better accuracy.

--> use the architecture **3 convolutional layers + 2 fc layers**

Optimal hyperparameter	Next step
• 3 convolutional layers • Learning rate 0.01 • Batch size 32 • 2 fully connected layers	How many filters in each convolutional layer?

3. HOW MANY FILTERS IN EACH CONV LAYER



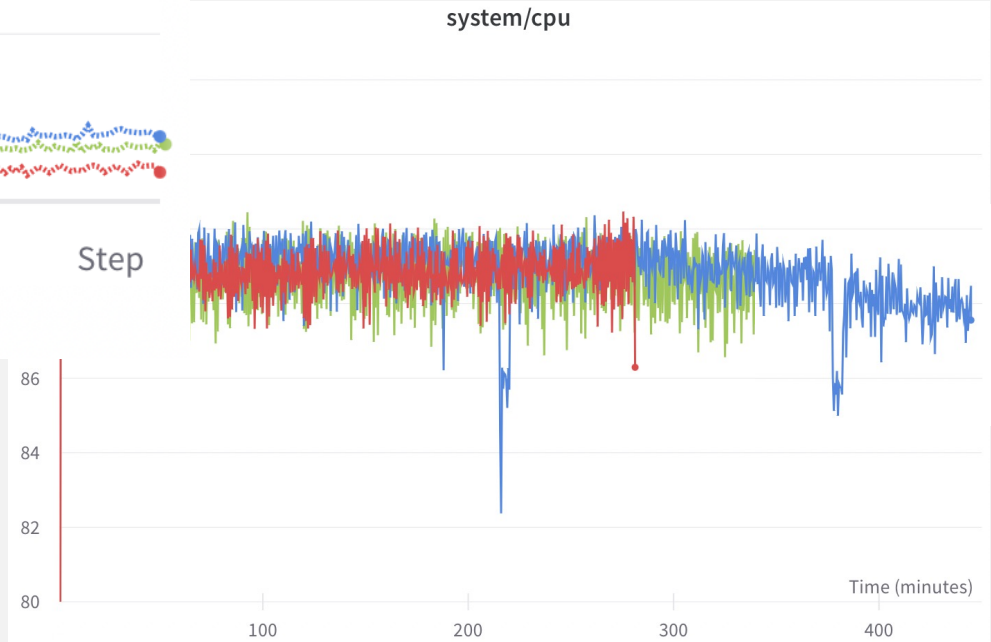
3 convolutional layers

nr. of filters:

- 16 - 32 - 128
- 32 - 64 - 256
- 64 - 128 - 512

Accuracy:

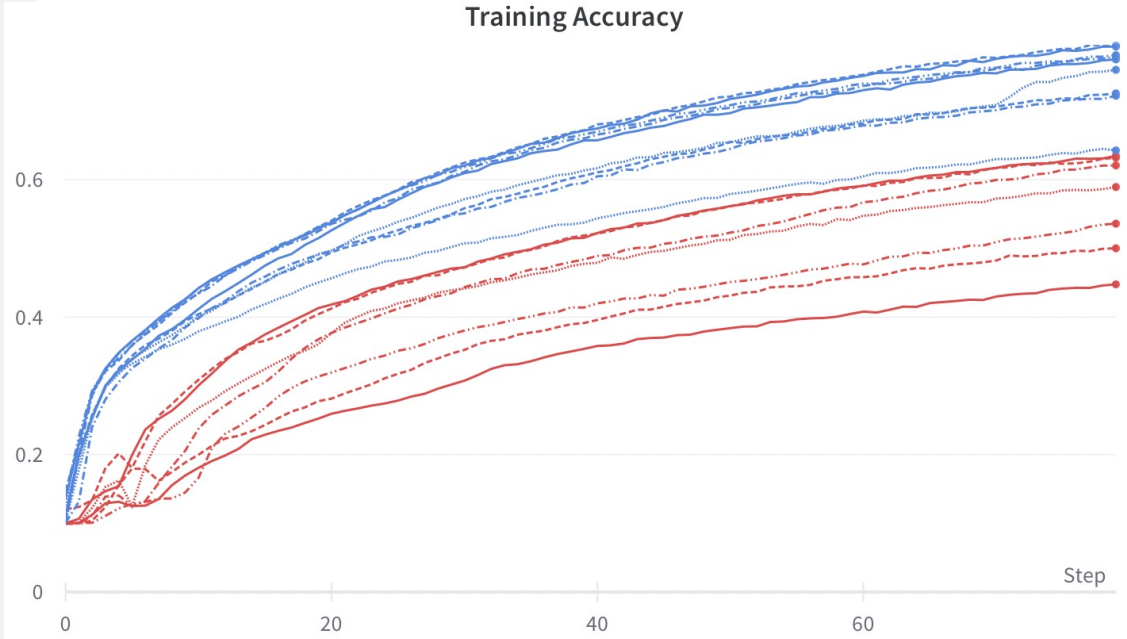
- train
- test
- difference

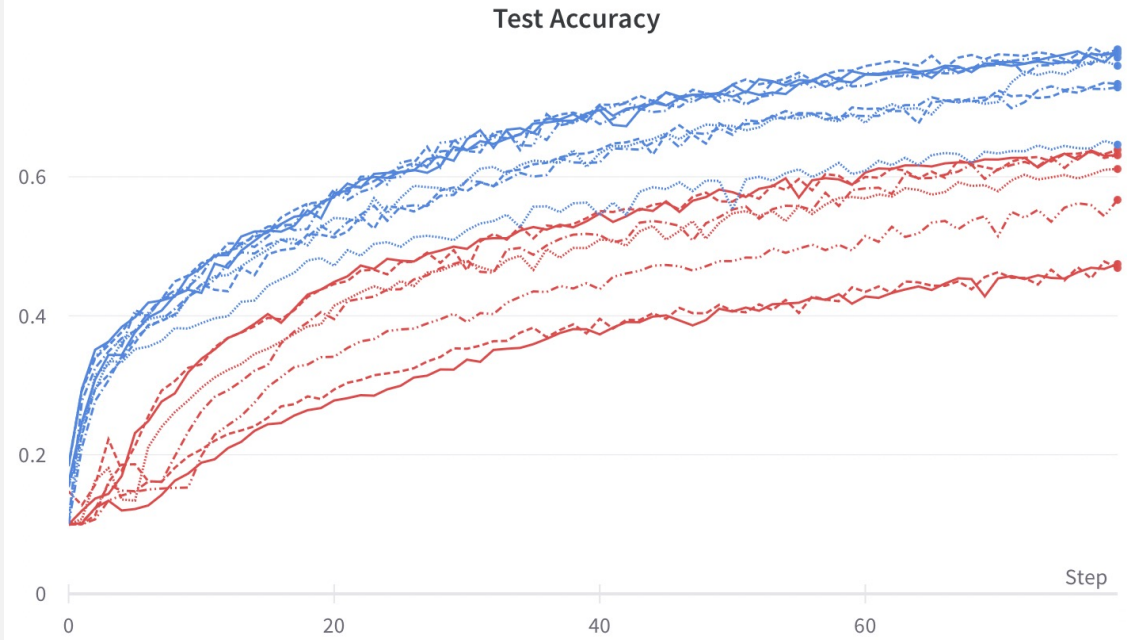


Number of filters in the three convolutional layers:
64 - 128 - 512

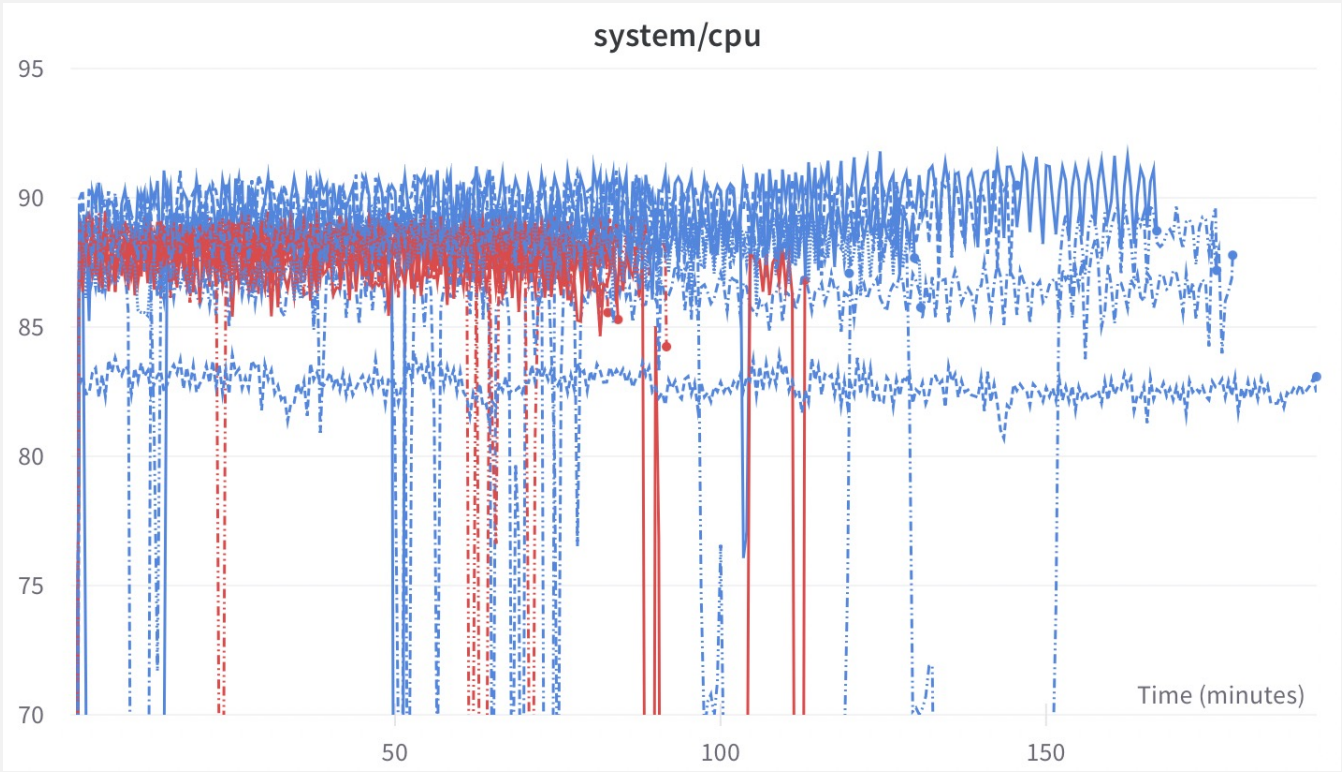
Optimal hyperparameter	Next step
• 3 convolutional layers • Learning rate 0.01 • Batch size 32 • 2 fully connected layers • 64 – 128 – 512 filters in convolutional layers	convolutional hyperparameter: - Kernel size - Stride - Padding - pooling

4. KERNEL SIZE & STRIDE & PADDING





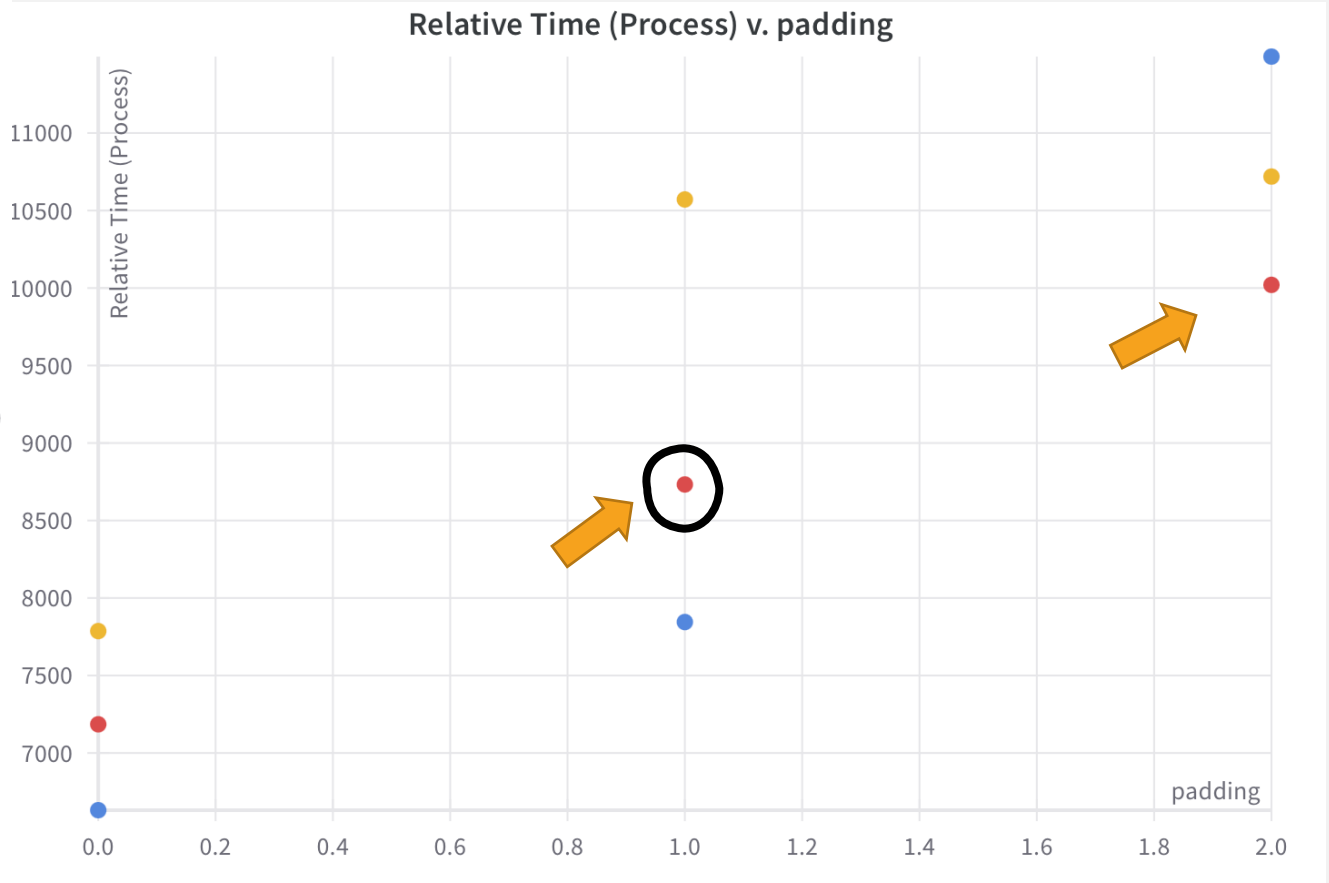
stride = 1
stride = 2



4. KERNEL SIZE & STRIDE & PADDING

stride = 1

- kernel size = 2
- kernel size = 3
- kernel size = 4



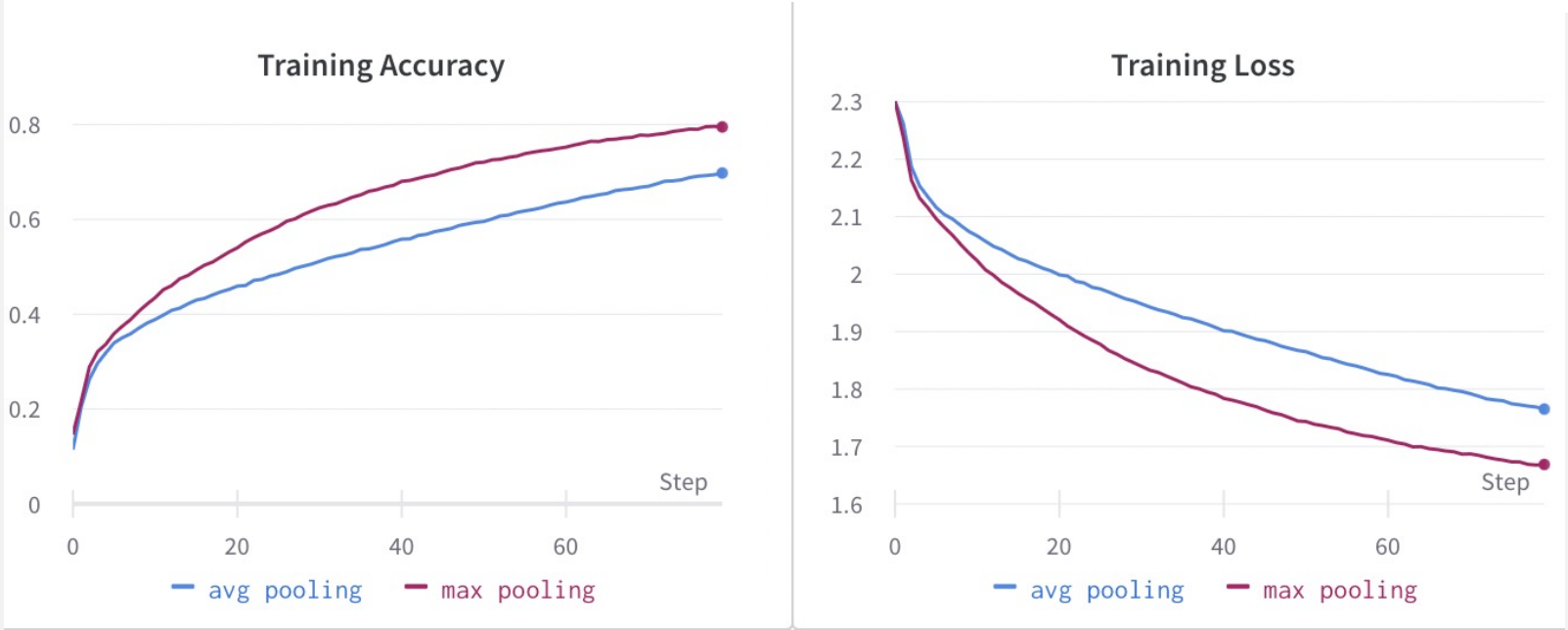
Accuracy:

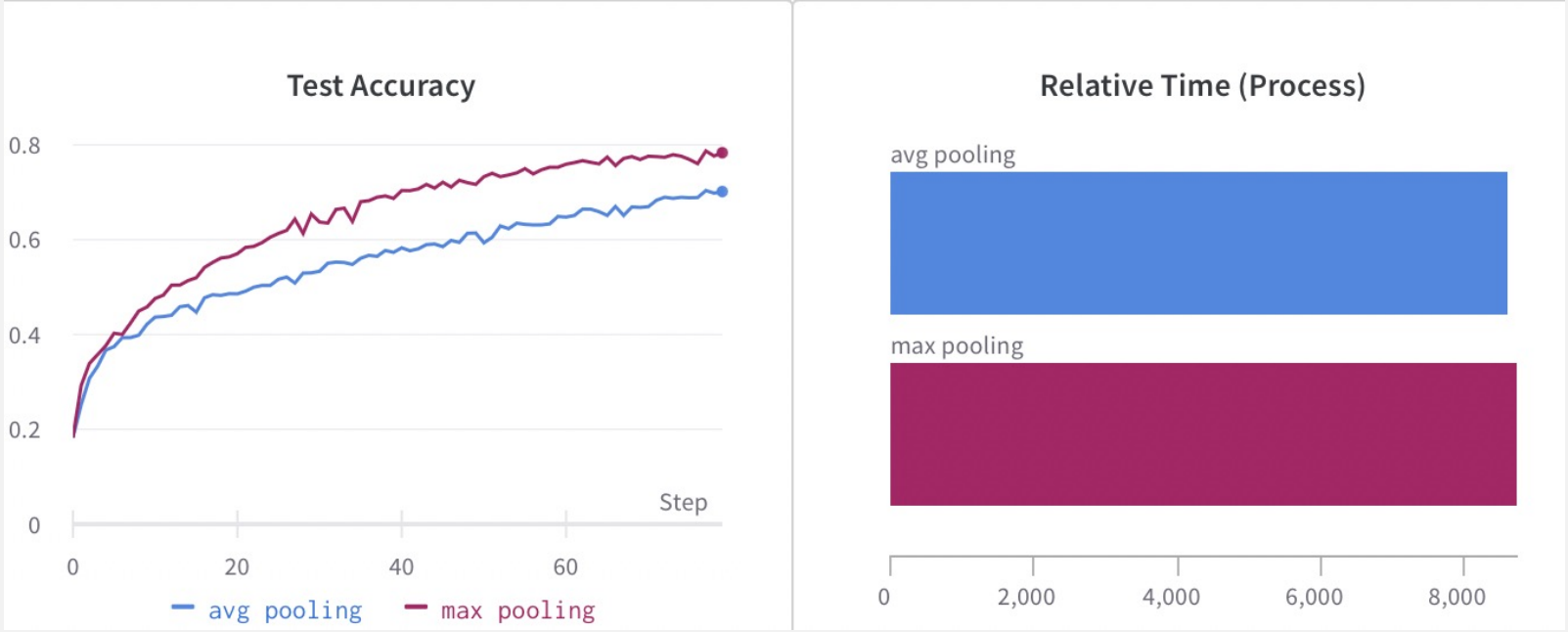
- kernel size = 4 & padding =1
- kernel size = 4 & padding =2

Computational time:

- kernel size = 4 & padding =1 is much shorter than kernel size = 4 & padding =2

POOLING TYPE



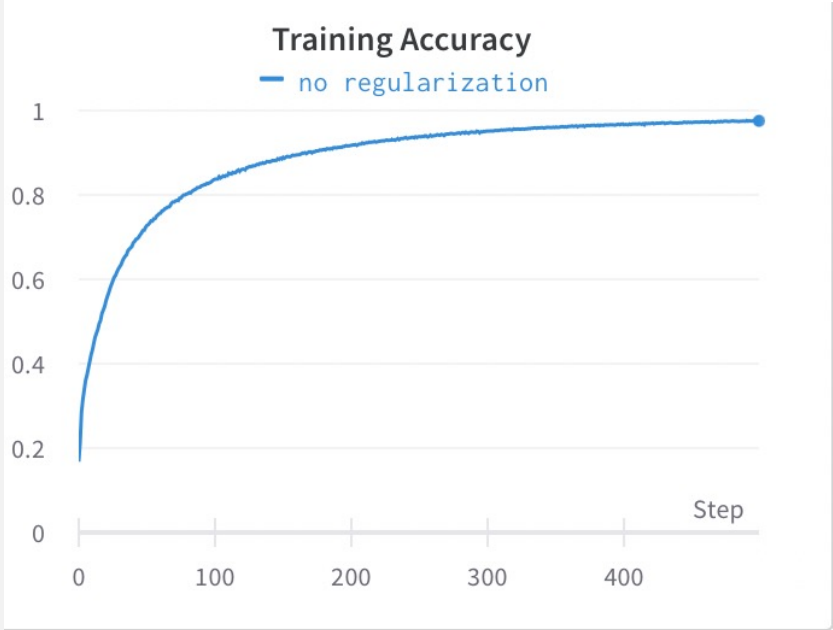


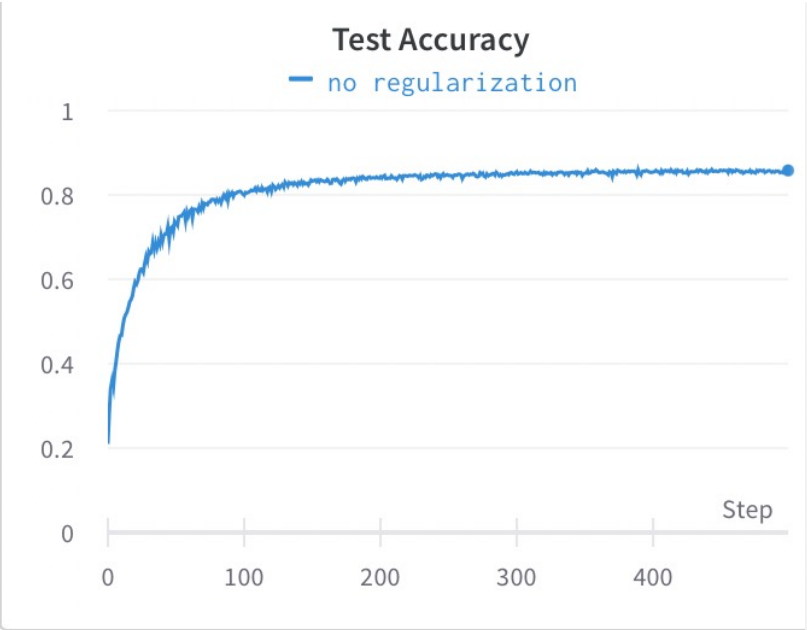
Kernel size = 4
Stride = 1
Padding = 1

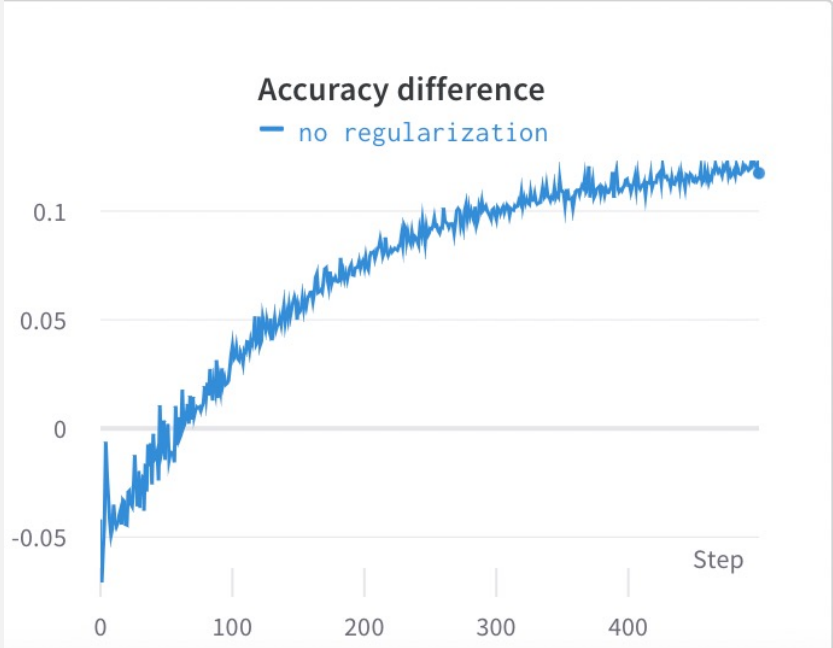
Max pooling

Optimal hyperparameter	Next step
• 3 convolutional layers • Learning rate 0.01 • Batch size 32 • 2 fully connected layers • 64 – 128 – 512 filters in convolutional layers • Convolutional hyperparameters - Kernel size = 4 - stride = 1 - padding = 1 - max pooling of size 2	Regularization? L1, L2, dropout?

NO REGULARIZATION (RECAP)





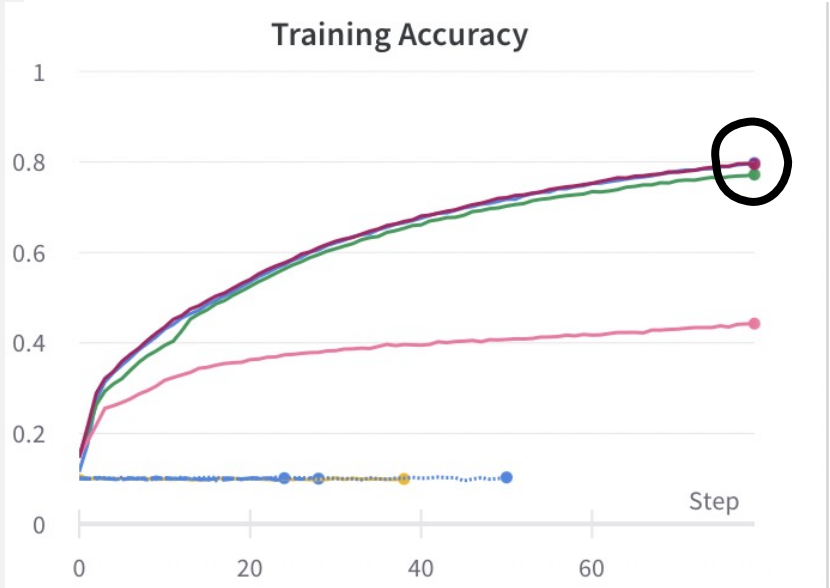


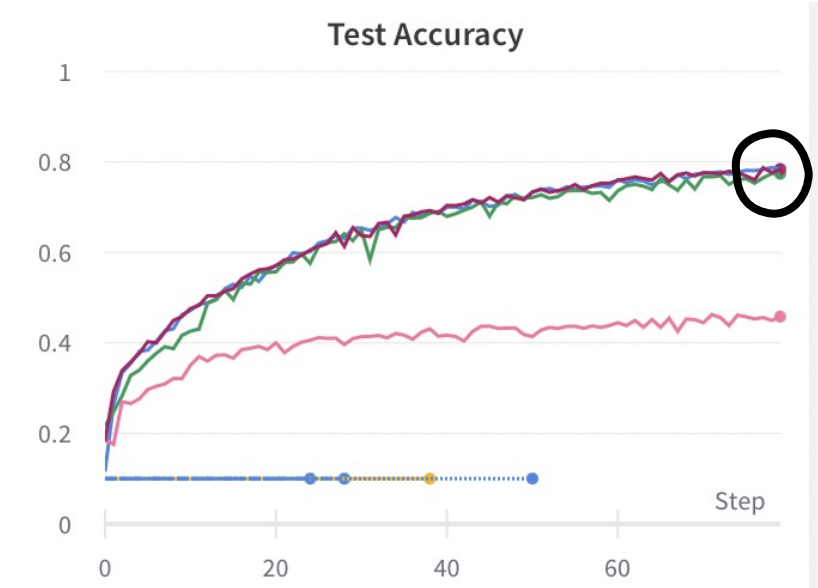
- the model reaches accuracy of trainset 0.9755, testset 0.858

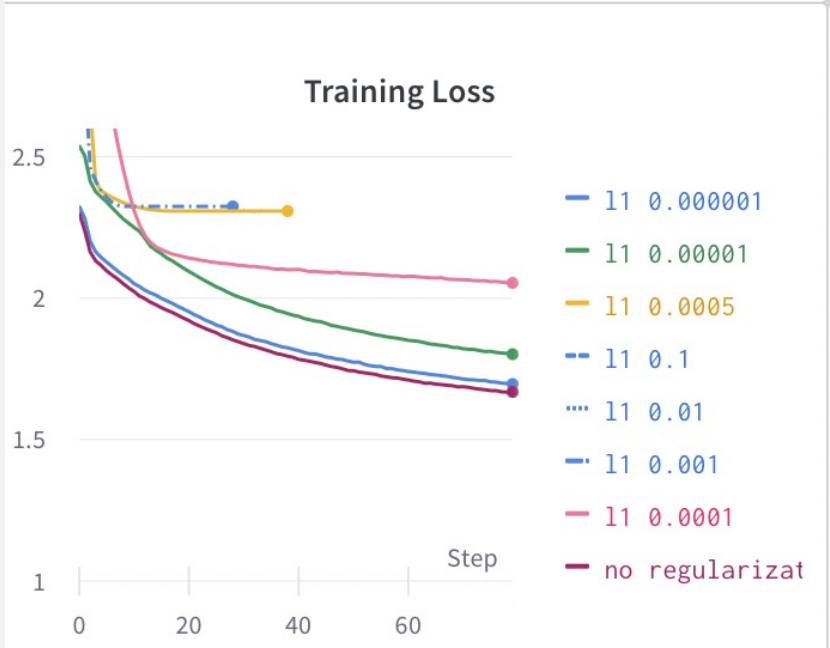
- the large accuracy difference (0.1165) indicated the model is overfitting

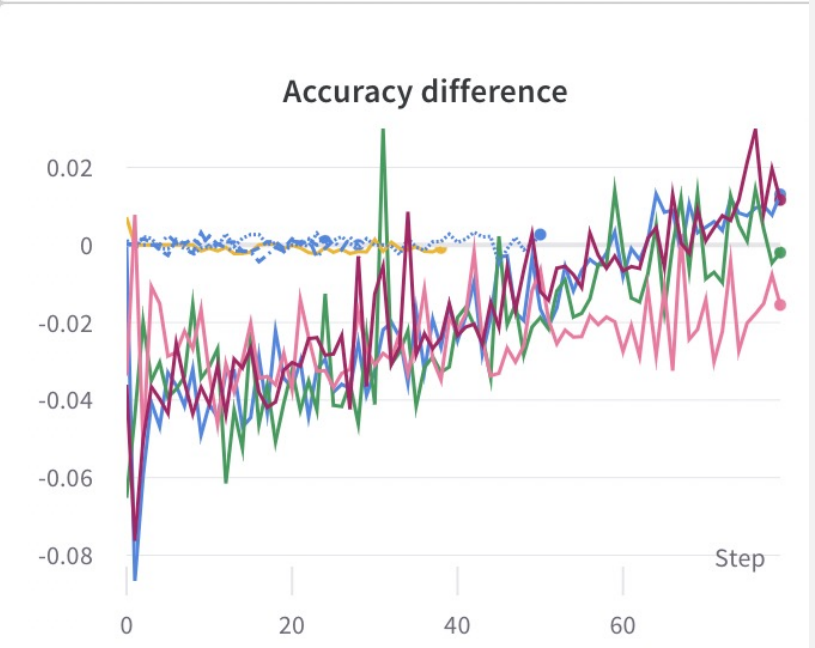
- So I will apply l1, l2 regularization and dropout to mitigate the overfitting in the next steps

5. L1 REGULARIZATION I



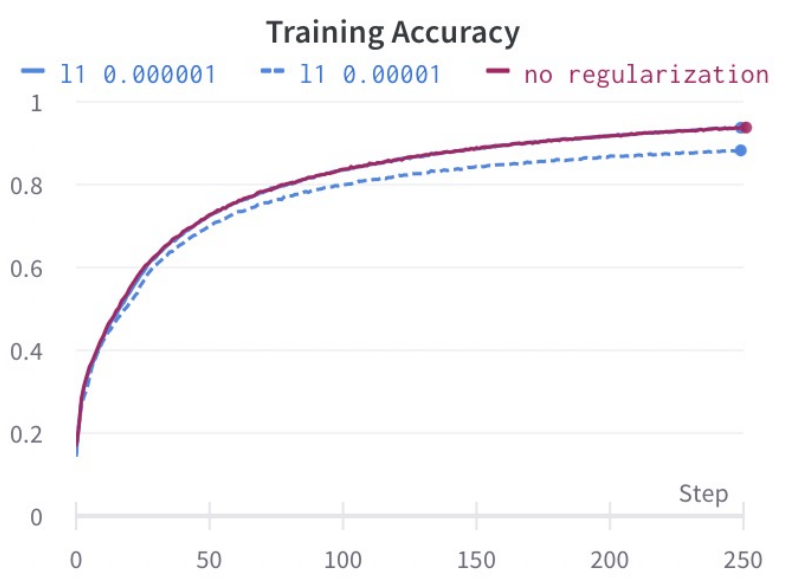


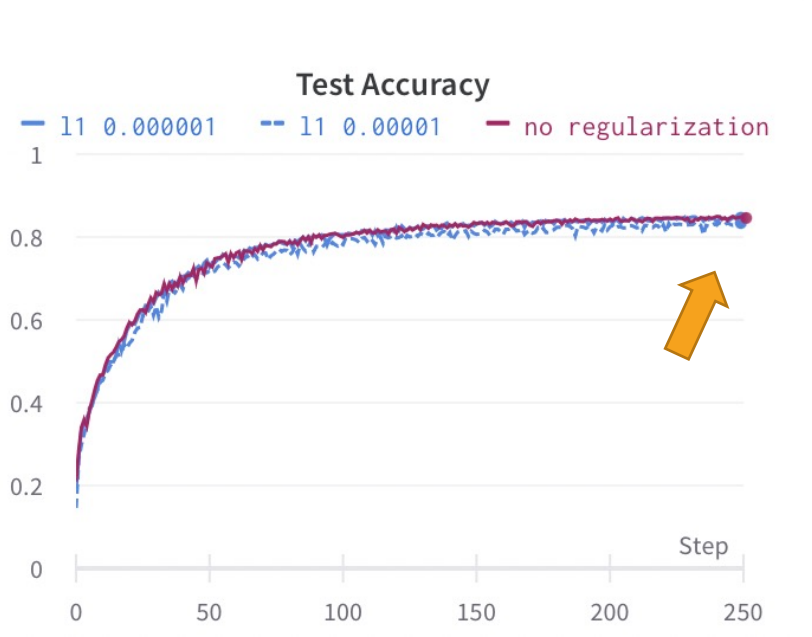


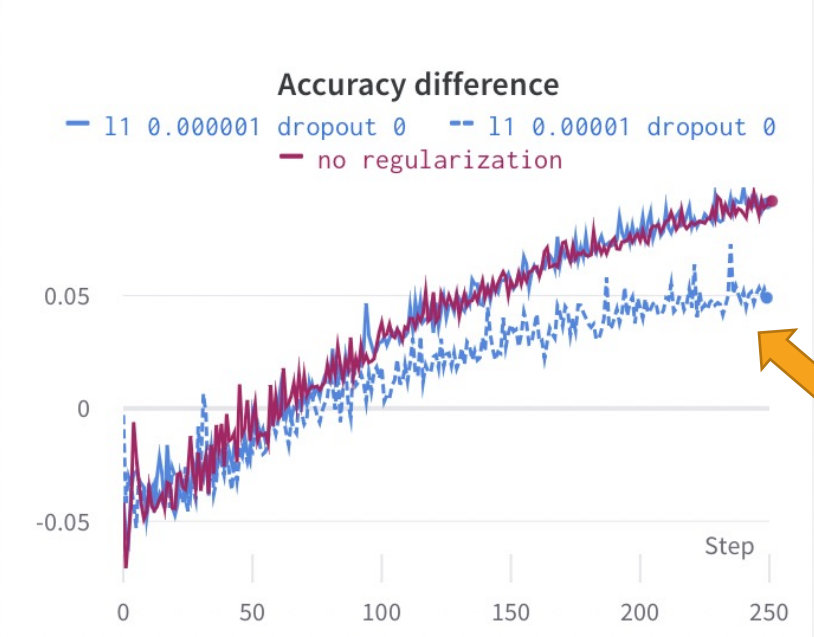


- L1 with strength 0.1, 0.01, 0.001, 0.0001, 0.0005, 0.00001, 0.000001
- Control: no regularization
- More epochs is needed to observe long term regularization effect

L1 REGULARIZATION II



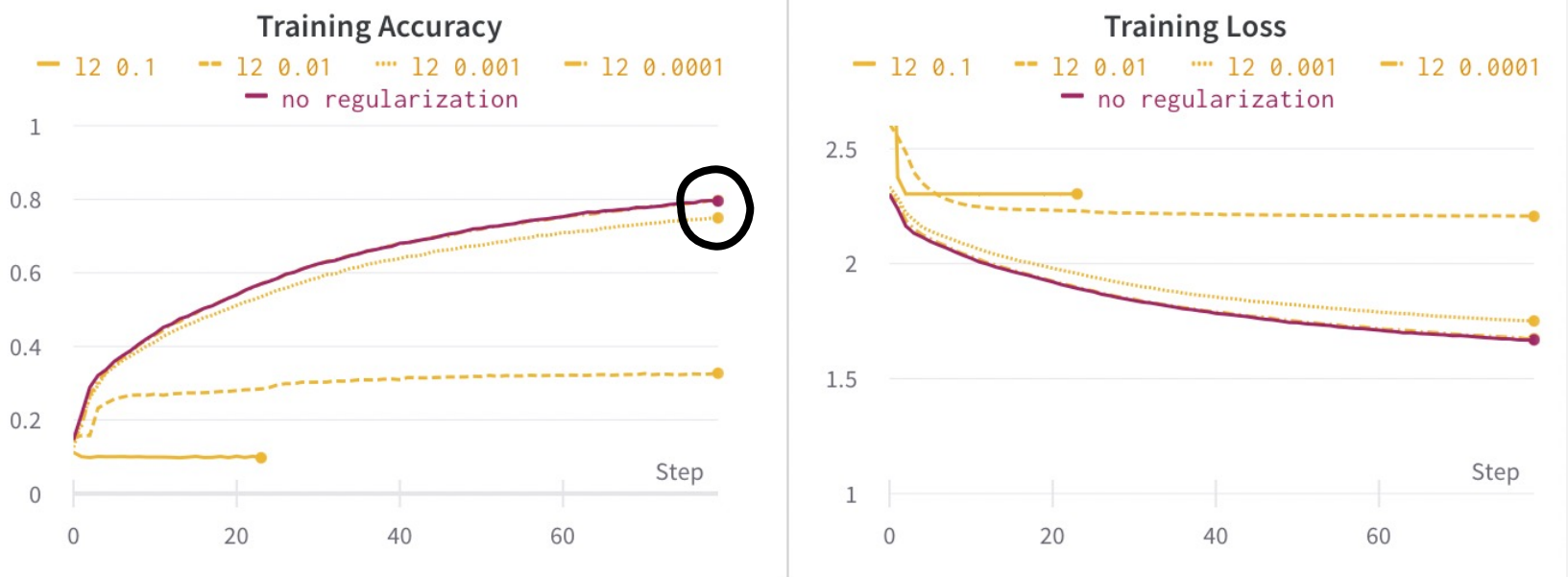


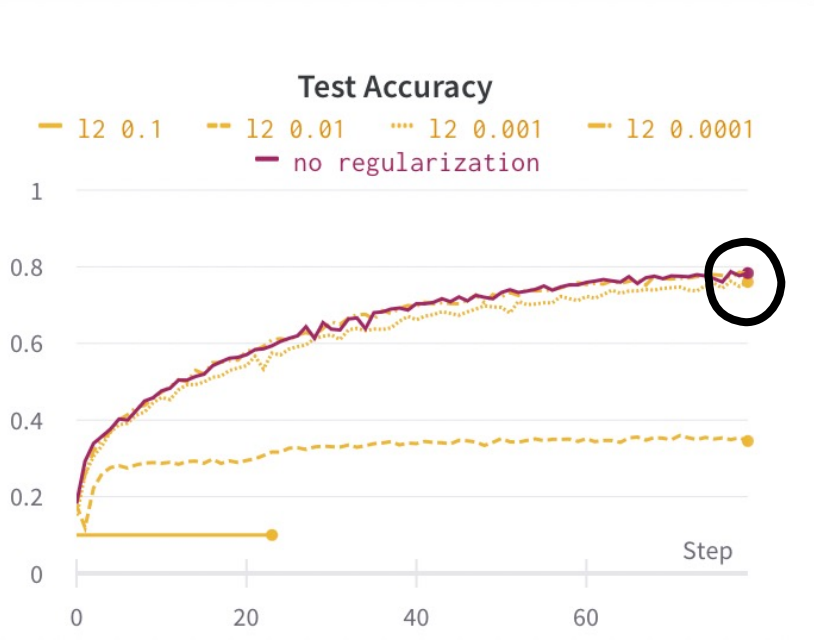


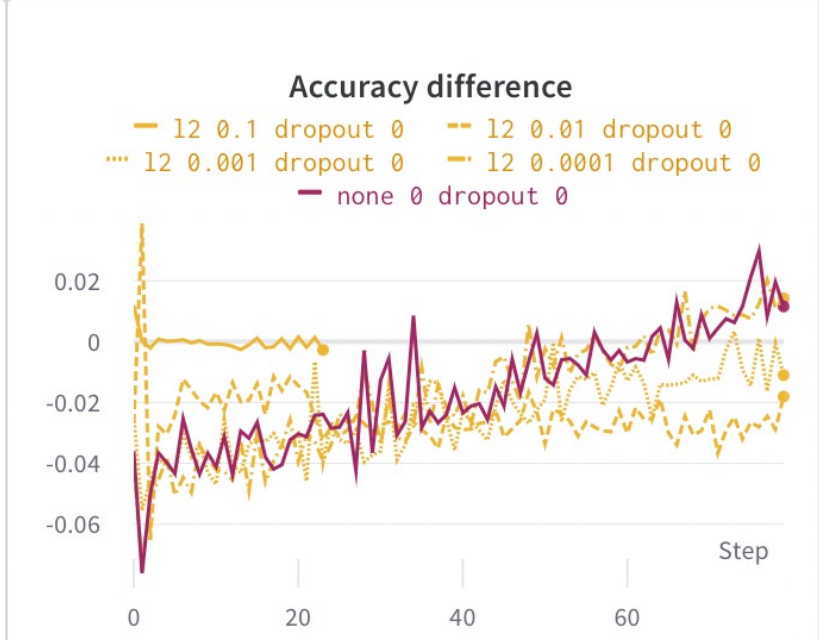
L1 0.00001

- Test accuracy is only slightly reduced
- Accuracy gap is reduced from around 0.09 to 0.05

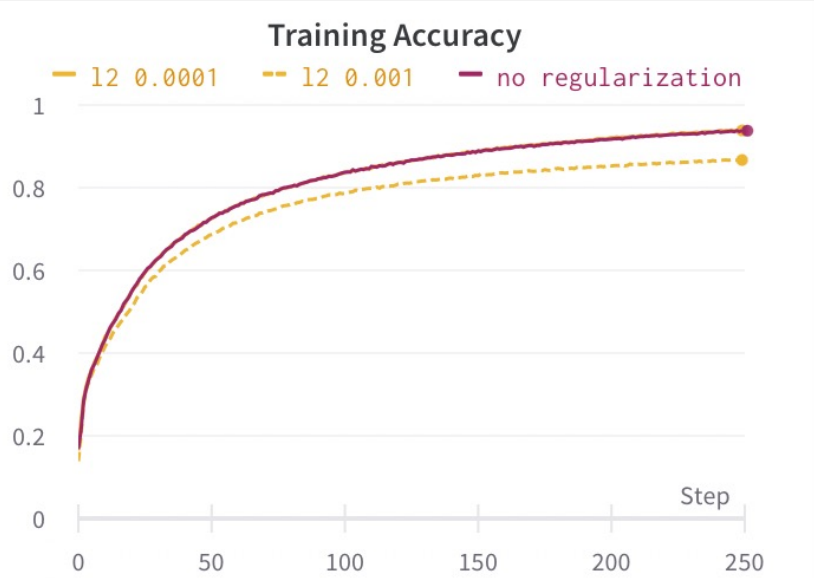
L2 REGULARIZATION I

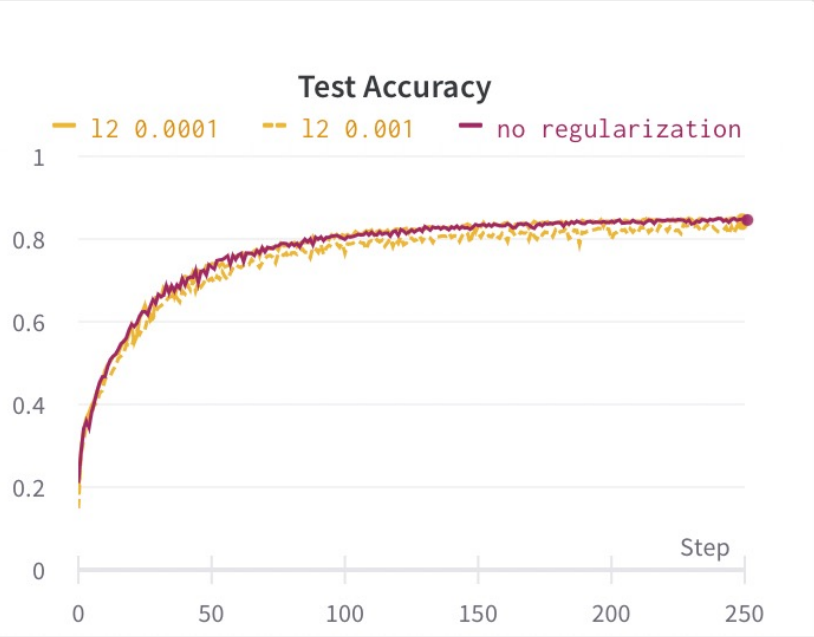


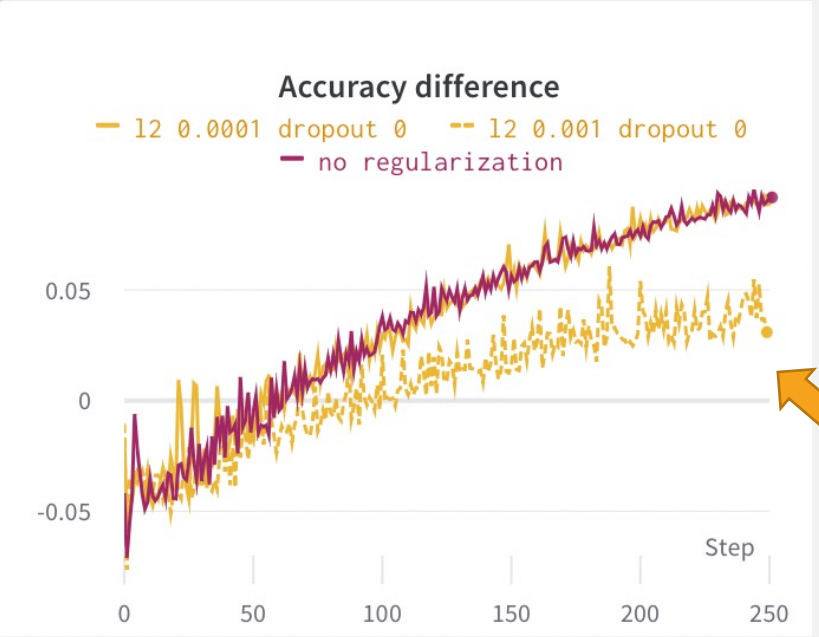




L2 REGULARIZATION II



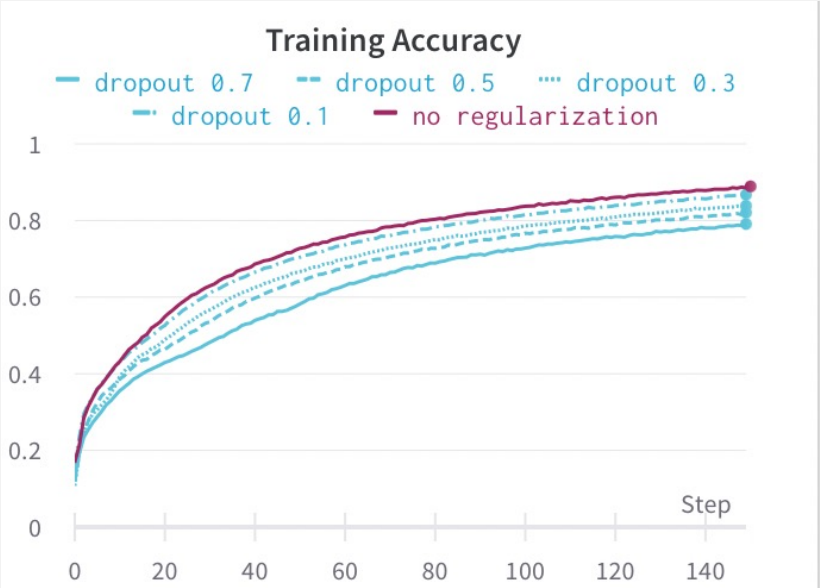


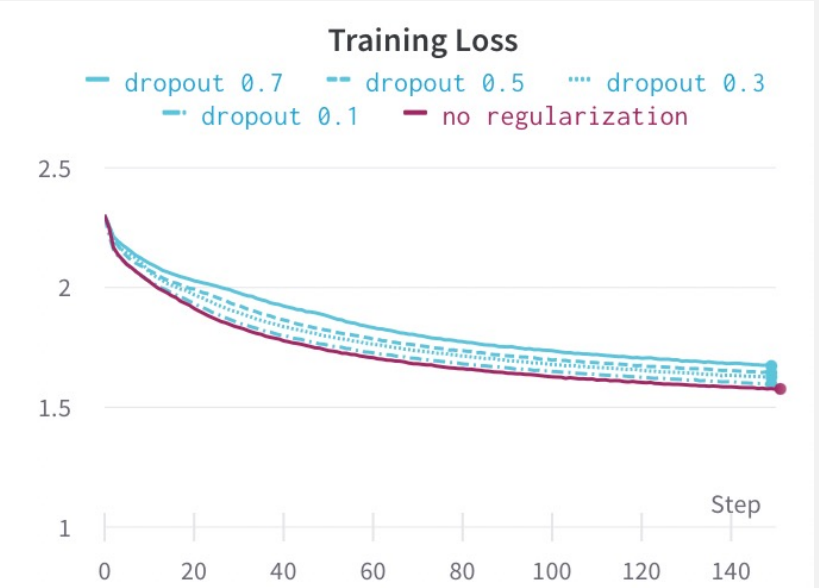


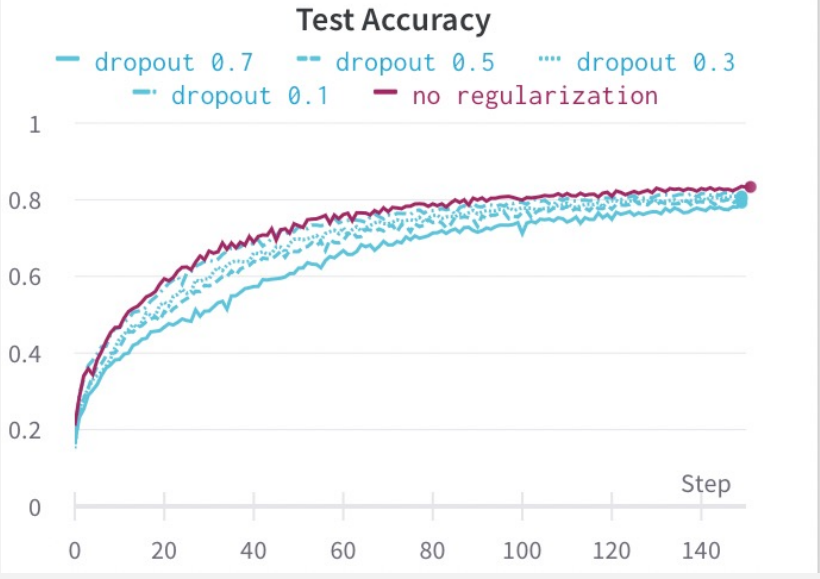
L2 0.001

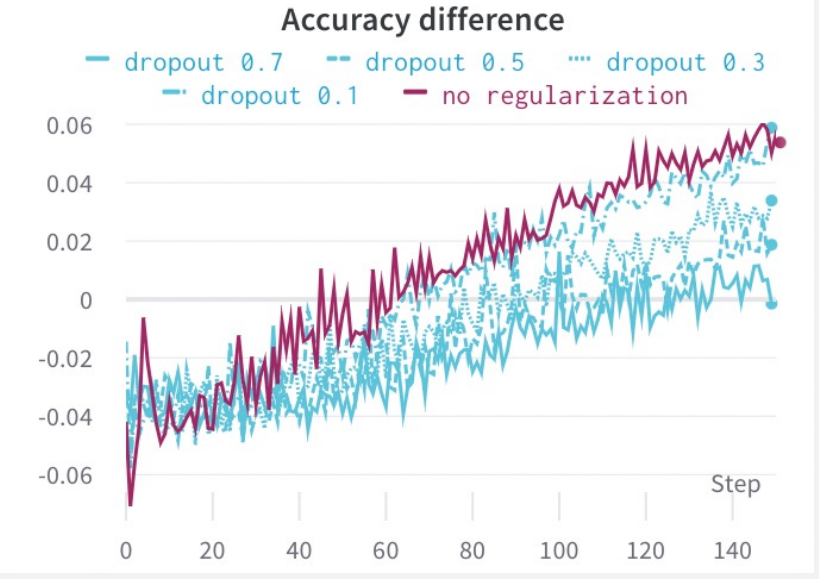
- Test accuracy is only slightly reduced
- Accuracy gap is reduced from around 0.09 to 0.035

DROPOUT





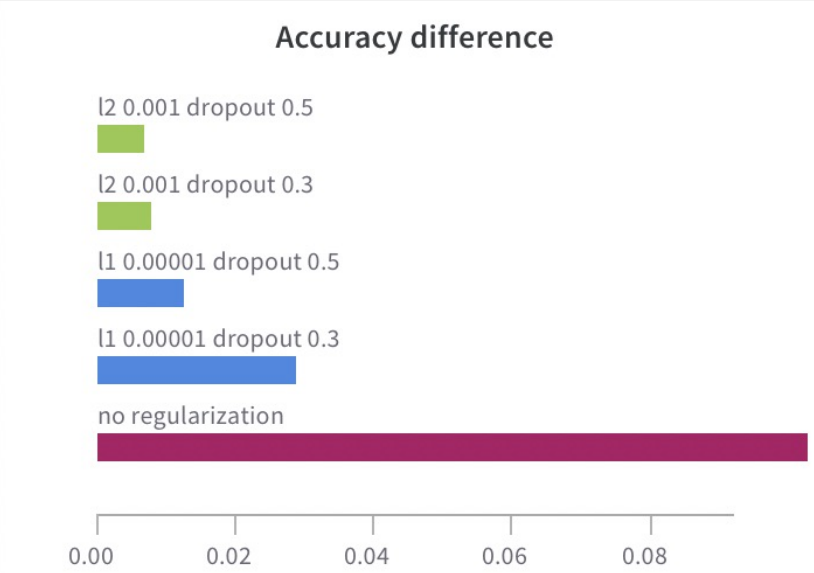


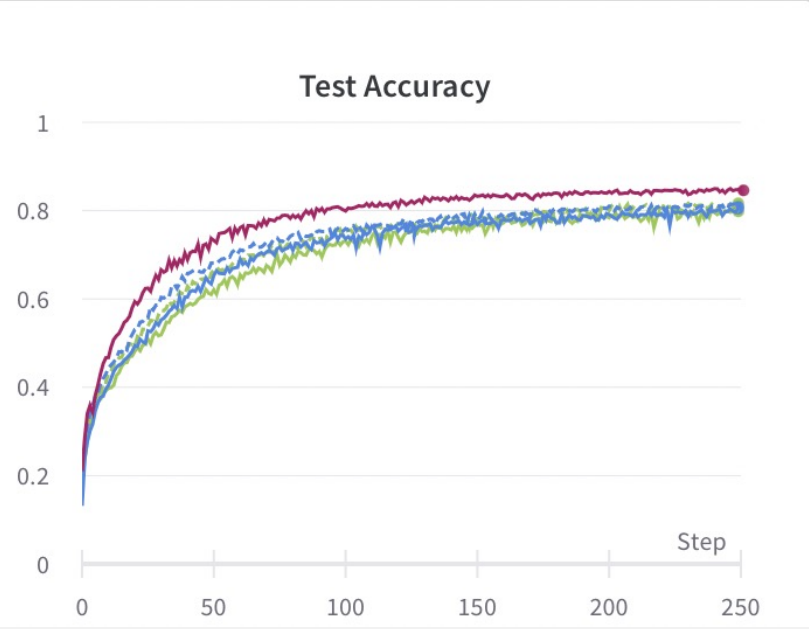


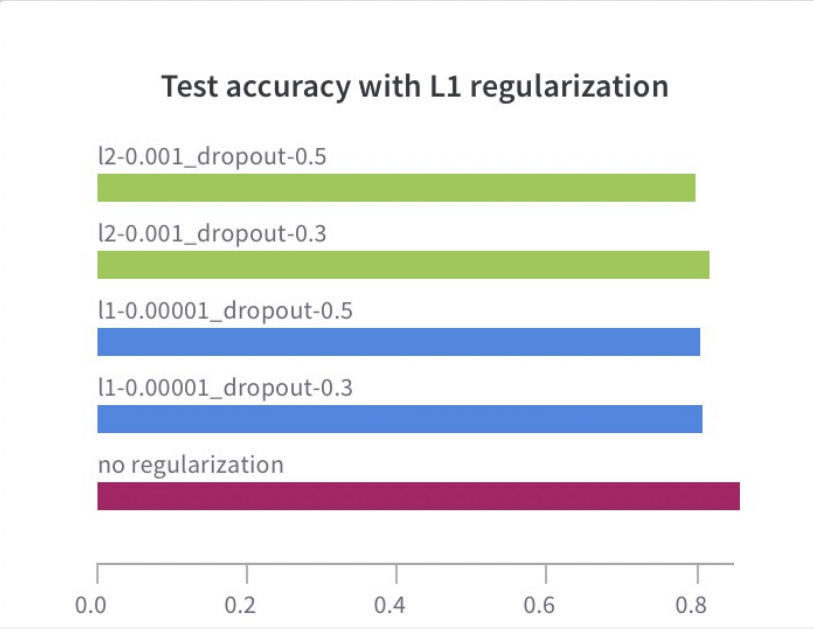
Dropout rate
0.1: performance is slightly reduced comparing to model without regularization. overfitting is not improved
0.7: train and test accuracy are similar, which indicates model is underfitting

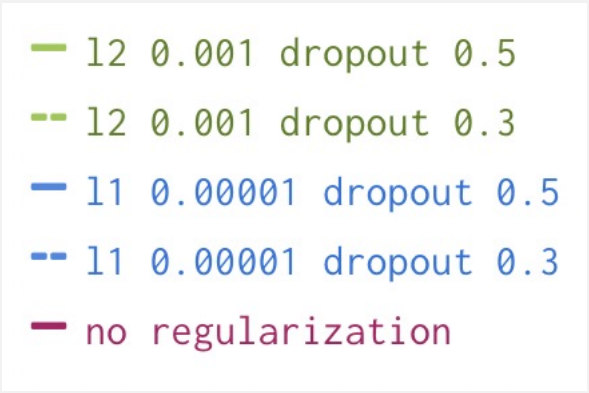
→ Drop out rate 0.3, 0.5

L1 + DROPOUT; L2 + DROPOUT

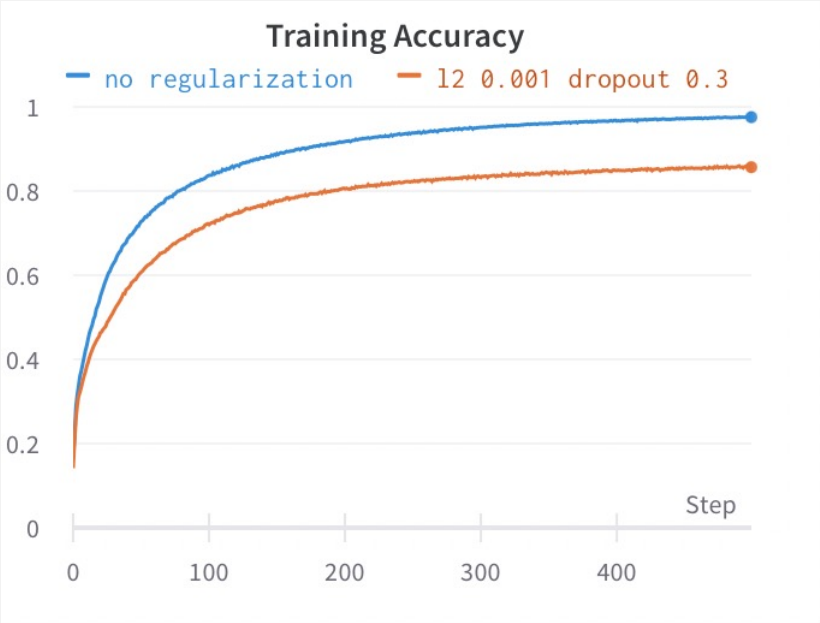


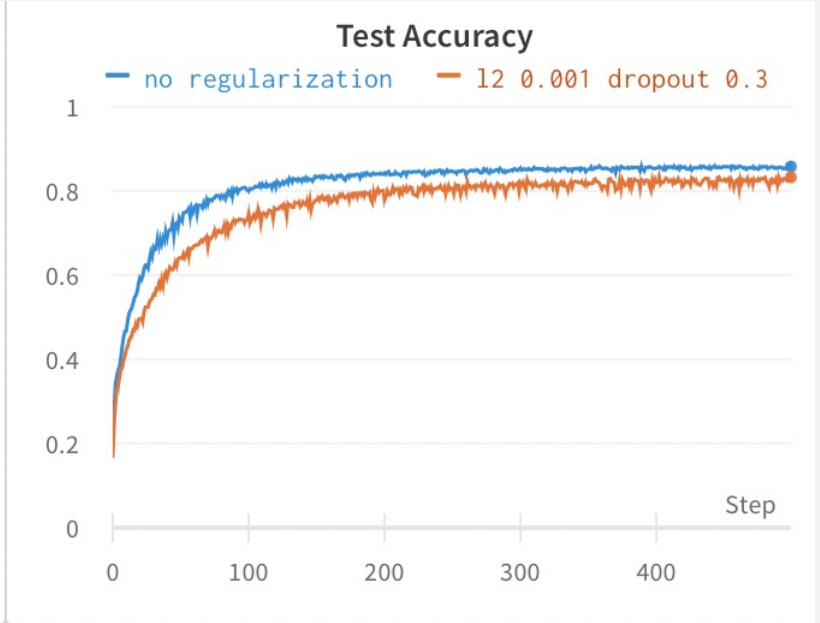


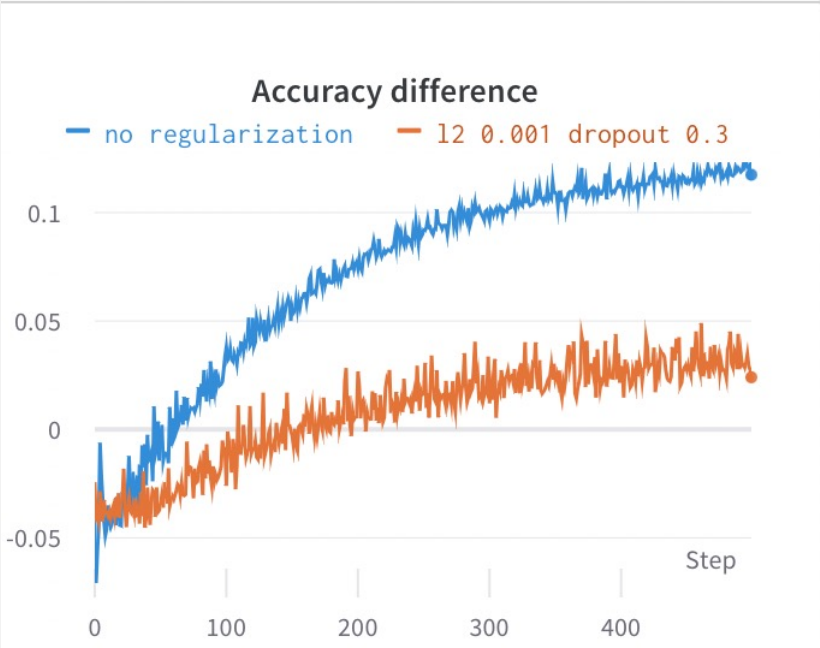




BEFORE VS AFTER REGULARIZATION



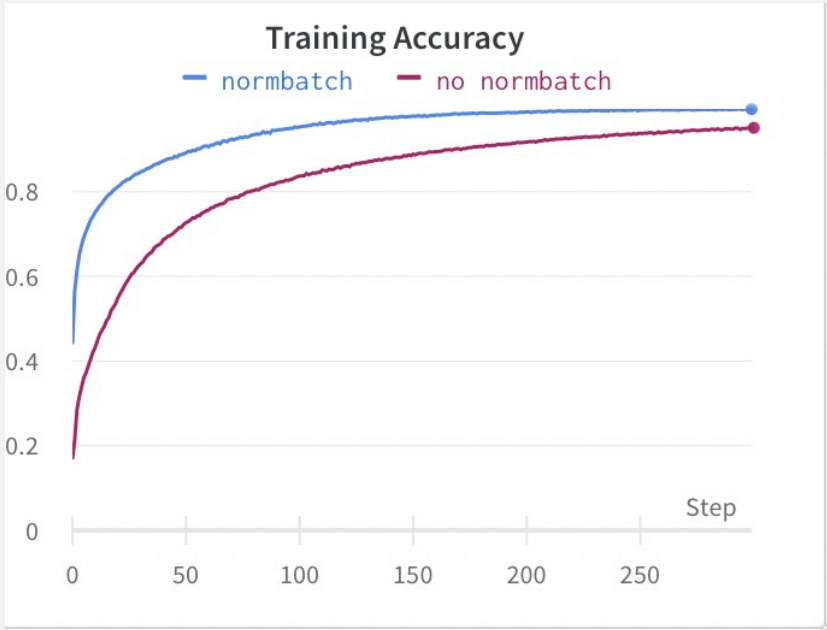


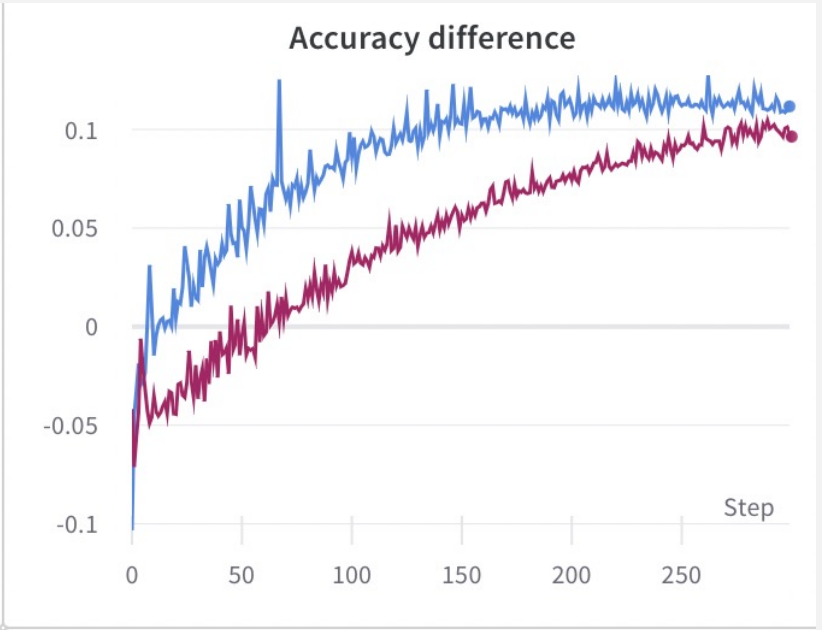


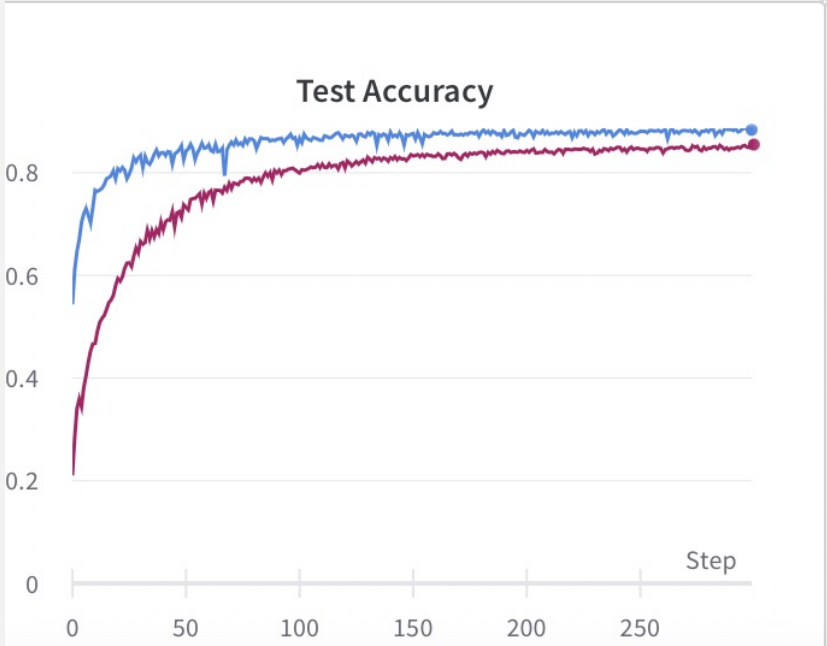
1. Accuracy on test set is only slightly reduced.
2. Accuracy difference is largely reduced from 0.12 to 0.03

Optimal hyperparameter	Next step
• 3 convolutional layers • Learning rate 0.01 • Batch size 32 • 2 fully connected layers • 64 – 128 – 512 filters in convolutional layers • Convolutional hyperparameters - Kernel size = 4 - stride = 1 - padding = 1 - max pooling of size 2 • L2 0.001 + dropout 0.3	Batch normalization

BATCHNORM







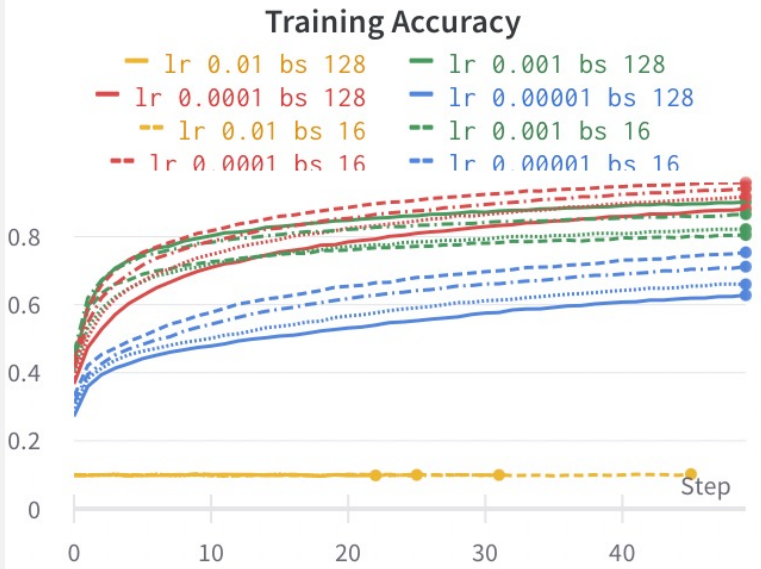
batch normalization on
- convolutional layers
- fully connected layers

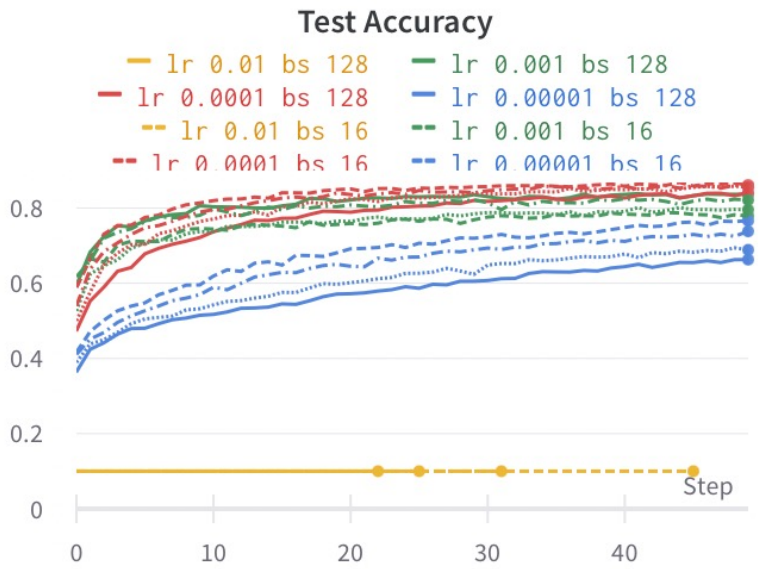
Improve learning efficiency:
- Reach a good accuracy with much less epochs

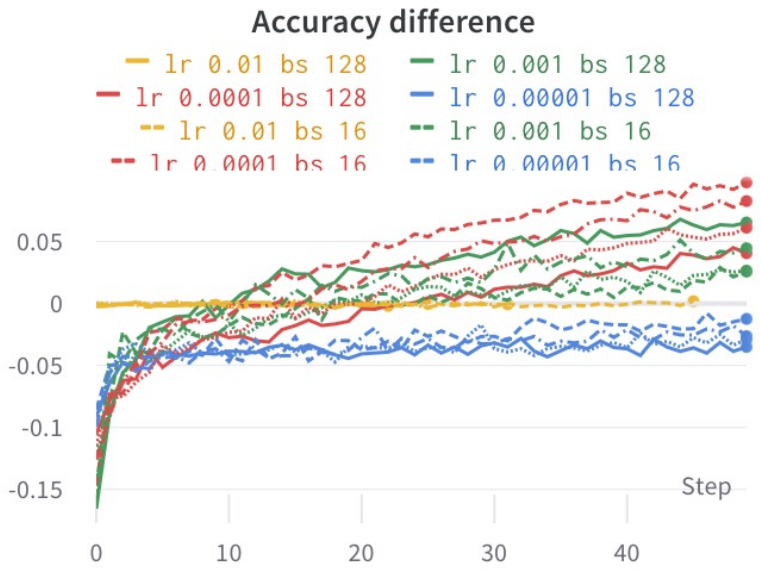
Optimal hyperparameter	Next step
• 3 convolutional layers • Learning rate 0.01 • Batch size 32 • 2 fully connected layers • 64 – 128 – 512 filters in convolutional layers • Convolutional hyperparameters - Kernel size = 4 - stride = 1 - padding = 1 - max pooling of size 2 • L2 0.001 + dropout 0.3	Optimizer: Adam vs SGD

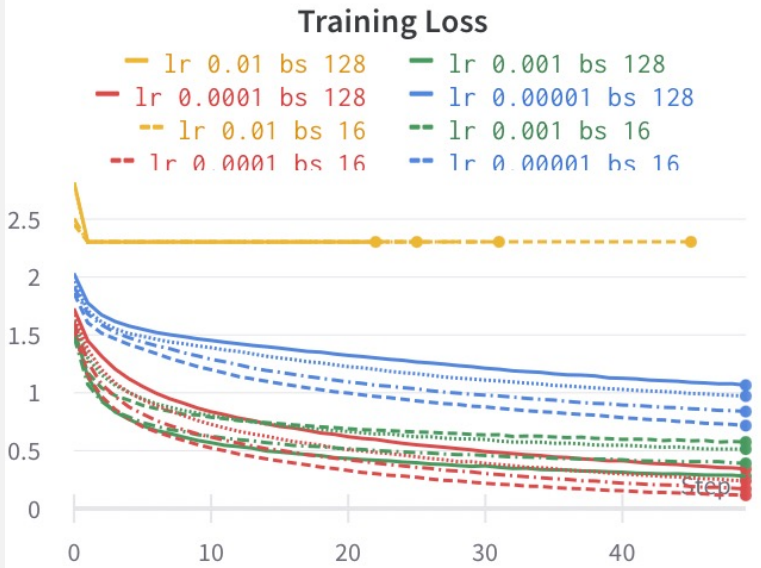
ADAM OPTIMIZER

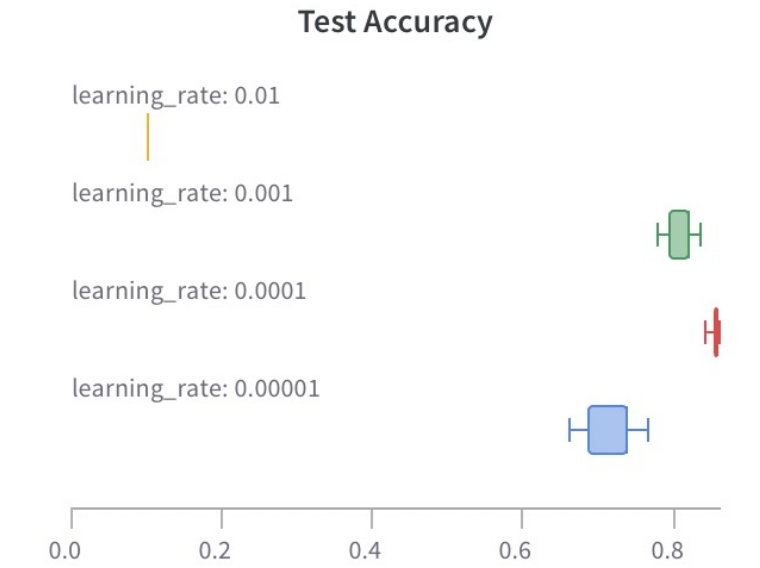
Learning rate 0.001
Batch size 64

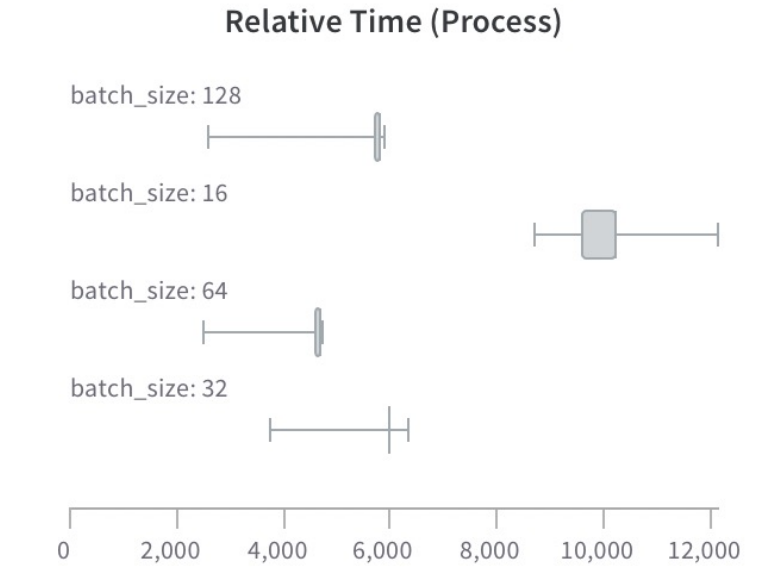




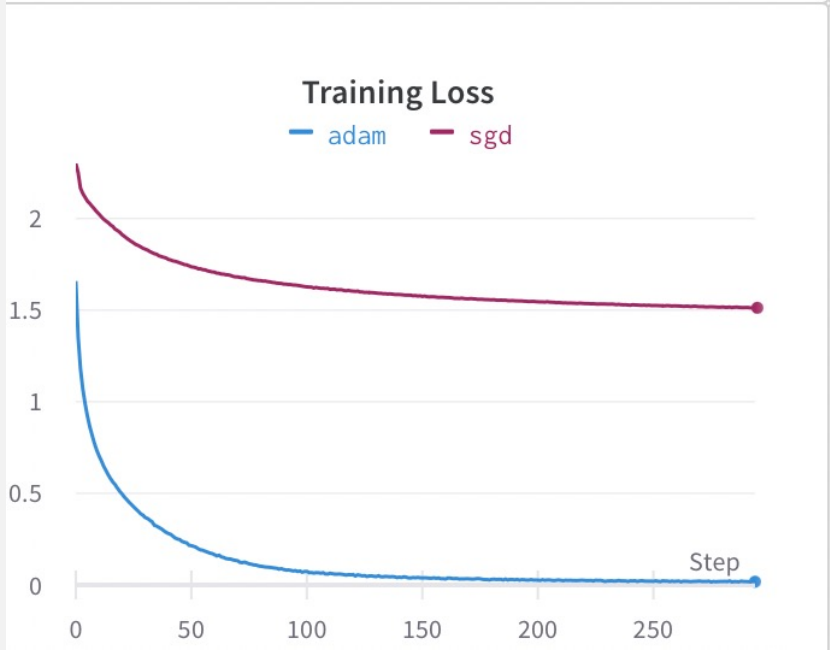


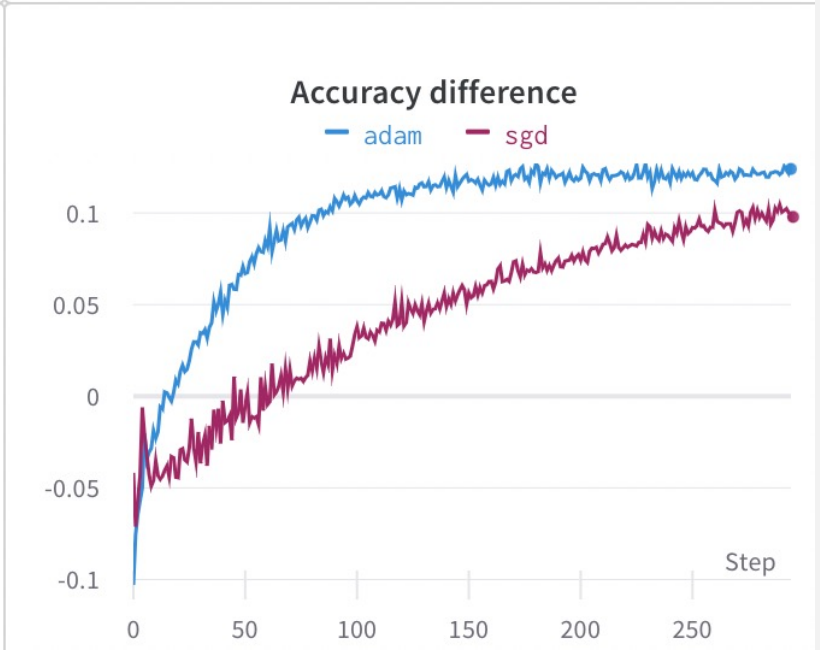


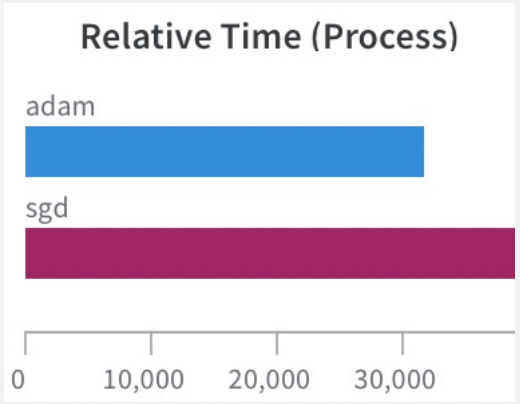




ADAM VS SGD







Adam adjust learning rate based on historical gradients:

- First moment: gradients mean -> direction
- second moment: gradients variance -> adjust learning rate based on the gradients scale

SUMMARY

- Adam optimizer is efficient due to its adaptive learning rate to each parameter
- Normbatch could stablize learning process and improve the learning efficiency by reducing the impact of batch-to-batch variations
- SGD needs fine hyperparameter tuning