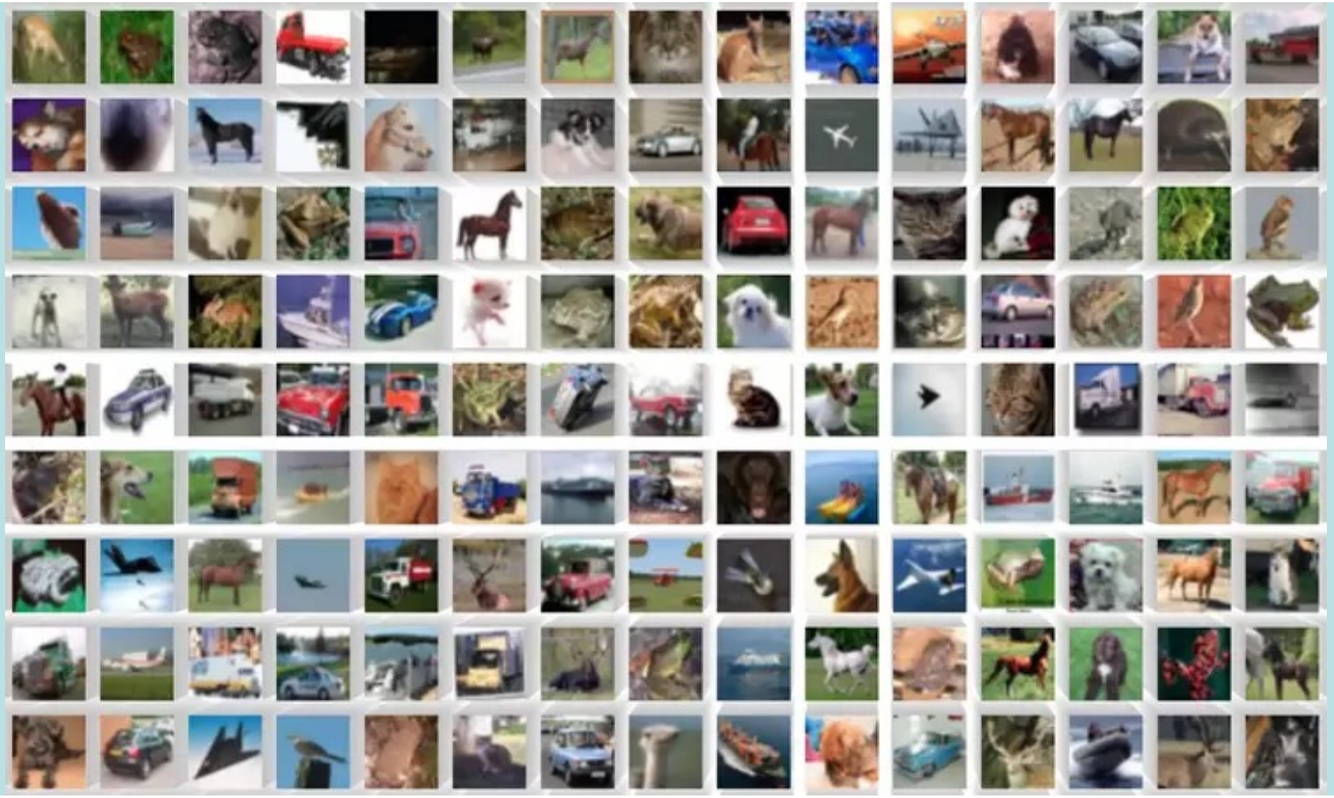
MC1: Hyperparameter & Model Tuning

Deep Learning SGDS

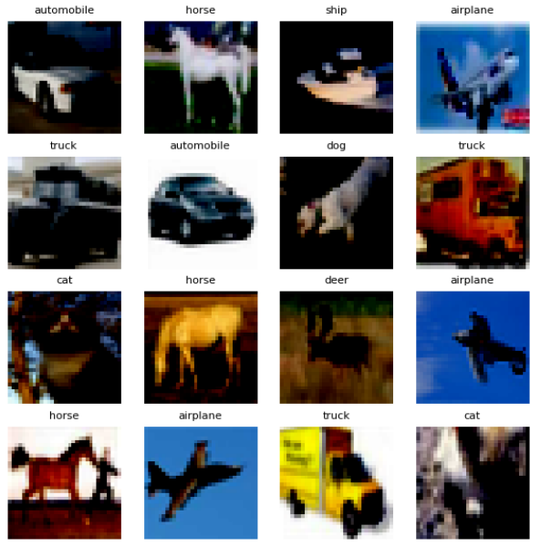


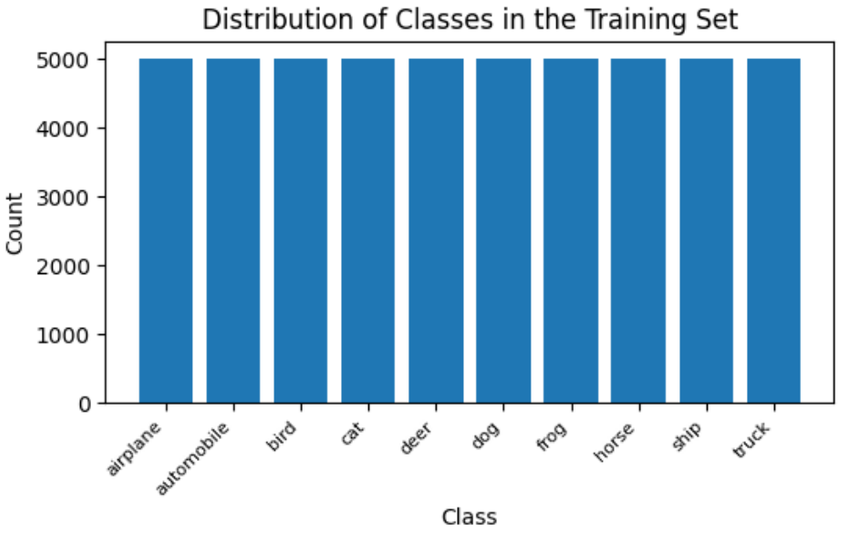
Overview

- Tools & Dataset
- Neural Networks Architecture & Results
- Summary & Conclusion

Tools & Dataset

- Pytorch for Model Training & Evaluation
- Weights & Biases for Logging
- Run on local M1 GPU
- CIFAR10 containing 50’000+10’000 32x32 color images
- 10 evenly distributed classes
- Cross Entropy Loss Function
- Accuracy for Evaluation



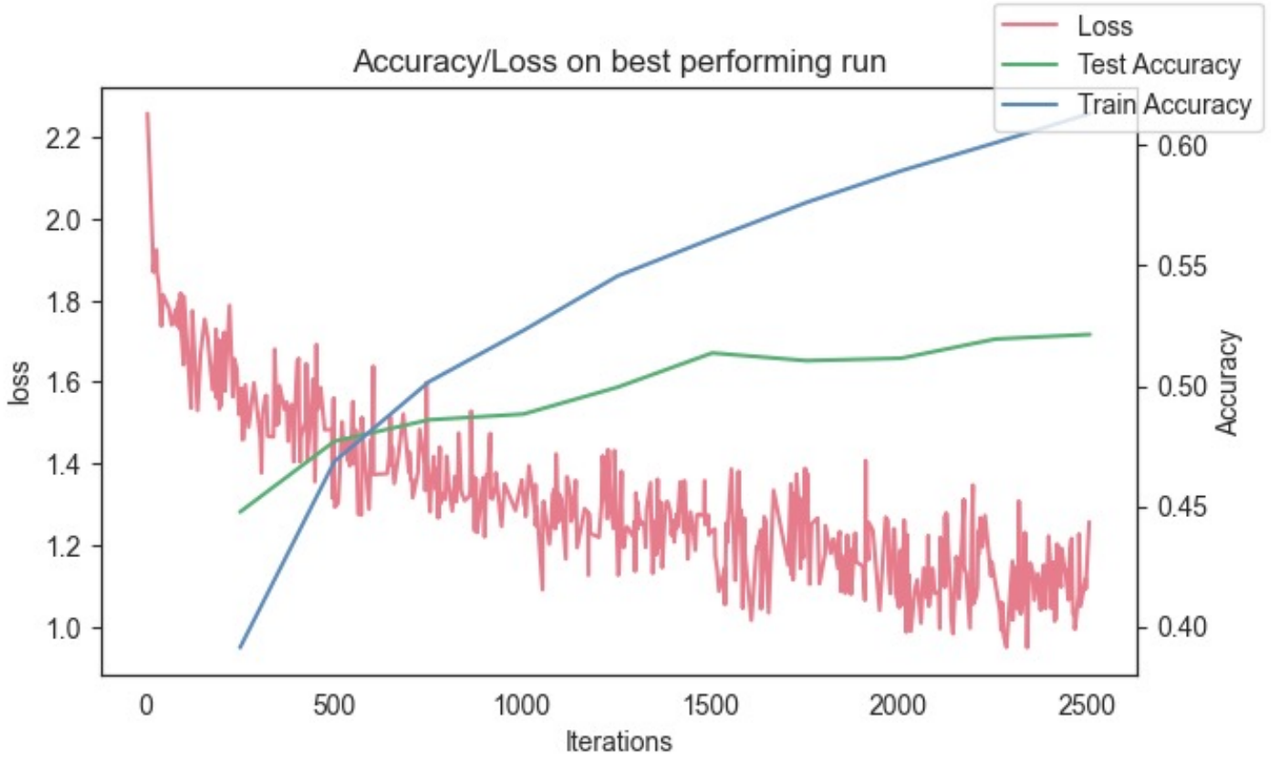


Neural Networks Architecture & Results

- Baseline Architecture
- MLP with Layers of size 128 and 64
- ReLu activation function
- Initialization of weights using Kaiming Initialization
- Batch Size of 20,50,100,200
- Learning Rate of 0.01, 0.1, 0.2

MLP Baseline

- Best Test Accuracy of **0.5214**
- Learning Rate: 0.1
- Batch Size: 200
- 402’250 Parameters

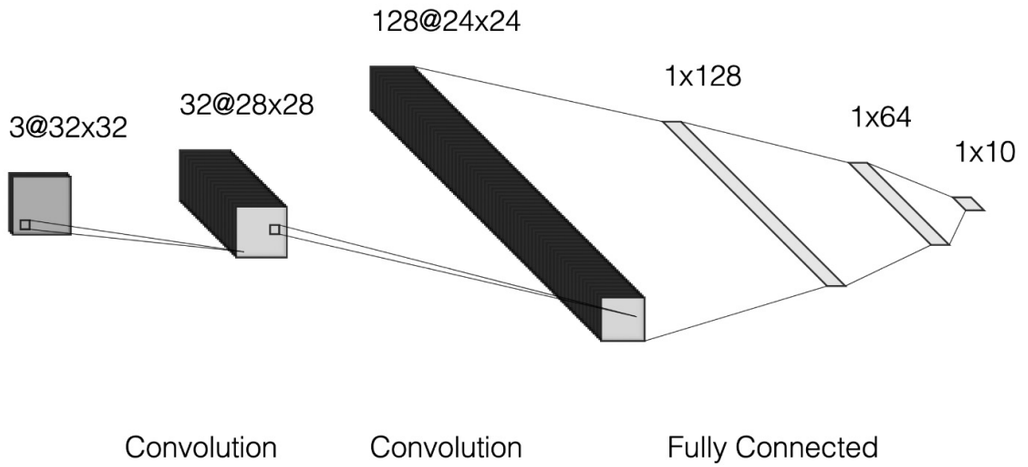


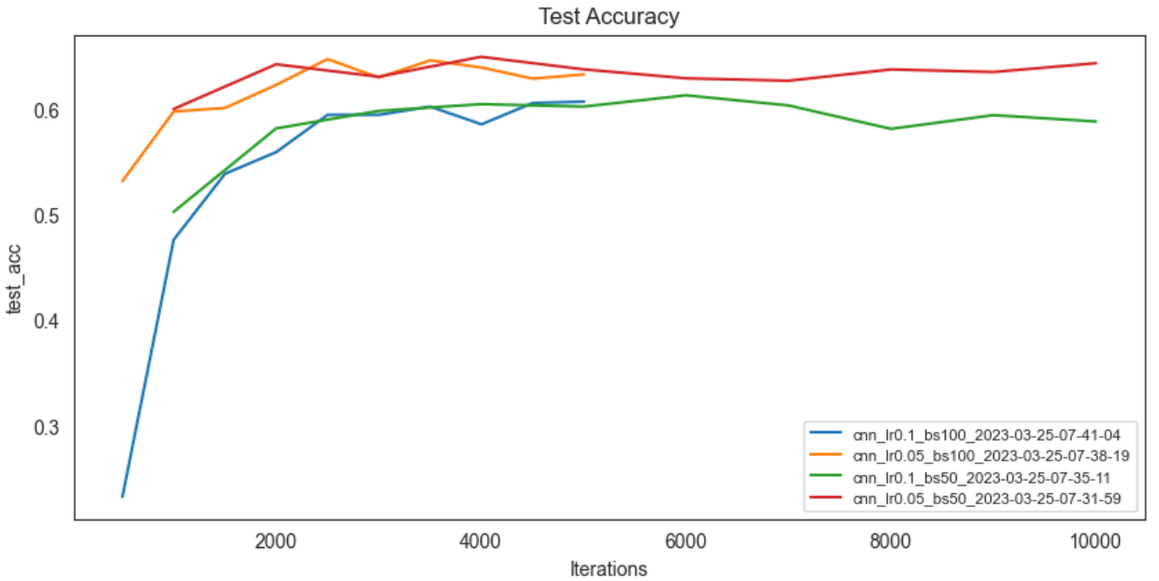
Added additional layer of 256

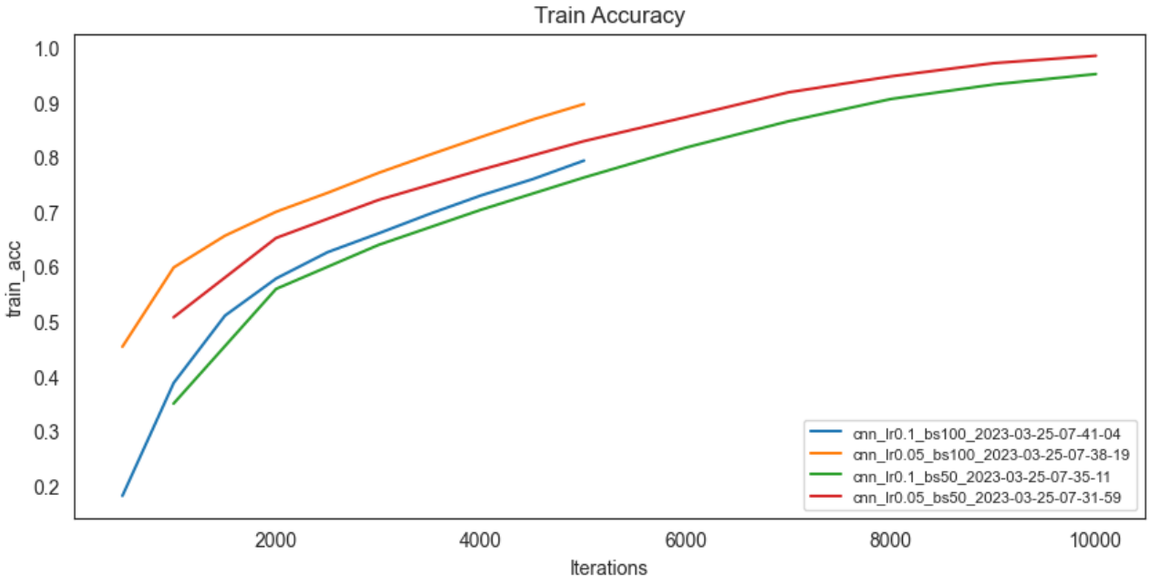
- Double the parameters
- Accuracy increase by 0.011

CNN Baseline

- Adding 2 conv. layers of size 32 and 128
- Kernel size 5
- Accuracy **0.64**

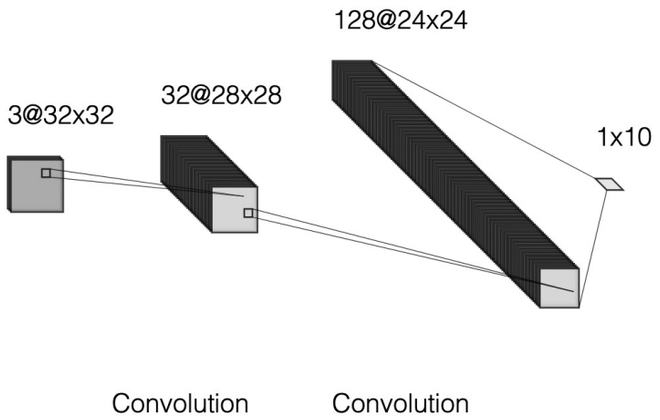


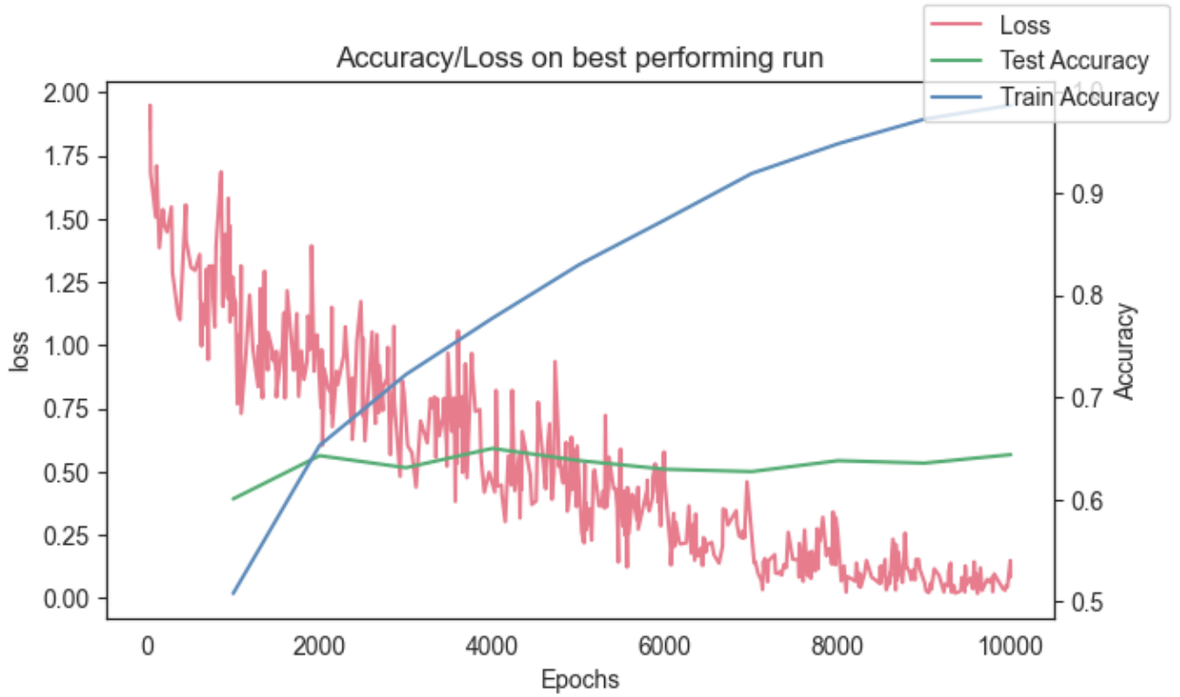




CNN – Remove Fully Connected Layers

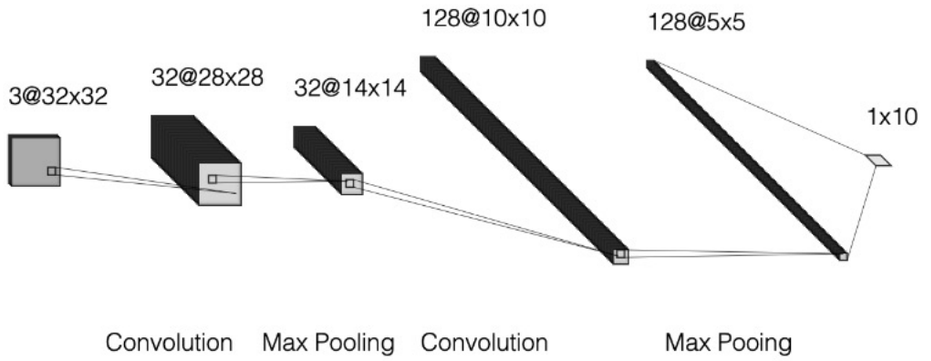
- Accuracy remained the same (0.64)
- Only 10% of parameter to previous model

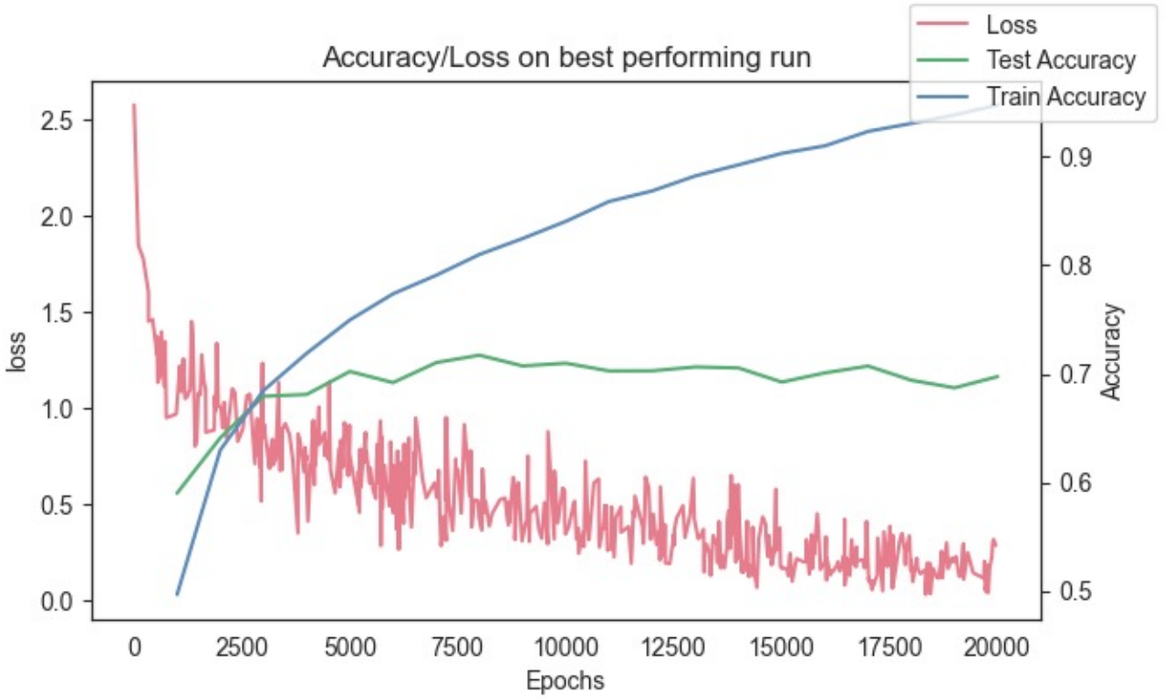




CNN – Max Pooling

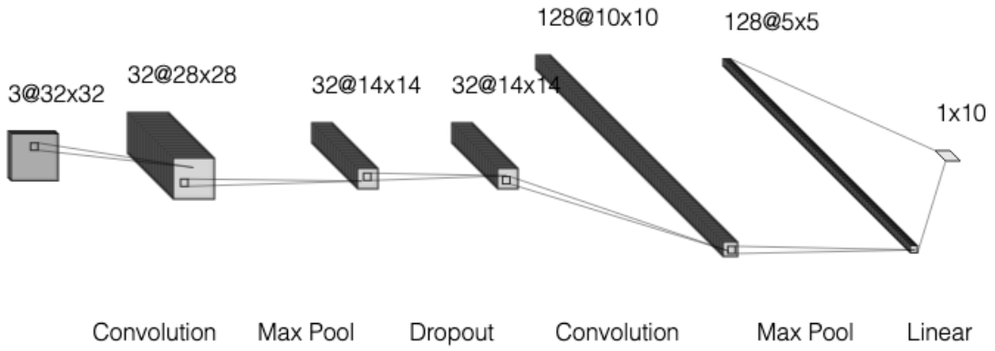
- Max Pooling of 2x2
- Accuracy of 0.69 (+0.05 to prev.)
- 136’970 Parameters (decrease again by 80%)





CNN – Dropout Regularization

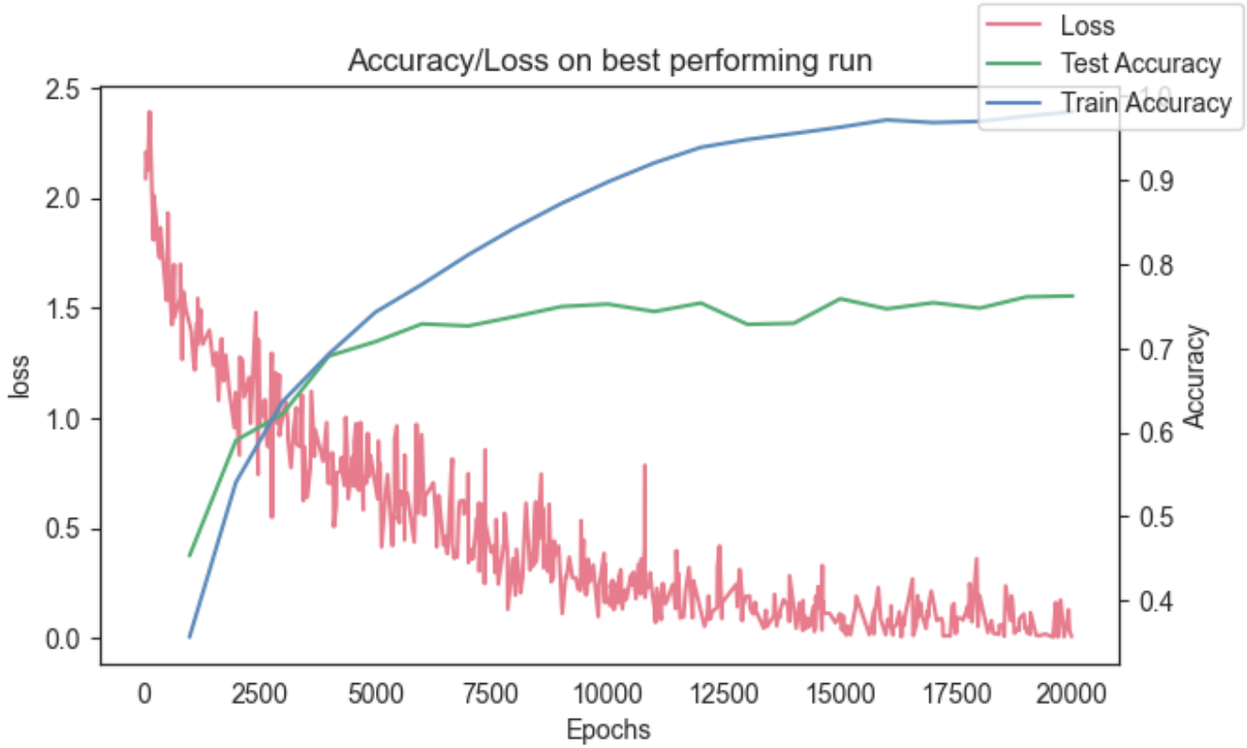
- Droupout after the first Conv. Layer
- Try to reduce overfitting of training data
- Accuracy of **0.714**



CNN – Increase layers

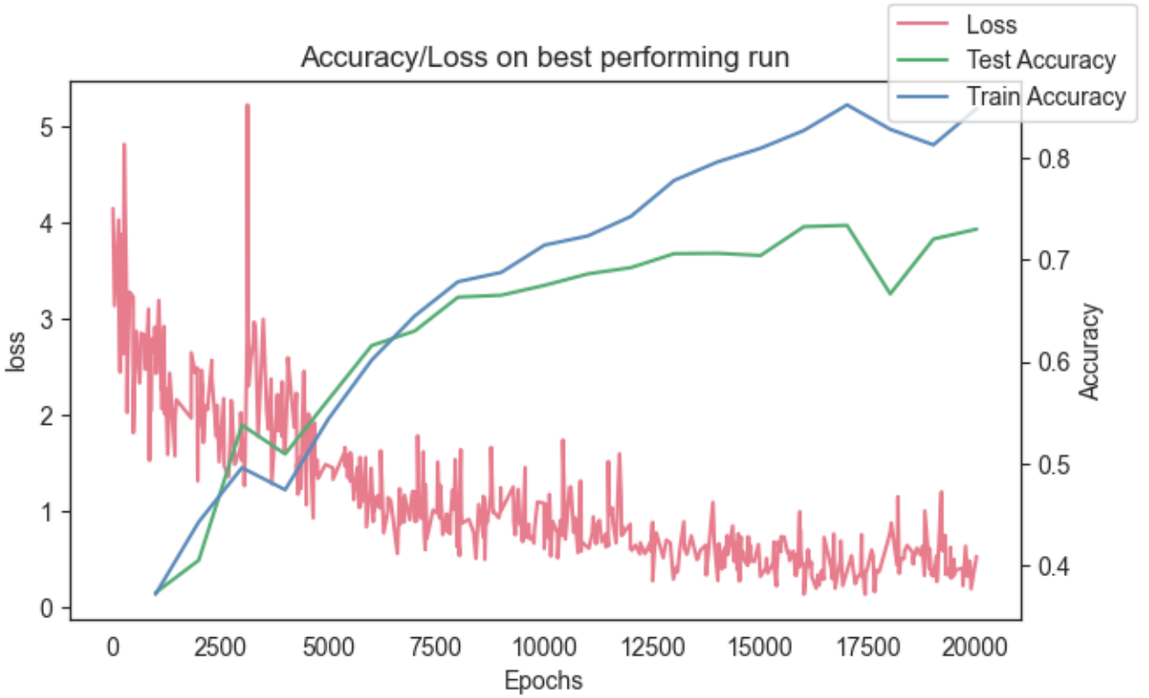
- Adding more Convolutional Layers
- Adding one linear layer
- Accuracy of **0.76**
- Decreasing kernel size didn't have a positive effect

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 30, 30]	4,864
Dropout-2	[-1, 64, 15, 15]	0
Conv2d-3	[-1, 128, 13, 13]	204,928
Conv2d-4	[-1, 256, 10, 10]	819,456
Conv2d-5	[-1, 256, 8, 8]	1,638,656
Linear-6	[-1, 4096]	51,384,320
Dropout-7	[-1, 4096]	0
Linear-8	[-1, 10]	40,970



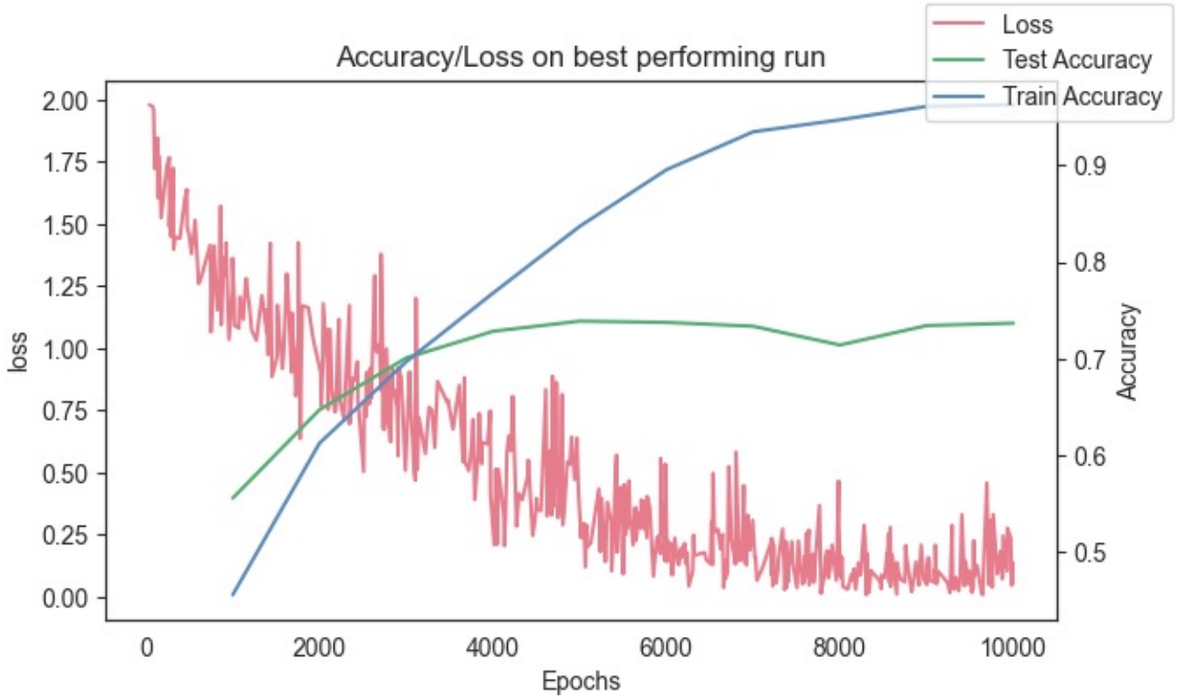
CNN – Batch Normalization

- Added batch normalization after each layer
- Expectation: Faster loss reduction; Batch size can be increased; Faster Training
- Assumtion: Network is not large enough to see effect of batch norm
- Accuracy: **0.73**



CNN – ADAM

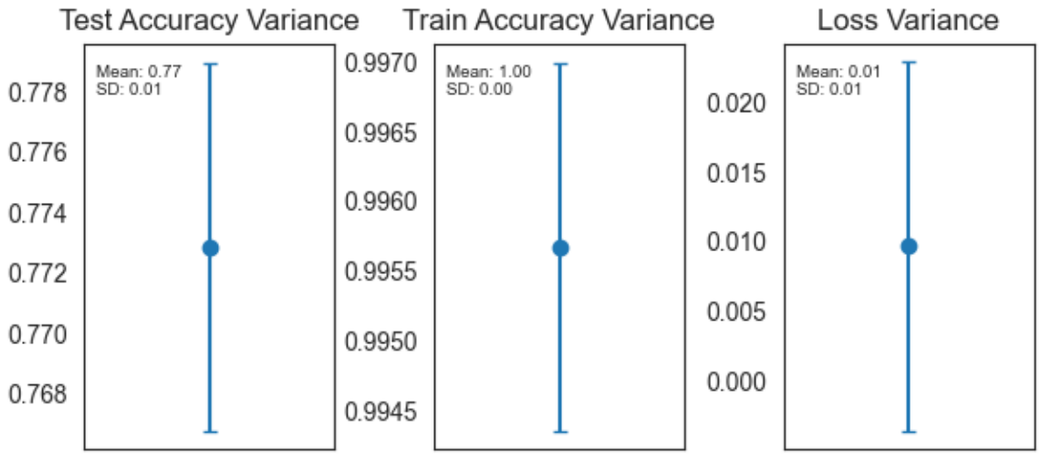
- Accuracy: 0.73
- ADAM is capable of finding local minimas in «deep valley» which is not possible with SGD



Results Overview

Type	Linear Layers	Convolutional Layers	Accuracy
MLP	128, 64		0.5214
MLP	256, 128, 64		0.532
CNN	128, 64	6, 16	0.5607
CNN (increase Conv Layers)	128, 64	32, 128	0.6529
CNN (remove linear Layers)	-	32, 128	0.6438
CNN (Max Pooling)	-	32, 128	0.6972
CNN (Max Pooling, Dropout)	-	32, 128	0.7142
CNN (Max Pooling, Dropout)	4096	64, 128, 256, 256	0.7621
CNN (Max Pooling, Dropout, adjusted Stride, Padding, Kernel Size)	4096	64, 128, 256, 256	0.7653
CNN (Max Pooling, Dropout, Batch Norm)	4096	64, 128, 256, 256	0.7302
CNN (ADAM Optimizer)	4096	64, 128, 256, 256	0.7366

Cross Validation / Variance



END