M124 Coursework 4

Vangelis Tsiatouras
cs2190021@di.uoa.gr

Jun 7, 2021

Neural Networks

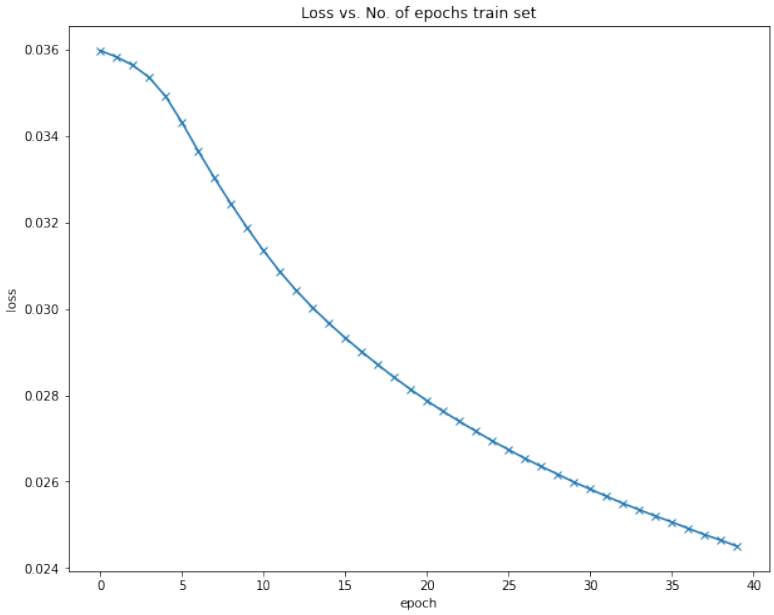
In this task we have to classify the images of CIFAR-10 dataset into 10 different classes. In order to achieve this it was required to implement two different types of neural networks; one fully connected network with 4 layers and one convolutional network with 2 convolutional layers and 3 fully connected layers.

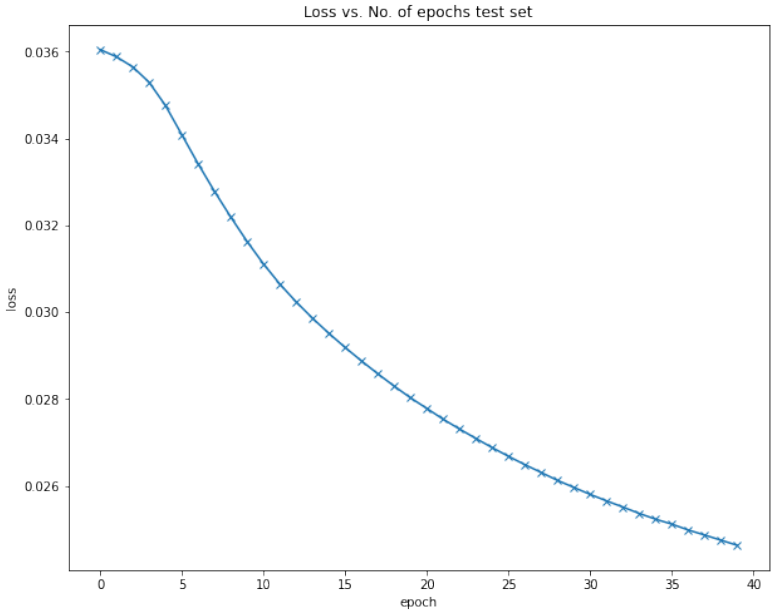
Regular Neural Networks

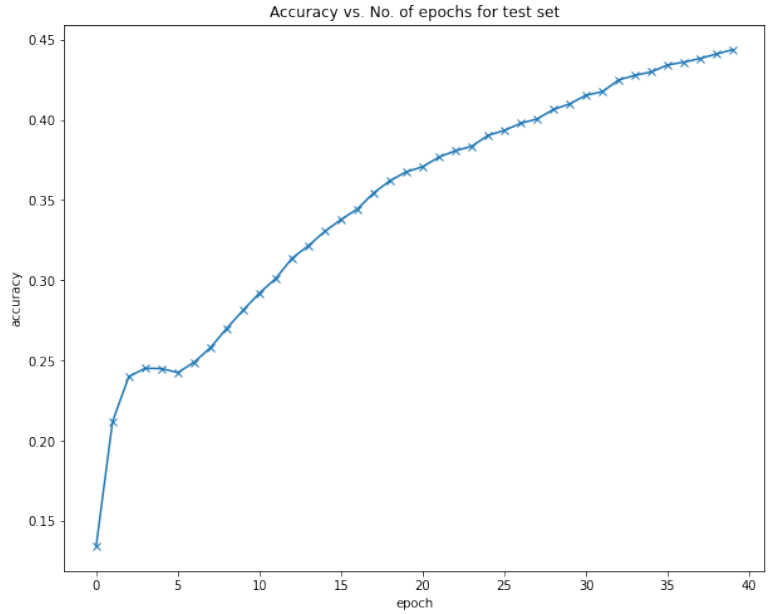
In this section we will report the usage and the performance of a simple fully connected neural network of 4 layers. Initially, lets define the architecture of the network that was used in the first experiments.

Layer	Input	Output
1	$3 \times 32 \times 32$	512
2	512	512
3	512	256
4	256	10

With learning rate of 0.001, momentum 0 and for 40 epochs we extracted the results below.

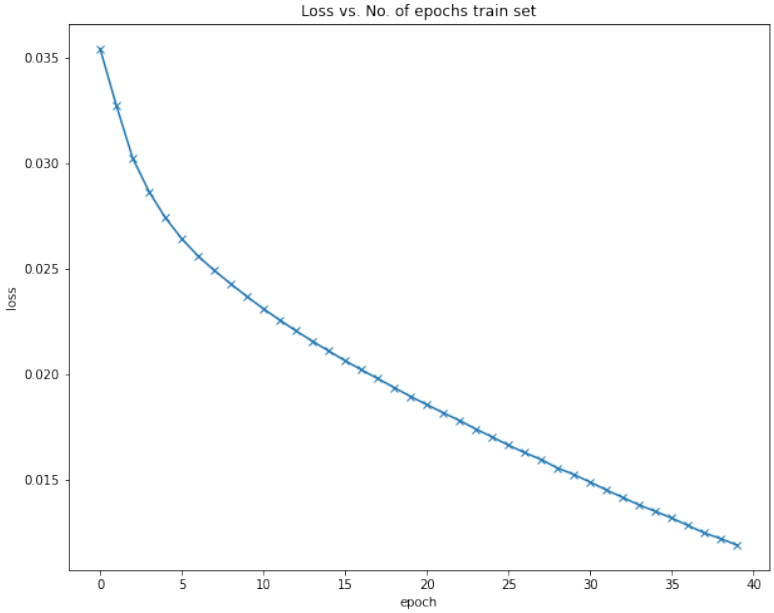


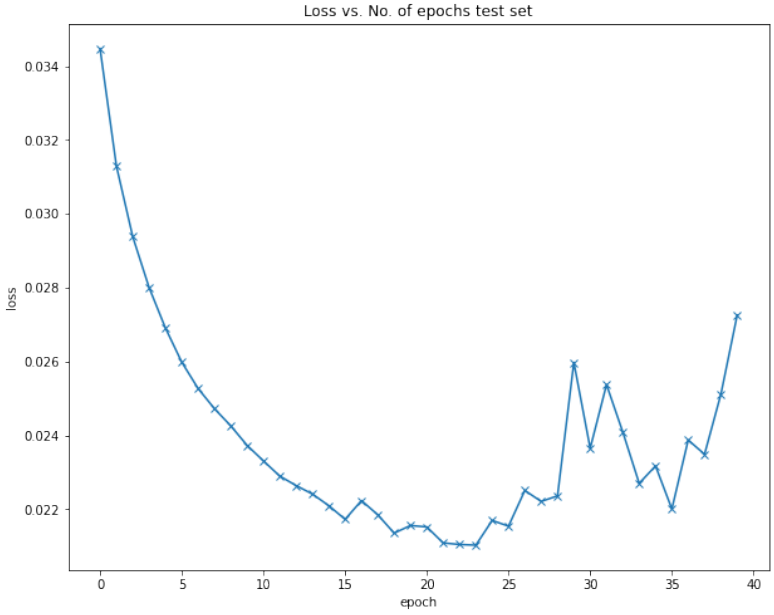


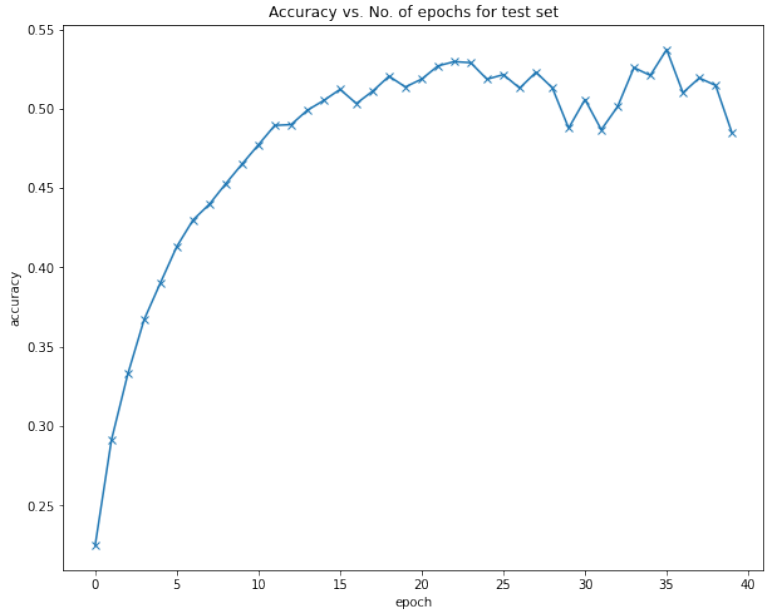


As it can be seen, the model didn't converged yet after 40 epochs and the accuracy is pretty low, approximately 44%. Due to the low learning rate and accuracy we will increase the learning rate in our next model.

With learning rate of 0.005, momentum 0 and for 40 epochs we extracted the results below.

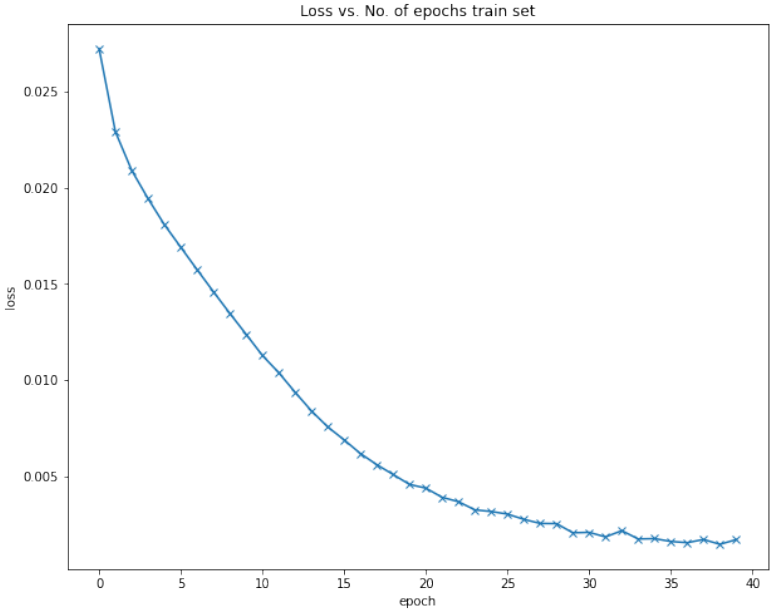


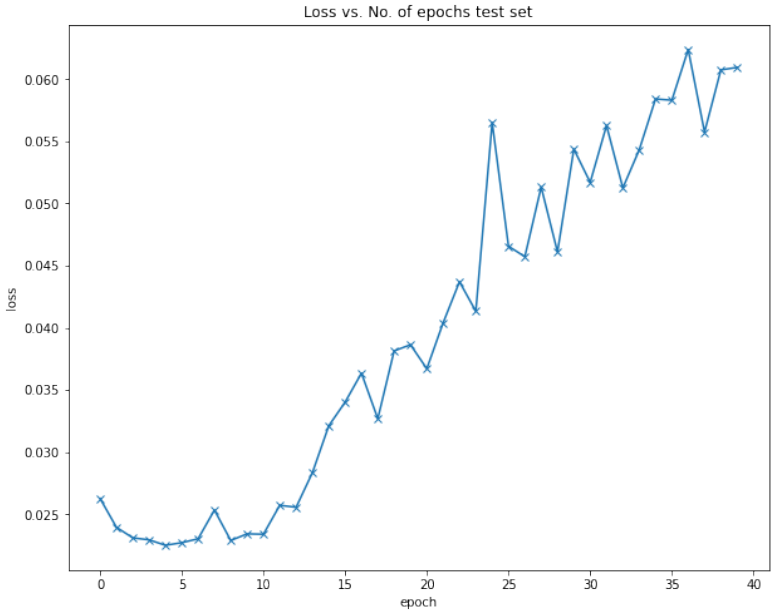


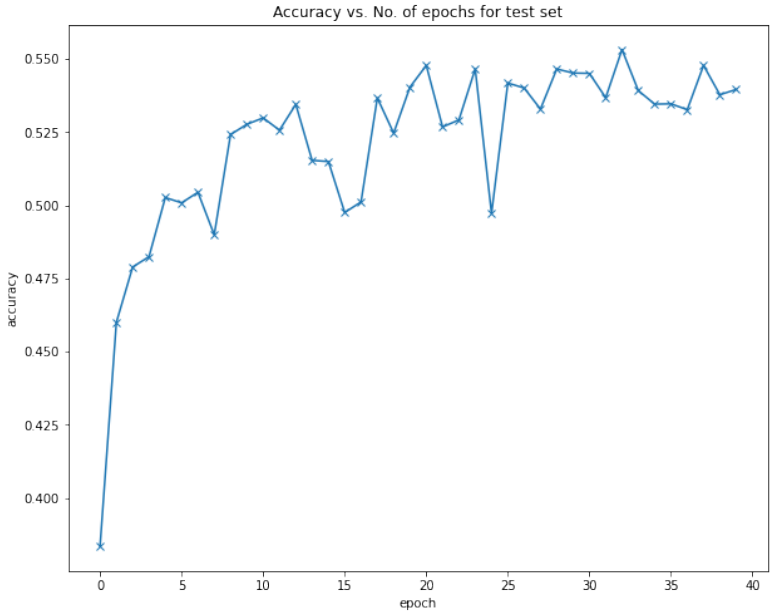


Much better results here. The model seems to be converged exactly at 40th epoch. A minor increase at the loss of test set, although it is very minimal and can be consider acceptable. The achieved accuracy is 48.5%.

With learning rate of 0.1, momentum 0 and for 40 epochs we extracted the results below.



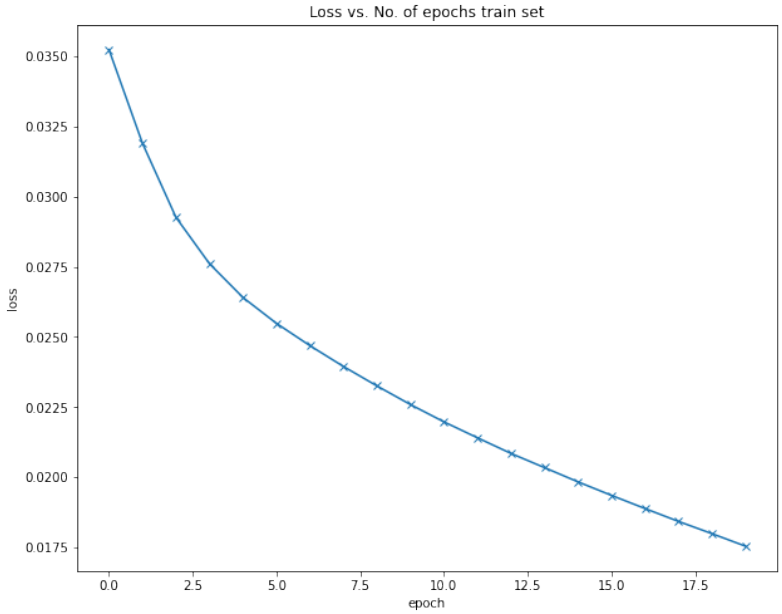


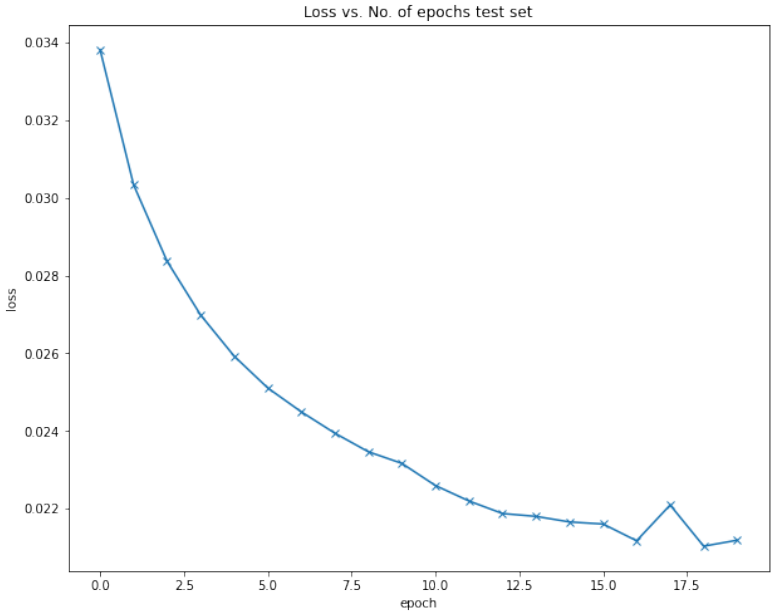


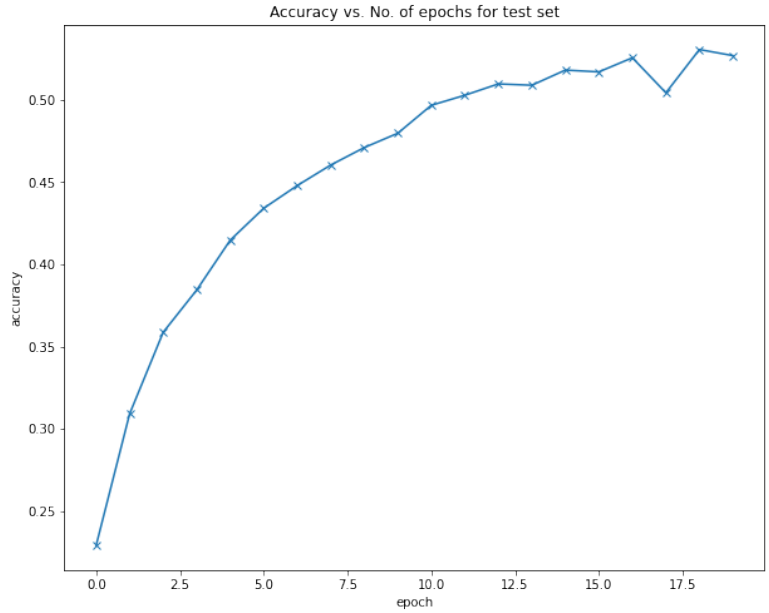
Here the things are starting to get pretty weird. First looking only the loss diagram for training set, it is noticed that after 25 epoch our model it is not learning anything new. Also, at the same epoch the test loss gets increased (still pretty low though, under 0.1). Although, the accuracy after epoch 10 is not stabilized, it is increased and dropped constantly. So, we are safe to conclude that this model may be yield better accuracy after epoch 40, but it doesn't generalize well because of the instability of accuracy results.

Using momentum

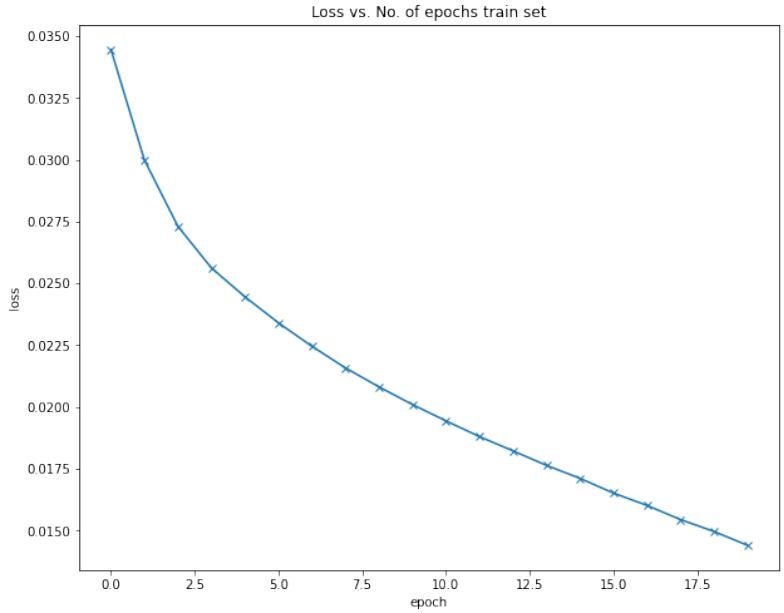
With learning rate of 0.005, momentum 0.2 and for 20 epochs we extracted the results below.

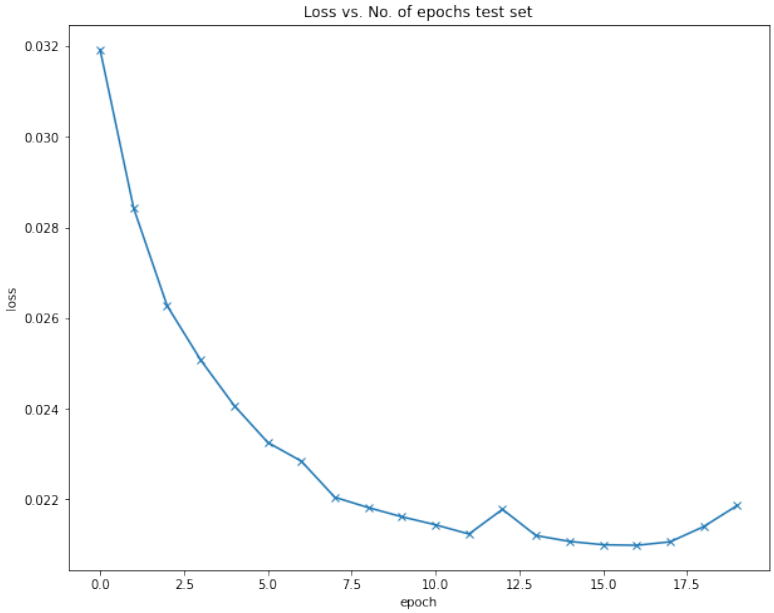


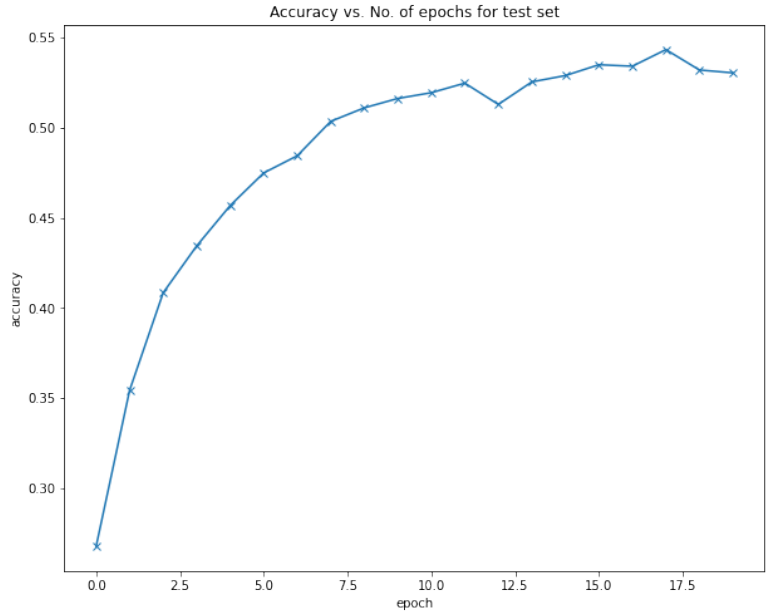




With learning rate of 0.005, momentum 0.5 and for 20 epochs we extracted the results below.







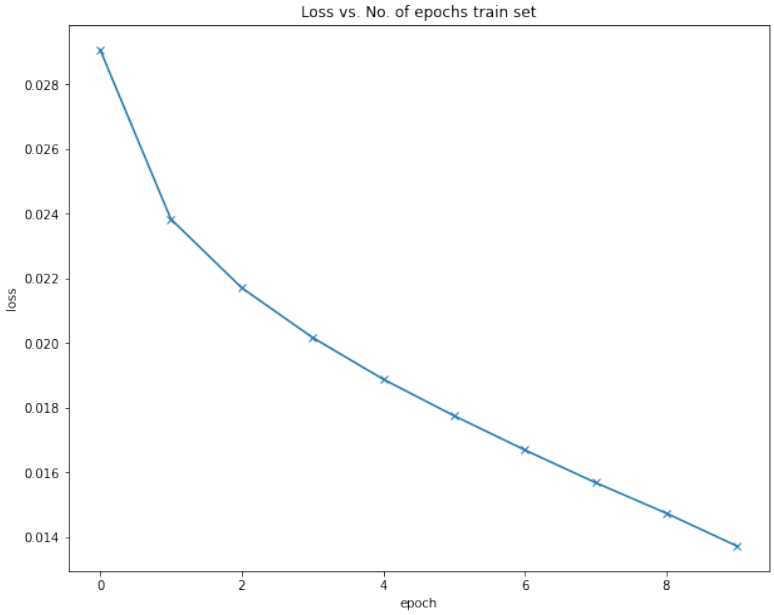
Both models show significant increased accuracy with low loss. Also it is important to mention that these models use fewer epochs, compared to the previous ones, in order to converge.

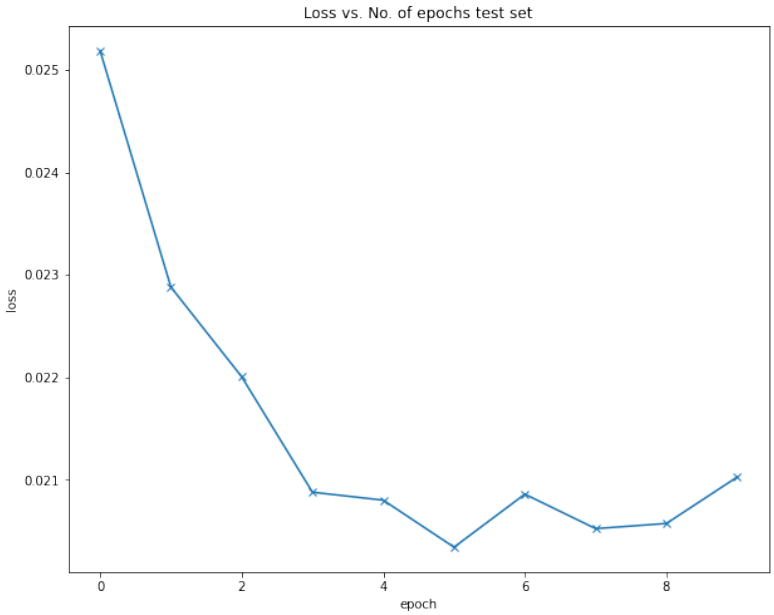
Using different Neural Network architectures

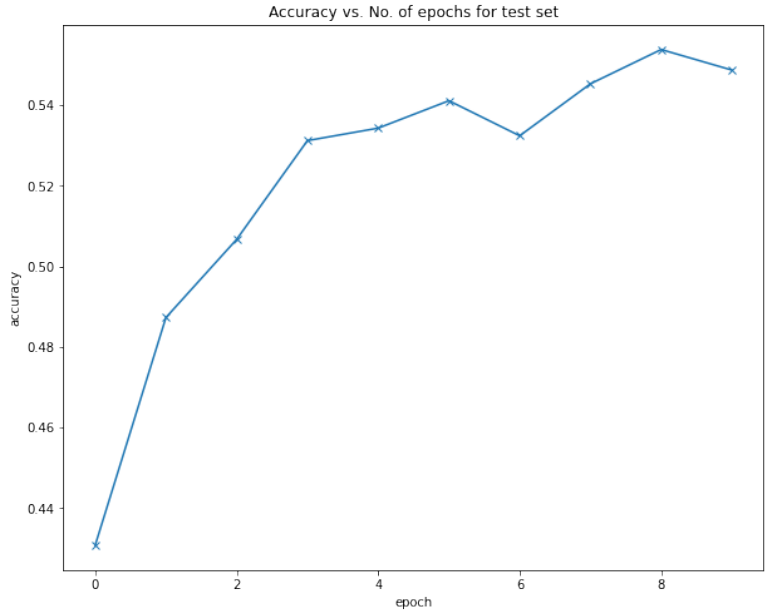
Model 1

Layer	Input	Output
1	$3 \times 32 \times 32$	512
2	512	512
3	512	512
4	512	10

With learning rate of 0.005, momentum 0.9 and for 10 epochs we extracted the results below.



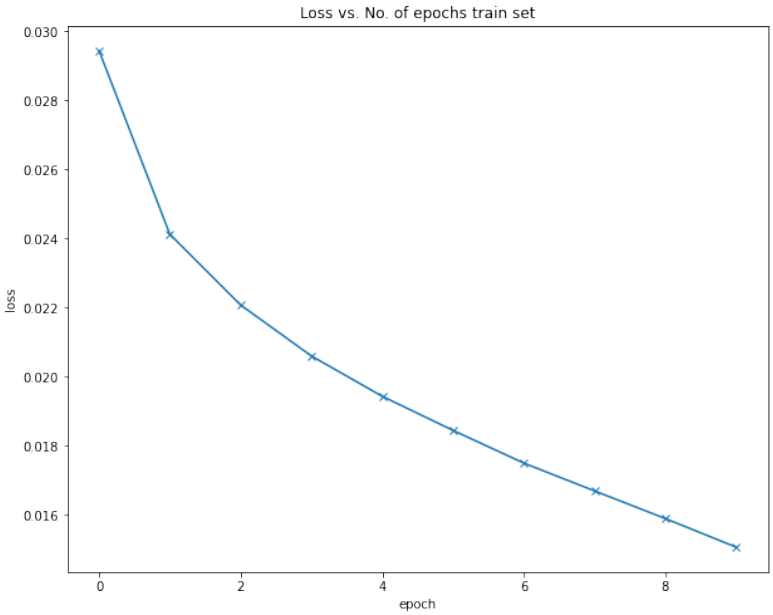


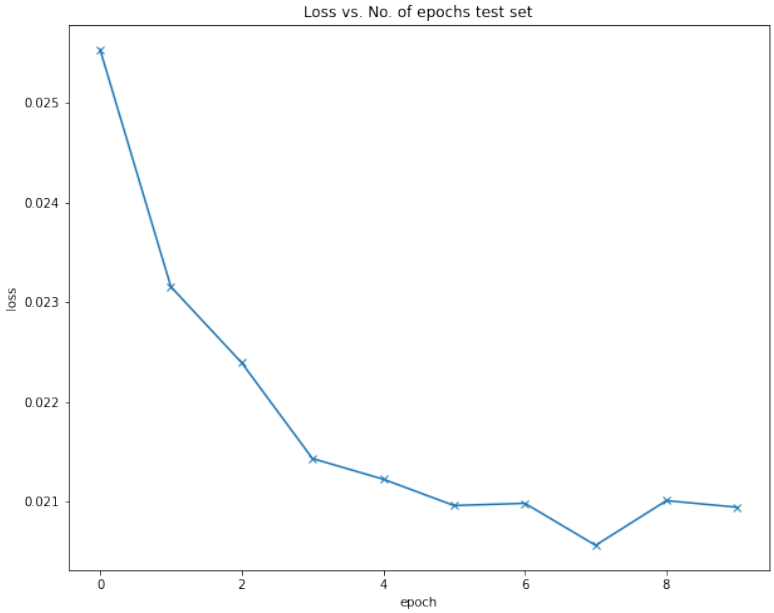


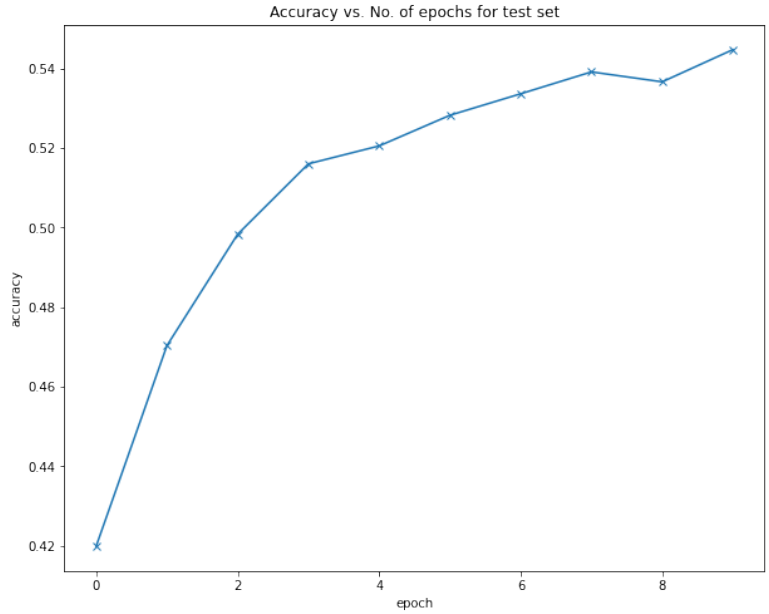
Model 2

Layer	Input	Output
1	$3 \times 32 \times 32$	256
2	256	256
3	256	256
4	256	10

With learning rate of 0.005, momentum 0.9 and for 10 epochs we extracted the results below.







Both models yield very good results (54-55% accuracy). Although, model 2 uses much fewer neurons and has slightly lower loss.

Parameters & Structure of Model 2

```
NeuralNetwork(
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (linear_relu_stack): Sequential(
    (0): Linear(in_features=3072, out_features=256, bias=True)
    (1): ReLU()
    (2): Linear(in_features=256, out_features=256, bias=True)
    (3): ReLU()
    (4): Linear(in_features=256, out_features=256, bias=True)
    (5): ReLU()
```

```
    (6): Linear(in_features=256, out_features=10, bias=True)
  )
)
Layer: linear_relu_stack.0.weight | Size: torch.Size([256, 3072]) |
Values : tensor([[-0.0080, -0.0160, -0.0078,  ...,  0.0095, -0.0156, -0.0174],
        [-0.0093, -0.0078,  0.0013,  ..., -0.0067, -0.0006, -0.0053]],
       device='cuda:0', grad_fn=<SliceBackward>)

Layer: linear_relu_stack.0.bias | Size: torch.Size([256]) |
Values : tensor([-0.1223,  0.0239], device='cuda:0', grad_fn=<SliceBackward>)

Layer: linear_relu_stack.2.weight | Size: torch.Size([256, 256]) |
Values : tensor([[-9.7534e-02,  3.8644e-02,  1.5474e-02,  1.3337e-03, -1.7496e-02,
         -5.7189e-02,  1.5584e-02, -1.3012e-02, -7.9383e-02,  2.0850e-02,
         -5.3131e-02, -7.2779e-02,  1.6253e-03, -3.0375e-02,  8.3476e-03,
         -3.3320e-02, -1.4722e-03,  4.7098e-02, -1.3773e-01,  2.5736e-02,
         -8.4082e-02,  9.4015e-02,  2.6907e-02,  3.0452e-02,  7.3696e-03,
          9.0398e-02, -6.5945e-02, -1.5053e-02, -6.7345e-02,  1.8929e-02,
          5.7111e-02, -2.2698e-02, -1.6301e-02, -2.4105e-02,  3.0407e-02,
         -3.2220e-02, -6.6195e-02, -7.4514e-02, -7.3647e-02, -8.3282e-02,
          2.3541e-02, -9.5963e-02, -1.1357e-02,  1.1110e-03,  6.3638e-02,
          1.0204e-03, -4.9167e-03, -1.4535e-02, -3.4906e-02,  2.4007e-02,
         -1.1595e-02, -1.8922e-02, -1.4080e-02, -5.4389e-03, -2.0153e-02,
         -1.1484e-01,  5.7974e-03, -1.5413e-02, -2.4864e-02,  1.7753e-02,
         -9.3060e-03, -8.0536e-03,  4.8404e-02,  2.2706e-02,  1.9977e-03,
         -1.5054e-01, -8.2152e-02,  5.1631e-03,  1.8237e-02,  7.1819e-02,
         -6.4134e-02,  5.1304e-03,  1.1996e-02, -5.6003e-02, -1.4776e-02,
          2.9870e-02,  4.2117e-02, -7.9605e-02,  6.8267e-02,  4.0236e-02,
         -8.5710e-03,  7.9148e-02,  1.0025e-02,  9.8662e-02,  4.8162e-02,
          9.5980e-03, -3.3493e-02, -2.1841e-02,  4.5546e-02,  7.2332e-02,
         -1.0785e-01,  5.8568e-03, -9.6206e-02, -5.0557e-02,  7.5758e-02,
          3.3499e-02, -6.5989e-02,  1.7522e-02,  5.4170e-02,  3.3046e-02,
         -5.1036e-02,  3.8568e-02, -6.8768e-02,  6.7419e-02, -7.0133e-02,
         -3.8767e-02,  4.0130e-02, -4.0568e-02, -7.7361e-02, -9.0302e-03,
          5.5867e-02,  5.0301e-02, -5.3556e-02,  3.6176e-03, -7.5198e-02,
         -8.6444e-02,  1.7042e-02, -6.3200e-02,  2.8703e-02, -3.3525e-02,
         -9.6322e-04, -5.8631e-02,  4.9685e-03, -1.6049e-02, -5.0778e-02,
          2.2719e-02, -2.5176e-02,  4.1991e-02,  7.3274e-02,  6.3264e-02,
          3.6763e-02, -2.9625e-02,  1.1636e-02,  1.3075e-02, -4.1778e-02,
          2.5655e-02, -1.0645e-01, -5.0733e-02, -2.8984e-02,  7.6907e-02,
         -6.8717e-02, -4.3484e-02, -2.6556e-02,  4.6903e-03, -3.1186e-02,
         -6.6008e-02,  7.9774e-02, -5.1618e-02,  9.5221e-02,  1.3957e-01,
         -4.2405e-03, -6.1760e-02, -3.0644e-02,  1.5218e-02,  1.1177e-01,
         -8.8837e-02,  6.8103e-03,  4.1482e-02, -1.6482e-02,  2.4143e-03,
          1.6744e-02,  5.2435e-02,  4.0706e-02, -1.5204e-02,  2.8412e-02,
          9.6491e-02, -3.1176e-02, -4.6111e-02,  2.4100e-02, -2.9350e-02,
          2.4993e-02,  6.4430e-02, -6.8131e-02, -2.0987e-02, -3.8314e-03,
          5.8650e-02, -4.5894e-02, -6.3041e-02,  4.1060e-02, -1.1087e-02,
          1.6940e-02,  2.7768e-02,  2.0722e-02,  4.9129e-02,  3.1684e-02,
         -8.6163e-02, -7.8687e-03,  2.4793e-02,  2.8889e-02,  4.9197e-02,
          6.5534e-02, -3.0960e-02, -4.0161e-02,  3.6407e-02, -1.4910e-02,
         -4.1086e-02,  1.5066e-02, -7.8714e-02,  2.7492e-02, -2.5275e-02,
         -1.6070e-02, -2.6640e-03, -6.8542e-02, -9.9332e-02,  2.2121e-02,
          3.1033e-03,  7.3896e-02,  6.8659e-02, -2.7359e-02, -3.1838e-02,
         -4.9435e-03, -1.1361e-02, -1.7937e-02,  1.4756e-02,  4.3467e-02,
         -1.1290e-01,  4.3941e-02,  3.4373e-03,  7.5076e-04,  1.3226e-02,
         -8.5801e-02,  1.0436e-02, -2.3772e-02, -6.4388e-03,  6.6940e-02,
         -5.7332e-02, -3.1610e-02,  1.0020e-01,  1.6130e-02, -4.0661e-02,
          5.5709e-02, -1.9392e-03, -9.9115e-02, -8.6887e-02, -2.2912e-02,
```

```
        -1.9881e-02,  1.6626e-03,  2.1535e-02,  3.1849e-02, -1.6014e-02,
        -6.3736e-02,  3.5986e-03,  8.9359e-02,  2.7874e-02, -4.5243e-02,
        -1.0221e-01,  1.7042e-02,  7.4243e-02,  3.2369e-02, -4.7487e-02,
        -6.5552e-02, -3.1043e-02,  3.2060e-02, -3.9953e-02,  9.0756e-02,
         3.3728e-02],
       [-4.9544e-03,  2.5412e-02, -1.0467e-03,  4.6002e-02,  3.3579e-02,
        -6.0399e-02,  6.7784e-02,  6.7397e-02,  1.3069e-02,  2.7377e-02,
        -8.0438e-02,  6.6454e-02, -4.0037e-02, -7.1357e-02, -5.7782e-03,
        -2.1622e-02,  4.8266e-02, -1.6776e-02, -1.5377e-02, -3.2074e-02,
        -7.8949e-02, -7.2614e-02,  5.6913e-02,  2.0063e-02, -2.9097e-02,
        -1.9080e-02,  3.6357e-02,  3.1460e-02,  7.0117e-02, -5.8871e-03,
         8.9051e-03,  9.7176e-03, -5.0935e-02,  3.2898e-04, -3.2202e-02,
        -3.3886e-02,  1.4996e-02,  4.1392e-02, -2.7345e-03, -1.4031e-02,
         4.4787e-02,  8.4784e-02,  2.3668e-02,  1.0655e-02, -1.4942e-02,
         2.8500e-02, -4.2024e-02, -5.9177e-03,  2.1382e-02, -6.6796e-02,
        -6.1542e-02, -6.6550e-02,  5.8421e-04,  1.2505e-02, -3.2381e-02,
        -3.6788e-05,  2.0196e-02,  7.7132e-02, -7.5778e-03,  9.2857e-03,
        -5.4176e-02,  1.3325e-03,  8.6710e-03, -6.6819e-02,  4.1780e-02,
        -1.2201e-03,  4.7864e-02, -5.9170e-02,  5.9393e-02, -4.9819e-02,
        -4.1783e-03, -2.5934e-02, -3.2215e-02,  3.3673e-02,  7.4453e-03,
         3.5756e-02,  3.8233e-02, -1.0265e-02, -3.2035e-02, -7.8619e-02,
        -5.2725e-02, -5.2916e-02,  1.3210e-02,  3.0154e-02, -3.3004e-02,
        -5.2641e-02,  1.0552e-02, -2.2826e-02,  7.5695e-02,  3.9069e-02,
        -6.2615e-02, -5.4530e-02,  7.2599e-02,  6.9074e-02,  4.5483e-02,
         4.1136e-03, -1.6302e-02, -6.6001e-02, -5.9871e-02, -4.8256e-02,
         5.5603e-03, -2.3998e-02, -3.4072e-02,  6.8036e-02, -4.1925e-03,
         3.4313e-02, -2.0698e-02, -3.5318e-03,  7.5283e-02, -5.4609e-02,
        -8.2217e-02, -5.9376e-02, -7.0741e-02,  2.5456e-02, -6.7390e-02,
        -6.3828e-02, -5.5151e-02, -7.7274e-02,  6.3541e-02, -2.1164e-02,
         9.2444e-02, -1.0186e-02, -1.6843e-02, -2.5397e-02,  3.2617e-02,
        -2.7204e-02, -9.1103e-03, -2.8299e-02,  1.1994e-02, -5.8573e-02,
         1.0705e-01,  1.7745e-03,  4.5733e-02,  4.0037e-02, -8.3288e-02,
        -5.1191e-02,  5.4136e-02, -3.8126e-03,  1.0227e-01,  3.0605e-02,
         4.2224e-02, -2.4213e-02, -4.9455e-02, -1.0587e-03, -4.0519e-02,
        -2.7464e-02, -8.2082e-03,  5.6022e-02, -4.2617e-02,  4.7498e-02,
         1.1560e-02, -7.9669e-02, -4.3946e-02, -1.5358e-02,  2.3710e-02,
         2.8546e-02, -4.1157e-02,  1.0265e-01, -1.7117e-02,  1.9700e-02,
         4.3758e-02, -4.3005e-02,  2.0150e-02, -3.4302e-02,  1.9143e-02,
        -1.5874e-02, -3.4097e-03, -7.3062e-02,  6.1042e-02,  1.3365e-02,
         6.4047e-03,  1.7861e-02, -4.2557e-02, -3.8428e-02, -1.0051e-02,
        -7.9629e-02,  4.8442e-02,  6.3668e-02,  7.1939e-02,  1.9762e-02,
        -5.3948e-02, -2.7576e-02, -6.8336e-02,  3.8941e-02, -4.3065e-02,
        -2.4526e-02, -5.7056e-02, -3.2465e-02, -5.6117e-02, -5.4689e-02,
         6.6972e-02, -2.5624e-02,  1.8059e-02,  3.4518e-03,  1.6543e-02,
        -4.7352e-02,  7.7258e-02, -2.2756e-02,  5.6622e-02,  6.3474e-02,
        -1.6873e-02, -3.6351e-02, -4.4462e-03,  2.3675e-02, -6.4086e-02,
         2.9051e-02, -8.5565e-02,  5.7233e-03,  4.6150e-02, -9.6459e-03,
         2.4288e-02,  3.0083e-02, -2.4591e-02,  1.6678e-02,  1.1529e-02,
         8.7966e-02, -1.7642e-02, -2.2044e-02, -2.3954e-02, -3.6923e-02,
         6.2527e-02,  5.3854e-03, -6.3792e-02,  5.9337e-02,  3.6011e-02,
         8.9410e-04,  4.8738e-02,  5.1241e-03, -2.9666e-02, -2.3300e-02,
        -8.4370e-03,  3.0430e-02,  1.8795e-02,  9.8683e-02,  6.0494e-02,
         5.0046e-02,  6.3881e-03,  5.9392e-03,  2.2400e-02, -1.7033e-02,
         3.5428e-02,  3.1373e-02, -5.7064e-02, -9.4956e-03,  2.9280e-02,
         4.8217e-02, -2.0317e-02, -1.1853e-03, -1.5088e-02,  6.3824e-02,
        -1.6339e-02, -2.8886e-03, -2.1975e-02,  8.3166e-03, -8.4001e-02,
        -2.7628e-02]], device='cuda:0', grad_fn=<SliceBackward>)

Layer: linear_relu_stack.2.bias | Size: torch.Size([256]) |
```

```
Values : tensor([0.3242, 0.1284], device='cuda:0', grad_fn=<SliceBackward>)

Layer: linear_relu_stack.4.weight | Size: torch.Size([256, 256]) |
Values : tensor([[ 6.5475e-02,  2.1689e-02,  6.2264e-02,  1.5349e-02,  3.4751e-02,
          3.6582e-02,  4.4164e-03, -4.5358e-02, -5.9497e-02,  2.4798e-02,
          4.4571e-02,  2.5079e-02,  7.6626e-03,  3.5595e-02,  2.0390e-02,
          5.4797e-02, -3.8524e-02,  1.9002e-03,  4.1051e-02,  1.0170e-02,
         -7.1793e-02,  1.8636e-02, -6.9012e-02,  6.4916e-02, -1.0923e-02,
         -1.3754e-02, -2.0203e-02, -6.5958e-03, -7.9364e-02, -8.1307e-03,
          3.1054e-02, -4.2885e-02,  2.6079e-02, -2.4215e-02, -1.2418e-02,
          3.4488e-02,  2.1412e-02,  5.9925e-02,  4.3921e-02, -5.3078e-03,
         -6.6985e-02, -1.7140e-02, -1.7205e-02, -7.2449e-02,  3.8688e-02,
         -2.1954e-02,  5.7907e-02, -5.8633e-02,  2.1126e-02,  4.5090e-02,
          1.5439e-02,  3.3244e-02, -8.9900e-03, -4.1404e-02, -4.0168e-02,
          2.9152e-02, -2.3751e-02,  4.2562e-02, -2.6335e-02,  2.6377e-02,
          9.9797e-02,  6.6389e-03,  3.3051e-02, -3.0088e-02, -1.0755e-02,
         -6.0980e-02, -4.4703e-02, -1.1498e-02, -6.7979e-02,  4.9851e-02,
          6.0864e-02,  6.2772e-02, -8.1709e-02, -7.3764e-02, -3.0996e-02,
          5.8367e-02,  4.5606e-02, -3.4979e-05,  3.3303e-03, -3.4303e-02,
         -2.2909e-02,  1.2094e-01,  4.2353e-02, -5.5755e-02,  5.1593e-02,
          4.0111e-02, -1.8679e-02,  2.7865e-02, -3.6162e-03, -4.0848e-02,
         -4.1058e-02,  2.5321e-02, -1.2797e-02,  2.7930e-02,  2.9206e-02,
         -5.3761e-02, -2.6441e-02, -5.4709e-02, -1.1532e-02,  6.0909e-02,
          5.0452e-02,  2.5450e-02, -2.2613e-02, -9.9204e-03, -1.9267e-02,
         -3.2856e-02, -2.0293e-02, -3.3358e-02,  2.0626e-02,  6.7938e-02,
          3.0848e-02, -1.5960e-02,  3.1098e-02, -8.8271e-02, -2.3700e-03,
         -1.9010e-02,  3.3124e-02, -1.0881e-02,  1.0217e-02,  5.5516e-02,
          3.9283e-02,  2.6314e-02, -1.9154e-02,  3.3046e-02,  2.8375e-02,
          3.3595e-02, -4.1947e-02, -2.7633e-02,  6.8116e-02,  4.4341e-02,
         -7.8427e-02, -3.5293e-02, -5.4839e-02,  2.7426e-02, -3.2433e-02,
          4.7775e-02,  2.6423e-02, -4.1777e-02, -3.1763e-02,  3.1231e-02,
         -3.8563e-02, -3.2844e-02,  5.4877e-02,  2.4917e-02,  3.6127e-02,
          5.3986e-02,  4.1356e-03,  1.9839e-02, -3.6515e-02,  4.4580e-02,
          6.2167e-02,  3.7465e-02, -3.2889e-02, -9.3711e-03, -3.5348e-02,
         -5.1326e-02,  1.5288e-02, -5.6403e-02, -4.5397e-02, -3.1711e-02,
          6.4270e-02, -2.0774e-02, -3.0223e-02,  3.9726e-02,  5.8036e-02,
          3.3163e-02,  1.3491e-03,  2.5625e-03, -3.8236e-03,  3.1634e-02,
         -3.8925e-02, -6.0084e-02, -2.4366e-02,  5.5028e-02, -3.9096e-02,
          1.1313e-02, -4.6563e-02,  7.7658e-03,  2.2289e-03,  1.2023e-02,
          5.5789e-03, -3.9301e-03,  1.5366e-02,  1.8380e-02, -5.8616e-02,
          7.5454e-03,  5.4410e-02, -3.1348e-02,  4.2190e-02, -2.9568e-02,
         -3.3054e-02,  1.8000e-02,  1.3795e-02, -4.5986e-02, -2.7809e-02,
          4.1988e-03, -2.2019e-02,  2.4513e-02, -3.4784e-02, -2.0138e-02,
         -3.2274e-02, -9.5009e-03, -2.6792e-02, -6.8220e-02, -4.3321e-02,
          1.8330e-02,  4.6284e-02,  5.1569e-02,  5.2036e-02,  5.7124e-02,
         -3.2827e-03, -8.1954e-02,  2.4811e-02, -4.1072e-02, -3.7215e-02,
          4.9814e-02, -4.6535e-02,  4.8706e-02, -2.2930e-03, -1.5620e-02,
         -3.8052e-02,  7.0442e-02, -1.0155e-02, -2.7545e-02, -6.3550e-02,
          5.8819e-03, -1.8232e-02, -4.2924e-02,  5.3286e-02,  5.6774e-02,
         -9.5159e-02,  1.0998e-02,  5.8305e-02,  2.2796e-02,  1.9002e-02,
          1.9579e-02,  1.3606e-02, -3.4262e-02,  1.2215e-02, -2.0437e-02,
          6.1526e-02, -7.2449e-03,  5.8147e-02,  3.8666e-02,  4.9544e-02,
          2.4447e-02, -2.7819e-02, -4.3814e-02, -2.3242e-02,  1.1990e-02,
         -2.0610e-02,  9.6210e-02,  5.5554e-02, -5.7511e-02,  8.0394e-02,
          7.1270e-02],
        [-2.0543e-02,  9.0369e-03,  3.6379e-02,  4.1339e-02, -5.3090e-02,
          1.7436e-02, -1.5313e-02, -5.9357e-03,  4.6669e-02,  3.1252e-02,
         -3.7329e-02, -4.5303e-02,  1.4869e-03, -7.0545e-03, -6.4160e-02,
         -4.9870e-02,  5.0483e-02, -2.0211e-02,  5.3524e-02, -4.4171e-02,
```

```
          4.0110e-04, -2.1049e-02, -6.1635e-02, -1.6148e-03, -3.1603e-02,
          5.3147e-02, -9.7137e-03, -5.6172e-02,  3.8696e-02,  7.7259e-03,
         -2.6432e-02,  6.0433e-02,  5.5322e-03, -5.1102e-02, -5.2334e-02,
         -1.3740e-02, -9.9910e-03, -2.1406e-02, -5.9838e-02, -5.8343e-02,
         -3.1160e-02,  2.0967e-02, -5.5823e-02,  2.7879e-02, -3.1152e-02,
         -4.5027e-02, -4.2783e-02,  4.3554e-02, -1.4915e-03, -5.3968e-02,
         -2.4750e-02, -4.8149e-02, -3.8810e-02,  1.3495e-02,  6.2433e-02,
          5.3553e-02, -4.3207e-02,  4.5292e-02,  4.0638e-02,  3.2601e-02,
          5.1236e-02, -2.2950e-02, -1.7294e-02,  4.4881e-02,  4.8364e-02,
          2.0258e-02,  1.7631e-02, -5.3558e-02,  4.6824e-02,  2.9402e-02,
          5.8698e-03,  1.0404e-02, -4.6136e-02,  3.8000e-02,  1.8309e-02,
         -5.2515e-02, -6.1717e-02, -1.3042e-02,  2.5651e-02,  2.8926e-02,
          6.4722e-03, -3.0960e-02, -4.9000e-02, -4.0460e-02, -2.6519e-03,
         -5.7394e-02, -3.4803e-03, -5.1981e-02, -1.7578e-02,  2.6001e-02,
         -1.9406e-03, -5.8027e-02,  6.5371e-02, -1.7279e-02,  1.7105e-02,
         -5.6636e-02,  4.8750e-02,  1.3589e-02, -2.6963e-02,  5.8048e-02,
         -2.7143e-03,  2.7088e-02,  4.8166e-02,  2.3917e-02, -3.5699e-02,
          1.9211e-02,  3.6718e-02, -1.3939e-02, -5.0199e-02, -3.0492e-02,
          2.4962e-02,  4.4694e-02,  4.2623e-02,  6.5076e-02, -7.1239e-03,
         -6.2378e-02,  4.1850e-02,  7.5280e-03,  3.1958e-02,  3.4814e-02,
         -2.1256e-02,  1.0536e-02,  2.0596e-02,  5.7349e-02, -4.0262e-02,
         -2.3537e-03, -3.2478e-02, -3.2510e-02,  4.3590e-03, -3.5247e-02,
          2.1319e-02, -3.3870e-02, -4.8529e-02, -5.9460e-02,  1.6531e-02,
          4.3185e-02,  4.1965e-02, -1.3178e-03,  8.1956e-03,  1.5425e-02,
          1.9563e-02, -3.9066e-02, -2.3421e-02, -7.6832e-04, -1.3113e-03,
          4.3769e-03, -4.4634e-02, -4.8208e-03, -6.0788e-03, -4.9198e-02,
          5.4599e-02,  2.1641e-02,  1.0053e-02,  2.5887e-02,  4.4726e-02,
         -3.5866e-03, -5.2378e-02,  1.4566e-02, -4.4668e-03, -3.7767e-02,
          6.3858e-02,  2.0254e-02,  1.2180e-02, -1.5126e-02,  2.1769e-02,
          4.9959e-02, -5.9914e-02, -4.7165e-02, -3.0020e-02,  1.3055e-02,
         -1.4515e-02, -1.8739e-02, -2.2179e-02, -1.4858e-02,  4.2260e-02,
         -4.5617e-02,  3.6358e-02, -1.0439e-02,  1.5261e-02,  5.1026e-02,
         -1.9347e-02,  6.9743e-04,  1.0554e-02, -7.5974e-03,  2.9472e-02,
          2.0781e-02, -4.8576e-02,  5.9925e-02,  2.5668e-02, -3.2934e-02,
         -5.6907e-02,  4.7669e-02, -1.0426e-02, -3.2948e-02,  4.5075e-02,
          1.2646e-02,  1.9606e-02, -3.7501e-03, -1.9256e-02, -5.1384e-02,
          3.5677e-02,  1.6388e-02, -5.5507e-02, -1.9679e-02, -3.3788e-02,
          5.7069e-03, -4.1289e-02, -2.8433e-02, -2.0294e-02,  1.5870e-02,
          1.4815e-02,  4.3730e-02,  9.8143e-03,  2.6137e-02,  4.6292e-02,
         -9.0445e-03, -1.4804e-02, -1.1857e-02, -5.2592e-02, -3.6178e-02,
          4.1310e-02,  4.1040e-02, -3.6277e-02,  1.1504e-02, -6.2477e-02,
          4.7341e-02,  3.1316e-02, -2.0110e-02,  6.2721e-02, -5.6489e-02,
          5.3216e-02,  1.7742e-02,  3.2654e-03, -5.2198e-02, -2.9930e-02,
          3.9245e-02, -3.1465e-02, -3.4528e-02,  6.3300e-02, -9.1536e-03,
         -5.8586e-02, -6.7407e-02,  3.9214e-02,  8.2263e-03, -1.2510e-02,
         -3.0088e-02,  4.8860e-02,  2.7311e-02, -1.7220e-02, -2.2785e-02,
          5.9307e-03,  2.0176e-02, -4.1201e-02, -4.4997e-02, -3.4786e-03,
         -5.6365e-02]], device='cuda:0', grad_fn=<SliceBackward>)

Layer: linear_relu_stack.4.bias | Size: torch.Size([256]) |
Values : tensor([ 0.0984, -0.0428], device='cuda:0', grad_fn=<SliceBackward>)

Layer: linear_relu_stack.6.weight | Size: torch.Size([10, 256]) |
Values : tensor([[-0.0595,  0.0515,  0.1977,  0.1285, -0.0592, -0.0857, -0.0159,  0.0740,
          0.0871,  0.0655, -0.1190,  0.0409, -0.0284,  0.0671, -0.1583, -0.0858,
          0.0225, -0.0036,  0.1020,  0.1037, -0.0446,  0.2170,  0.2657,  0.1947,
          0.1003, -0.2101,  0.0315,  0.0293, -0.1172,  0.0568, -0.2951,  0.0099,
          0.2190, -0.1097,  0.0422, -0.0540,  0.0297, -0.0140, -0.1175, -0.1166,
         -0.0581, -0.0513, -0.1202,  0.1970,  0.0521,  0.2092, -0.1037, -0.0414,
```

```
  -0.0643,  0.0342, -0.2252,  0.0340, -0.0783,  0.2205, -0.0967, -0.0196,
  -0.1317, -0.0761,  0.2337, -0.1774,  0.0907,  0.1222, -0.1507, -0.0590,
   0.0264,  0.0460,  0.1450,  0.1883,  0.0191,  0.0261,  0.0667,  0.0402,
  -0.1017,  0.2929,  0.0076,  0.0649, -0.0063, -0.0022, -0.1797,  0.0561,
   0.2146,  0.0687,  0.0177, -0.1443,  0.0082, -0.1121,  0.0423,  0.0861,
   0.0391, -0.1443,  0.1872, -0.1330, -0.0713, -0.0525, -0.1096,  0.0961,
  -0.3214,  0.0077,  0.0895, -0.2200,  0.1057, -0.1721,  0.0956, -0.0851,
   0.0114, -0.1640,  0.0376, -0.1849, -0.0673, -0.0046, -0.1362, -0.0509,
   0.0118,  0.0404, -0.0167, -0.1715, -0.1816,  0.0607,  0.0428, -0.0829,
   0.0832, -0.0291,  0.1113, -0.0066, -0.1031,  0.1371,  0.0068,  0.2365,
  -0.1469, -0.0849,  0.2663, -0.1782,  0.0168,  0.0576, -0.0825, -0.0070,
   0.0529, -0.0670,  0.0919,  0.2349,  0.1654,  0.0136, -0.0470, -0.0287,
  -0.0170, -0.0049,  0.0115,  0.0711,  0.1189,  0.2038,  0.0988, -0.0102,
  -0.1874, -0.0067,  0.1265, -0.1497,  0.0751,  0.0196, -0.1352,  0.0488,
   0.0168, -0.0623,  0.0210, -0.0965,  0.0648,  0.0010,  0.1076,  0.0249,
   0.0100, -0.0434, -0.0444,  0.0273, -0.0115,  0.0706, -0.1882, -0.0757,
   0.0014, -0.0131,  0.0448,  0.0479, -0.0075, -0.0722,  0.0070, -0.0345,
   0.0894, -0.0344, -0.0341, -0.0911, -0.0274, -0.0999, -0.0902, -0.1053,
   0.0632,  0.0634, -0.0310,  0.0482,  0.0634, -0.1458, -0.0766, -0.0712,
   0.2581, -0.0814, -0.0263,  0.0268, -0.1628,  0.1065,  0.0287, -0.0371,
  -0.0357,  0.0778, -0.2194, -0.0330,  0.1977, -0.0491, -0.0554,  0.1672,
  -0.0328,  0.1429, -0.0746,  0.0666,  0.0255,  0.0370, -0.0522,  0.2427,
   0.0313, -0.0793, -0.1204,  0.0089,  0.0872,  0.0339,  0.0065, -0.0142,
   0.0269,  0.0670,  0.1821, -0.0529,  0.1547, -0.0638, -0.1171,  0.0828,
  -0.0263, -0.0115, -0.0168, -0.2609, -0.1830,  0.0416,  0.0978,  0.0044,
   0.1513,  0.0694, -0.1271,  0.0126,  0.1339,  0.0117, -0.0673,  0.1927],
 [-0.1559,  0.0446, -0.1291,  0.2966,  0.0430,  0.0502, -0.0278,  0.1207,
  -0.0222,  0.0250, -0.0091, -0.0546, -0.0445,  0.0622, -0.0659, -0.1002,
  -0.1528,  0.0845,  0.0170, -0.0595, -0.0371,  0.0941, -0.2367, -0.0237,
  -0.0034, -0.2364,  0.1029,  0.0569,  0.2358,  0.0053, -0.0987,  0.0589,
  -0.1874, -0.0857, -0.1195, -0.0075, -0.1220, -0.0465,  0.1164,  0.0617,
  -0.0787,  0.1384, -0.1053,  0.0102, -0.0305, -0.1545,  0.0148,  0.0763,
   0.1132,  0.0460, -0.0695, -0.0011,  0.0866,  0.1276,  0.1342,  0.2992,
  -0.0035, -0.1182, -0.1018,  0.0497, -0.0590,  0.1462,  0.0181,  0.0014,
  -0.0610, -0.0650, -0.0575,  0.1687,  0.0167,  0.0378,  0.1810, -0.0034,
  -0.1730, -0.0037,  0.0082,  0.0656, -0.0281,  0.0181,  0.2935,  0.0498,
   0.0242,  0.0234,  0.0118,  0.1478, -0.1918, -0.1605, -0.0680, -0.0966,
  -0.0141, -0.0232, -0.1619,  0.0993, -0.0075,  0.0620, -0.1718,  0.2215,
   0.2068, -0.0375,  0.0967, -0.1573, -0.1294,  0.1845, -0.2244,  0.1236,
  -0.0216, -0.0814,  0.0059, -0.1400, -0.0370,  0.0089,  0.0457, -0.1002,
  -0.0690, -0.0566,  0.3203,  0.1937,  0.1166,  0.1334, -0.0023, -0.0485,
   0.0426,  0.0530, -0.0820, -0.0777,  0.0187,  0.1439, -0.0970, -0.2915,
  -0.0705,  0.1111,  0.0581, -0.1795,  0.0168, -0.1902, -0.1312,  0.0637,
   0.0835,  0.1004, -0.1071,  0.0338,  0.0950,  0.0631, -0.0815, -0.0395,
   0.0116,  0.0141, -0.0667, -0.1073,  0.0062, -0.2298,  0.0748,  0.0444,
  -0.1140,  0.0447, -0.1171, -0.0854, -0.0446, -0.0475, -0.1872,  0.0428,
  -0.0193,  0.1203, -0.1247, -0.0914, -0.0354,  0.0125, -0.1562,  0.0358,
  -0.2237, -0.2530, -0.0334, -0.0041,  0.1621, -0.0863, -0.1397, -0.1397,
  -0.0441, -0.0749, -0.1124,  0.0410, -0.0644, -0.0346,  0.1223, -0.0775,
  -0.0876,  0.0258,  0.2494, -0.0215,  0.1008, -0.1584, -0.2060, -0.1731,
   0.0360, -0.1153,  0.1433, -0.0269,  0.0738,  0.0910, -0.0483,  0.0495,
   0.1164,  0.0161, -0.0543,  0.0021,  0.1273, -0.1122,  0.0220,  0.0360,
  -0.0449, -0.0850, -0.1836,  0.0494, -0.1026, -0.0327, -0.0079, -0.2854,
  -0.2188,  0.0717,  0.0156,  0.1395,  0.0209, -0.1074, -0.1360, -0.0339,
  -0.1298,  0.1644,  0.2165, -0.1036,  0.1317, -0.0077, -0.1124, -0.0985,
  -0.1393,  0.1043,  0.0844, -0.2128, -0.1962, -0.2126,  0.1718,  0.2001,
  -0.0663, -0.2537, -0.0344, -0.2177,  0.0702, -0.0152,  0.1041, -0.1869,
   0.0447,  0.1323, -0.1035, -0.0412,  0.2845, -0.0653, -0.0406, -0.0892]],
device='cuda:0', grad_fn=<SliceBackward>)
```

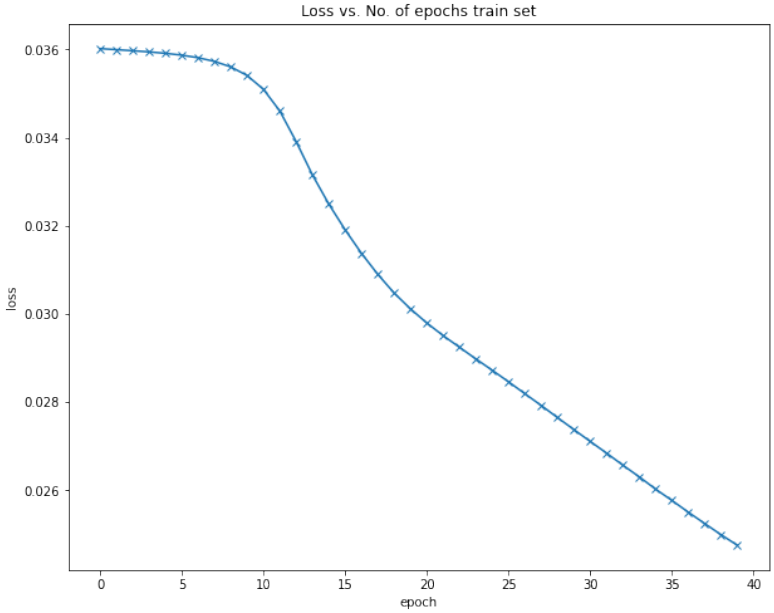
```
Layer: linear_relu_stack.6.bias | Size: torch.Size([10]) |
Values : tensor([ 0.0344, -0.4291], device='cuda:0', grad_fn=<SliceBackward>)
```

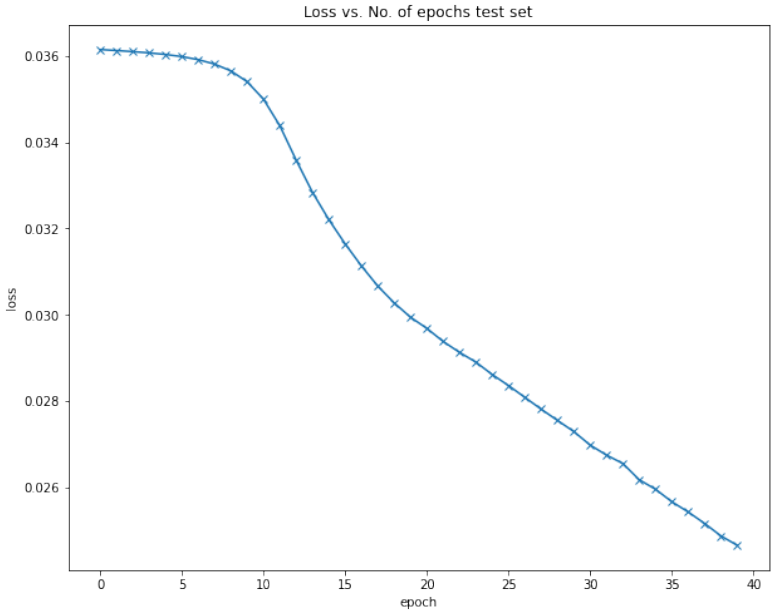
Convolutional Neural Networks

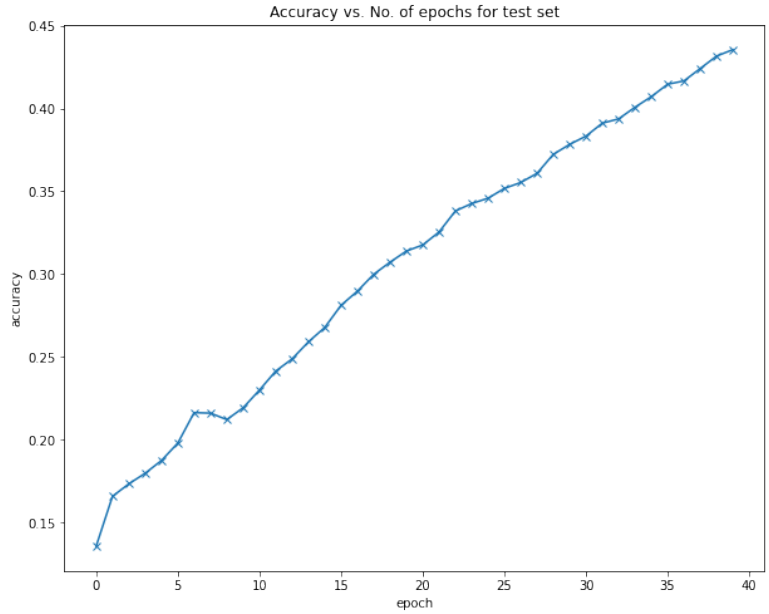
In this section we will report the usage and the performance of a LeNet-5 convolutional neural network. Initially, lets define the architecture of the network that was used in the first experiments.

Layer	Input	Output	Kernel Size	Type
1	3	16	3×3	Convolutional
2	16	32	3×3	Convolutional
3	$6 \times 6 \times 32$	512	-	Fully Connected
4	512	256	-	Fully Connected
5	256	10	-	Fully Connected

With learning rate of 0.001, momentum 0 and for 40 epochs we extracted the results below.

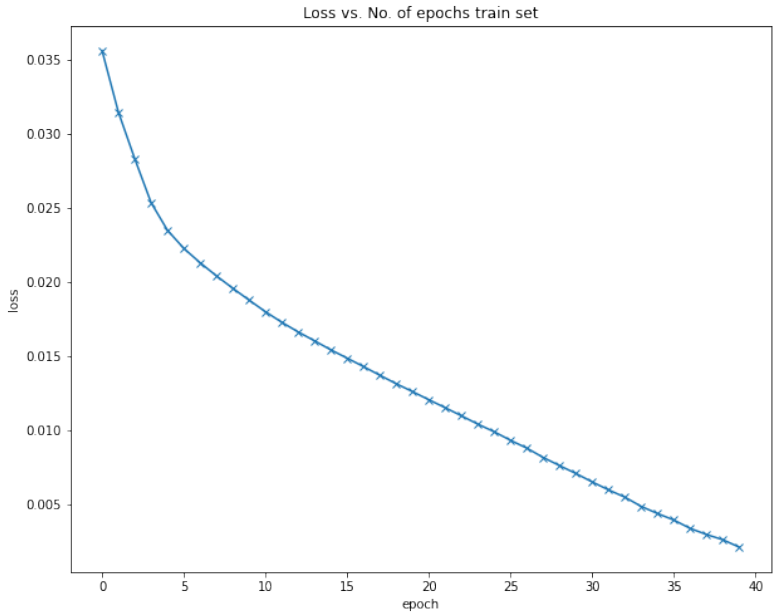


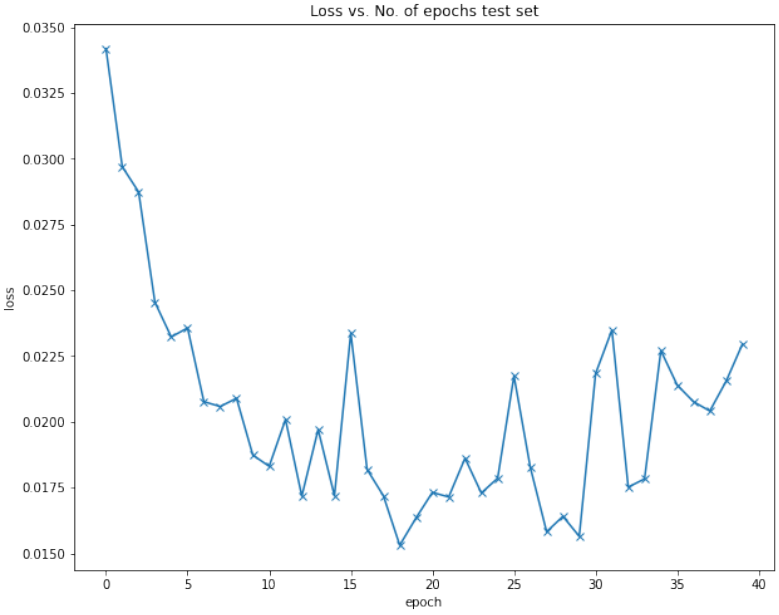


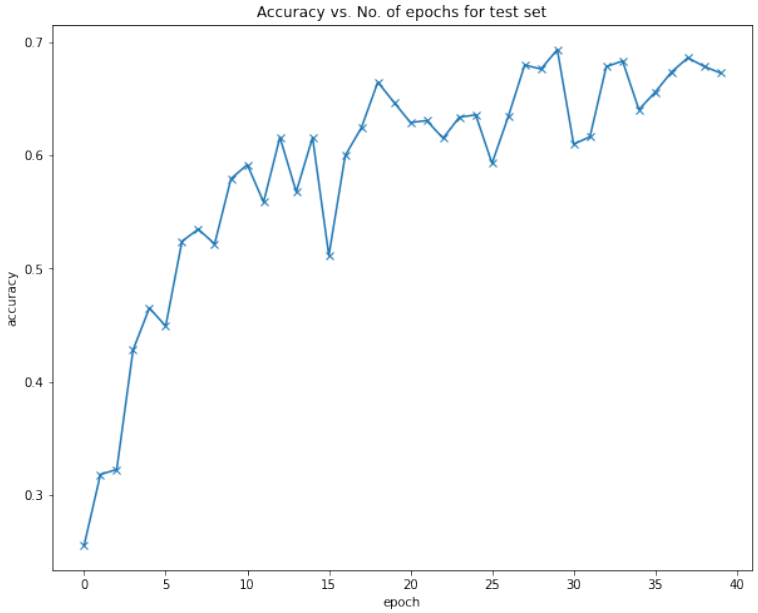


As it can be seen, the model didn't converged yet after 40 epochs and the accuracy is pretty low, approximately 44%. Due to the low learning rate and accuracy we will increase the learning rate in our next model. (similar behaviour with the simple NN)

With learning rate of 0.01, momentum 0 and for 40 epochs we extracted the results below.

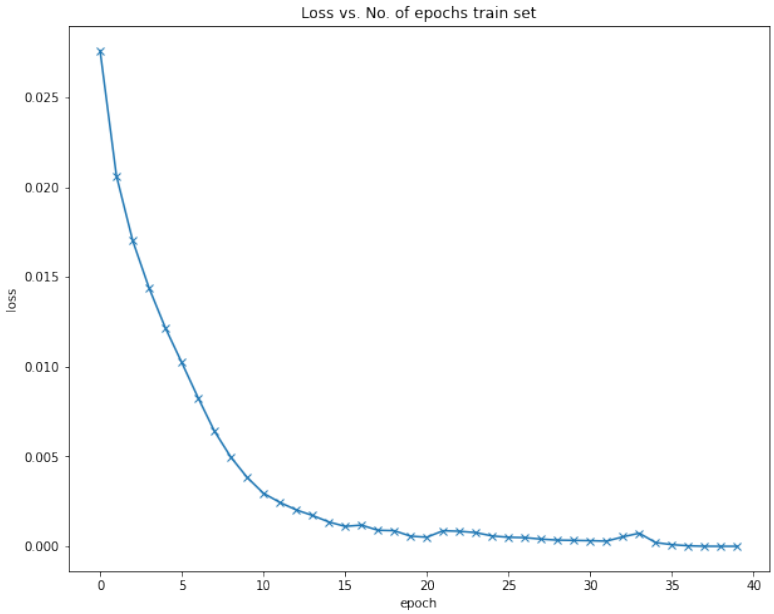


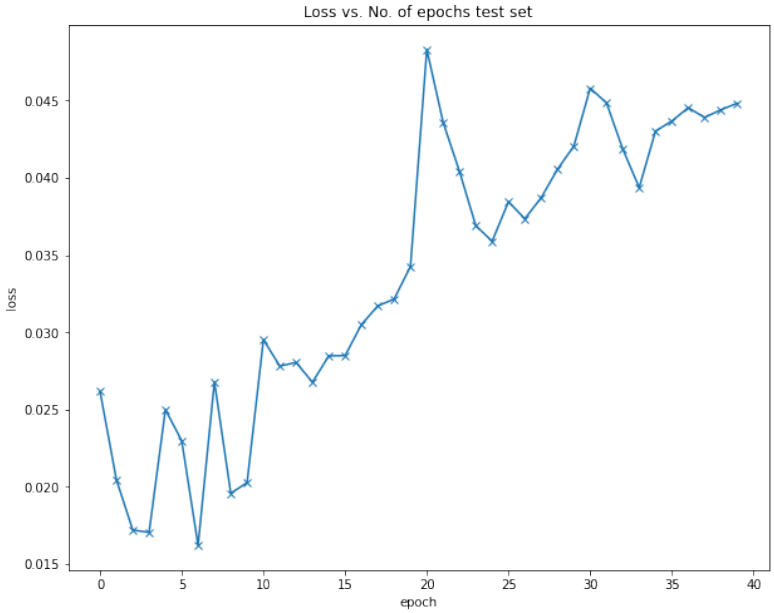


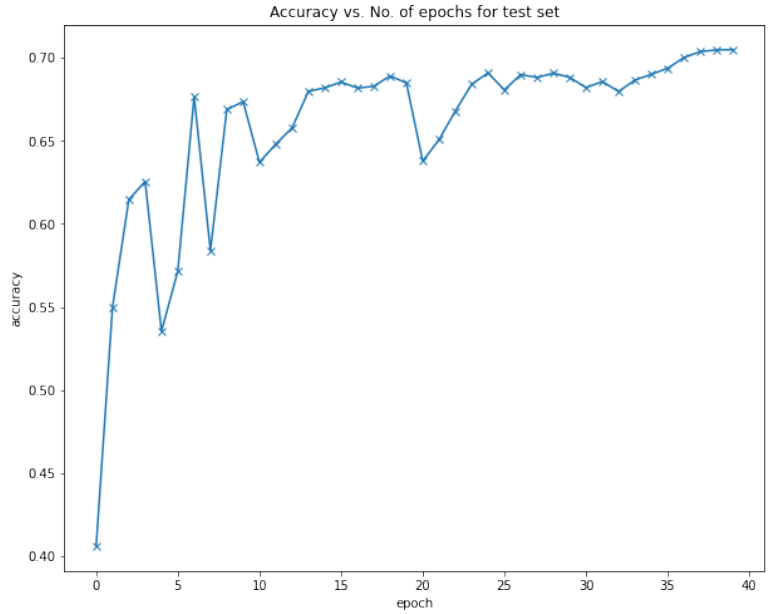


With accuracy of 67.3% and loss of 0.022 this learning rate seems to be very optimal.

With learning rate of 0.1, momentum 0 and for 40 epochs we extracted the results below.



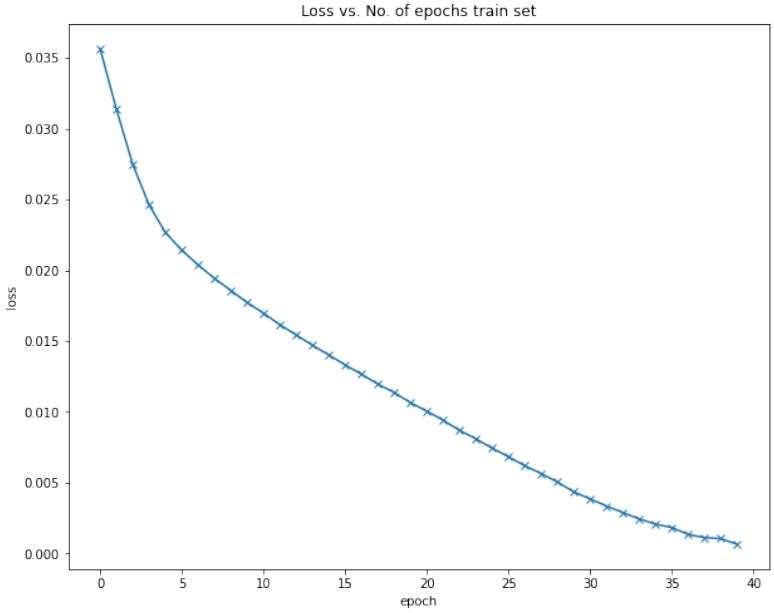


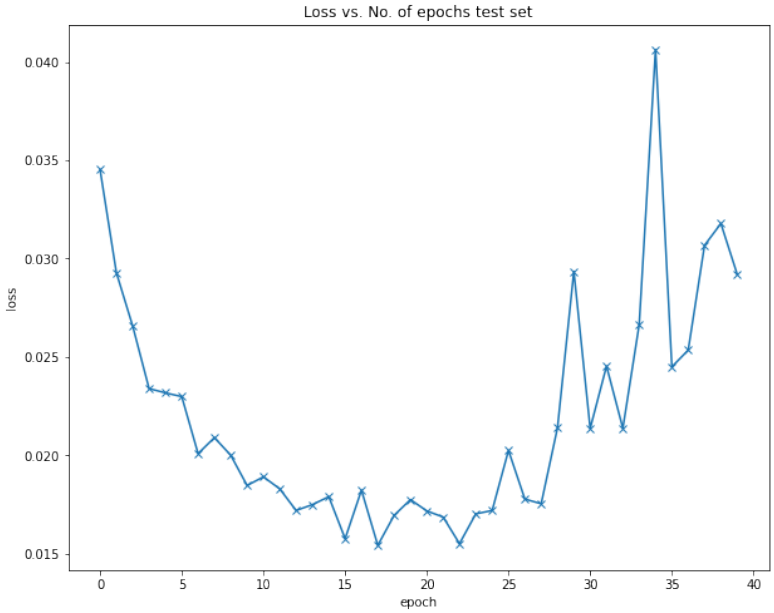


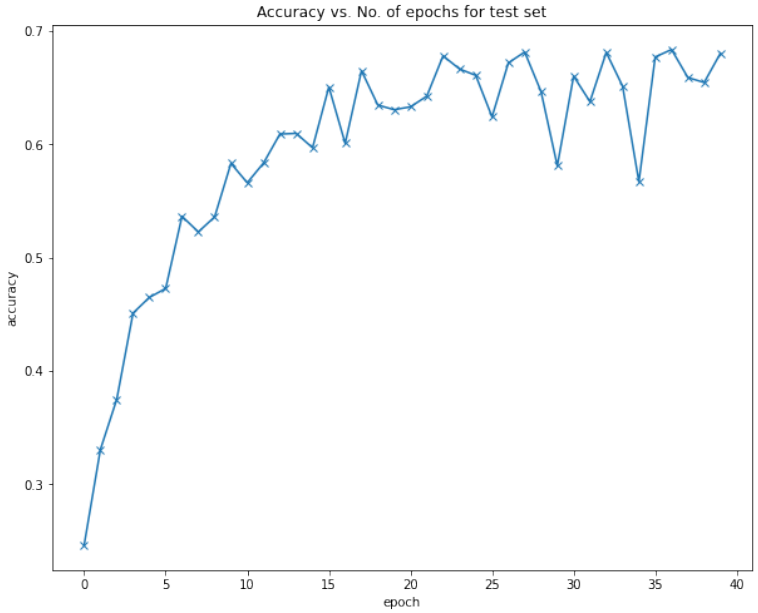
The model after epoch 20 is completely converged. The loss for both train and test is stabilized and the accuracy is constant at 71%. Despite the increased accuracy our choice of preference will be learning rate 0.01 because of the low loss.

Using momentum

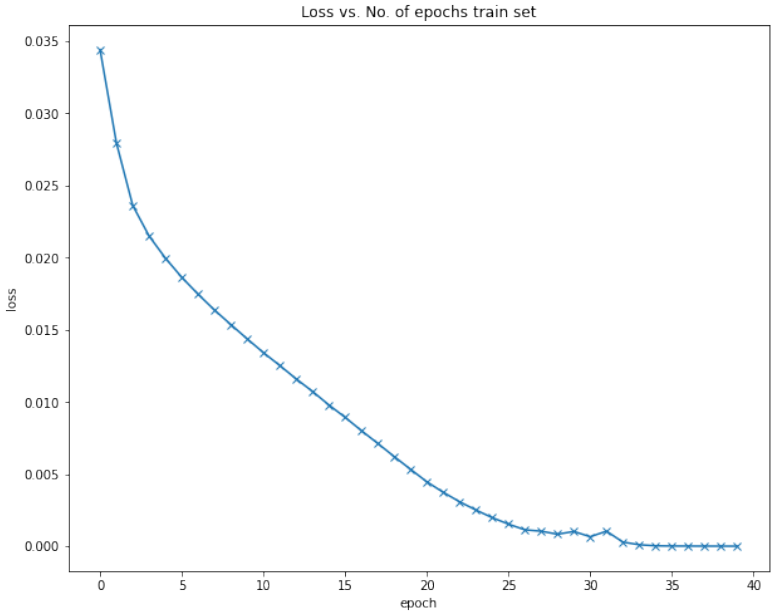
With learning rate of 0.01, momentum 0.2 and for 40 epochs we extracted the results below.

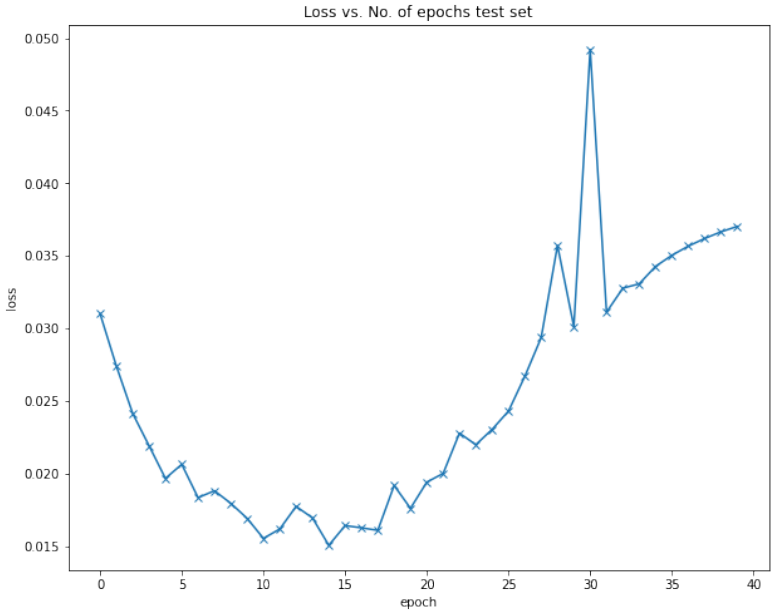


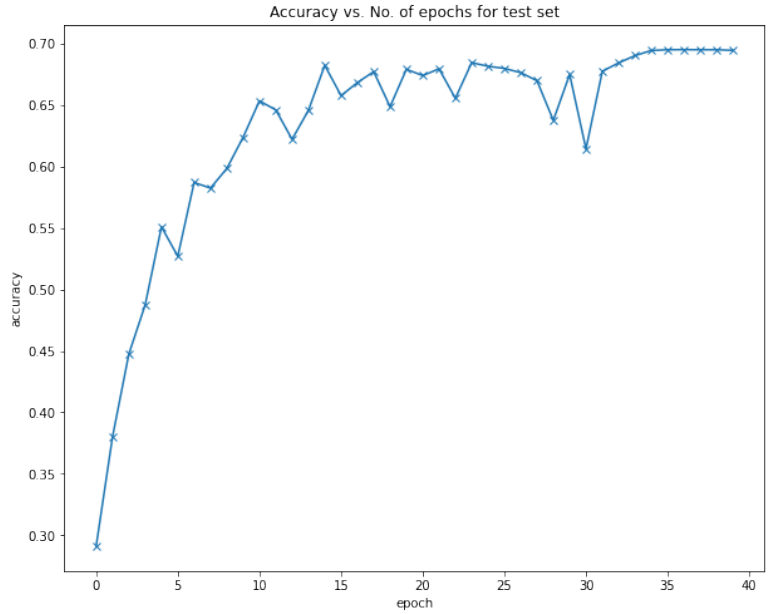




With learning rate of 0.01, momentum 0.5 and for 40 epochs we extracted the results below.







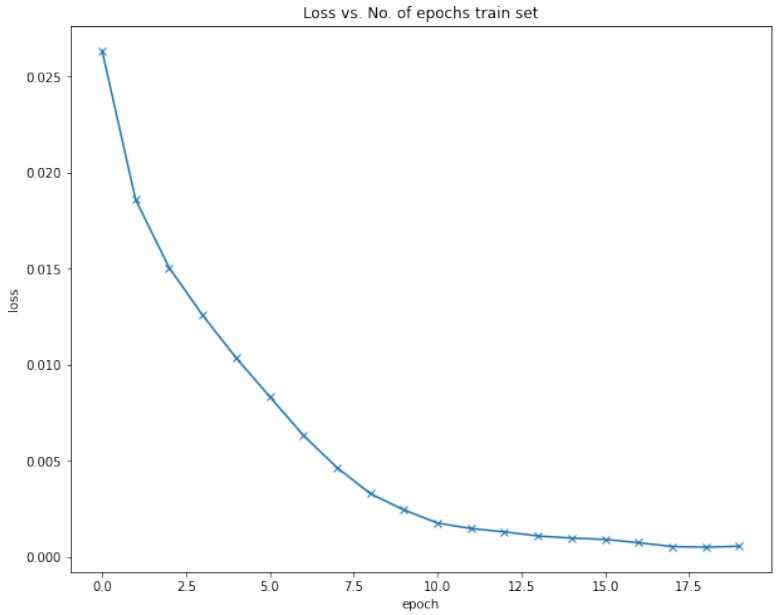
Amazing results here. With learning rate of 0.01 we achieved almost 70%. Up next, by further tuning the CNN architecture we will extract even more accurate models.

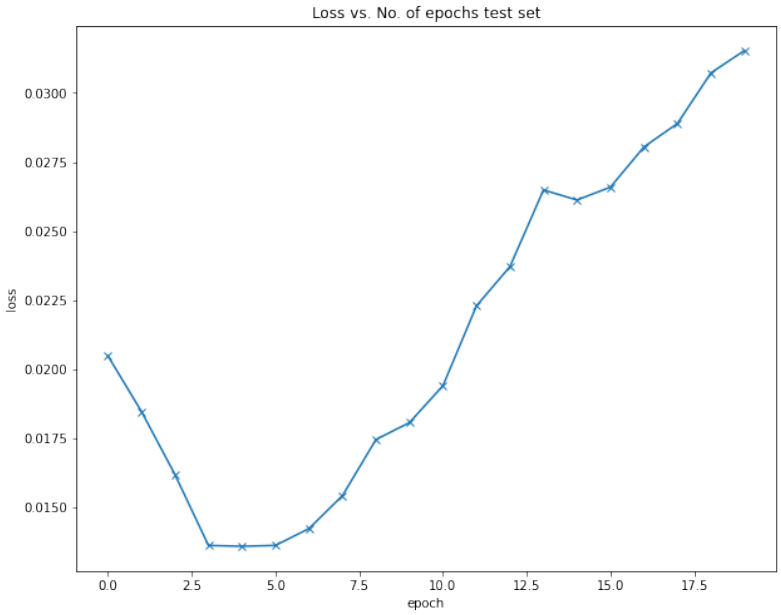
Using different Convolutional Neural Network architectures

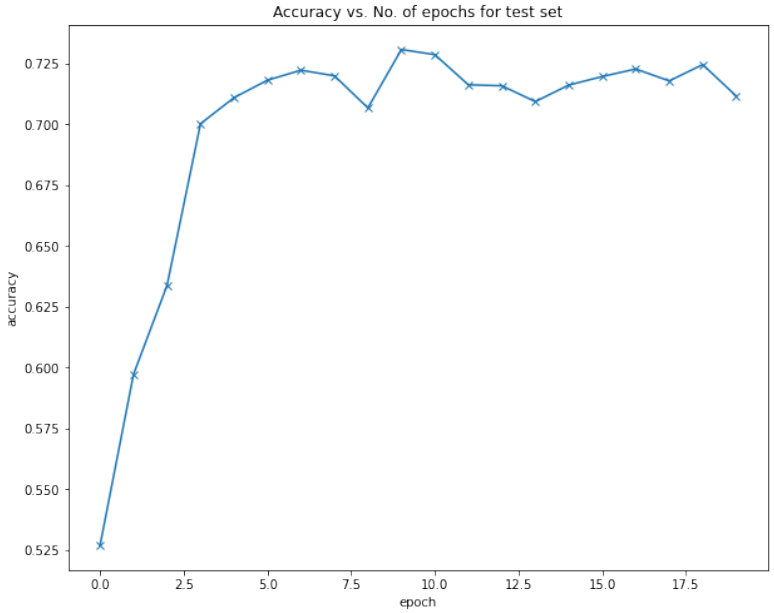
Model 1

Layer	Input	Output	Kernel Size	Type
1	3	32	3×3	Convolutional
2	32	64	3×3	Convolutional
3	$6 \times 6 \times 64$	512	-	Fully Connected
4	512	256	-	Fully Connected
5	256	10	-	Fully Connected

With learning rate of 0.01, momentum 0.9 and for 20 epochs we extracted the results below.



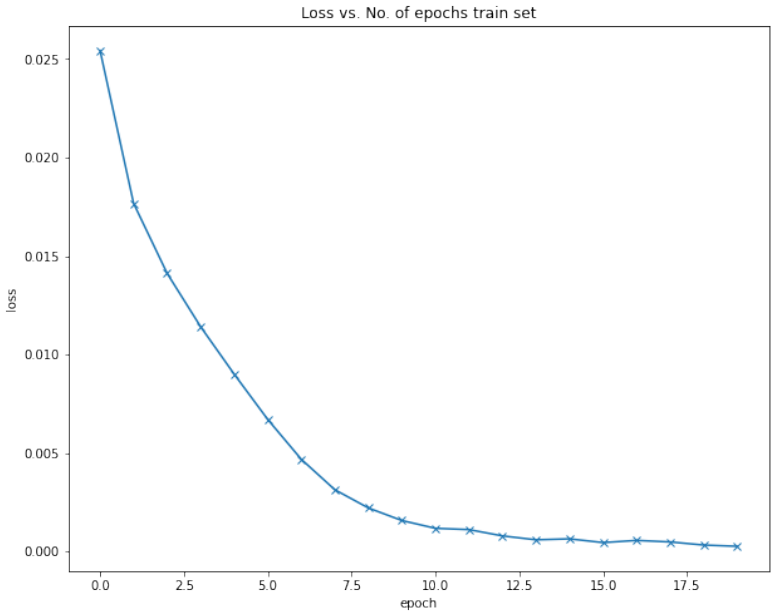


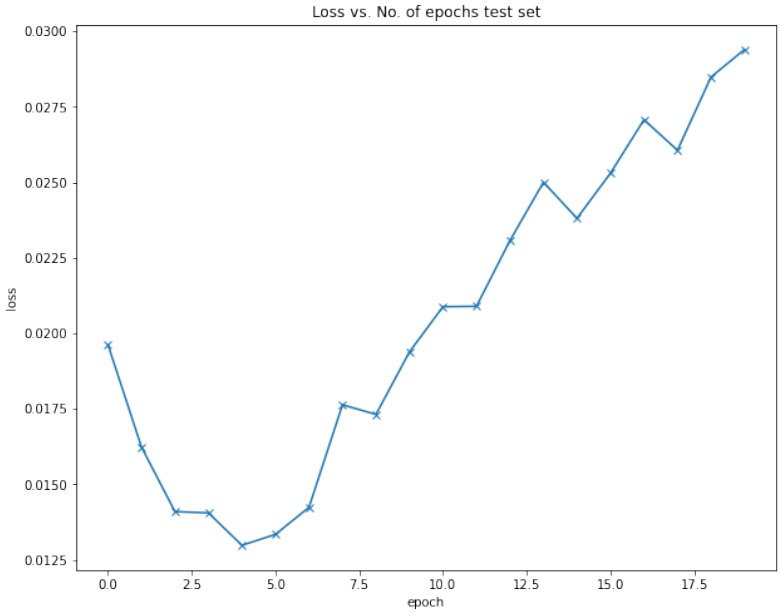


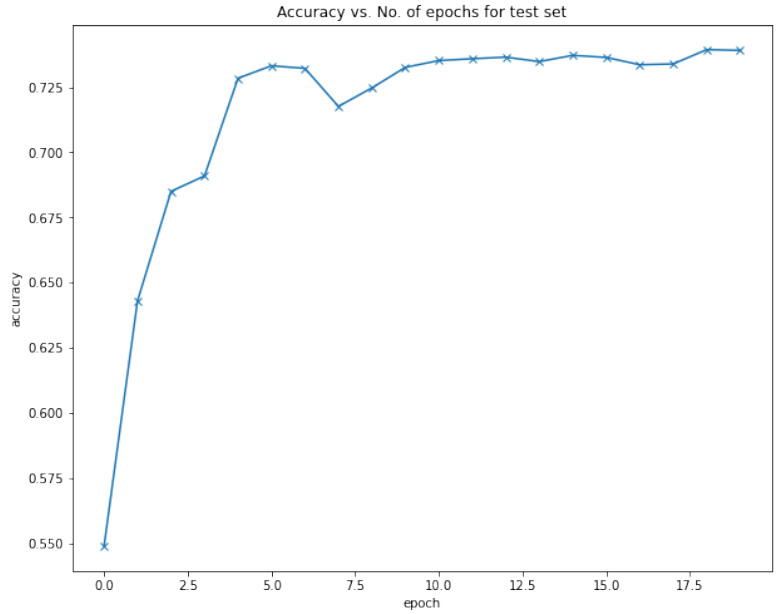
Model 2

Layer	Input	Output	Kernel Size	Type
1	3	64	3×3	Convolutional
2	64	128	3×3	Convolutional
3	$6 \times 6 \times 128$	512	-	Fully Connected
4	512	256	-	Fully Connected
5	256	10	-	Fully Connected

With learning rate of 0.01, momentum 0.9 and for 20 epochs we extracted the results below.



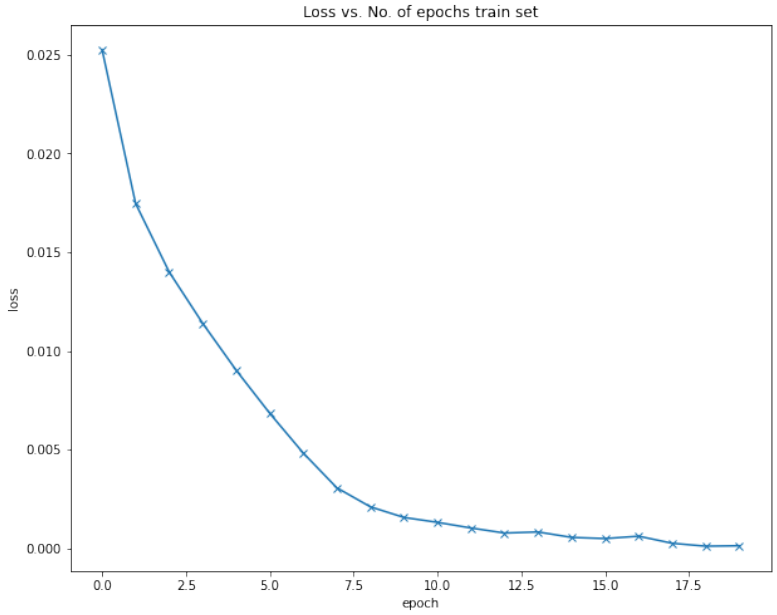

Loss vs. No. of epochs test set
loss
epoch
0.0300
0.0275
0.0250
0.0225
0.0200
0.0175
0.0150
0.0125
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

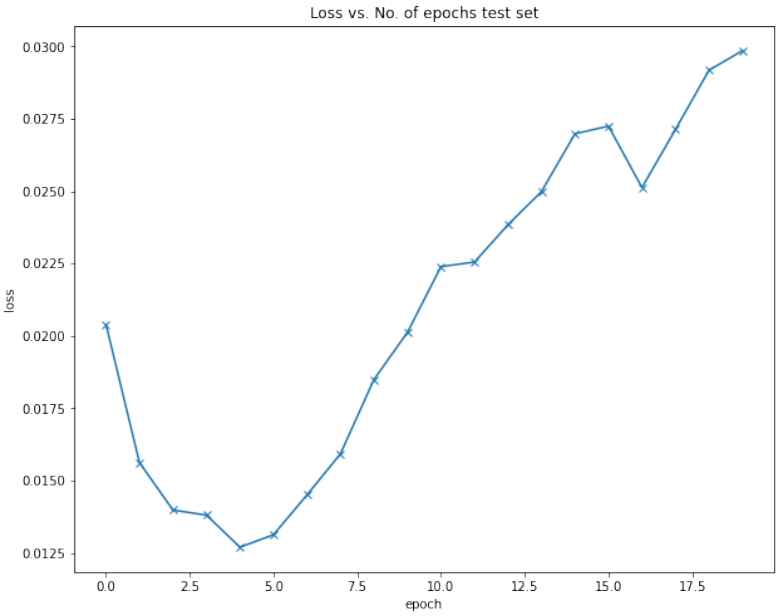

Accuracy vs. No. of epochs for test set
accuracy
epoch
0.725
0.700
0.675
0.650
0.625
0.600
0.575
0.550
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

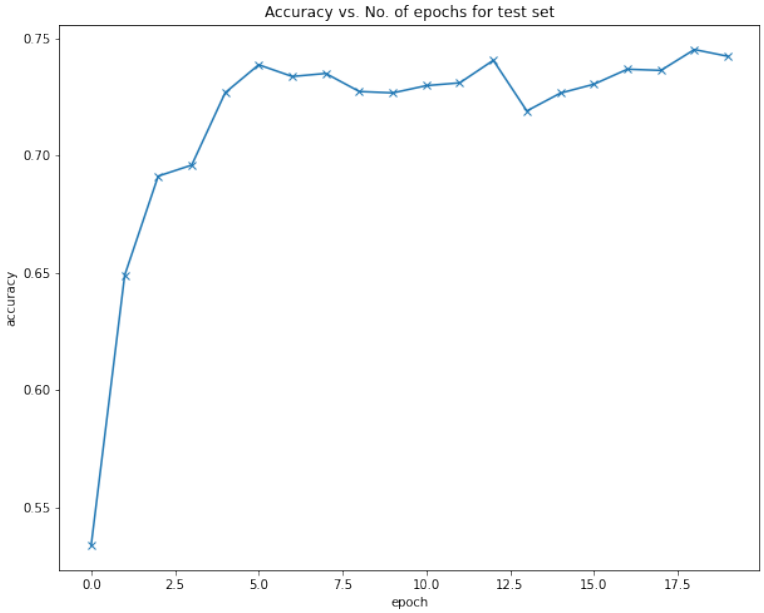
Model 3

Layer	Input	Output	Kernel Size	Type
1	3	64	3×3	Convolutional
2	64	128	3×3	Convolutional
3	$6 \times 6 \times 128$	512	-	Fully Connected
4	512	512	-	Fully Connected
5	512	10	-	Fully Connected

With learning rate of 0.01, momentum 0.9 and for 20 epochs we extracted the results below.

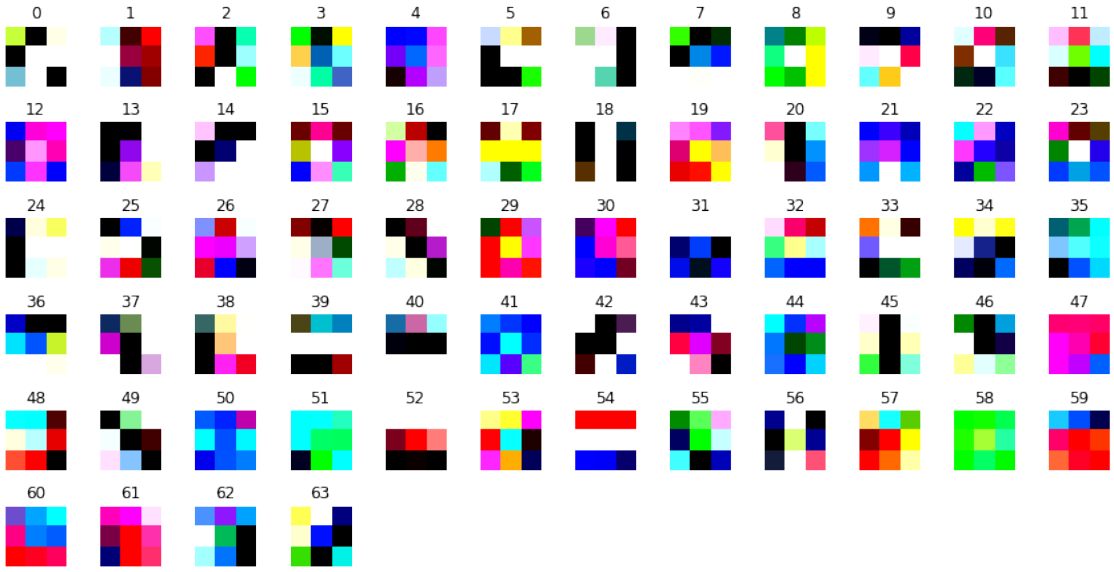






Model 3 almost achieved 75% accuracy with 0.029 loss. Thus, Convolutional Neural Networks extract much better results contrast to simple Neural Networks in terms of accuracy, despite the slightly worse loss (0.006 difference). So, it is assumed that Convolutional Neural Networks in this dataset are far more generalizable and accurate. In the next and final section are reported some details about the Model 3, such as the filters of the 1st Convolutional Layer and the number of the parameters of the model.

1st Convolutional Layer Filters of Model 3



Some observations about the filters of the 1st Convolutional Layer. All the filters have 3x3 size. It is reasonable because 1st Layer uses kernel size 3. Also in this plot all the channels of each filter are merged in one image. Typically, each filter has 3 channels in our example here.

Parameters & Structure of Model 3

```
LeNet5(
  (conv1): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1))
  (conv2): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1))
  (dense1): Linear(in_features=4608, out_features=512, bias=True)
  (dense2): Linear(in_features=512, out_features=512, bias=True)
  (dense3): Linear(in_features=512, out_features=10, bias=True)
)
Layer: conv1.weight | Size: torch.Size([64, 3, 3, 3]) |
Values : tensor([[[[ 1.1613e-01, -6.6382e-01,  2.2235e-01],
          [-4.3903e-01,  3.6674e-01,  3.0718e-01],
          [-1.1842e-02,  5.0024e-01, -4.9846e-01]],

         [[ 2.7742e-01, -5.5835e-01,  2.5999e-01],
          [-5.2689e-01,  3.9068e-01,  2.3969e-01],
          [ 1.0426e-01,  4.1482e-01, -3.9785e-01]],

         [[-1.0622e-01, -6.5661e-01,  1.6680e-01],
          [-4.7030e-01,  3.6906e-01,  5.9653e-01],
          [ 1.3447e-01,  3.9757e-01, -4.1021e-01]]],


        [[[ 5.7686e-02, -7.4600e-02,  3.2427e-01],
          [ 1.5932e-01,  2.5113e-02,  3.3065e-02],
          [ 1.1854e-01, -1.3990e-01,  2.5269e-04]],

         [[ 1.4450e-01, -1.6092e-01, -3.6803e-01],
          [ 3.2265e-01, -1.6007e-01, -3.9560e-01],
          [ 3.2381e-01, -1.3106e-01, -3.8797e-01]],

         [[ 5.4927e-01, -1.6494e-01, -4.7163e-01],
          [ 4.5427e-01, -7.9116e-02, -4.9526e-01],
          [ 5.7612e-01, -1.9121e-02, -1.7629e-01]]]], device='cuda:0',
       grad_fn=<SliceBackward>)

Layer: conv1.bias | Size: torch.Size([64]) |
Values : tensor([-0.0714, -0.1103], device='cuda:0', grad_fn=<SliceBackward>)

Layer: conv2.weight | Size: torch.Size([128, 64, 3, 3]) |
Values : tensor([[[[ 0.0384,  0.0230, -0.0448],
          [-0.0264,  0.1392,  0.0255],
          [-0.0693, -0.0021, -0.0910]],

         [[ 0.0248, -0.0169, -0.0475],
          [ 0.0630, -0.0457, -0.0642],
          [ 0.0410, -0.0959, -0.0654]],

         [[-0.0065,  0.0510, -0.0076],
          [-0.0374,  0.0827, -0.0432],
          [ 0.0065,  0.0082, -0.0169]],

         ...,

         [[-0.0302, -0.0460,  0.0172],
          [-0.0654, -0.0738, -0.0595],
          [ 0.0073, -0.0327, -0.0651]],

         [[ 0.0573,  0.0022,  0.0056],
```

```
           [ 0.0725, -0.0312,  0.0146],
           [-0.0091, -0.0304,  0.0418]],

          [[ 0.0322, -0.0235,  0.0526],
           [ 0.0421, -0.0642, -0.0499],
           [-0.0012, -0.0771, -0.0511]]],


         [[[ 0.0843,  0.0471,  0.0968],
           [ 0.0041,  0.0713, -0.0082],
           [ 0.0050, -0.0334,  0.0192]],

          [[-0.0637, -0.0874,  0.0177],
           [ 0.0630, -0.0670,  0.0886],
           [ 0.1132, -0.0385, -0.0421]],

          [[ 0.0322,  0.0232,  0.0284],
           [-0.0268,  0.0616, -0.0249],
           [ 0.0165, -0.0027,  0.0199]],

          ...,

          [[ 0.0253, -0.0594, -0.0691],
           [-0.0725,  0.0274, -0.0830],
           [-0.0401,  0.1113,  0.0393]],

          [[-0.0205, -0.0852, -0.0542],
           [-0.0486, -0.1571, -0.0896],
           [-0.0670, -0.1026, -0.0583]],

          [[-0.0100, -0.0496,  0.0287],
           [ 0.0076, -0.0337,  0.1031],
           [ 0.0677, -0.0043, -0.0102]]]], device='cuda:0',
       grad_fn=<SliceBackward>)

Layer: conv2.bias | Size: torch.Size([128]) |
Values : tensor([-0.1550, -0.2623], device='cuda:0', grad_fn=<SliceBackward>)

Layer: dense1.weight | Size: torch.Size([512, 4608]) |
Values : tensor([[-0.0149, -0.0117,  0.0028,  ..., -0.0374, -0.0406, -0.0104],
        [ 0.0255,  0.0011,  0.0050,  ..., -0.0056, -0.0105,  0.0020]],
       device='cuda:0', grad_fn=<SliceBackward>)

Layer: dense1.bias | Size: torch.Size([512]) |
Values : tensor([-0.0032, -0.0022], device='cuda:0', grad_fn=<SliceBackward>)

Layer: dense2.weight | Size: torch.Size([512, 512]) |
Values : tensor([[-0.0323, -0.0370, -0.0157,  ..., -0.0283, -0.0193, -0.0290],
        [ 0.0093, -0.0569,  0.0306,  ...,  0.0079, -0.0342,  0.0071]],
       device='cuda:0', grad_fn=<SliceBackward>)

Layer: dense2.bias | Size: torch.Size([512]) |
Values : tensor([ 3.5731e-05, -1.4163e-02], device='cuda:0', grad_fn=<SliceBackward>)

Layer: dense3.weight | Size: torch.Size([10, 512]) |
Values : tensor([[ 0.1028,  0.2802, -0.0054,  ...,  0.0916, -0.1682, -0.0255],
        [ 0.0267,  0.0042,  0.0410,  ...,  0.0161,  0.0796,  0.0652]],
       device='cuda:0', grad_fn=<SliceBackward>)
```

```
Layer: dense3.bias | Size: torch.Size([10]) |
Values : tensor([-0.1679, -0.4982], device='cuda:0', grad_fn=<SliceBackward>)
```