I trained 5 networks with different structures using both 1,000 and 60,000 samples until convergence. The models are tested for every 10 epochs (for 1000 samples) and every epoch (for 60000 samples).

Model		Accuracy							
		60,000 samples				1,000 samples			
		E-1	E-2	E-3	Mean	E-10	E-20	E-30	Mean
1) LeNet	plain	0.957	0.974	0.978	0.970	0.826	0.891	0.906	0.874
	with Dropout2d	0.913	0.947	0.957	0.939	0.748	0.837	0.864	0.816
	with BatchNorm2d	0.983	0.990	0.993	0.989	0.934	0.946	0.949	0.943
2) 2FC		0.832	0.851	0.859	0.847	0.763	0.786	0.790	0.780
3) 3FC		0.921	0.943	0.954	0.939	0.628	0.726	0.749	0.701
4) 3C + 2FC		0.986	0.993	0.995	0.991	0.532	0.795	0.881	0.736
5) 4C + 2FC		0.975	0.986	0.989	0.983	0.836	0.918	0.931	0.895

Note: E-1 means epoch no. 1, E-2 means epoch no. 2, and so on.

By using LeNet, adding Dropout2d can reduce overfitting, which results in a worse accuracy. Adding BatchNorm2d can speed up learning and thus increase accuracy. During training of a neural network, the distribution of the input values of each layer is affected by all layers that come before it. This variability reduces training speed. Batch normalization resolves this variability and speeds up learning. Similar LeNet results can be seen for both 60,000 samples and 1,000 samples.

By comparing the results of 2FC and 3FC, adding a fully connected layer can increase the accuracy in the case of 60,000 samples but decrease the accuracy in the case of 1,000 samples.

By comparing the results of 3C + 2FC and 4C + 2FC, adding a convolutional layer decreases the accuracy in the case of 60,000 samples but can increase the accuracy significantly in the case of 1,000 samples.

Details of loss function and optimizer:

```
loss_func = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)
```

Details of models:

LeNet

```
(conv1): Conv2d(1, 20, kernel_size=(5, 5), stride=(1, 1))
(conv2): Conv2d(20, 50, kernel_size=(5, 5), stride=(1, 1))
(fc1): Linear(in_features=800, out_features=500, bias=True)
(fc2): Linear(in_features=500, out_features=10, bias=True)
(relu): ReLU()
```

LeNet (with BatchNorm2d)

```
(conv1): Conv2d(1, 20, kernel_size=(5, 5), stride=(1, 1))
(bn1): BatchNorm2d(20, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
(conv2): Conv2d(20, 50, kernel_size=(5, 5), stride=(1, 1))
(bn2): BatchNorm2d(50, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
(fc1): Linear(in_features=800, out_features=500, bias=True)
(fc2): Linear(in_features=500, out_features=10, bias=True)
(relu): ReLU()
```

LeNet (with Dropout2d)

```
(conv1): Conv2d(1, 20, kernel_size=(5, 5), stride=(1, 1))
(conv1_drop): Dropout2d(p=0.5, inplace=False)
(conv2): Conv2d(20, 50, kernel_size=(5, 5), stride=(1, 1))
(conv2_drop): Dropout2d(p=0.5, inplace=False)
(fc1): Linear(in_features=800, out_features=500, bias=True)
(fc2): Linear(in_features=500, out_features=10, bias=True)
(relu): ReLU()
```

2FC

```
(fc1): Linear(in_features=784, out_features=512, bias=True)
(fc2): Linear(in_features=512, out_features=10, bias=True)
(relu): ReLU()
```

3FC

```
(fc1): Linear(in_features=784, out_features=512, bias=True)
(fc2): Linear(in_features=512, out_features=128, bias=True)
(fc3): Linear(in_features=128, out_features=10, bias=True)
(relu): ReLU()
```

3C+2FC

```
(conv1): Conv2d(1, 20, kernel_size=(5, 5), stride=(1, 1))
(bn1): BatchNorm2d(20, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
(conv2): Conv2d(20, 50, kernel_size=(5, 5), stride=(1, 1))
(bn2): BatchNorm2d(50, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
```

```
(conv3): Conv2d(50, 100, kernel_size=(3, 3), stride=(1, 1))
(bn3): BatchNorm2d(100, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
(fc1): Linear(in_features=100, out_features=50, bias=True)
(fc2): Linear(in_features=50, out_features=10, bias=True)
(relu): ReLU()
```

4C+2FC

```
(conv1): Conv2d(1, 10, kernel_size=(5, 5), stride=(1, 1))
(bn1): BatchNorm2d(10, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
(conv2): Conv2d(10, 20, kernel_size=(3, 3), stride=(1, 1))
(bn2): BatchNorm2d(20, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
(conv3): Conv2d(20, 50, kernel_size=(2, 2), stride=(1, 1))
(bn3): BatchNorm2d(50, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
(conv4): Conv2d(50, 100, kernel_size=(2, 2), stride=(1, 1))
(bn4): BatchNorm2d(100, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
(fc1): Linear(in_features=100, out_features=50, bias=True)
(fc2): Linear(in_features=50, out_features=10, bias=True)
(relu): ReLU()
```