Identify Hand Written Digits

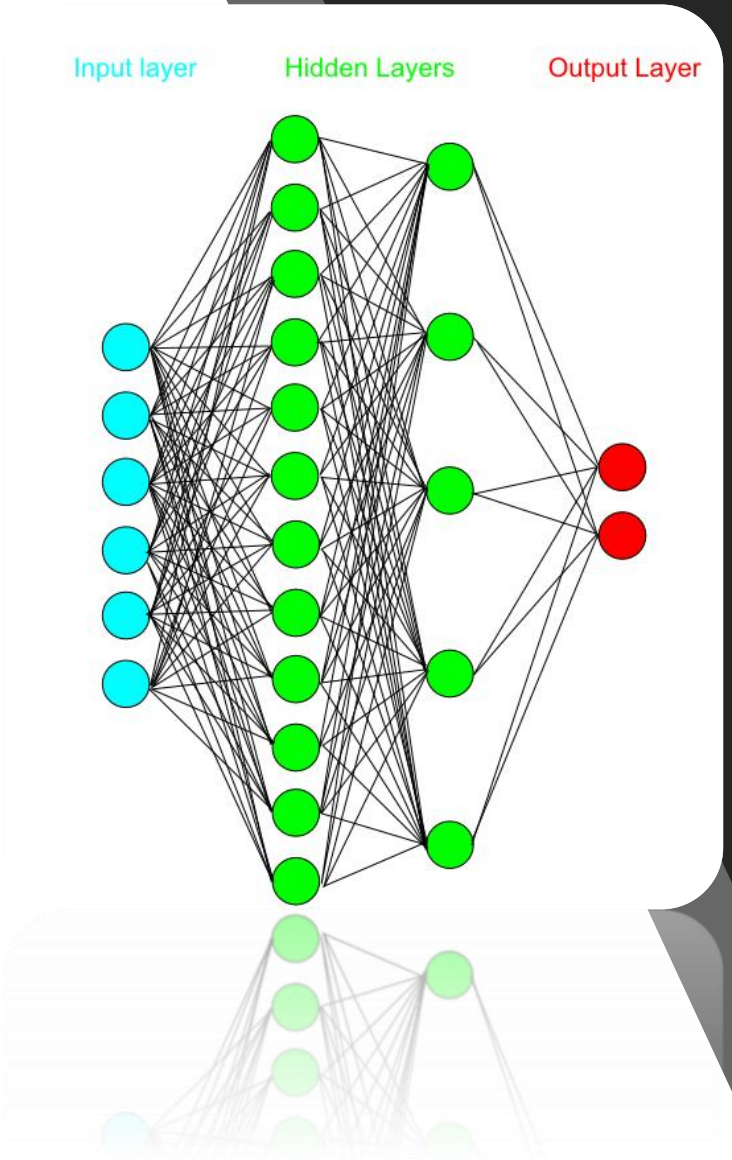
Group 8

Walia, Neeraj, Zhen Zhang

MNIST DATASET

- Data contains grey scaled images of hand written digits from 0 to 9.
- Each image size is 28 * 28 pixels
- Each pixel has a number between 0 to 255 attached to it. The number identifies the darkness of the pixel.
- Training data set has 785 columns with first column as label which describes the hand written digit.
- We got the data from kaggle website.
- Data is already divided into training and test data sets.
- For training data set there are 42000 rows of input.
- For test data set there are 28000 rows of data.

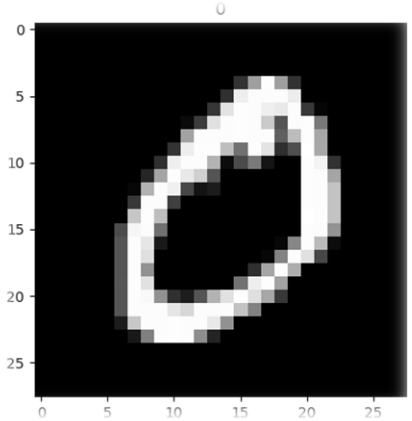
APPROACH

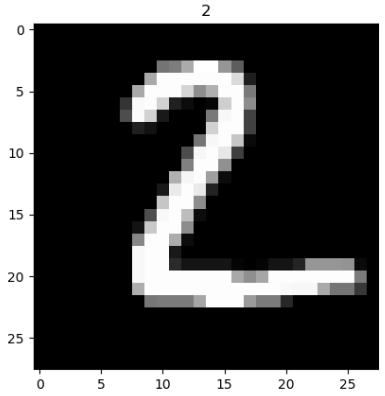


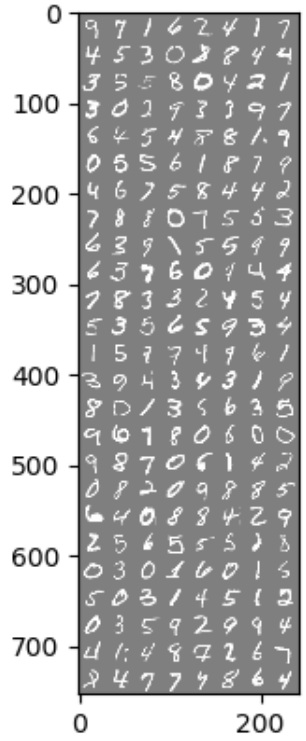
- We built a multi – layer network architecture for this problem. And we tested the number of layers in the network to achieve best accuracy with least complex model.
  - Along with layers, we have changed other variables to test the accuracy like batch size, epoch etc.
- We have designed a Multilayered Perceptron (MLP) and Convolution network for this project and compared their accuracy.
- Pytorch framework is used for this project.

DATA OVERVIEW

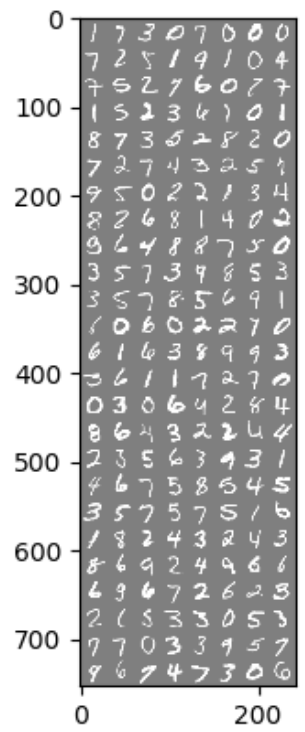
```
training image:torch.Size([60000, 28, 28])
training label:torch.Size([60000])
testing image:torch.Size([10000, 28, 28])
testing label:torch.Size([10000])
```







Randomly Training Samples



Randomly Testing Samples

MULTILAYERED PERCEPTRON

- We have finalized a two-layer network architecture with the following parameters. We chose two layered network because adding more layers was not improving the accuracy of the results.
  - Epoch_size = 50
  - Image_dimen = 1
  - Image_size = 28
  - Hidden_size = 200
  - Batch_size = 200
  - Class_num = 10

We have used CrossEntropyLoss() criterion as this is a classification problem. This criterion helps in training data with number of classes.

We also test with nn.NLLLoss() criterion but did not found any difference in accuracy or performance.

MULTILAYERED PERCEPTRON

optim.SGD() is used to optimize the algorithm. Following are the options selected for this optimizer:

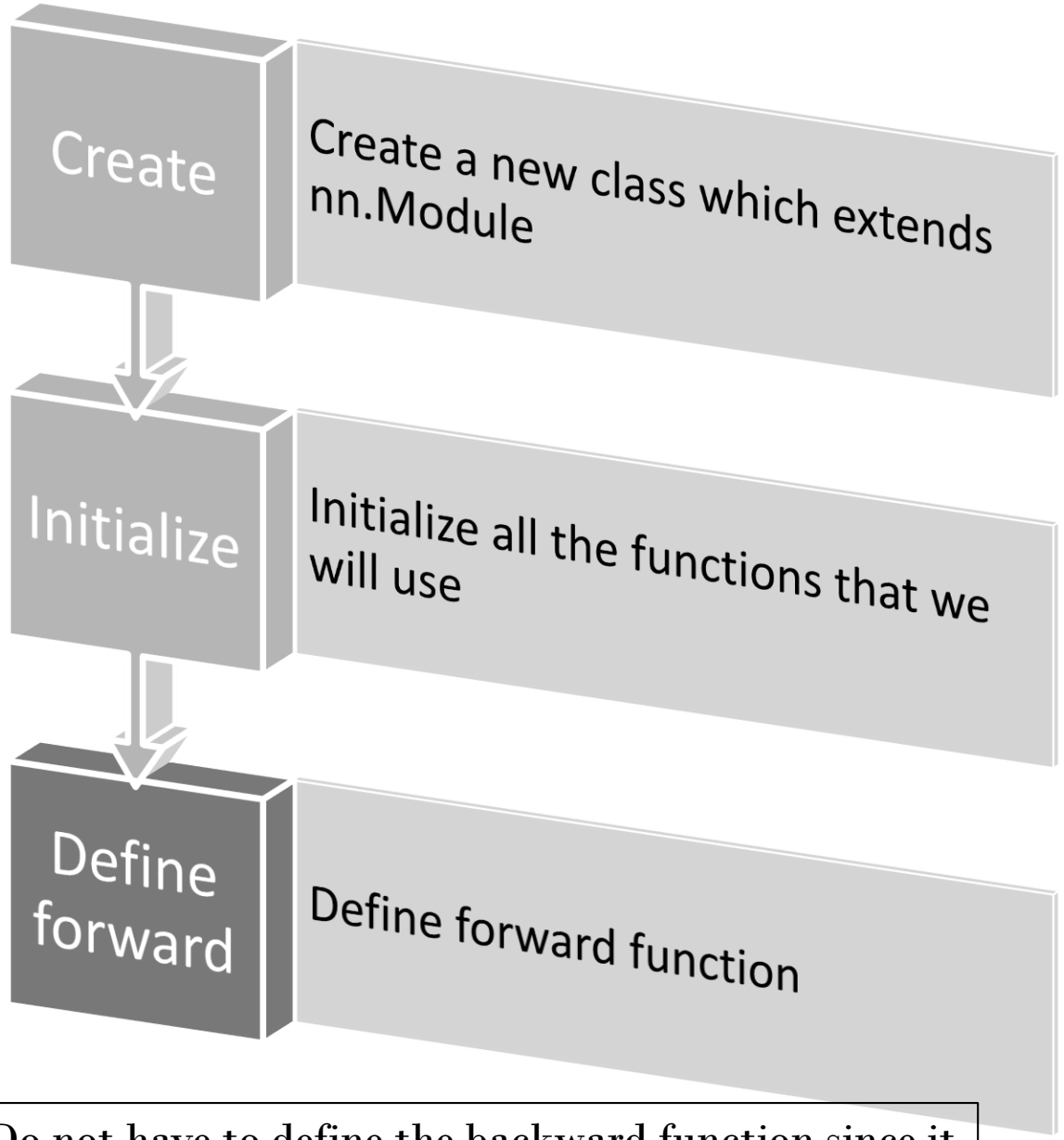
Learning rate = 0.1	Momentum = 0.9

Model got trained using the above mentioned parameters on training data set and its accuracy was test on test dataset.

Accuracy = 98%

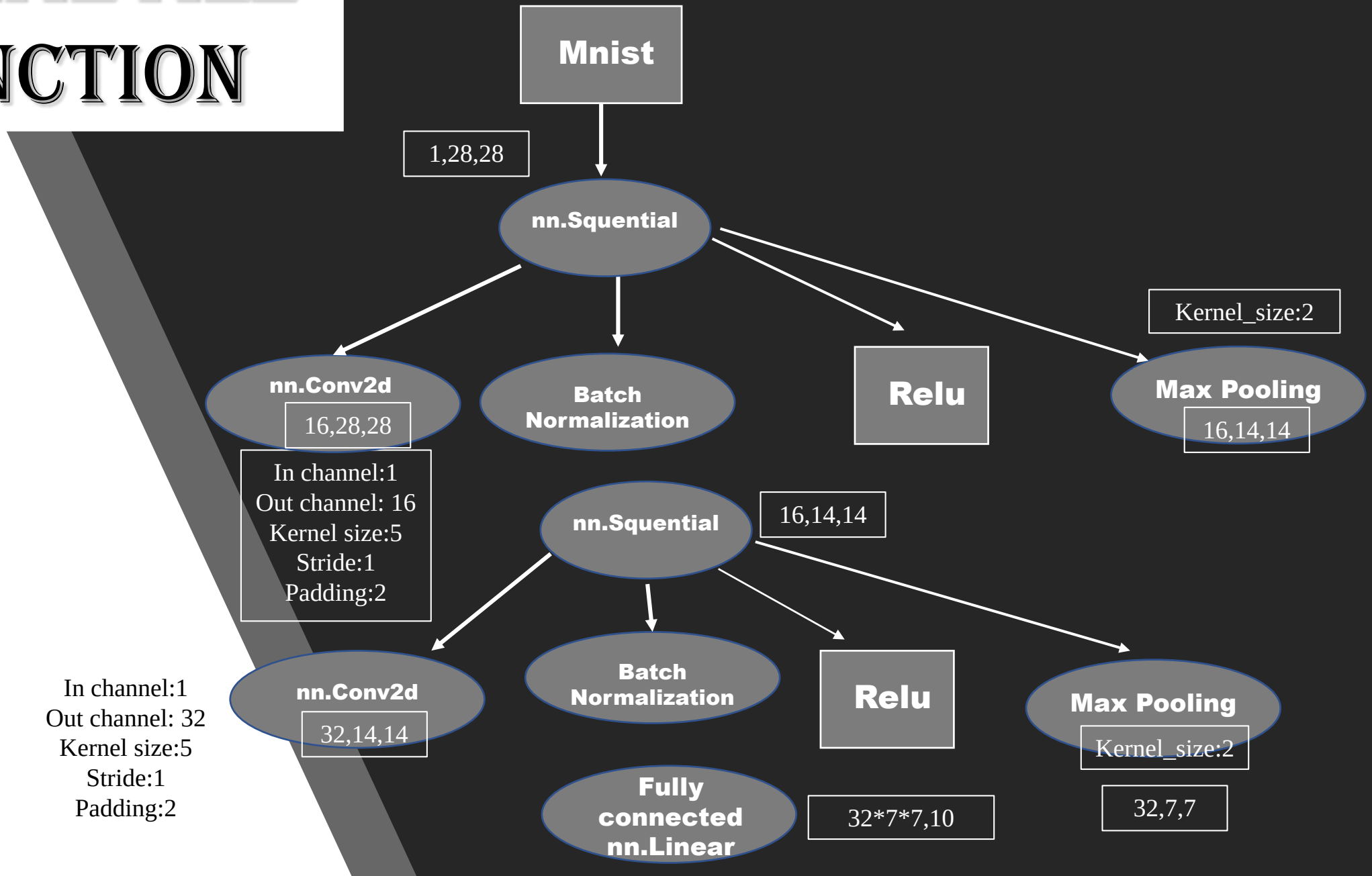
Define a Convolution Neural Network

batch_size = 200
learning_rate = 0.1
epoch_size = 50
hidden_layers=100

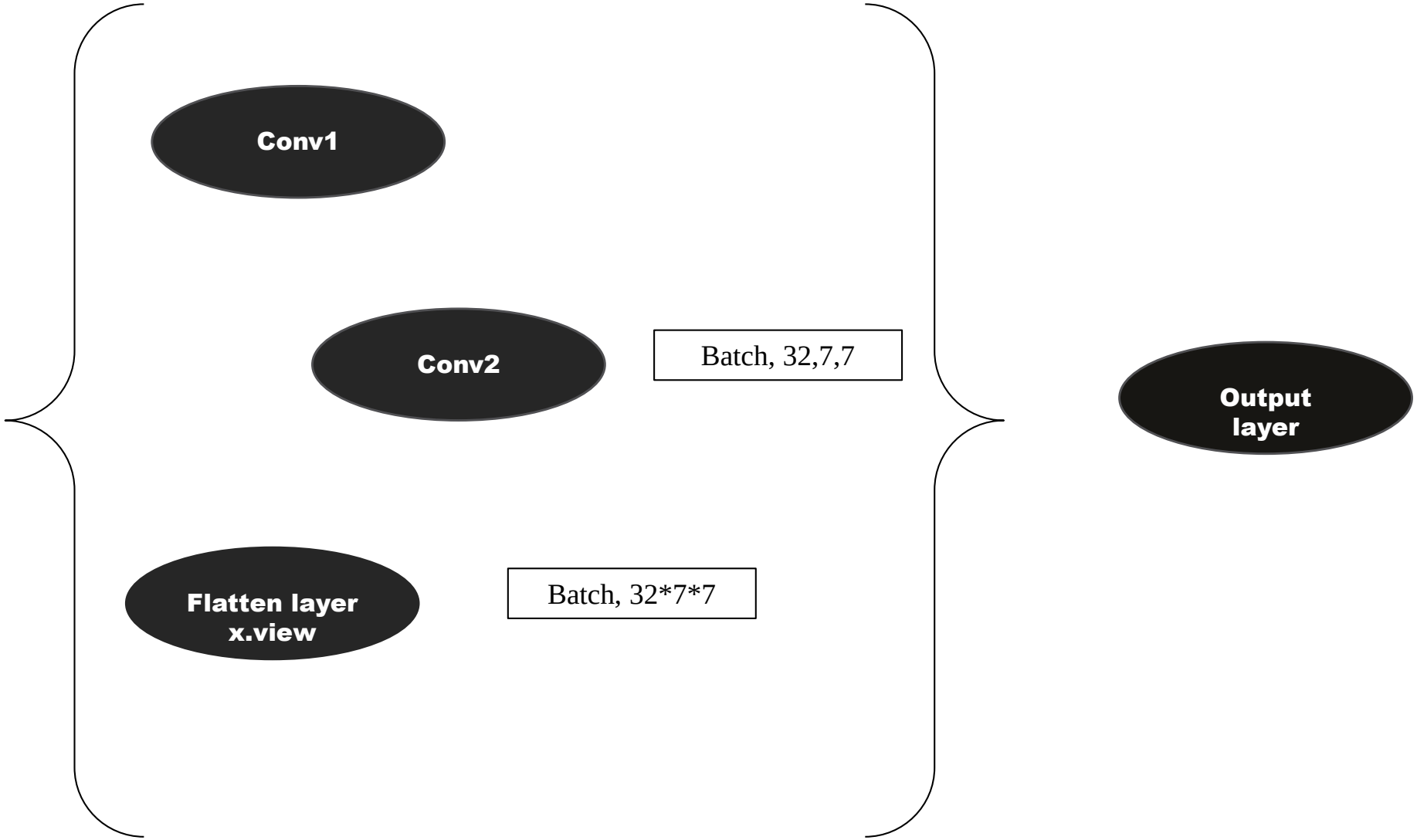


Do not have to define the backward function since it automatically determined by the autograd package.

INITIAL ALL FUNCTION



DEFINE FORWARD FUNCTION



CONVOLUTION NEURAL NETWORK

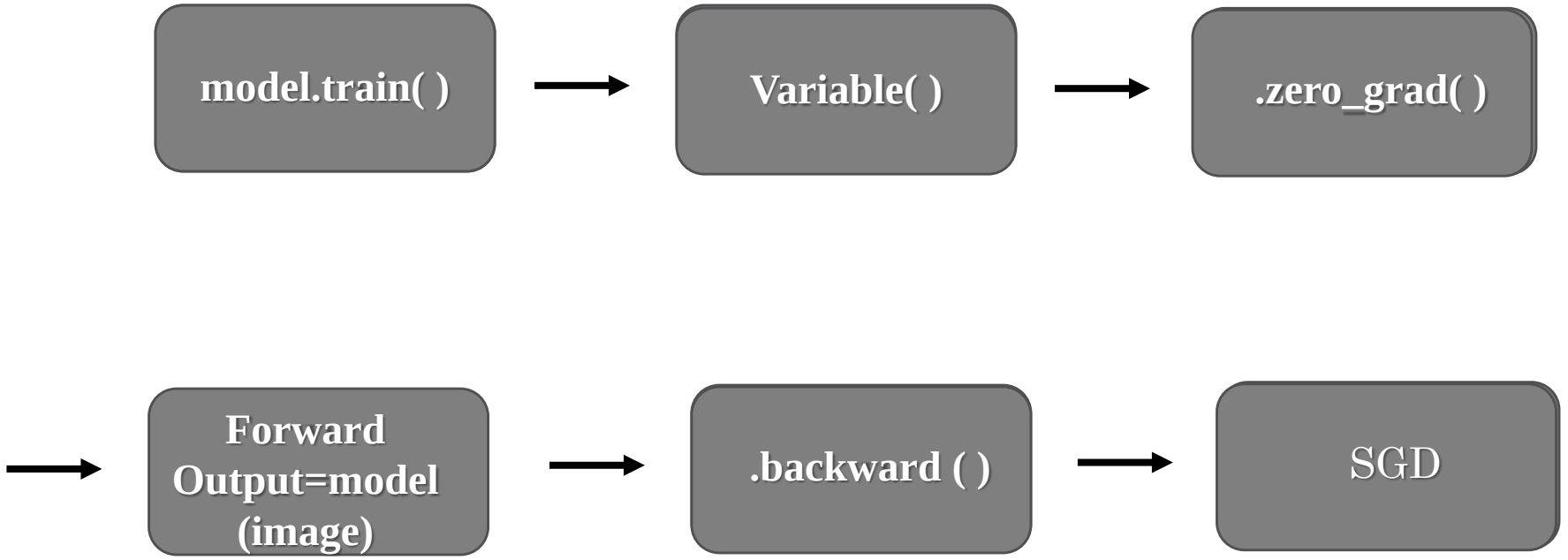
Net Architecture

```
CNN(
  (layer1): Sequential(
    (0): Conv2d(1, 16, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), dilation=(1, 1), ceil_mode=False)
  )
```

```
  (layer2): Sequential(
    (0): Conv2d(16, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), dilation=(1, 1), ceil_mode=False)
  )
```

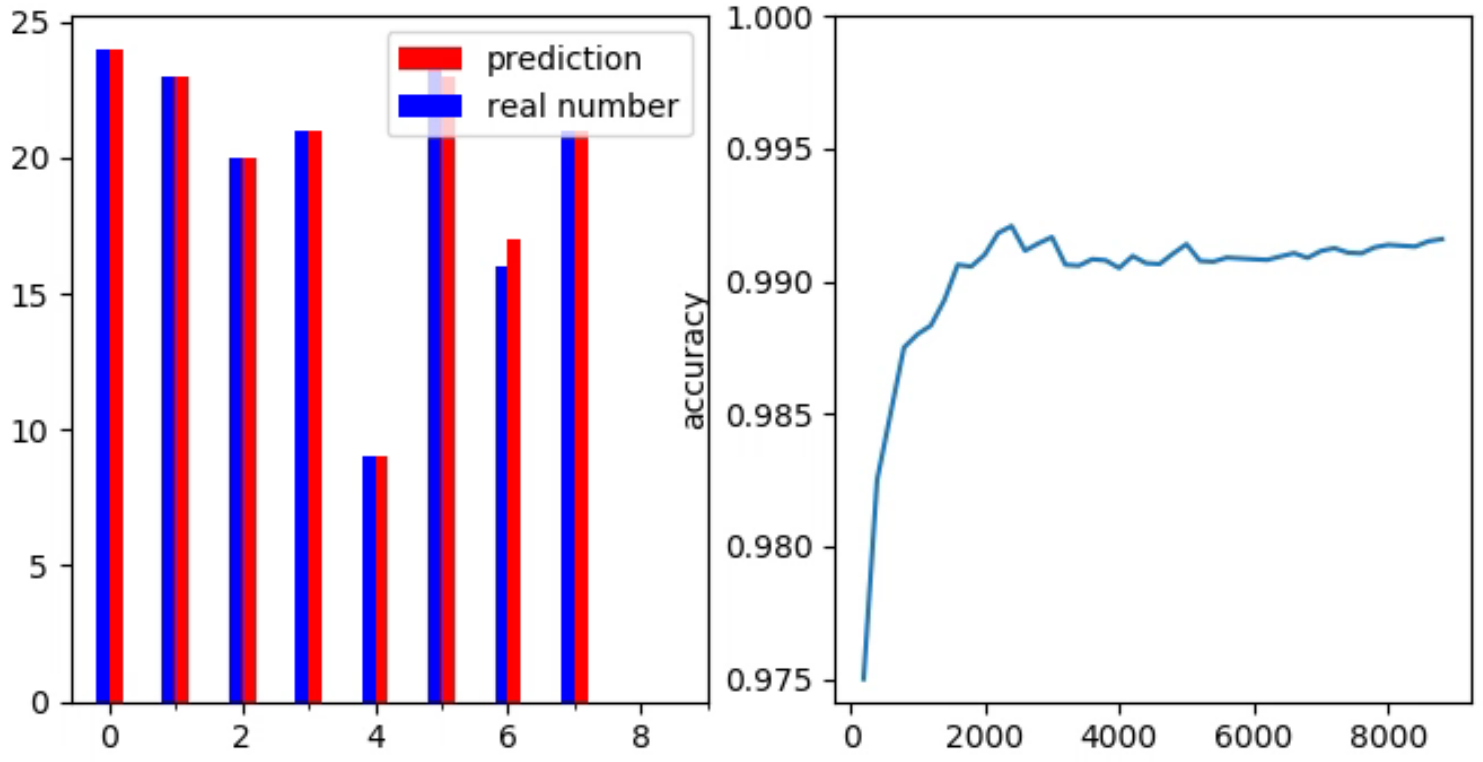
TRAINING NETWORK

Optimizer: Stochastic Gradient Descent
Loss: Cross entropy

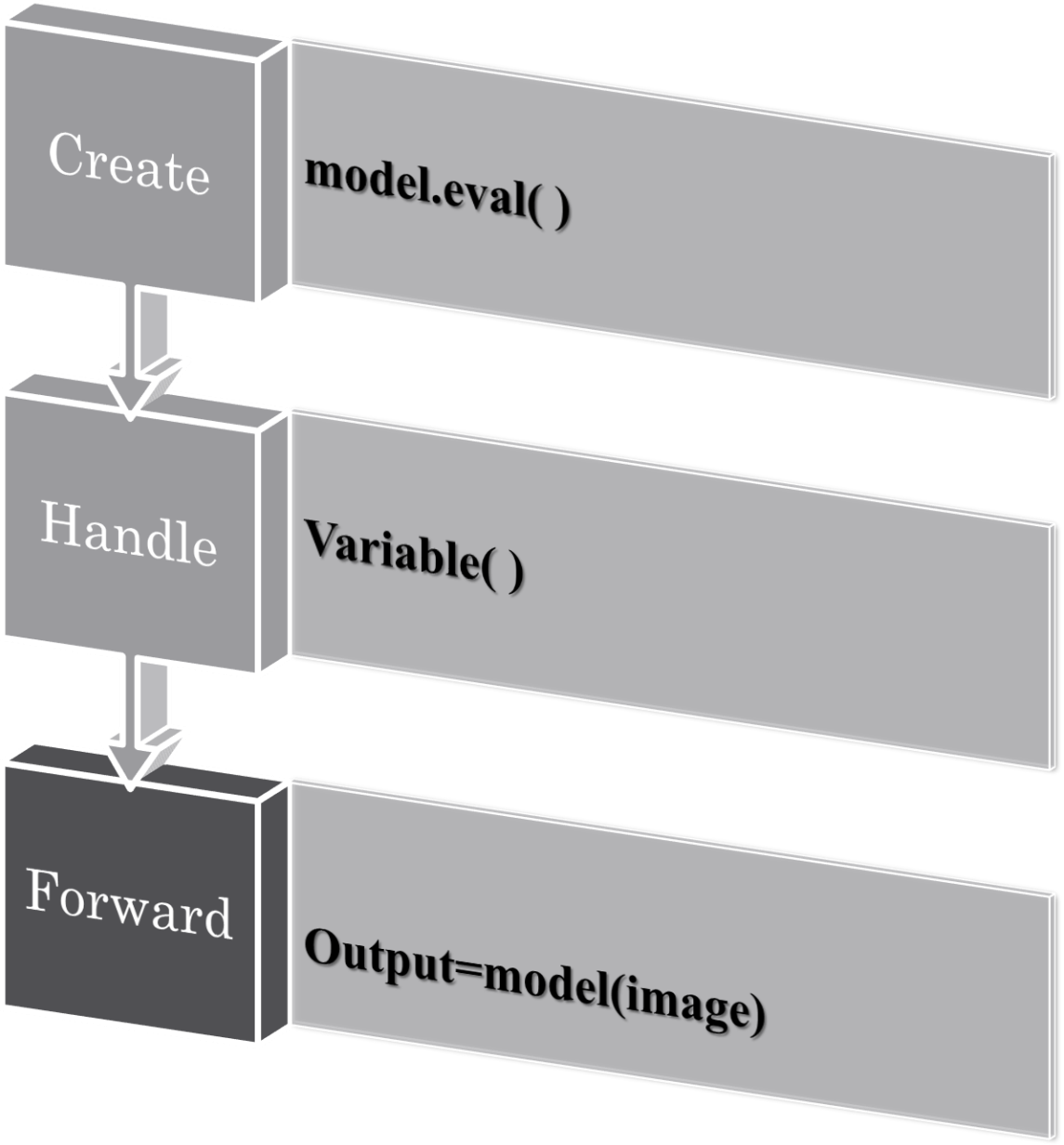


- Torch.optim
- weight = weight - learning_rate * gradient

Training Results



TESTING OUR TRAINED NETWORK



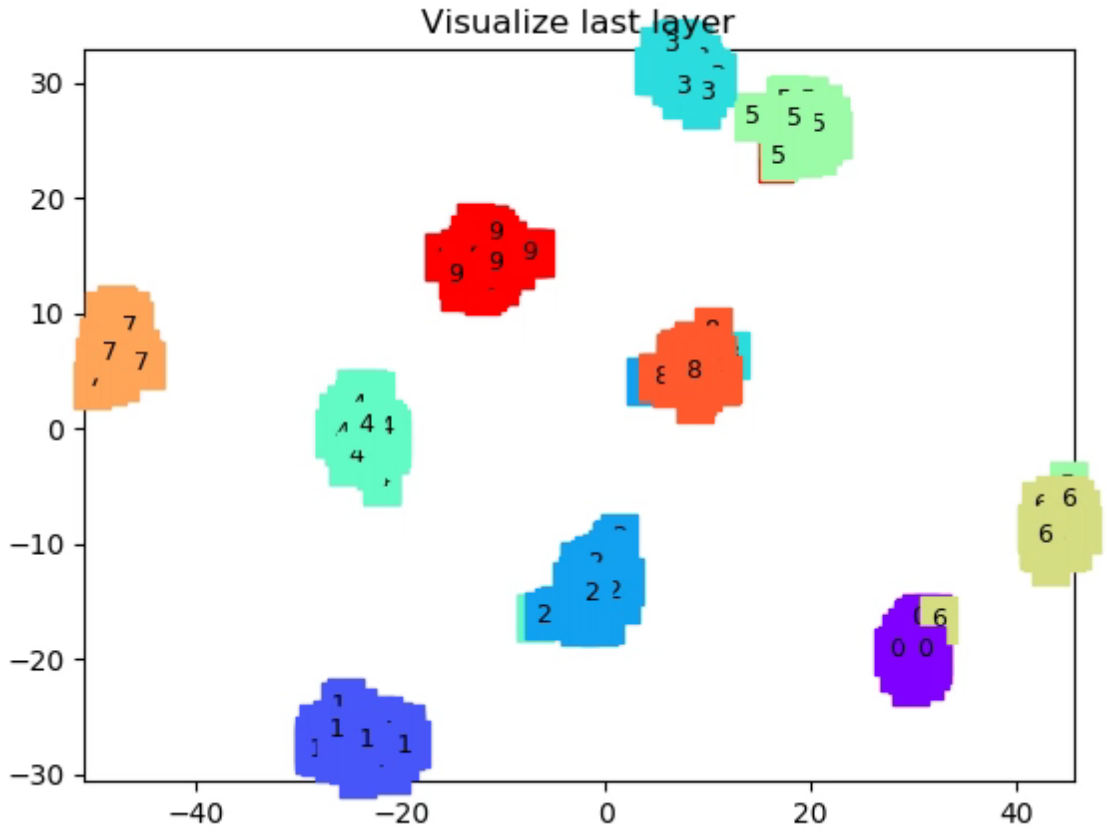
TESTING RESULTS

```
Test Accuracy of the model on the 10000 test images: 98 %
Training Epoch 9
[9,    50] loss: 0.009
[9,   100] loss: 0.010
[9,   150] loss: 0.008
[9,   200] loss: 0.010
[9,   250] loss: 0.018
[9,   300] loss: 0.012
Training Completed
Test Accuracy of the model on the 10000 test images: 99 %
Training Epoch 10
[10,    50] loss: 0.007
[10,   100] loss: 0.009
[10,   150] loss: 0.009
[10,   200] loss: 0.008
[10,   250] loss: 0.011
[10,   300] loss: 0.009
Training Completed
Test Accuracy of the model on the 10000 test images: 99 %
Training Epoch 11
[11,    50] loss: 0.008
[11,   100] loss: 0.004
[11,   150] loss: 0.006
[11,   200] loss: 0.008
[11,   250] loss: 0.009
[11,   300] loss: 0.010
Training Completed
```

Average Accuracy of the model on the 10000 test images is 99%

```
Accuracy for number 0 is 0.9948979591836735
Accuracy for number 1 is 0.9982378854625551
Accuracy for number 2 is 0.9932170542635659
Accuracy for number 3 is 0.992079207920792
Accuracy for number 4 is 0.9938900203665988
Accuracy for number 5 is 0.9932735426008968
Accuracy for number 6 is 0.9906054279749478
Accuracy for number 7 is 0.9873540856031129
Accuracy for number 8 is 0.9917864476386037
Accuracy for number 9 is 0.9861248761149654
```

Visualization



- **Almost every digit goes to their own group**

CONCLUSION

- We got accuracy of MLP network is rated as 98% and of the convolution network at 99%.
- This shows that both networks are highly accurate in reading the hand written digit and with further tweaking they can perform at almost 100% accuracy.
- So according to our testing both network are suitable for this project of reading hand written digits

End....

Thank You