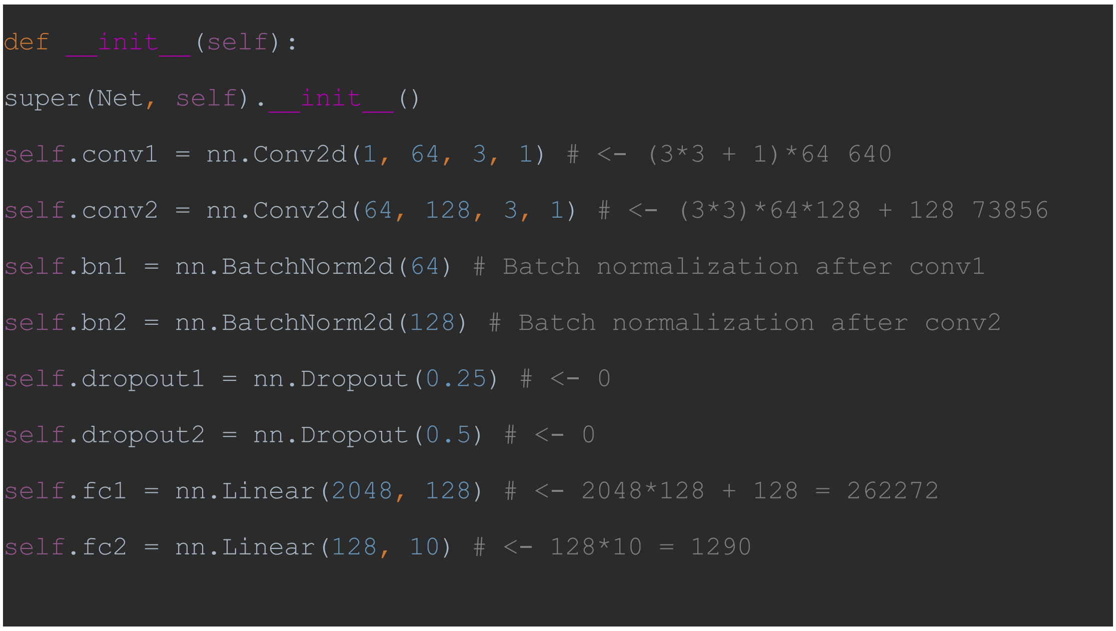
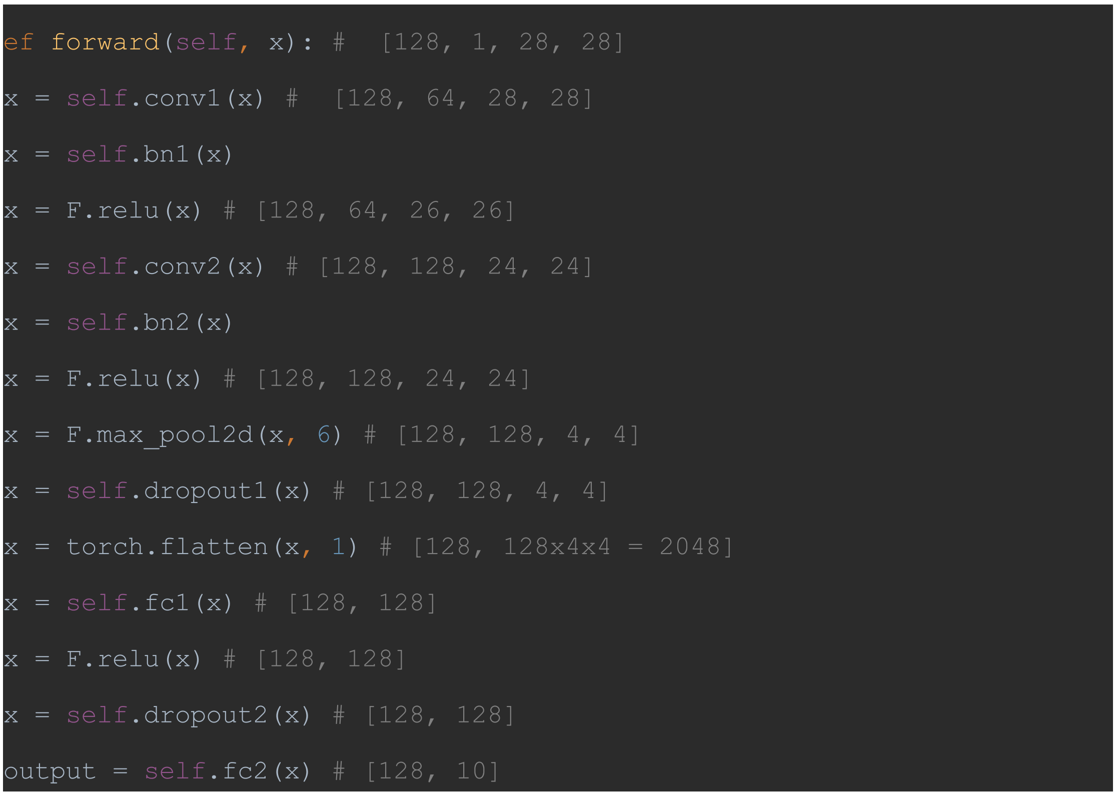
- Calculate number of weights on each layer



```
def __init__(self):
super(Net, self).__init__()
self.conv1 = nn.Conv2d(1, 64, 3, 1) # <- (3*3 + 1)*64 640
self.conv2 = nn.Conv2d(64, 128, 3, 1) # <- (3*3)*64*128 + 128 73856
self.bn1 = nn.BatchNorm2d(64) # Batch normalization after conv1
self.bn2 = nn.BatchNorm2d(128) # Batch normalization after conv2
self.dropout1 = nn.Dropout(0.25) # <- 0
self.dropout2 = nn.Dropout(0.5) # <- 0
self.fc1 = nn.Linear(2048, 128) # <- 2048*128 + 128 = 262272
self.fc2 = nn.Linear(128, 10) # <- 128*10 = 1290
```

- Calculate shape of tensors before and after each layer



```
ef forward(self, x): #  [128, 1, 28, 28]
x = self.conv1(x) #  [128, 64, 28, 28]
x = self.bn1(x)
x = F.relu(x) # [128, 64, 26, 26]
x = self.conv2(x) # [128, 128, 24, 24]
x = self.bn2(x)
x = F.relu(x) # [128, 128, 24, 24]
x = F.max_pool2d(x, 6) # [128, 128, 4, 4]
x = self.dropout1(x) # [128, 128, 4, 4]
x = torch.flatten(x, 1) # [128, 128x4x4 = 2048]
x = self.fc1(x) # [128, 128]
x = F.relu(x) # [128, 128]
x = self.dropout2(x) # [128, 128]
output = self.fc2(x) # [128, 10]
```

- Make model overfit the data. Show loss curves with overfit

Додав функціонал зменшення тренувального датасету до 200 прикладів кожного з класів

```
train_dataset = datasets.MNIST('./mnsit-dataset', train=True,
download=True,
```

```
if overfit:

samples_per_class = 200

# Define the transformation

transform = transforms.Compose([transforms.ToTensor(),
transforms.Normalize((0.5,), (0.5,))])

full_train_dataset = train_dataset

# Create a list to store the selected indices

selected_indices = []

# Iterate through each class and select samples

for class_label in range(10):

class_indices = [i for i, (_, label) in enumerate(full_train_dataset)
if label == class_label]

selected_indices.extend(random.sample(class_indices,
samples_per_class))

# Create a Subset of the training dataset with the selected indices

train_dataset = torch.utils.data.Subset(full_train_dataset,
selected_indices)
```

Результати плачевні

```
tensor([0.0000, 0.0000, 0.0019, 0.0000, 0.0000, 0.0000, 1.0000, 0.0000, 0.0010,        0.0000])

tensor([[   0,    0,    0,    0,    0,    0,  980,    0,    0,    0],

        [   0,    0,    0,    0,    0,    0, 1135,    0,    0,    0],

        [   0,    0,    2,    0,    0,    0, 1029,    0,    1,    0],

        [   2,    0,    0,    0,    0,    0, 1008,    0,    0,    0],

        [   5,    0,    5,    0,    0,    1,  967,    0,    4,    0],

        [   2,    0,    1,    0,    0,    0,  888,    0,    1,    0],

        [   0,    0,    0,    0,    0,    0,  958,    0,    0,    0],

        [  18,    0,    2,    0,    0,    1, 1006,    0,    0,    1],

        [   0,    0,    0,    0,    0,    0,  973,    0,    1,    0],

        [  24,    0,    1,    0,    0,    3,  974,    0,    7,    0]])
```

Test set: Average loss: 0.0181, Accuracy: 961/10000 (10%)

- Reduce model complexity (number parameters) with keeping accuracy add batch norm as well

Змінами вдалось зменшити кількість пераметрів **з 1,200,074 до 338,058 без впливу на точність**

Без батч нормалізації

Accuracy by class
tensor([0.9908, 0.9885, 0.9506, 0.9802, 0.9786, 0.9821, 0.9656, 0.9494, 0.9610, 0.9564])
Confusion matrix

```
tensor([[ 971,    0,    0,    0,    1,    0,    5,    1,    2,    0],
        [   0, 1122,    4,    2,    1,    1,    3,    1,    1,    0],
        [   5,    4,  981,   11,    3,    1,    0,   14,   13,    0],
        [   0,    0,    3,  990,    0,    6,    0,    3,    3,    5],
        [   1,    1,    2,    0,  961,    0,    3,    2,    1,   11],
        [   2,    1,    0,    5,    0,  876,    4,    1,    3,    0],
        [  10,    3,    1,    0,    5,    9,  925,    0,    5,    0],
        [   0,    3,   20,    4,    0,    1,    0,  976,    1,   23],
        [   5,    2,    4,    4,    4,    2,    5,    4,  936,    8],
        [   5,    3,    1,    8,    9,    4,    0,    7,    7,  965]])
```

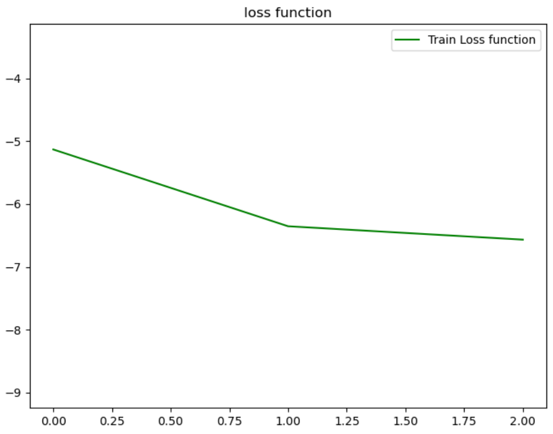
Test set: Average loss: **0.0007**, Accuracy: **9703/10000 (97%)**

Total params: 338,058

Trainable params: 338,058

Params size (MB): 1.29

Estimated Total Size (MB): 2.20



З батч нормалізацією

Accuracy by class

```
tensor([0.9949, 0.9956, 0.9874, 0.9881, 0.9898, 0.9922, 0.9812, 0.9776, 0.9856, 0.9693])
```

Confusion matrix

```
tensor([[ 975,    0,    0,    0,    0,    1,    2,    1,    1,    0],

        [   0, 1130,    4,    0,    0,    1,    0,    0,    0,    0],

        [   3,    2, 1019,    2,    0,    0,    0,    4,    2,    0],

        [   0,    0,    2,  998,    0,    4,    0,    2,    3,    1],

        [   0,    1,    2,    0,  972,    0,    1,    0,    2,    4],

        [   2,    0,    0,    3,    0,  885,    2,    0,    0,    0],

        [   6,    4,    0,    0,    2,    4,  940,    0,    2,    0],

        [   0,    2,   15,    0,    1,    1,    0, 1005,    1,    3],

        [   2,    1,    2,    2,    2,    1,    1,    1,  960,    2],

        [   3,    3,    0,    3,    6,    4,    0,    5,    7,  978]])
```

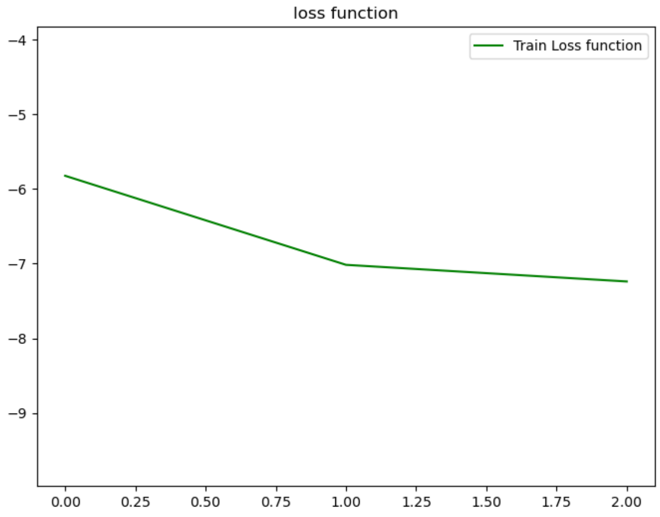
Test set: Average loss: **0.0003**, Accuracy: **9862/10000 (99%)**

Total params: 338,442

Trainable params: 338,442

Params size (MB): 1.29

Estimated Total Size (MB): 3.10



Висновок: застосування Батч нормалізації позитивно вплинула на точність, а саме маємо ріст на 2% до 99%