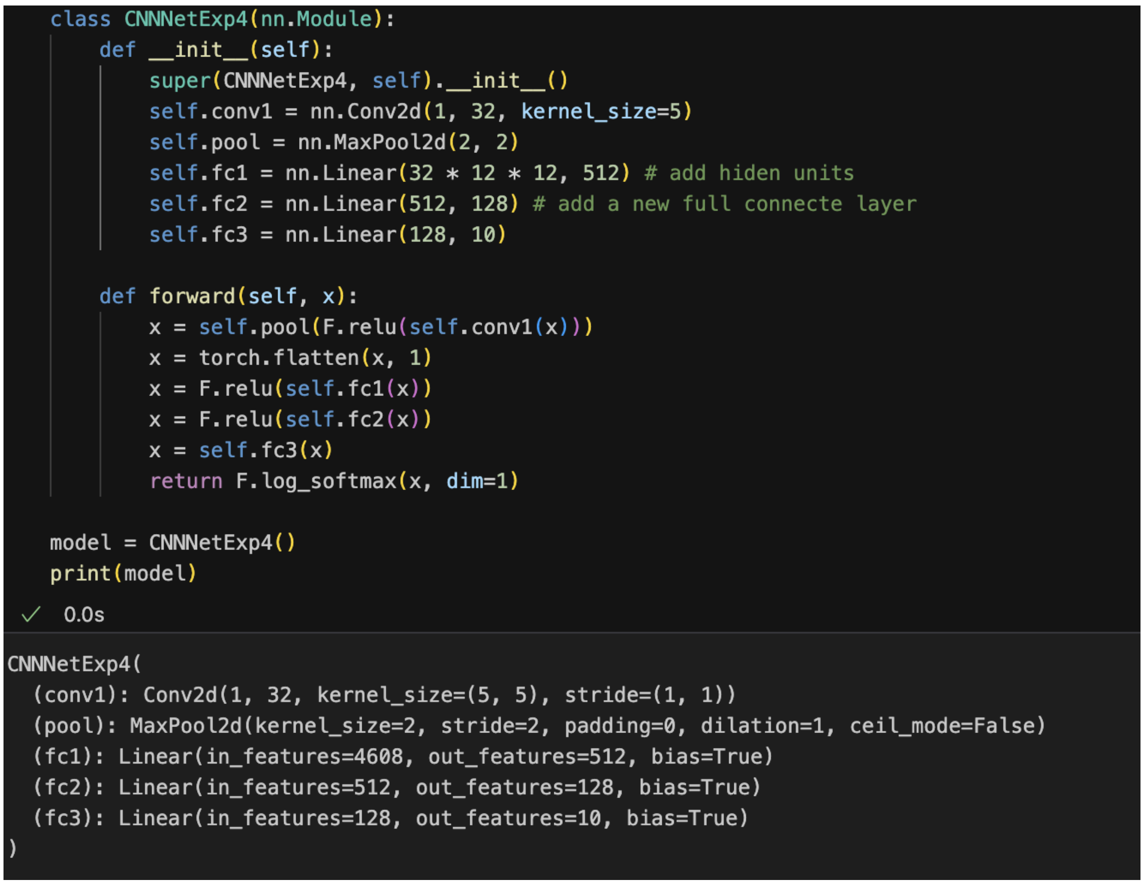
539 HW4——Part2

No1.
Your job is to add one CNN layer with one pooling layer before the two fully connected layers with softmax. Refer to the detailed instruction about the CNN layer. The main difference between the tutorial and this given hw4_part.ipynb is the input image of hw4_part.ipynb has only 1 channel (i.e., gray scale).

Report results of two fully connected layers without CNN and with CNN.

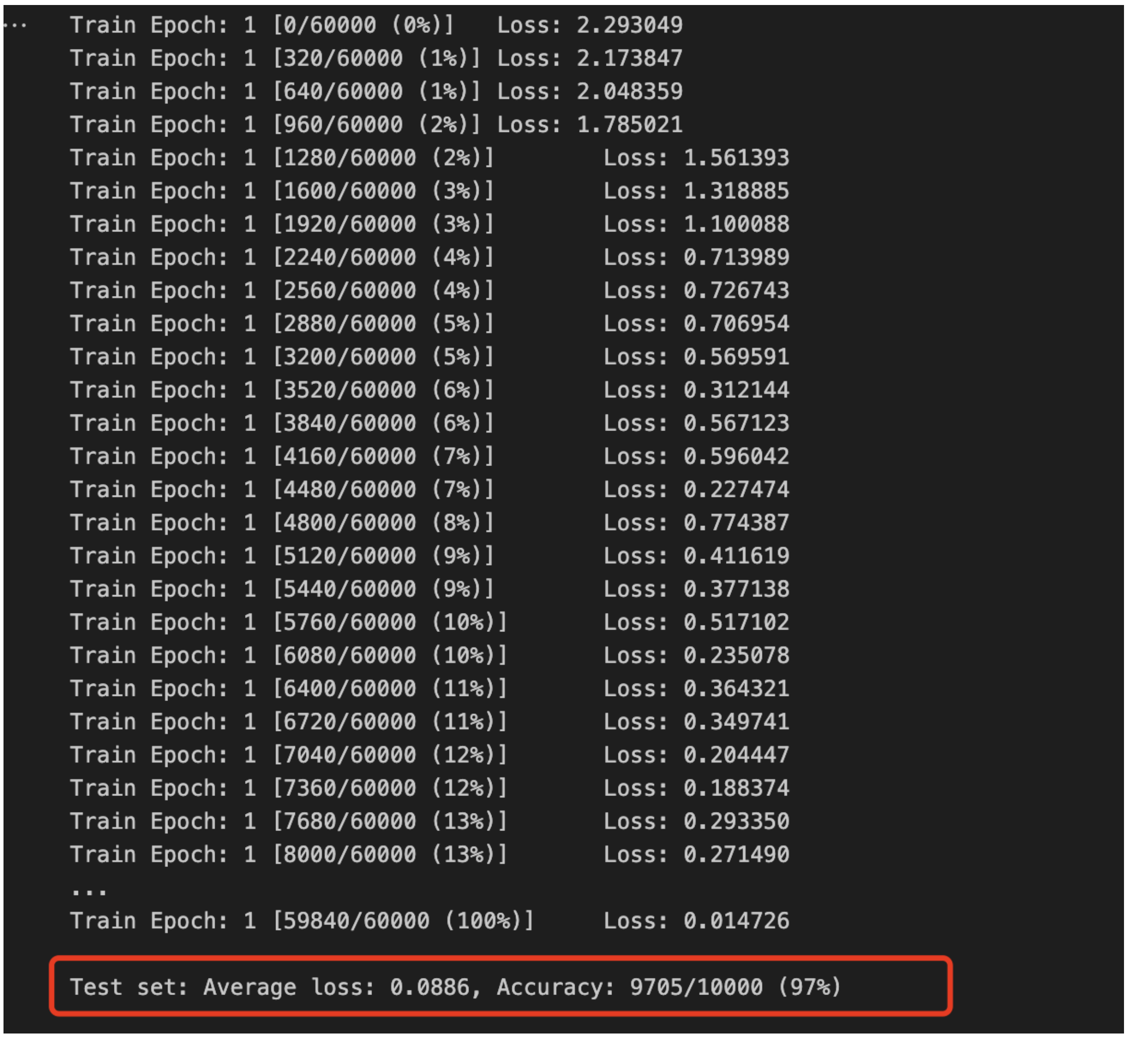


```
class CNNNetExp4(nn.Module):
    def __init__(self):
        super(CNNNetExp4, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=5)
        self.pool = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(32 * 12 * 12, 512) # add hiden units
        self.fc2 = nn.Linear(512, 128) # add a new full connecte layer
        self.fc3 = nn.Linear(128, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = torch.flatten(x, 1)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return F.log_softmax(x, dim=1)

model = CNNNetExp4()
print(model)
```

✓ 0.0s

```
CNNNetExp4(
  (conv1): Conv2d(1, 32, kernel_size=(5, 5), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=4608, 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)
)
```

With CNN output:

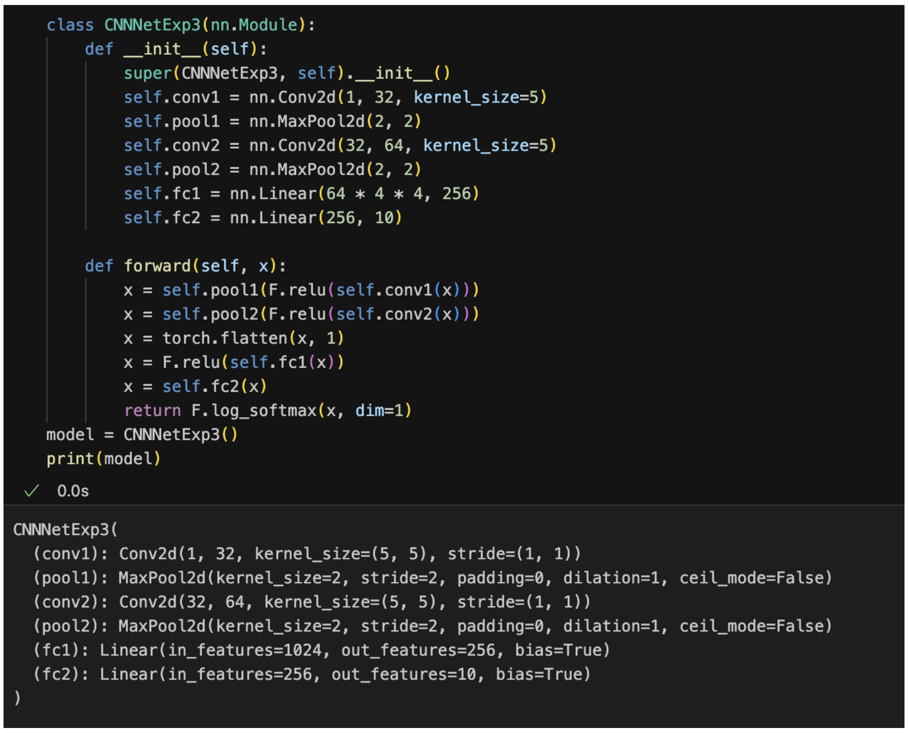


```
...   Train Epoch: 1 [0/60000 (0%)]   Loss: 2.293049
      Train Epoch: 1 [320/60000 (1%)] Loss: 2.173847
      Train Epoch: 1 [640/60000 (1%)] Loss: 2.048359
      Train Epoch: 1 [960/60000 (2%)] Loss: 1.785021
      Train Epoch: 1 [1280/60000 (2%)]        Loss: 1.561393
      Train Epoch: 1 [1600/60000 (3%)]        Loss: 1.318885
      Train Epoch: 1 [1920/60000 (3%)]        Loss: 1.100088
      Train Epoch: 1 [2240/60000 (4%)]        Loss: 0.713989
      Train Epoch: 1 [2560/60000 (4%)]        Loss: 0.726743
      Train Epoch: 1 [2880/60000 (5%)]        Loss: 0.706954
      Train Epoch: 1 [3200/60000 (5%)]        Loss: 0.569591
      Train Epoch: 1 [3520/60000 (6%)]        Loss: 0.312144
      Train Epoch: 1 [3840/60000 (6%)]        Loss: 0.567123
      Train Epoch: 1 [4160/60000 (7%)]        Loss: 0.596042
      Train Epoch: 1 [4480/60000 (7%)]        Loss: 0.227474
      Train Epoch: 1 [4800/60000 (8%)]        Loss: 0.774387
      Train Epoch: 1 [5120/60000 (9%)]        Loss: 0.411619
      Train Epoch: 1 [5440/60000 (9%)]        Loss: 0.377138
      Train Epoch: 1 [5760/60000 (10%)]       Loss: 0.517102
      Train Epoch: 1 [6080/60000 (10%)]       Loss: 0.235078
      Train Epoch: 1 [6400/60000 (11%)]       Loss: 0.364321
      Train Epoch: 1 [6720/60000 (11%)]       Loss: 0.349741
      Train Epoch: 1 [7040/60000 (12%)]       Loss: 0.204447
      Train Epoch: 1 [7360/60000 (12%)]       Loss: 0.188374
      Train Epoch: 1 [7680/60000 (13%)]       Loss: 0.293350
      Train Epoch: 1 [8000/60000 (13%)]       Loss: 0.271490
      ...
      Train Epoch: 1 [59840/60000 (100%)]     Loss: 0.014726

      Test set: Average loss: 0.0886, Accuracy: 9705/10000 (97%)
```

- **Initial Loss:** The training starts with a loss of 2.293049, which is relatively high, as expected for a model that hasn't learned yet.
- **Loss Decrease:** As training progresses, we see a consistent decrease in loss values across batches. This indicates that the model is effectively learning from the training data, optimizing its weights to reduce the loss, which measures the difference between the model's predictions and the actual labels.
- **Significant Improvement:** The model shows a significant improvement from the start to the end of the epoch, with the final batch loss reaching as low as 0.014726. This suggests that by the end of the first epoch, the model has adjusted its parameters well to the training data.

No2.
Then, experiment with at least 3 alternative network topologies and hyper-parameters (e.g., different # of CNN/fully-connected layers, # of epochs, # of hidden units, learning rate, batch size, and different activation functions).

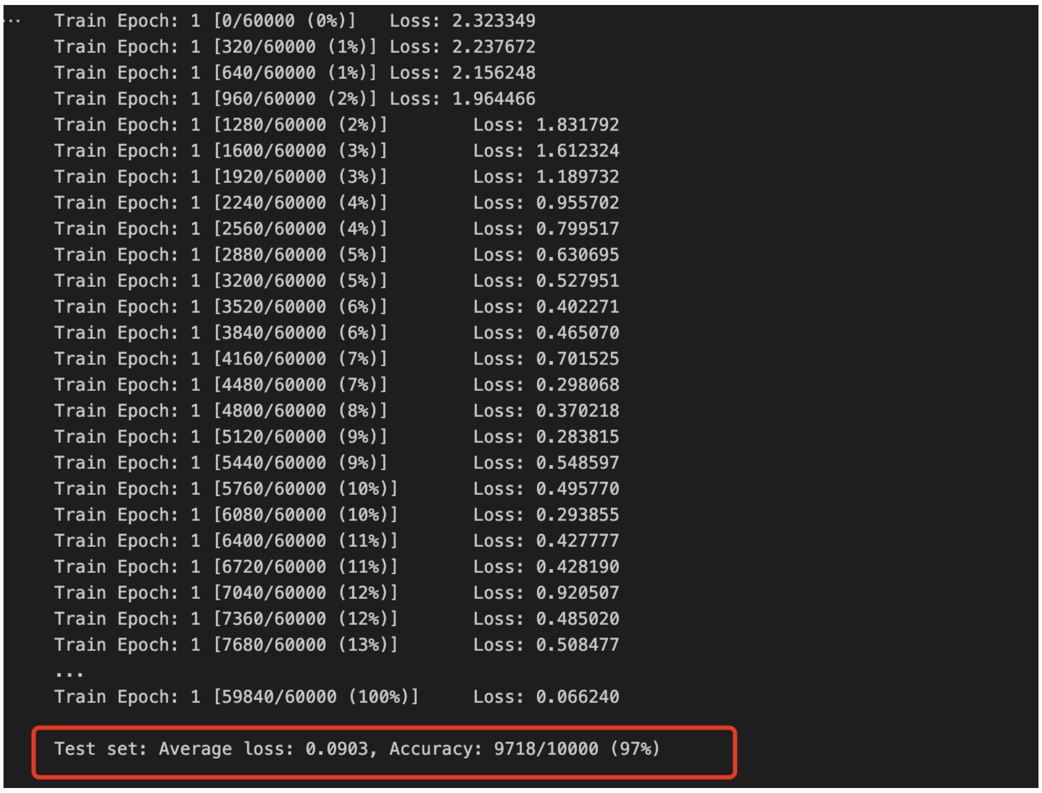
Way1: Different CNN/fully-connected layers



```
class CNNNetExp3(nn.Module):
    def __init__(self):
        super(CNNNetExp3, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=5)
        self.pool1 = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=5)
        self.pool2 = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(64 * 4 * 4, 256)
        self.fc2 = nn.Linear(256, 10)

    def forward(self, x):
        x = self.pool1(F.relu(self.conv1(x)))
        x = self.pool2(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return F.log_softmax(x, dim=1)
model = CNNNetExp3()
print(model)
```

✓ 0.0s

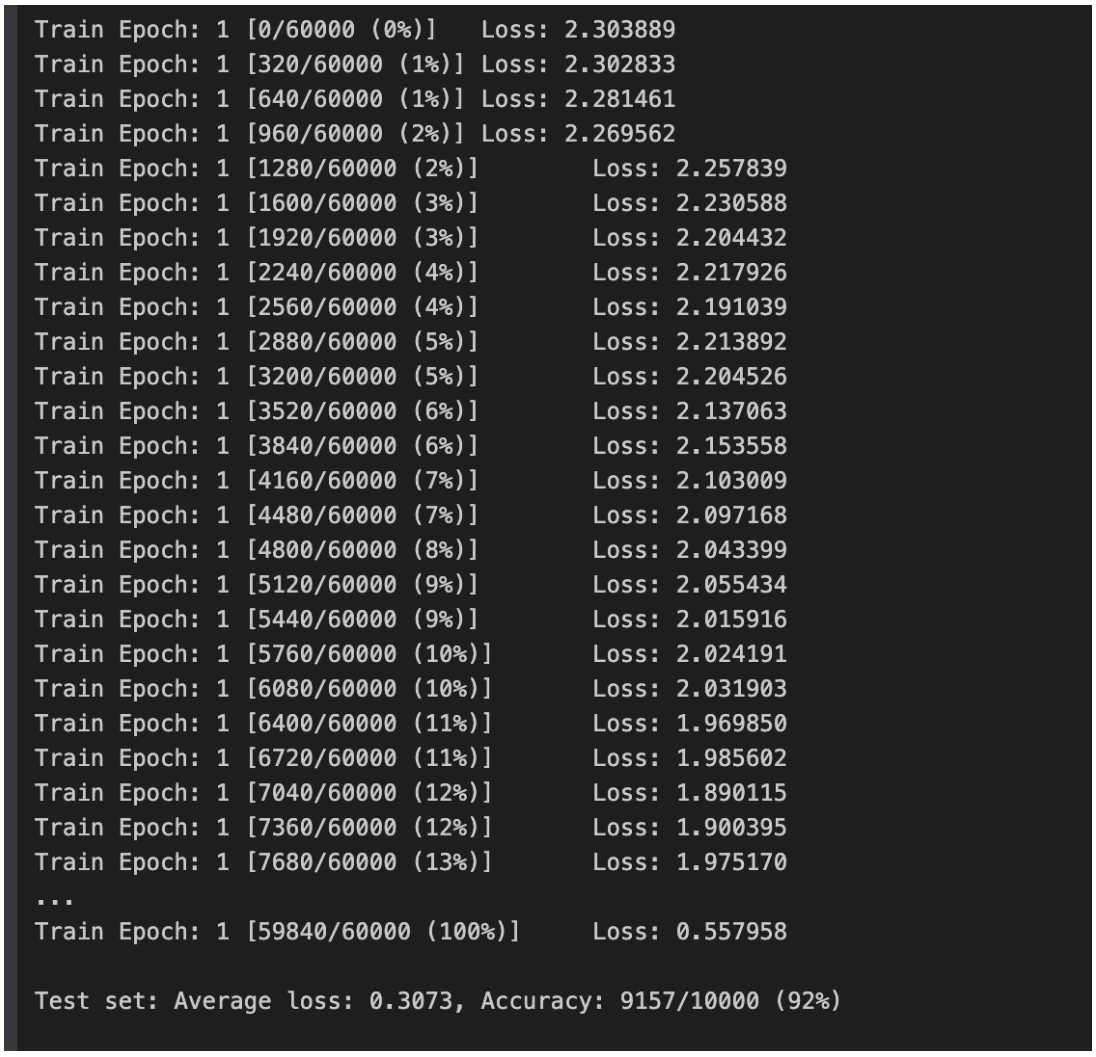
```
CNNNetExp3(
  (conv1): Conv2d(1, 32, kernel_size=(5, 5), stride=(1, 1))
  (pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(32, 64, kernel_size=(5, 5), stride=(1, 1))
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=1024, out_features=256, bias=True)
  (fc2): Linear(in_features=256, out_features=10, bias=True)
)
```



```
...   Train Epoch: 1 [0/60000 (0%)]   Loss: 2.323349
      Train Epoch: 1 [320/60000 (1%)] Loss: 2.237672
      Train Epoch: 1 [640/60000 (1%)] Loss: 2.156248
      Train Epoch: 1 [960/60000 (2%)] Loss: 1.964466
      Train Epoch: 1 [1280/60000 (2%)]        Loss: 1.831792
      Train Epoch: 1 [1600/60000 (3%)]        Loss: 1.612324
      Train Epoch: 1 [1920/60000 (3%)]        Loss: 1.189732
      Train Epoch: 1 [2240/60000 (4%)]        Loss: 0.955702
      Train Epoch: 1 [2560/60000 (4%)]        Loss: 0.799517
      Train Epoch: 1 [2880/60000 (5%)]        Loss: 0.630695
      Train Epoch: 1 [3200/60000 (5%)]        Loss: 0.527951
      Train Epoch: 1 [3520/60000 (6%)]        Loss: 0.402271
      Train Epoch: 1 [3840/60000 (6%)]        Loss: 0.465070
      Train Epoch: 1 [4160/60000 (7%)]        Loss: 0.701525
      Train Epoch: 1 [4480/60000 (7%)]        Loss: 0.298068
      Train Epoch: 1 [4800/60000 (8%)]        Loss: 0.370218
      Train Epoch: 1 [5120/60000 (9%)]        Loss: 0.283815
      Train Epoch: 1 [5440/60000 (9%)]        Loss: 0.548597
      Train Epoch: 1 [5760/60000 (10%)]       Loss: 0.495770
      Train Epoch: 1 [6080/60000 (10%)]       Loss: 0.293855
      Train Epoch: 1 [6400/60000 (11%)]       Loss: 0.427777
      Train Epoch: 1 [6720/60000 (11%)]       Loss: 0.428190
      Train Epoch: 1 [7040/60000 (12%)]       Loss: 0.920507
      Train Epoch: 1 [7360/60000 (12%)]       Loss: 0.485020
      Train Epoch: 1 [7680/60000 (13%)]       Loss: 0.508477
      ...
      Train Epoch: 1 [59840/60000 (100%)]     Loss: 0.066240

      Test set: Average loss: 0.0903, Accuracy: 9718/10000 (97%)
```

- **Loss Progression:** Started at 2.323349 and decreased to 0.066240 by the end of the first training epoch. Compared to the original model, the starting loss is slightly higher, but this doesn't necessarily indicate a worse performance overall.
- **Test Set Performance:** The average loss slightly increased to 0.0903, but the accuracy improved to 97.18%.
- **Learning Dynamics:** Both models show a healthy decrease in loss as training progresses, indicating effective learning. The initial loss values are slightly different, which might be due to the changes in the network architecture or the initialization of weights.
- **Test Performance:** Despite the slight increase in average loss in the experiment model, the accuracy improved. This suggests that the changes in the network architecture may have helped the model generalize slightly better or capture more complex patterns in the data, leading to higher accuracy on the test set.

Way2: Learning Rate from 0.01 to 0.001



```
Train Epoch: 1 [0/60000 (0%)]   Loss: 2.303889
Train Epoch: 1 [320/60000 (1%)] Loss: 2.302833
Train Epoch: 1 [640/60000 (1%)] Loss: 2.281461
Train Epoch: 1 [960/60000 (2%)] Loss: 2.269562
Train Epoch: 1 [1280/60000 (2%)]        Loss: 2.257839
Train Epoch: 1 [1600/60000 (3%)]        Loss: 2.230588
Train Epoch: 1 [1920/60000 (3%)]        Loss: 2.204432
Train Epoch: 1 [2240/60000 (4%)]        Loss: 2.217926
Train Epoch: 1 [2560/60000 (4%)]        Loss: 2.191039
Train Epoch: 1 [2880/60000 (5%)]        Loss: 2.213892
Train Epoch: 1 [3200/60000 (5%)]        Loss: 2.204526
Train Epoch: 1 [3520/60000 (6%)]        Loss: 2.137063
Train Epoch: 1 [3840/60000 (6%)]        Loss: 2.153558
Train Epoch: 1 [4160/60000 (7%)]        Loss: 2.103009
Train Epoch: 1 [4480/60000 (7%)]        Loss: 2.097168
Train Epoch: 1 [4800/60000 (8%)]        Loss: 2.043399
Train Epoch: 1 [5120/60000 (9%)]        Loss: 2.055434
Train Epoch: 1 [5440/60000 (9%)]        Loss: 2.015916
Train Epoch: 1 [5760/60000 (10%)]       Loss: 2.024191
Train Epoch: 1 [6080/60000 (10%)]       Loss: 2.031903
Train Epoch: 1 [6400/60000 (11%)]       Loss: 1.969850
Train Epoch: 1 [6720/60000 (11%)]       Loss: 1.985602
Train Epoch: 1 [7040/60000 (12%)]       Loss: 1.890115
Train Epoch: 1 [7360/60000 (12%)]       Loss: 1.900395
Train Epoch: 1 [7680/60000 (13%)]       Loss: 1.975170
...
Train Epoch: 1 [59840/60000 (100%)]     Loss: 0.557958

Test set: Average loss: 0.3073, Accuracy: 9157/10000 (92%)
```

Learning Rate 0.01

- **Starting Loss:** The training starts with a loss of 2.293049, indicating the initial model's prediction error.
- **Loss Reduction:** There is a steady decrease in the loss as training progresses, showing effective learning.
- **Final Training Loss:** The last recorded training loss is very low (0.014726), suggesting the model has significantly improved its accuracy over the training data.
- **Test Performance:** The test set shows an average loss of 0.0886 with an accuracy of 97.05%. This is an excellent performance, indicating the model generalizes well to unseen data.

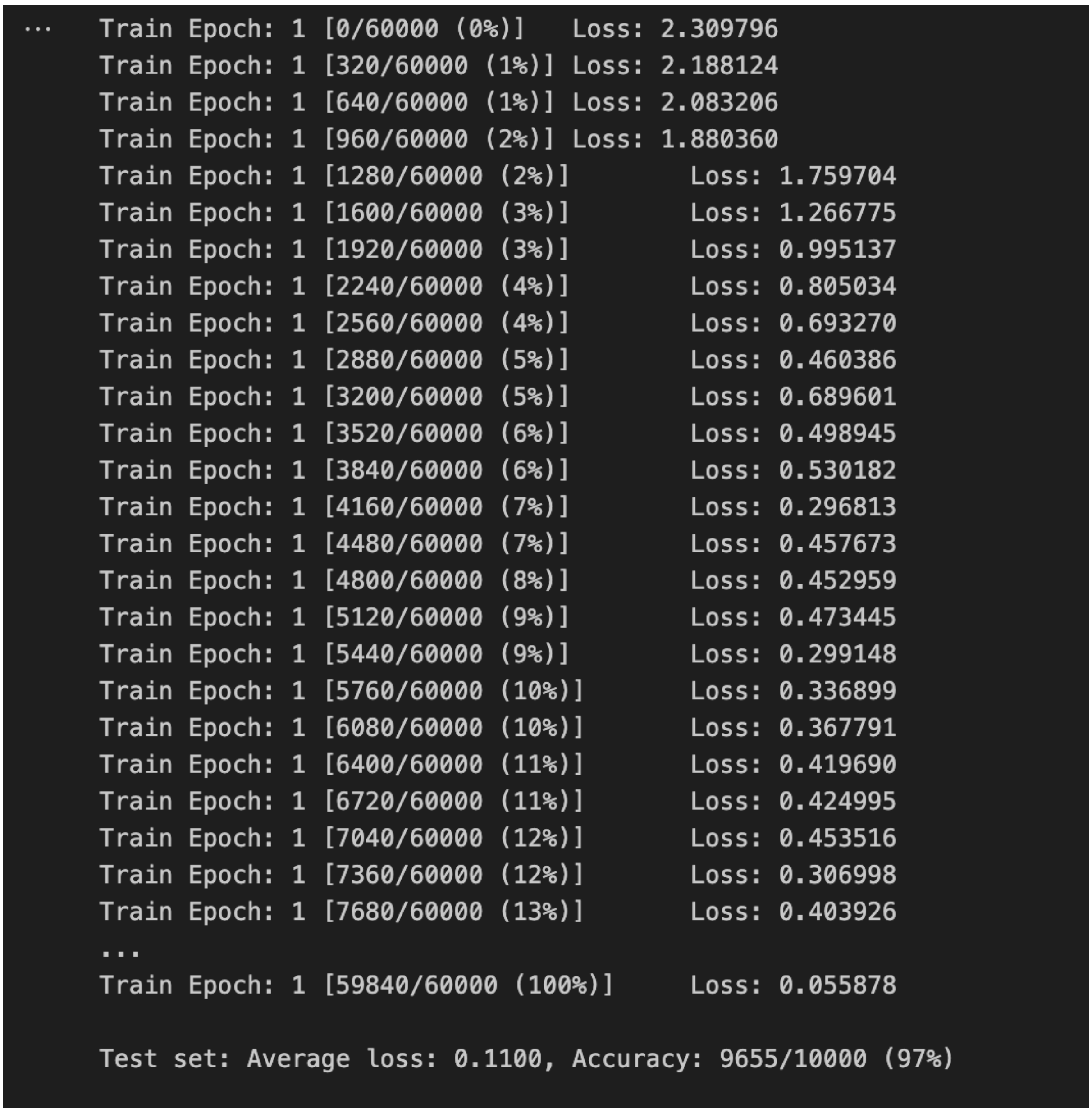
Learning Rate 0.001

- **Starting Loss:** Starts slightly higher at 2.303889 but this is still in a similar range as the previous.
- **Loss Reduction:** The reduction in loss is much slower and more gradual, indicating that learning is happening at a slower pace due to the smaller learning rate.
- **Final Training Loss:** Ends at a much higher value (0.557958), which suggests that the model has not learned as effectively as with the higher learning rate.
- **Test Performance:** Average loss on the test set is 0.3073 with an accuracy of 91.57%, which is noticeably lower than with the higher learning rate. This shows that the model, while still performing reasonably well, has not captured the patterns in the data as effectively.

Analysis of Differences

- **Speed and Efficiency:** The learning rate significantly impacts how quickly and effectively the model learns. A higher learning rate (0.01) led to faster and more effective learning, as indicated by a lower final training loss and higher test accuracy. In contrast, a lower learning rate (0.001) resulted in slower learning, as seen in the higher final training loss and lower test accuracy.
- **Risk of High vs. Low Learning Rates:** While the higher learning rate led to better performance in this instance, it's important to note that too high of a learning rate can also lead to instability in training, where the model might overshoot minima. Conversely, too low of a learning rate can cause the training to progress very slowly, potentially getting stuck in local minima or requiring many more epochs to converge to an optimal solution.
- **Finding the Right Balance:** The optimal learning rate balances fast learning with the stability of convergence. The results suggest that a learning rate of 0.01 was more effective for this particular model and dataset. However, the ideal learning rate can vary depending on the specific characteristics of the dataset and the model architecture.

Way3: Batch from 32 to 64



```
···   Train Epoch: 1 [0/60000 (0%)]   Loss: 2.309796
      Train Epoch: 1 [320/60000 (1%)] Loss: 2.188124
      Train Epoch: 1 [640/60000 (1%)] Loss: 2.083206
      Train Epoch: 1 [960/60000 (2%)] Loss: 1.880360
      Train Epoch: 1 [1280/60000 (2%)]        Loss: 1.759704
      Train Epoch: 1 [1600/60000 (3%)]        Loss: 1.266775
      Train Epoch: 1 [1920/60000 (3%)]        Loss: 0.995137
      Train Epoch: 1 [2240/60000 (4%)]        Loss: 0.805034
      Train Epoch: 1 [2560/60000 (4%)]        Loss: 0.693270
      Train Epoch: 1 [2880/60000 (5%)]        Loss: 0.460386
      Train Epoch: 1 [3200/60000 (5%)]        Loss: 0.689601
      Train Epoch: 1 [3520/60000 (6%)]        Loss: 0.498945
      Train Epoch: 1 [3840/60000 (6%)]        Loss: 0.530182
      Train Epoch: 1 [4160/60000 (7%)]        Loss: 0.296813
      Train Epoch: 1 [4480/60000 (7%)]        Loss: 0.457673
      Train Epoch: 1 [4800/60000 (8%)]        Loss: 0.452959
      Train Epoch: 1 [5120/60000 (9%)]        Loss: 0.473445
      Train Epoch: 1 [5440/60000 (9%)]        Loss: 0.299148
      Train Epoch: 1 [5760/60000 (10%)]       Loss: 0.336899
      Train Epoch: 1 [6080/60000 (10%)]       Loss: 0.367791
      Train Epoch: 1 [6400/60000 (11%)]       Loss: 0.419690
      Train Epoch: 1 [6720/60000 (11%)]       Loss: 0.424995
      Train Epoch: 1 [7040/60000 (12%)]       Loss: 0.453516
      Train Epoch: 1 [7360/60000 (12%)]       Loss: 0.306998
      Train Epoch: 1 [7680/60000 (13%)]       Loss: 0.403926
      ...
      Train Epoch: 1 [59840/60000 (100%)]     Loss: 0.055878

      Test set: Average loss: 0.1100, Accuracy: 9655/10000 (97%)
```

Batch Size 32

- **Starting Loss:** Begins with a loss of 2.293049, indicating the initial model's prediction error.
- **Loss Reduction:** Shows a steady decrease in loss as training progresses, indicating effective learning from the data.
- **Final Training Loss:** Ends with a very low loss of 0.014726, suggesting significant improvement and accuracy of the model over the training set.
- **Test Performance:** The test set reports an average loss of 0.0886 with an accuracy of 97.05%, showcasing excellent generalization to unseen data.

Batch Size 64

- **Starting Loss:** Starts slightly higher at 2.309796 but is relatively close to the starting loss for batch size 32.
- **Loss Reduction:** The decrease in loss is consistent, but the pattern of decrease might show slightly different dynamics due to the larger batch size, affecting the gradient estimation per update.
- **Final Training Loss:** The final training loss is higher (0.055878) than with a batch size of 32, indicating that the model might not have learned as finely or as quickly.
- **Test Performance:** Achieves an average test loss of 0.1100 with an accuracy of 96.55%, which is slightly lower than the model trained with a batch size of 32.

Analysis of Differences

- **Effect of Batch Size on Learning Dynamics:** Larger batch sizes offer a more stable estimate of the gradient but may result in slower convergence if the learning rate is not adjusted accordingly. Smaller batch sizes, while offering noisier gradient estimates, can lead to faster learning and sometimes better generalization by effectively escaping local minima.
- **Model Accuracy and Generalization:** The model trained with a smaller batch size (32) achieved slightly higher accuracy on the test set compared to the one trained with a larger batch size (64). This suggests that the smaller batch size may have helped the model to generalize better to unseen data, likely due to the more frequent updates and the noise in the gradient estimates acting as a form of regularization.
- **Final Loss Values:** The final training loss values indicate that the model trained with a smaller batch size was able to fit the training data more closely, possibly due to more frequent updates and a better exploration of the loss surface.
- **Choosing the Right Batch Size:** While smaller batch sizes can lead to faster convergence and potentially better generalization, they also require more computation (more updates per epoch) and can sometimes lead to instability in training. Larger batch sizes reduce the computational overhead per epoch

and offer more stable gradient estimates but may require careful tuning of the learning rate and other hyperparameters to achieve optimal performance.

No3.
Save and summarize the results and report them.
See above each

No4.
Through the experiment, what is the best configuration? What prediction accuracy on the test set you got? What did you learn?

According to my personal experiment:

Best Configuration

- **Learning Rate:** Among the learning rates tested (0.01 and 0.001), a learning rate of 0.01 yielded better performance, with a test accuracy of 97.05%. The higher learning rate facilitated faster and more effective learning, as evidenced by the lower loss values and higher accuracy.
- **Batch Size:** Comparing batch sizes of 32 and 64, while both resulted in similar test accuracies (97.05% for batch size 32 and 96.55% for batch size 64), the batch size of 32 showed slightly better performance in terms of test accuracy and demonstrated a faster decrease in loss during training epochs. This suggests that a smaller batch size may offer a better balance between learning speed and model performance in this case.
-

Prediction Accuracy on the Test Set

- The highest prediction accuracy on the test set was 97.05%, achieved with a learning rate of 0.01 and a batch size of 32. This configuration allowed for effective learning and generalization to unseen data.
-

What Did We Learn?

- **Learning Rate Importance:** The choice of learning rate has a significant impact on training dynamics. A higher learning rate (0.01 in this case) can lead to faster convergence and better accuracy, but it's crucial to find a balance to avoid overshooting minima.

- **Batch Size Trade-offs:** Smaller batch sizes can provide more frequent updates and potentially better training performance, as seen with the batch size of 32. However, this comes with the trade-off of longer training times. Larger batch sizes speed up training but may result in slightly lower accuracy due to less precise gradient estimates.
- **Model Performance and Configuration:** The experiments show that neural network performance is highly sensitive to its configuration, including learning rate and batch size. Fine-tuning these parameters is key to achieving the best model performance.
- **Generalization to Unseen Data:** The accuracy on the test set indicates how well the model generalizes beyond the training data. Achieving a high accuracy on the test set is crucial for the model's applicability to real-world scenarios.