ST311 Summative assignment

https://colab.research.google.com/drive/1aUJZukmQD4Kv6IVgt-7e3nUiZKsKkvr6?usp=sharing

Preface:

- *All data values will be given rounded to 3 decimal places when in writing*
- *Validation accuracy will be used throughout in order to give the best idea of how the models and techniques work relative to eachother*

1.A

Importing and gpu check

Importing necessary libraries

```
import torch
import torch.nn as nn
import matplotlib.pyplot as plt
from torchvision import datasets, transforms
import torch.optim as optim   # SGD
import time
```

Checking if a GPU is available. The program will run using the GPU if it's available, otherwise it will use the CPU.

```
cuda = torch.cuda.is_available()
cuda   # returns a Boolean indicating availability of GPU
```

```
False
```

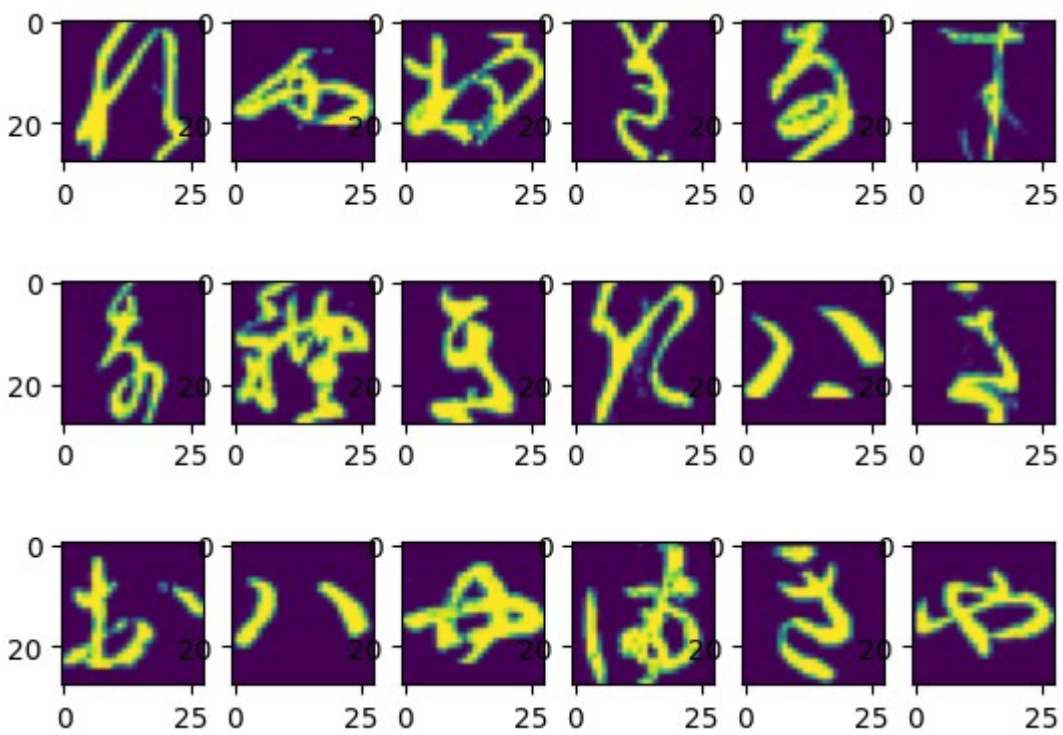
The Kuzushiji-MNIST data:

getting and viewing the data

```
mnist_images = datasets.KMNIST('data', train=True, download=True)   # downloading the data

for k, (image, label) in enumerate(mnist_images):   # viewing some of the data
    if k >= 18:
        break
    plt.subplot(3, 6, k+1)
    plt.imshow(image)
```

```
Downloading http://codh.rois.ac.jp/kmnist/dataset/kmnist/train-images-idx3-ubyte.gz
Downloading https://codh.rois.ac.jp/kmnist/dataset/kmnist/train-images-idx3-ubyte.gz to data/KM
100%|██████████| 18.2M/18.2M [00:20<00:00, 903kB/s]
```

```
Extracting data/KMNIST/raw/train-images-idx3-ubyte.gz to data/KMNIST/raw

Downloading http://codh.rois.ac.jp/kmnist/dataset/kmnist/train-labels-idx1-ubyte.gz
Downloading https://codh.rois.ac.jp/kmnist/dataset/kmnist/train-labels-idx1-ubyte.gz to data/KM
100%|██████████| 29.5k/29.5k [00:00<00:00, 179kB/s]
Extracting data/KMNIST/raw/train-labels-idx1-ubyte.gz to data/KMNIST/raw

Downloading http://codh.rois.ac.jp/kmnist/dataset/kmnist/t10k-images-idx3-ubyte.gz
Downloading https://codh.rois.ac.jp/kmnist/dataset/kmnist/t10k-images-idx3-ubyte.gz to data/KMN
100%|██████████| 3.04M/3.04M [00:03<00:00, 967kB/s]
Extracting data/KMNIST/raw/t10k-images-idx3-ubyte.gz to data/KMNIST/raw

Downloading http://codh.rois.ac.jp/kmnist/dataset/kmnist/t10k-labels-idx1-ubyte.gz
Downloading https://codh.rois.ac.jp/kmnist/dataset/kmnist/t10k-labels-idx1-ubyte.gz to data/KMN
100%|██████████| 5.12k/5.12k [00:00<00:00, 10.6MB/s]
Extracting data/KMNIST/raw/t10k-labels-idx1-ubyte.gz to data/KMNIST/raw
```



Convert to images to tensor format. The data has been downloaded in the previous code chunk `download=True`, so no need to download again here.

```
mnist_train = datasets.KMNIST('data', train=True, transform=transforms.ToTensor())
mnist_train = list(mnist_train)   # creating training data
mnist_val = datasets.KMNIST('data', train=False, transform=transforms.ToTensor())
mnist_val = list(mnist_val)   # creating validation data

print(mnist_train[0][0].shape)   # checking image size
```

```
torch.Size([1, 28, 28])
```

Creating the models

```
# Plain NN (3-layers)

model = nn.Sequential(nn.Flatten(),
                      nn.Linear(28*28, 256),
```

```
                        nn.ReLU(),
                        nn.Linear(256, 256),
                        nn.ReLU(),
                        nn.Linear(256, 10))

# NN with dropout (after ReLU)

model_drop = nn.Sequential(nn.Flatten(),
                           nn.Linear(28*28, 256),
                           nn.ReLU(),
                           nn.Dropout(),   # keeping p=0.5
                           nn.Linear(256, 256),
                           nn.ReLU(),
                           nn.Linear(256, 10))

# NN with batchnorm (before the ReLU)

model_bn = nn.Sequential(nn.Flatten(),
                         nn.Linear(28*28, 256),
                         nn.BatchNorm1d(256),
                         nn.ReLU(),
                         nn.Linear(256, 256),
                         nn.BatchNorm1d(256),
                         nn.ReLU(),
                         nn.Linear(256, 10))

# 3-layer NN with dropout and batchnorm

model_both = nn.Sequential(nn.Flatten(),
                           nn.Linear(28*28, 256),
                           nn.BatchNorm1d(256),
                           nn.ReLU(),
                           nn.Dropout(),   # keeping p=0.5
                           nn.Linear(256, 256),
                           nn.BatchNorm1d(256),
                           nn.ReLU(),
                           nn.Linear(256, 10))

# 5-layer NN with dropout and batchnorm (created after deciding to use dropout and batchnorm)

model_five_layer = nn.Sequential(nn.Flatten(),
                                 nn.Linear(28*28, 512),
                                 nn.BatchNorm1d(512),
                                 nn.ReLU(),
                                 nn.Dropout(),   # keeping p=0.5
                                 nn.Linear(512, 512),
                                 nn.BatchNorm1d(512),
                                 nn.ReLU(),
                                 nn.Dropout(),
                                 nn.Linear(512, 256),
                                 nn.BatchNorm1d(256),
                                 nn.ReLU(),
                                 nn.Dropout(),
                                 nn.Linear(256, 256),
                                 nn.BatchNorm1d(256),
                                 nn.ReLU(),
                                 nn.Linear(256, 10))
```

Initialising weights

Initialising weights

```
# chosen init to help avoid vanishing/exploding gradient problem and improve efficiency
def init_weights(m):
    "Initialise weights to the linear layer"
    if type(m) == nn.Linear:
        # nn.init.normal_(tensor=m.weight, mean=0.0,std=0.001)
        # nn.init.xavier_normal_(tensor=m.weight)
        # nn.init.xavier_uniform_(tensor=m.weight)
        # nn.init.kaiming_normal_(tensor=m.weight)
        nn.init.kaiming_uniform_(tensor=m.weight)
          # to avoid the dead neuron problem (as ReLU outputs zero for negative number)

# applying weight initialisation to the models
model.apply(init_weights)
model_drop.apply(init_weights)
model_bn.apply(init_weights)
model_both.apply(init_weights)
model_five_layer.apply(init_weights)
```

```
Sequential(
  (0): Flatten(start_dim=1, end_dim=-1)
  (1): Linear(in_features=784, out_features=512, bias=True)
  (2): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (3): ReLU()
  (4): Dropout(p=0.5, inplace=False)
  (5): Linear(in_features=512, out_features=512, bias=True)
  (6): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (7): ReLU()
  (8): Dropout(p=0.5, inplace=False)
  (9): Linear(in_features=512, out_features=256, bias=True)
  (10): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (11): ReLU()
  (12): Dropout(p=0.5, inplace=False)
  (13): Linear(in_features=256, out_features=256, bias=True)
  (14): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (15): ReLU()
  (16): Linear(in_features=256, out_features=10, bias=True)
)
```

Setting up device to do computation (device is either cpu or gpu)

```
# choosing between gpu and cpu depending on gpu availability (cuda)
device = torch.device("cuda" if cuda else "cpu")

# transfer model to device (transferring data to device in later code chunk)
model.to(device)
model_drop.to(device)
model_bn.to(device)
model_both.to(device)
model_five_layer.to(device)
```

```
Sequential(
  (0): Flatten(start_dim=1, end_dim=-1)
  (1): Linear(in_features=784, out_features=512, bias=True)
  (2): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (3): ReLU()
  (4): Dropout(p=0.5, inplace=False)
  (5): Linear(in_features=512, out_features=512, bias=True)
  (6): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
```

```
    (7): ReLU()
    (8): Dropout(p=0.5, inplace=False)
    (9): Linear(in_features=512, out_features=256, bias=True)
    (10): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): ReLU()
    (12): Dropout(p=0.5, inplace=False)
    (13): Linear(in_features=256, out_features=256, bias=True)
    (14): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (15): ReLU()
    (16): Linear(in_features=256, out_features=10, bias=True)
  )
```

Training

Creating training function

```
def train(model, data, batch_size=64, num_epochs=1, lr_choice=0.01, grad_clip=1.0):

    # Dataloader creates and iterator and fetches minibatches as needed
    train_loader = torch.utils.data.DataLoader(data,batch_size=batch_size,
                                               shuffle=True)

    # Pick loss and optimiser to use in training
    ## Inialise loss
    criterion = nn.CrossEntropyLoss()   # chosen loss function as it's very good for image classific

    ## Initialise optimizer (SGD vs SGD with momentum(=0.9) vs AdamW)
    # optimizer = optim.SGD(model.parameters(), weight_decay=0.01, lr=lr_choice)
    # optimizer = optim.SGD(model.parameters(), weight_decay=0.01, lr=lr_choice, momentum=0.9)
    optimizer = optim.AdamW(model.parameters(), lr=lr_choice)   # weight_decay=0.01 by default
      # performs best - porbably due to the adaptive learning rate and weight decay

    # Store evaluation metrics
    epch, losses, train_acc, val_acc = [], [], [], []

    # Training loop
    n = 0 # the number of epochs

    start_time = time.time()

    for epoch in range(num_epochs):

        model.train()   # training mode
        running_loss = 0.0

        # Iterate over the minibatches
        for imgs, labels in train_loader:

            # Transfer data to device
            imgs, labels = imgs.to(device), labels.to(device)

            # Forward pass: outputs logits for each image and computes the loss value
            out = model(imgs)
            loss = criterion(out, labels)

            running_loss += loss.item()   # record sum of losses over mini-batches in the current ep
```

```
            # Backward pass and update
            optimizer.zero_grad()                                   # reset gradients
            loss.backward()                                         # backprop
            nn.utils.clip_grad_norm_(model.parameters(), max_norm=grad_clip)   # clipping gradient
            optimizer.step()                                        # update each parameter




        # End of an epoch
        n += 1
        epch.append(n)
        losses.append(running_loss)
        train_acc.append(get_accuracy(model, train=True))   # compute training accuracy
        val_acc.append(get_accuracy(model, train=False))    # compute validation accuracy
        end_time = time.time()

        training_metric = (
          f'Epoch {epoch+1}/{num_epochs}, Training loss(ave. loss per minibatch): {round(running_los
          f'Time: {round(end_time - start_time,5)}s')
        print(training_metric)

    # Plotting
    plt.title("Training Curve")
    plt.plot(epch, losses, label="Train")
    plt.xlabel("epochs")
    plt.ylabel("Loss")
    plt.show()

    plt.title("Training Curve")
    plt.plot(epch, train_acc, label="Train")
    plt.plot(epch, val_acc, label="Validation")
    plt.xlabel("epochs")
    plt.ylabel("Training Accuracy")
    plt.legend(loc='best')
    plt.show()

    print("Final Training Accuracy: {}".format(train_acc[-1]))
    print("Final Validation Accuracy: {}".format(val_acc[-1]))
```

`get_accuracy` helper function

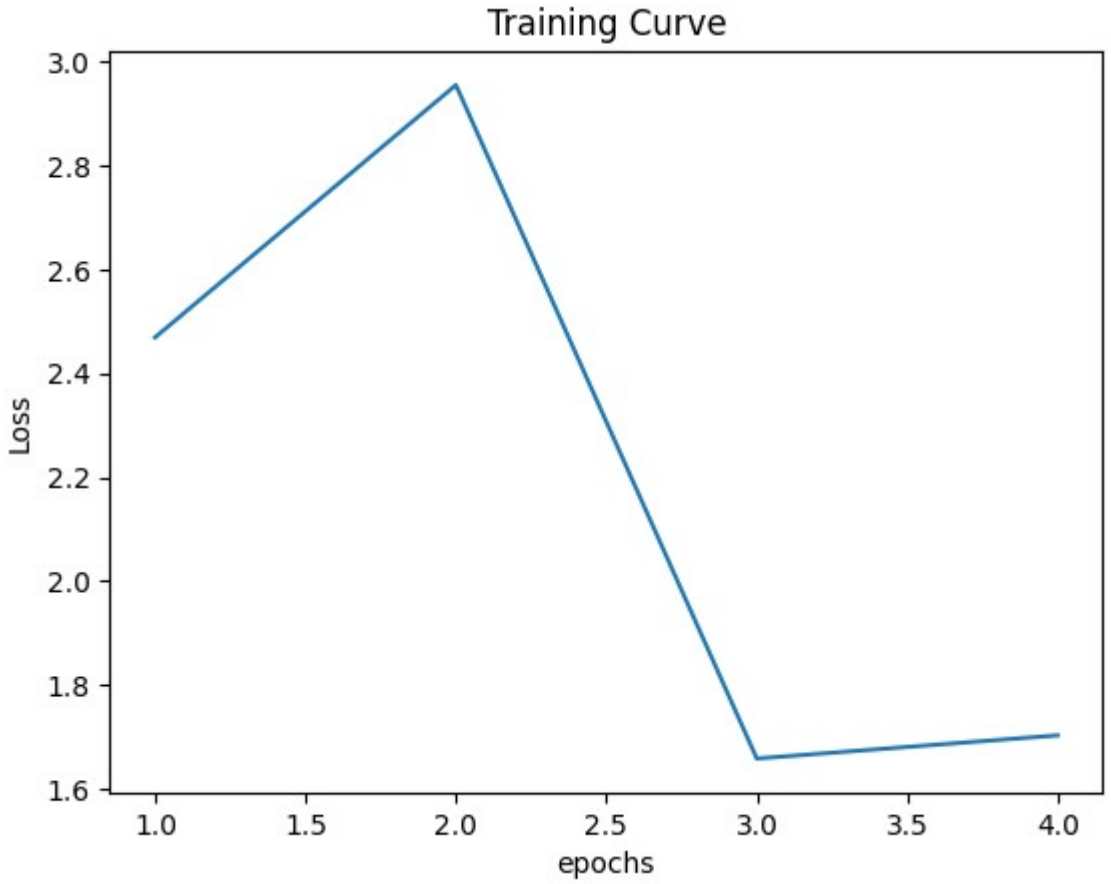
```
## to turn the probabilities into a discrete prediction, we will take the digit
## with the highest probability. Because of the way softmax is computed, the
## digit with the highest probability is the same as the digit with the
## (pre-activation) output value.

def get_accuracy(model, train=False, batch_size = 64):
    """
    Computes the accuracy of the training data (train=True), and validation
    data (train=False). Accuracy is defined as number correct predictions/ Total
    number of predictions made.

    This function is called from inside the training loop.

    Returns accuracy, a scalar.
    """
```

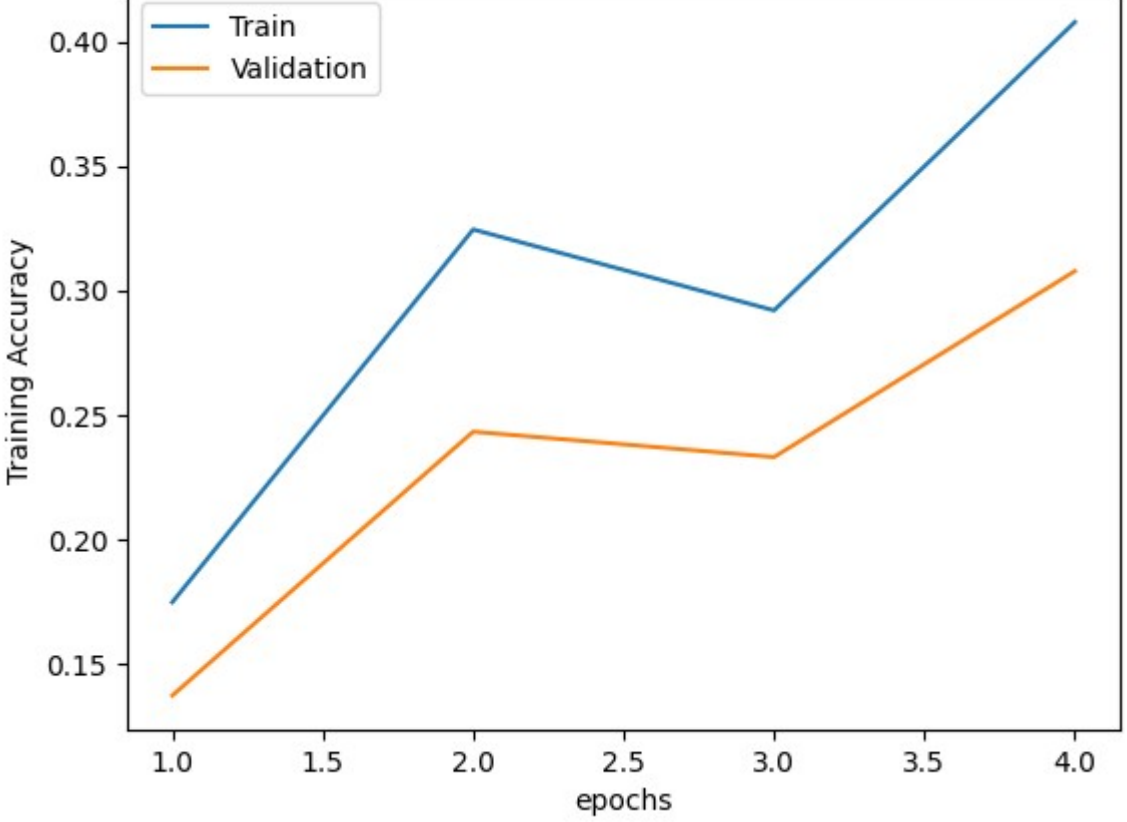
```
    model.eval()   # set model to evaluation mode

    data = mnist_train if train else mnist_val

    correct, total = 0, 0

    with torch.no_grad():
          for imgs, labels in torch.utils.data.DataLoader(data, batch_size=batch_size):
              # Transfer data to device
              imgs, labels = imgs.to(device), labels.to(device)

              output = model(imgs)
              pred = output.argmax(dim=1)   # index of class with highest probability
              correct += (pred == labels).sum().item()
              total += labels.size(0)

    return correct / total
```

Debugging

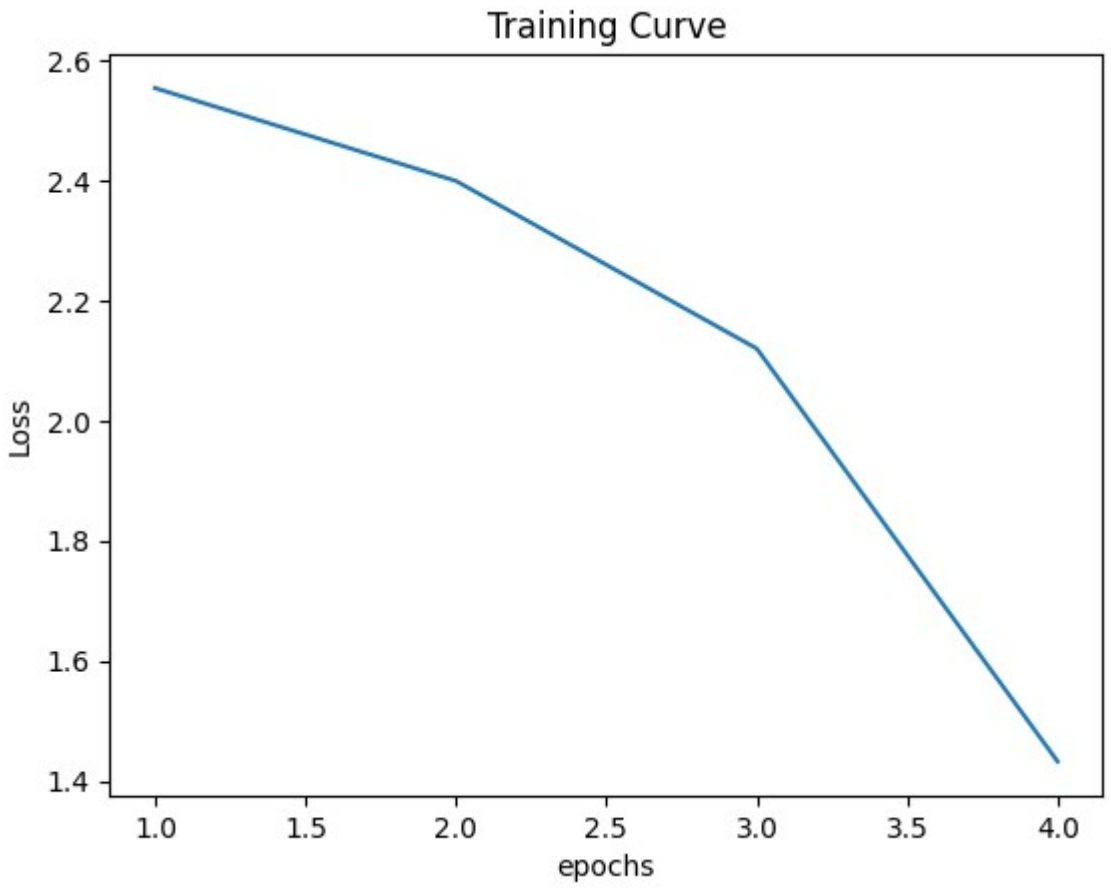
```
debug_data = mnist_train[:64] #sample size = batch size
train(model, debug_data, num_epochs=4)
train(model_drop, debug_data, num_epochs=4)
train(model_bn, debug_data, num_epochs=4)
train(model_both, debug_data, num_epochs=4)
train(model_five_layer, debug_data, num_epochs=4)
```

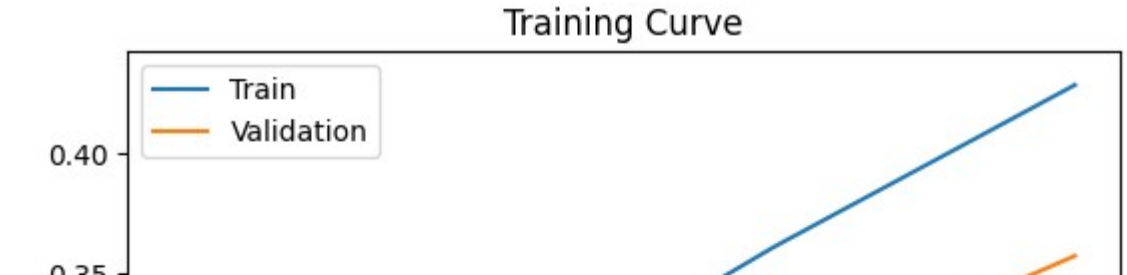
```
Epoch 1/4, Training loss(ave. loss per minibatch): 2.47,Time: 0.80083s
Epoch 2/4, Training loss(ave. loss per minibatch): 2.95,Time: 1.59872s
Epoch 3/4, Training loss(ave. loss per minibatch): 1.66,Time: 2.38816s
Epoch 4/4, Training loss(ave. loss per minibatch): 1.7,Time: 3.17428s
```

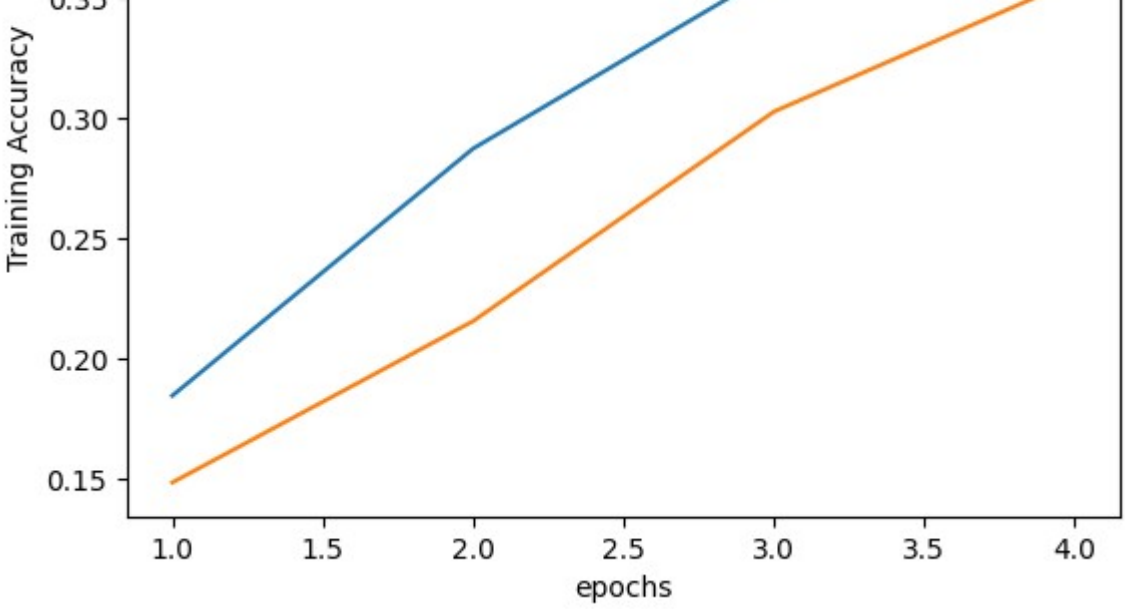




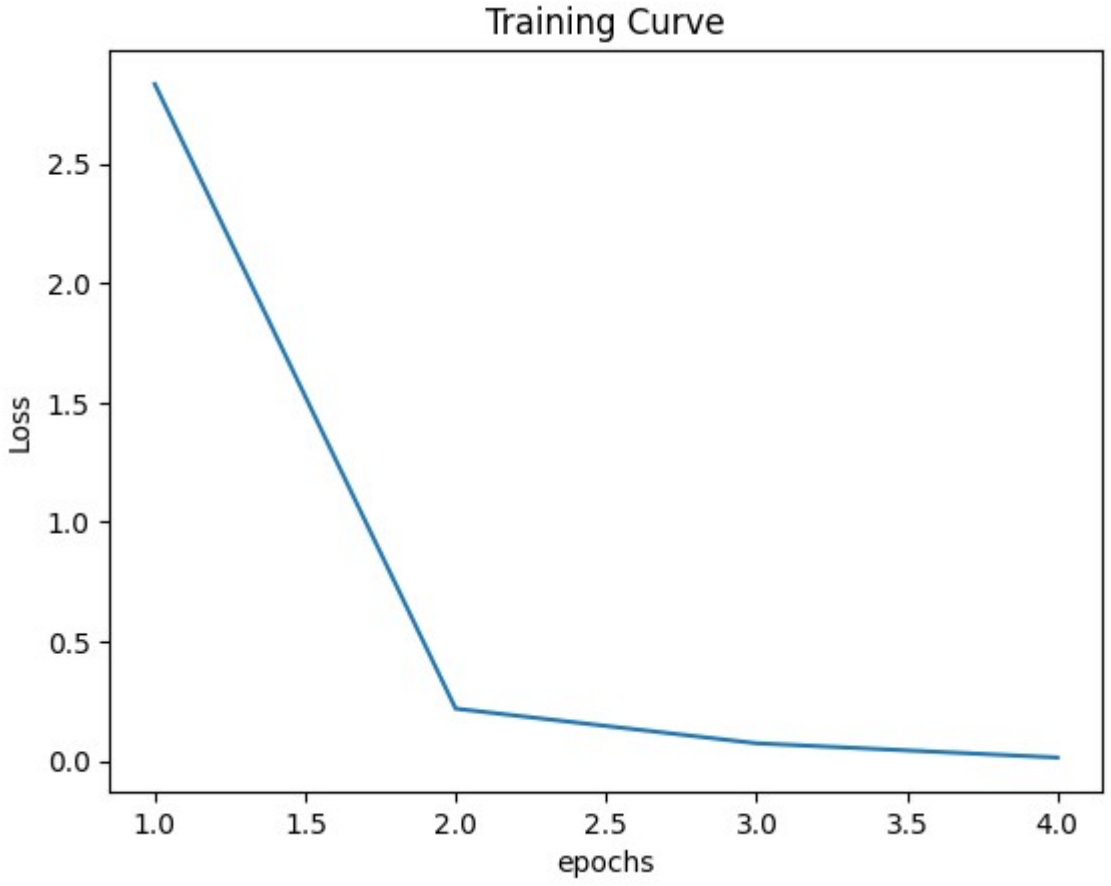

```
Final Training Accuracy: 0.40775
Final Validation Accuracy: 0.3077
Epoch 1/4, Training loss(ave. loss per minibatch): 2.55,Time: 0.79995s
Epoch 2/4, Training loss(ave. loss per minibatch): 2.4,Time: 1.58859s
Epoch 3/4, Training loss(ave. loss per minibatch): 2.12,Time: 2.41242s
Epoch 4/4, Training loss(ave. loss per minibatch): 1.43,Time: 3.23652s
```

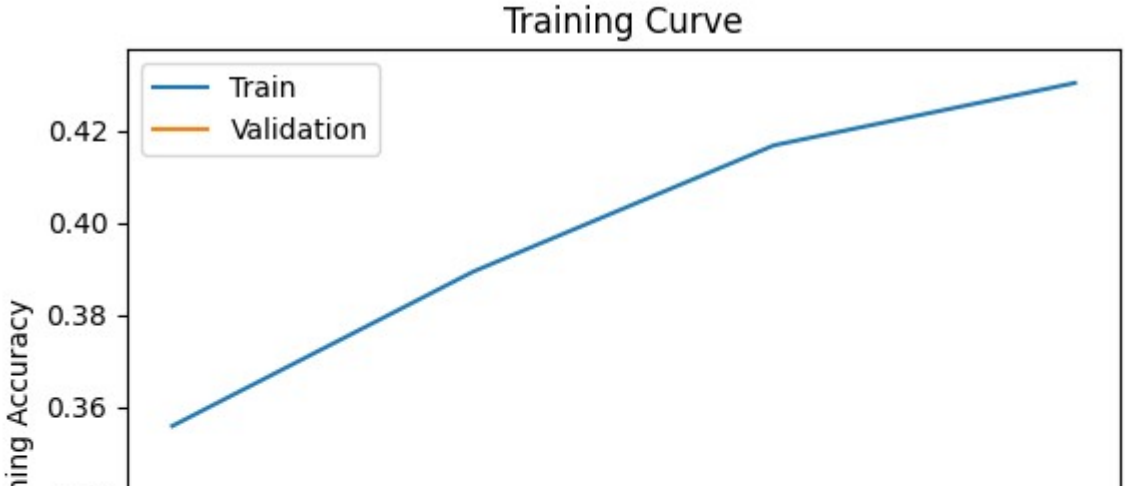


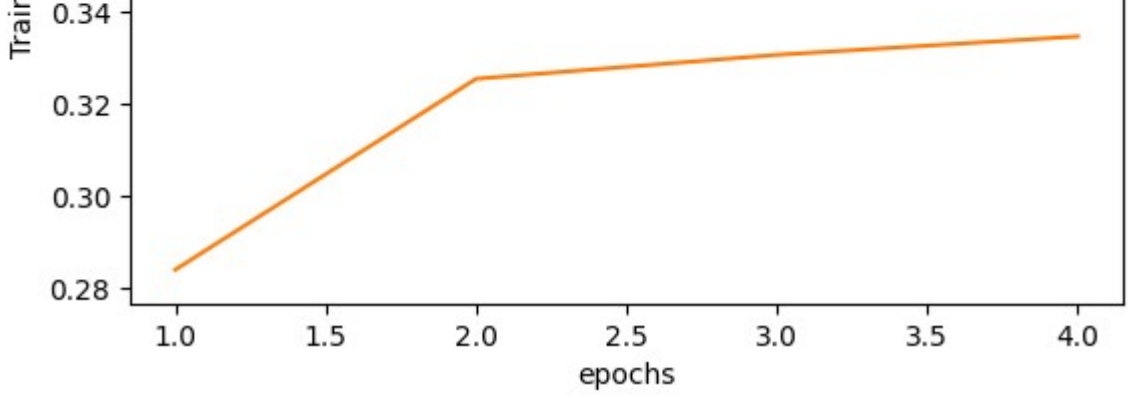




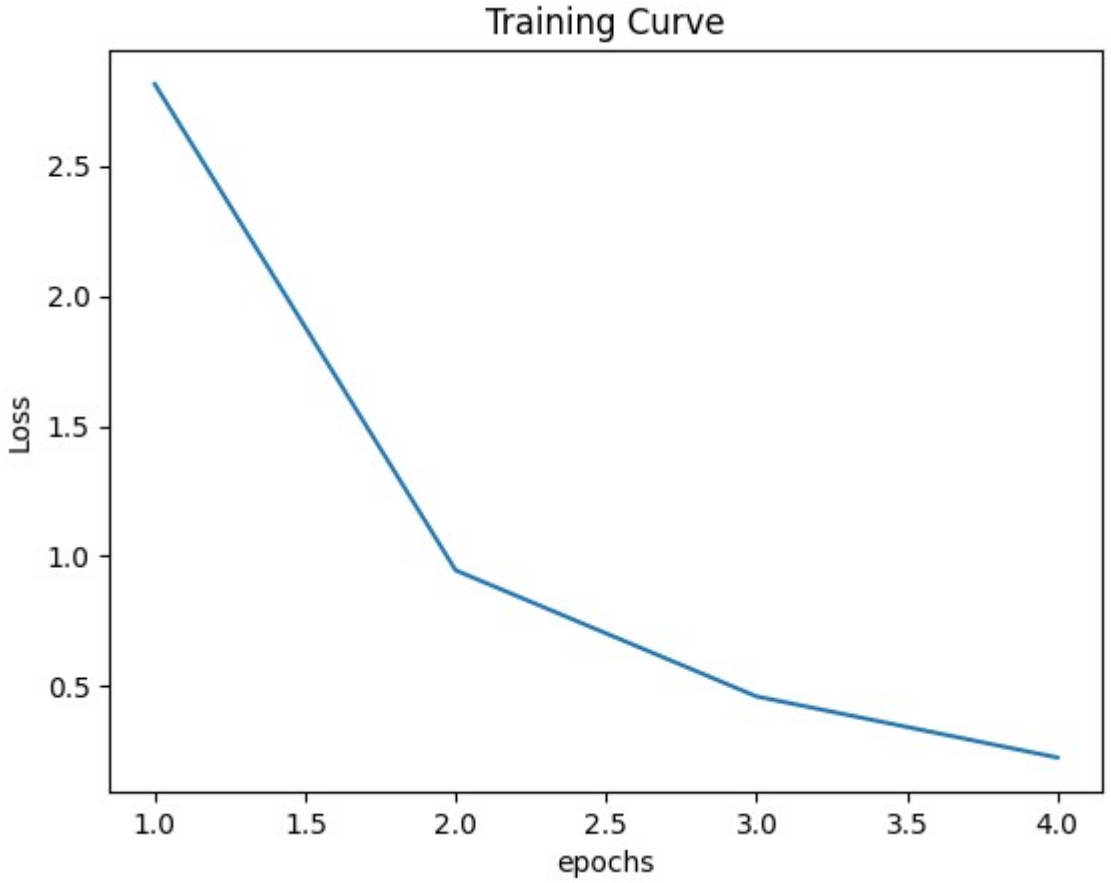

```
Final Training Accuracy: 0.4288166666666667
Final Validation Accuracy: 0.3576
Epoch 1/4, Training loss(ave. loss per minibatch): 2.83,Time: 0.88929s
Epoch 2/4, Training loss(ave. loss per minibatch): 0.22,Time: 1.84241s
Epoch 3/4, Training loss(ave. loss per minibatch): 0.07,Time: 3.03097s
Epoch 4/4, Training loss(ave. loss per minibatch): 0.01,Time: 3.89259s
```

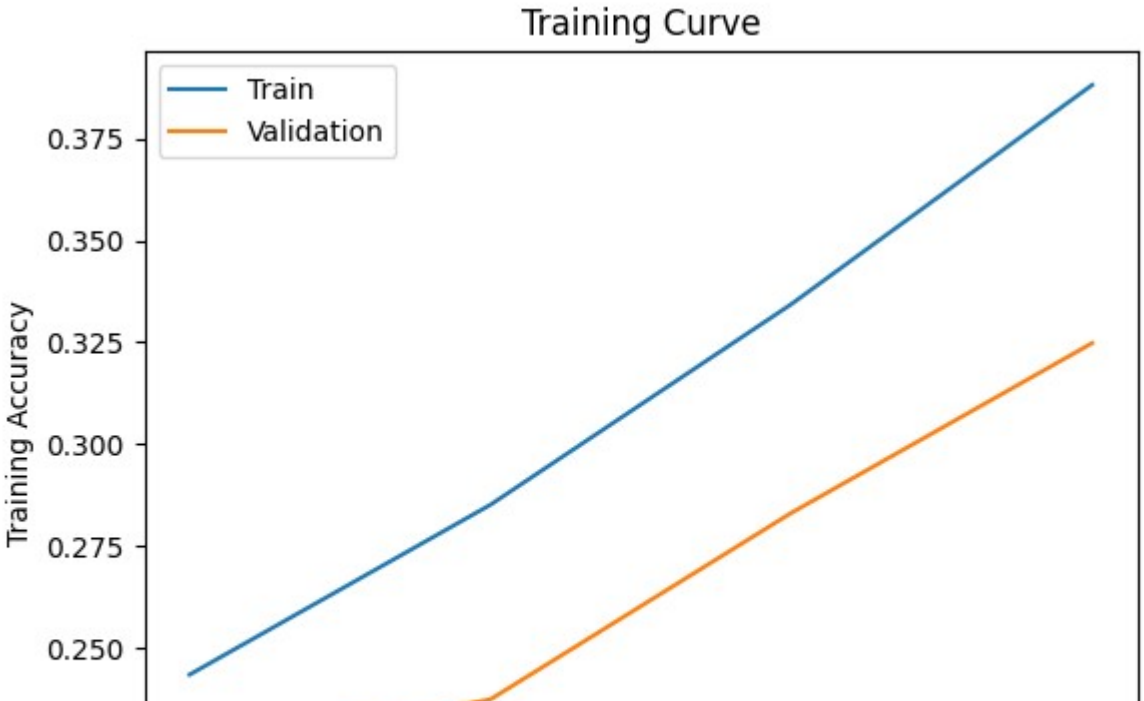


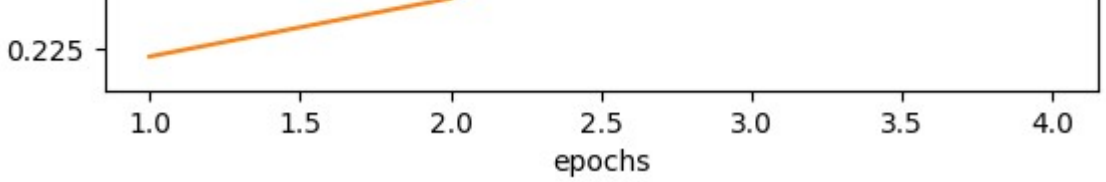




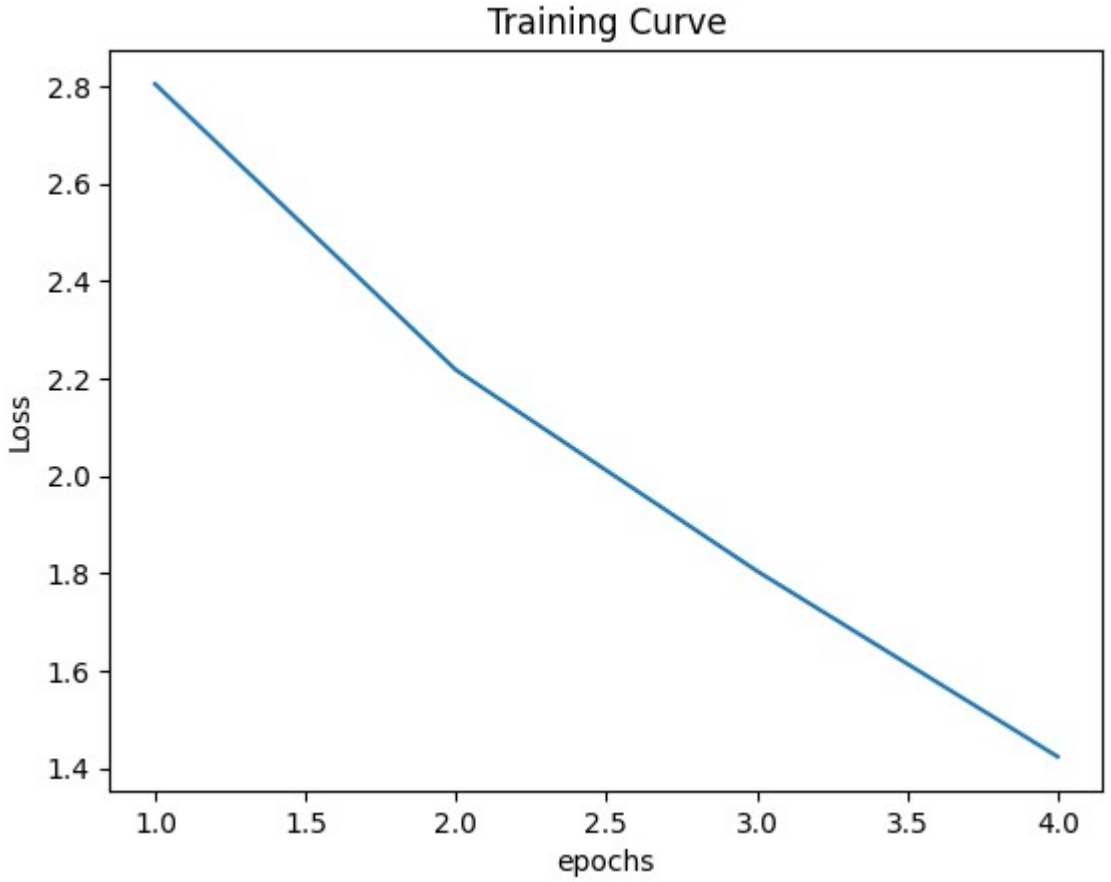

```
Final Training Accuracy: 0.43033333333333335
Final Validation Accuracy: 0.3348
Epoch 1/4, Training loss(ave. loss per minibatch): 2.81,Time: 0.88224s
Epoch 2/4, Training loss(ave. loss per minibatch): 0.95,Time: 1.74217s
Epoch 3/4, Training loss(ave. loss per minibatch): 0.46,Time: 2.598s
Epoch 4/4, Training loss(ave. loss per minibatch): 0.23,Time: 3.46244s
```

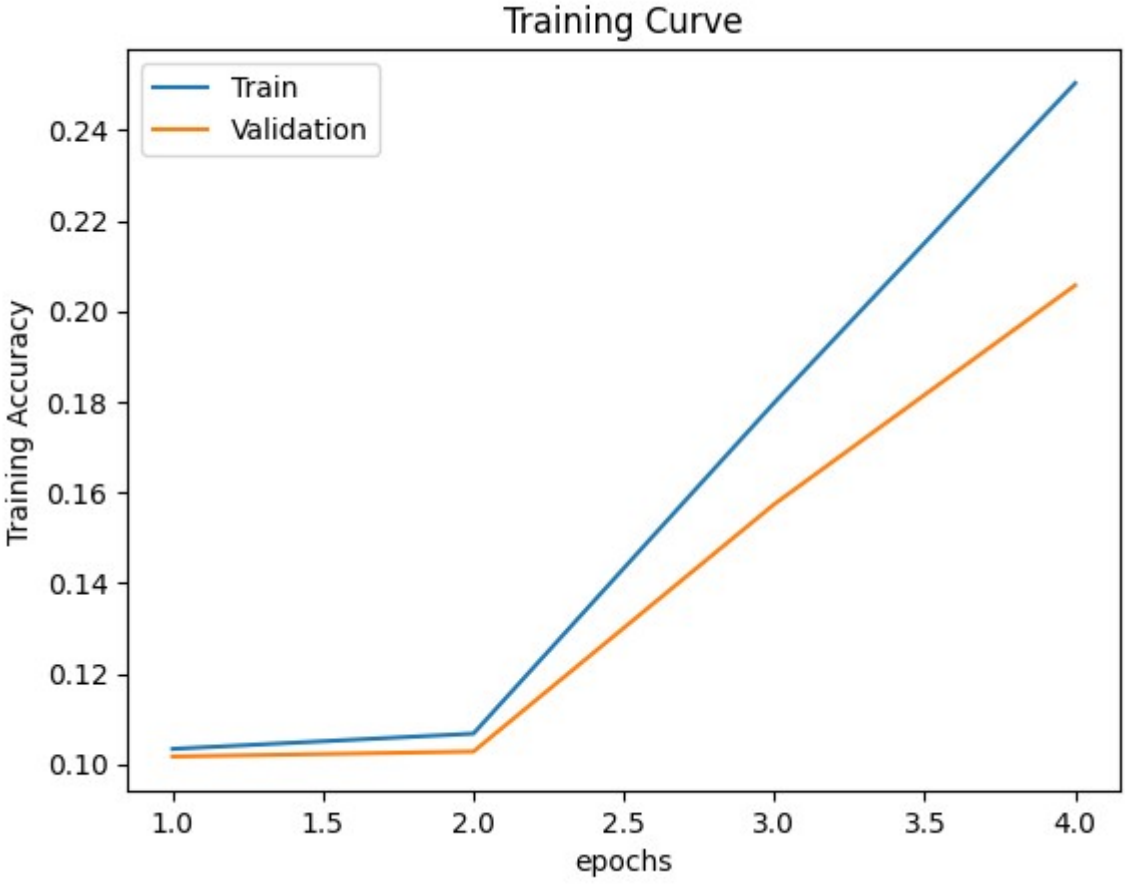







```
Final Training Accuracy: 0.3881833333333333
Final Validation Accuracy: 0.3248
Epoch 1/4, Training loss(ave. loss per minibatch): 2.8,Time: 2.31861s
Epoch 2/4, Training loss(ave. loss per minibatch): 2.22,Time: 4.65799s
Epoch 3/4, Training loss(ave. loss per minibatch): 1.8,Time: 7.37713s
Epoch 4/4, Training loss(ave. loss per minibatch): 1.42,Time: 9.69797s
```





Final Training Accuracy: 0.25025

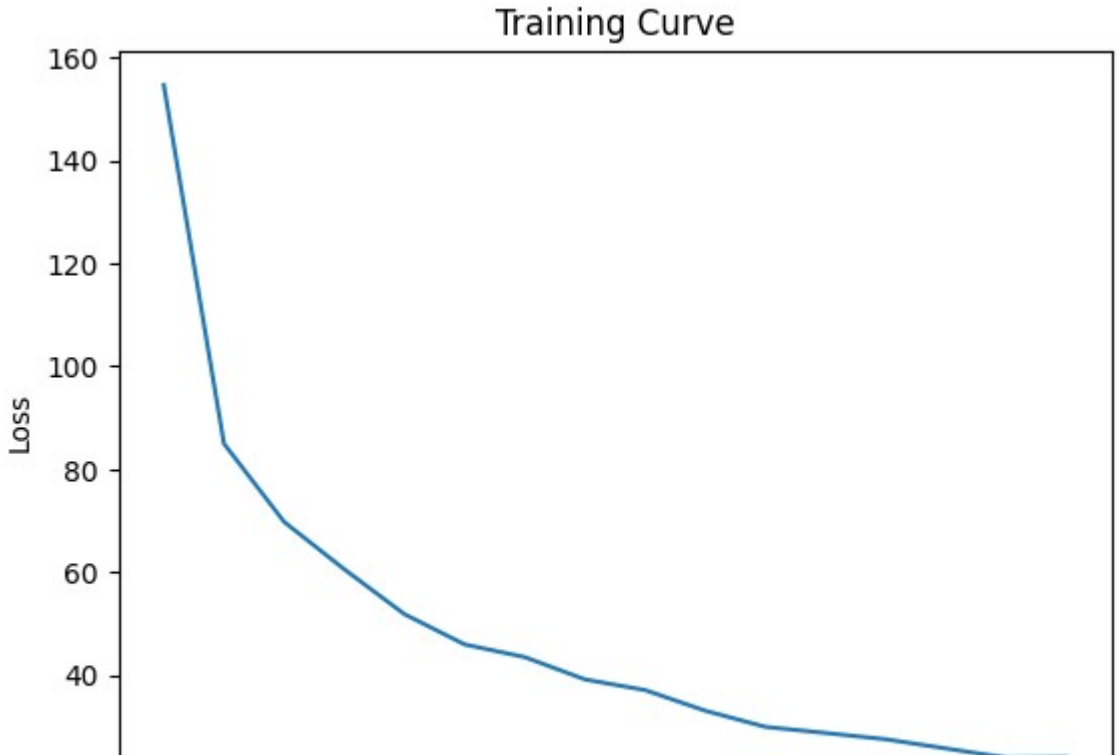
```
Final Training Accuracy: 0.25035
Final Validation Accuracy: 0.2057
```

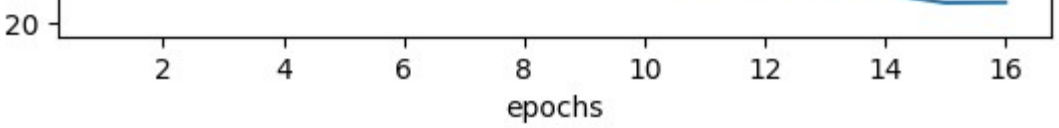
Hyper parameter Tuning (on all models by replacing model_both with other models)

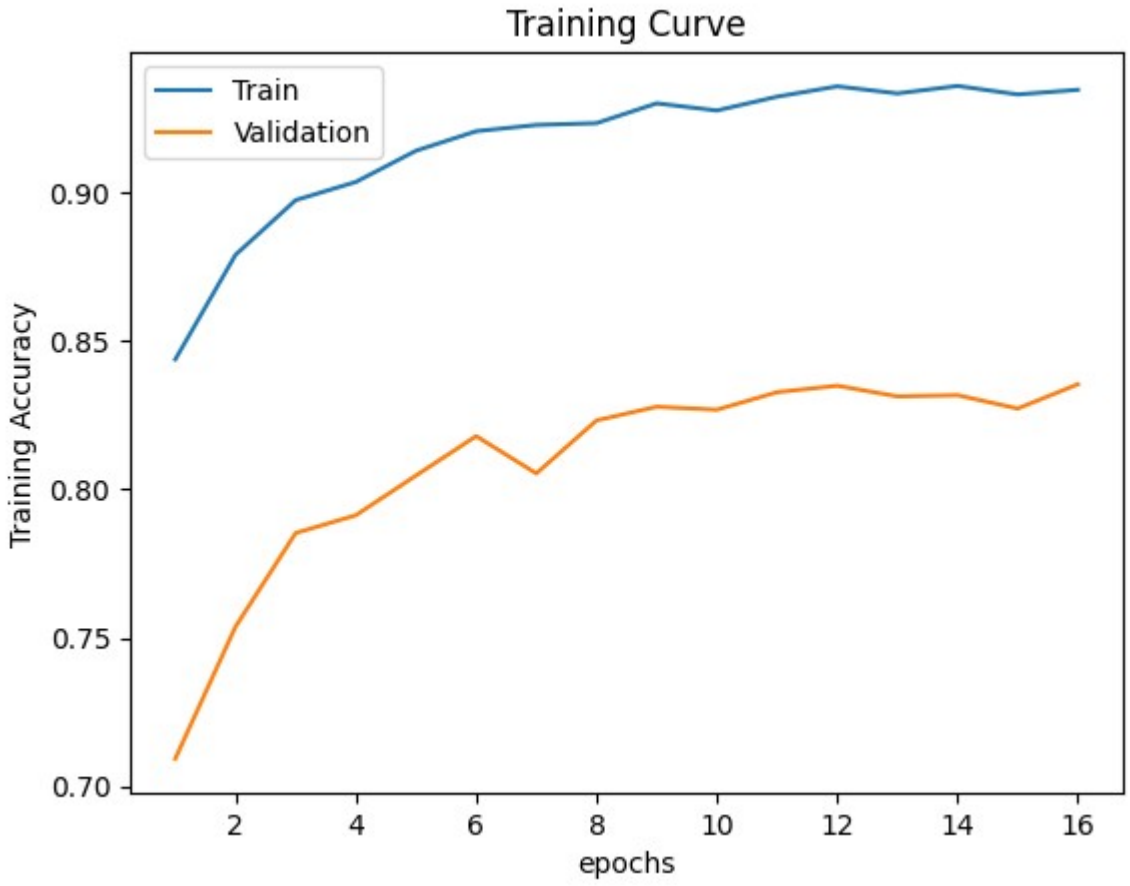
```
# creating a function to reset the models' learned parameters
def reset_mod_params(model):
  for layer in model.children():
    if hasattr(layer, 'reset_parameters'):
      layer.reset_parameters()
```

```
rate_optns = [0.0005, 0.001, 0.005, 0.01, 0.05, 0.1]   # best validation accuracy at 0.001
reset_mod_params(model_both)

for rate in rate_optns:
  print('rate: ' + str(rate))
  train(model_both, mnist_train[:10000], num_epochs=16, lr_choice=rate)
  reset_mod_params(model_both)
```

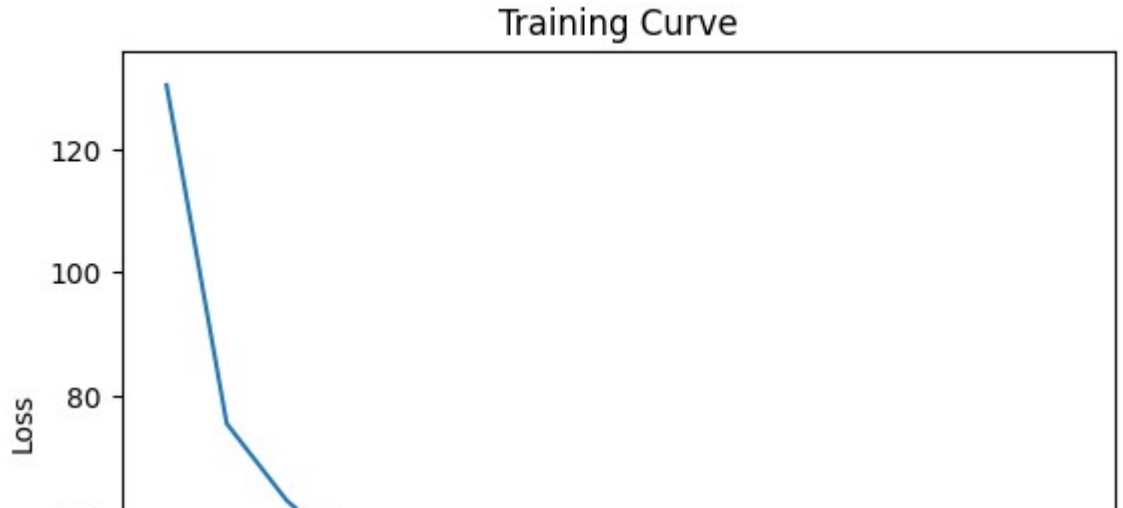
```
rate: 0.0005
Epoch 1/16, Training loss(ave. loss per minibatch): 0.99,Time: 1.4269s
Epoch 2/16, Training loss(ave. loss per minibatch): 0.54,Time: 2.9333s
Epoch 3/16, Training loss(ave. loss per minibatch): 0.44,Time: 4.42505s
Epoch 4/16, Training loss(ave. loss per minibatch): 0.39,Time: 5.83763s
Epoch 5/16, Training loss(ave. loss per minibatch): 0.33,Time: 7.23743s
Epoch 6/16, Training loss(ave. loss per minibatch): 0.29,Time: 8.65248s
Epoch 7/16, Training loss(ave. loss per minibatch): 0.28,Time: 10.44575s
Epoch 8/16, Training loss(ave. loss per minibatch): 0.25,Time: 11.82713s
Epoch 9/16, Training loss(ave. loss per minibatch): 0.24,Time: 13.19148s
Epoch 10/16, Training loss(ave. loss per minibatch): 0.21,Time: 14.6208s
Epoch 11/16, Training loss(ave. loss per minibatch): 0.19,Time: 15.99928s
Epoch 12/16, Training loss(ave. loss per minibatch): 0.18,Time: 17.40833s
Epoch 13/16, Training loss(ave. loss per minibatch): 0.18,Time: 18.82747s
Epoch 14/16, Training loss(ave. loss per minibatch): 0.16,Time: 20.33395s
Epoch 15/16, Training loss(ave. loss per minibatch): 0.15,Time: 22.01293s
Epoch 16/16, Training loss(ave. loss per minibatch): 0.15,Time: 23.38857s
```

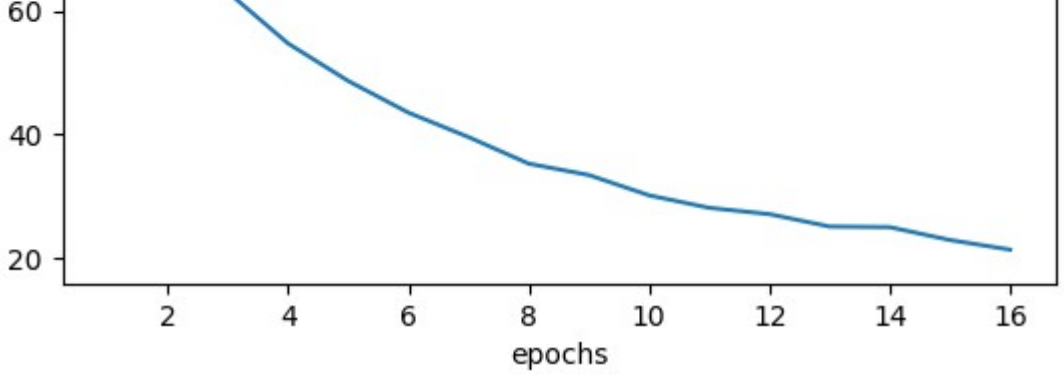


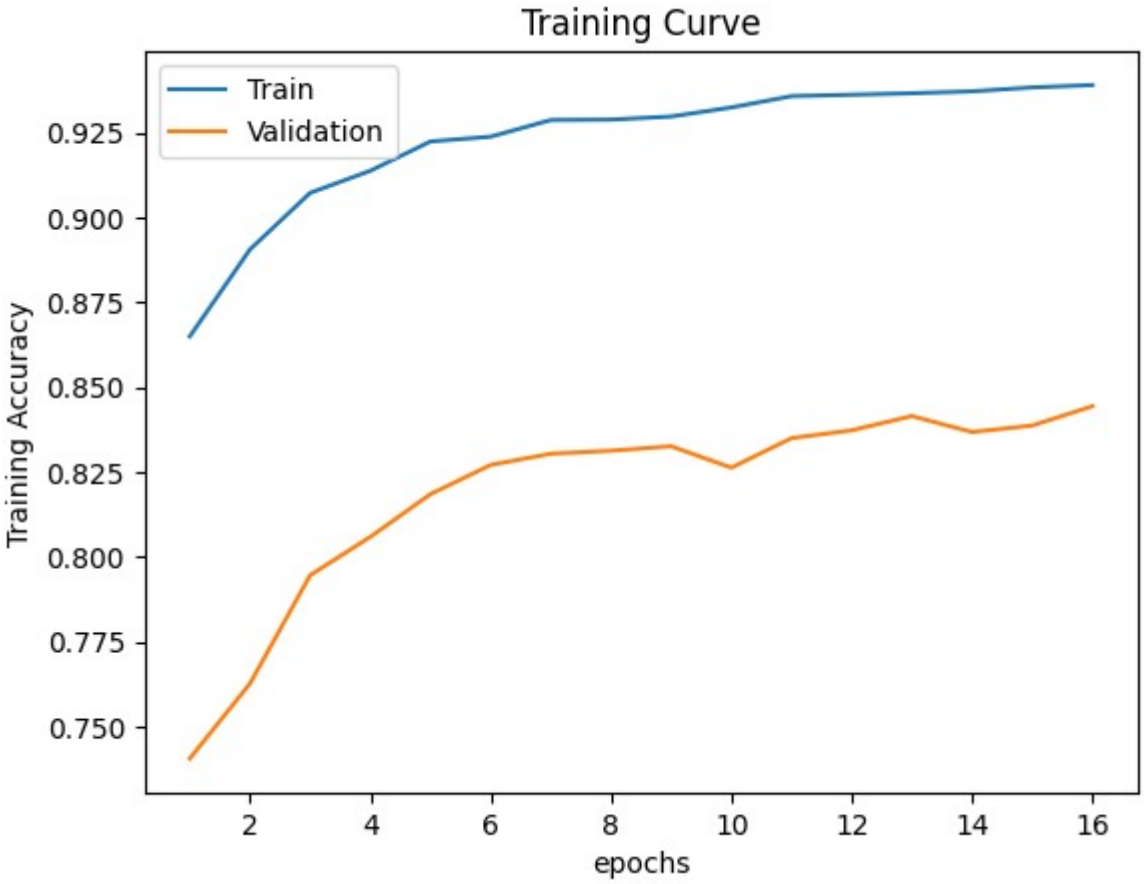




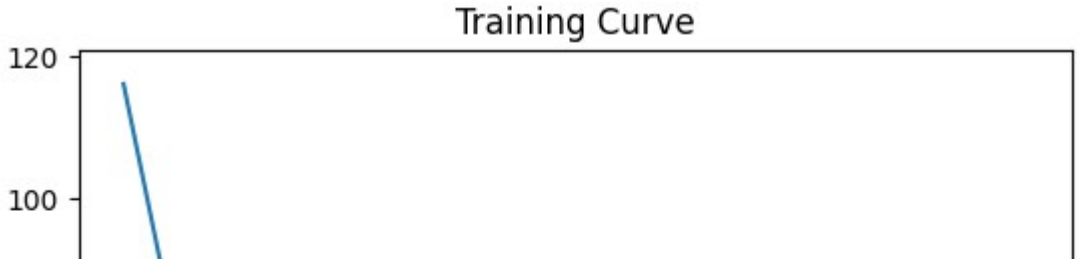

```
Final Training Accuracy: 0.9348
Final Validation Accuracy: 0.8355
rate: 0.001
Epoch 1/16, Training loss(ave. loss per minibatch): 0.83,Time: 1.41046s
Epoch 2/16, Training loss(ave. loss per minibatch): 0.48,Time: 2.76182s
Epoch 3/16, Training loss(ave. loss per minibatch): 0.4,Time: 4.1856s
Epoch 4/16, Training loss(ave. loss per minibatch): 0.35,Time: 5.57275s
Epoch 5/16, Training loss(ave. loss per minibatch): 0.31,Time: 6.96606s
Epoch 6/16, Training loss(ave. loss per minibatch): 0.28,Time: 8.67981s
Epoch 7/16, Training loss(ave. loss per minibatch): 0.25,Time: 10.17807s
Epoch 8/16, Training loss(ave. loss per minibatch): 0.22,Time: 11.57157s
Epoch 9/16, Training loss(ave. loss per minibatch): 0.21,Time: 12.98773s
Epoch 10/16, Training loss(ave. loss per minibatch): 0.19,Time: 14.3842s
Epoch 11/16, Training loss(ave. loss per minibatch): 0.18,Time: 15.83784s
Epoch 12/16, Training loss(ave. loss per minibatch): 0.17,Time: 17.29845s
Epoch 13/16, Training loss(ave. loss per minibatch): 0.16,Time: 18.74126s
Epoch 14/16, Training loss(ave. loss per minibatch): 0.16,Time: 20.51433s
Epoch 15/16, Training loss(ave. loss per minibatch): 0.15,Time: 21.93311s
Epoch 16/16, Training loss(ave. loss per minibatch): 0.14,Time: 23.3387s
```

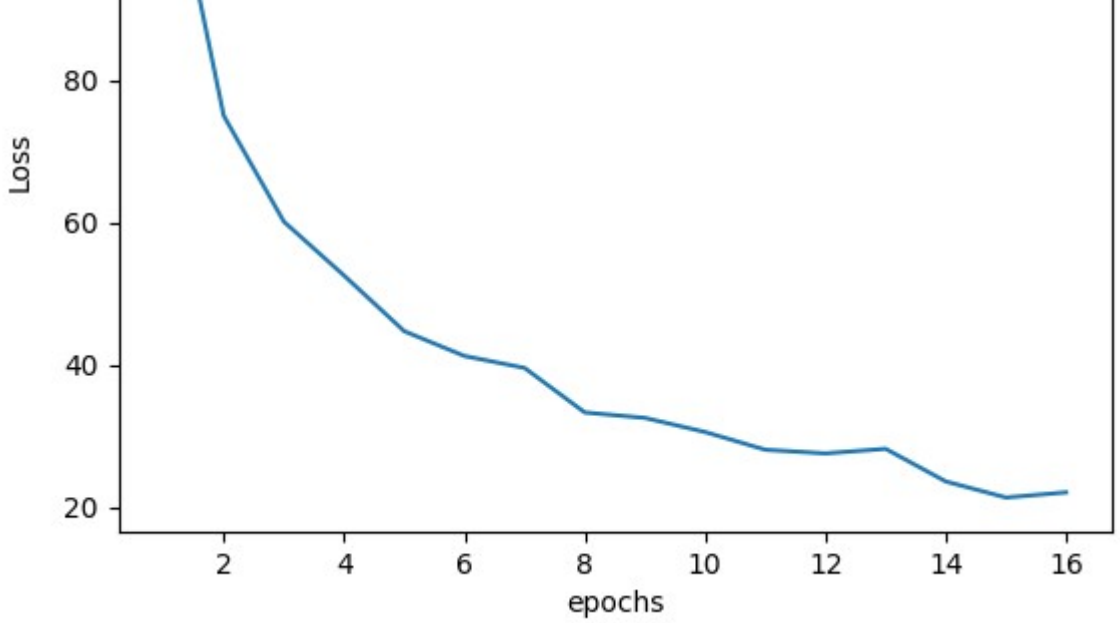


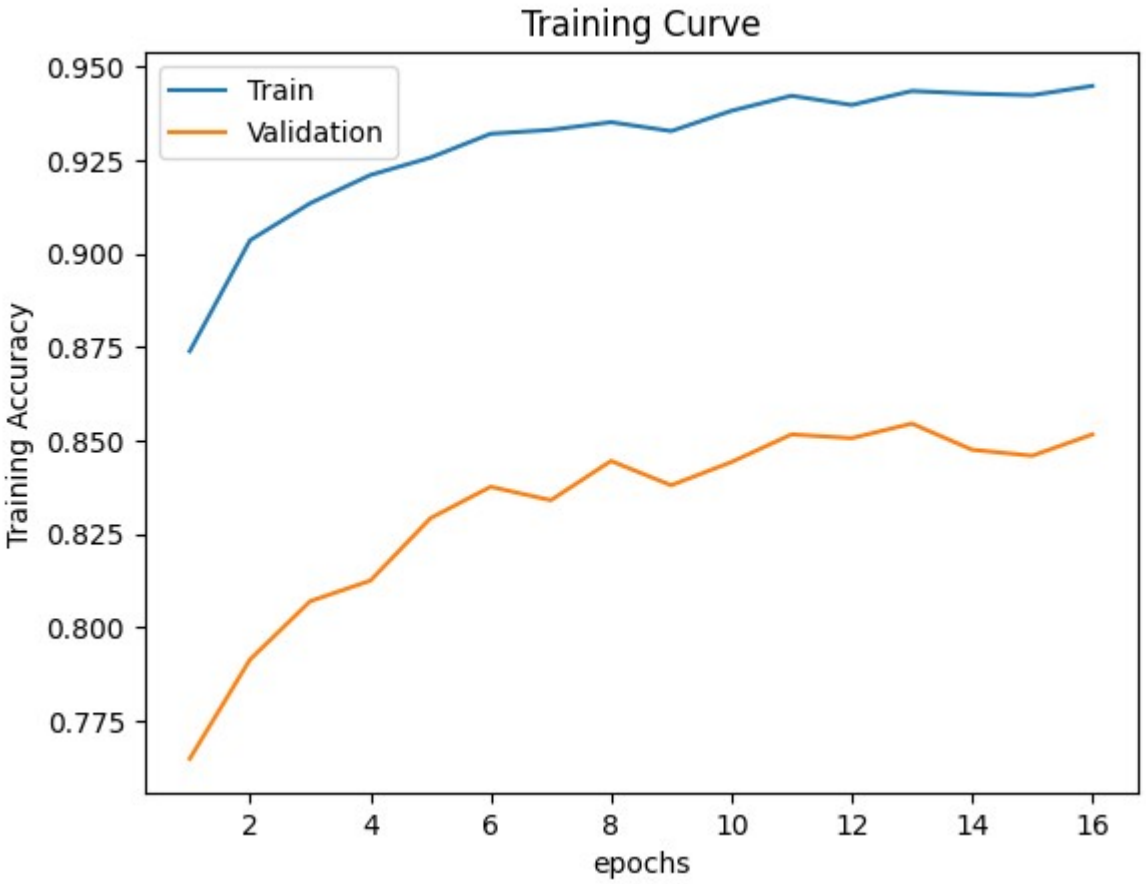




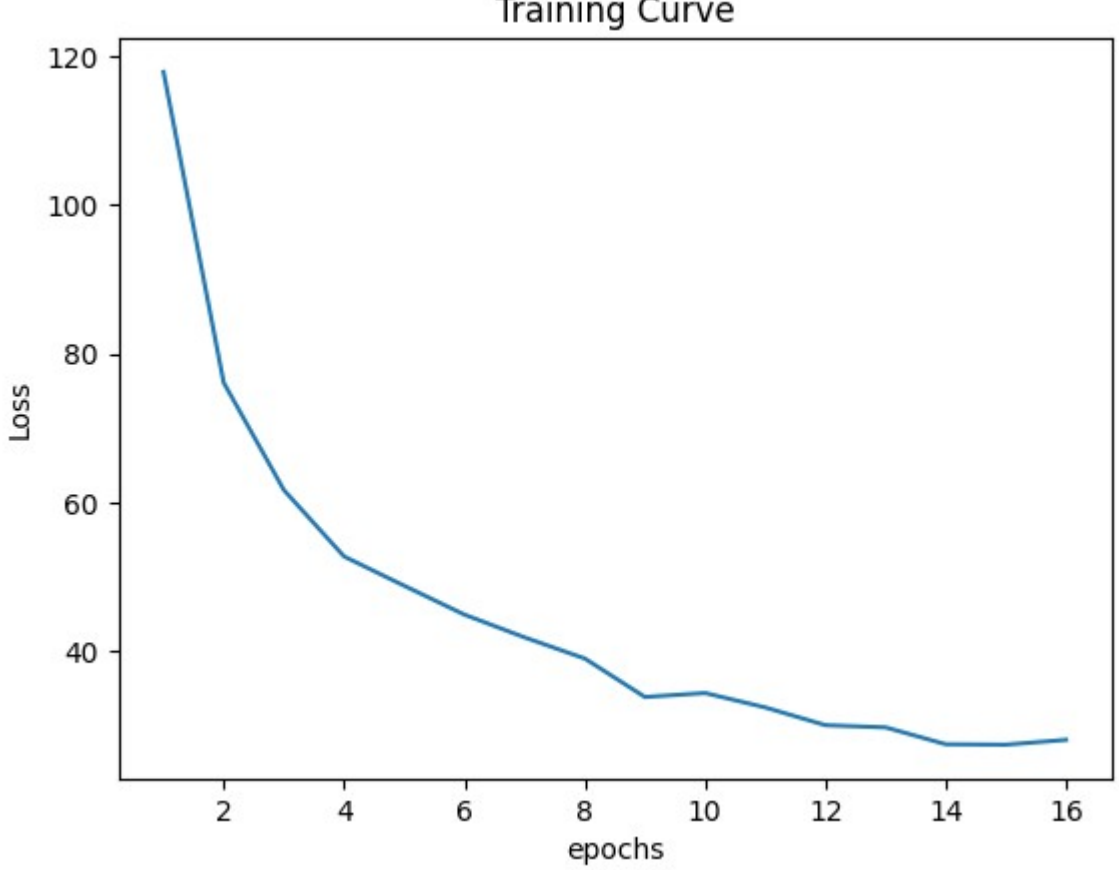

```
Final Training Accuracy: 0.9390333333333334
Final Validation Accuracy: 0.8444
rate: 0.005
Epoch 1/16, Training loss(ave. loss per minibatch): 0.74,Time: 1.34809s
Epoch 2/16, Training loss(ave. loss per minibatch): 0.48,Time: 2.74717s
Epoch 3/16, Training loss(ave. loss per minibatch): 0.38,Time: 4.15061s
Epoch 4/16, Training loss(ave. loss per minibatch): 0.33,Time: 5.52304s
Epoch 5/16, Training loss(ave. loss per minibatch): 0.28,Time: 7.05397s
Epoch 6/16, Training loss(ave. loss per minibatch): 0.26,Time: 8.72711s
Epoch 7/16, Training loss(ave. loss per minibatch): 0.25,Time: 10.12915s
Epoch 8/16, Training loss(ave. loss per minibatch): 0.21,Time: 11.48966s
Epoch 9/16, Training loss(ave. loss per minibatch): 0.21,Time: 12.90417s
Epoch 10/16, Training loss(ave. loss per minibatch): 0.19,Time: 14.28974s
Epoch 11/16, Training loss(ave. loss per minibatch): 0.18,Time: 15.69817s
Epoch 12/16, Training loss(ave. loss per minibatch): 0.18,Time: 17.10614s
Epoch 13/16, Training loss(ave. loss per minibatch): 0.18,Time: 18.77899s
Epoch 14/16, Training loss(ave. loss per minibatch): 0.15,Time: 20.35399s
Epoch 15/16, Training loss(ave. loss per minibatch): 0.14,Time: 21.78458s
Epoch 16/16, Training loss(ave. loss per minibatch): 0.14,Time: 23.1705s
```

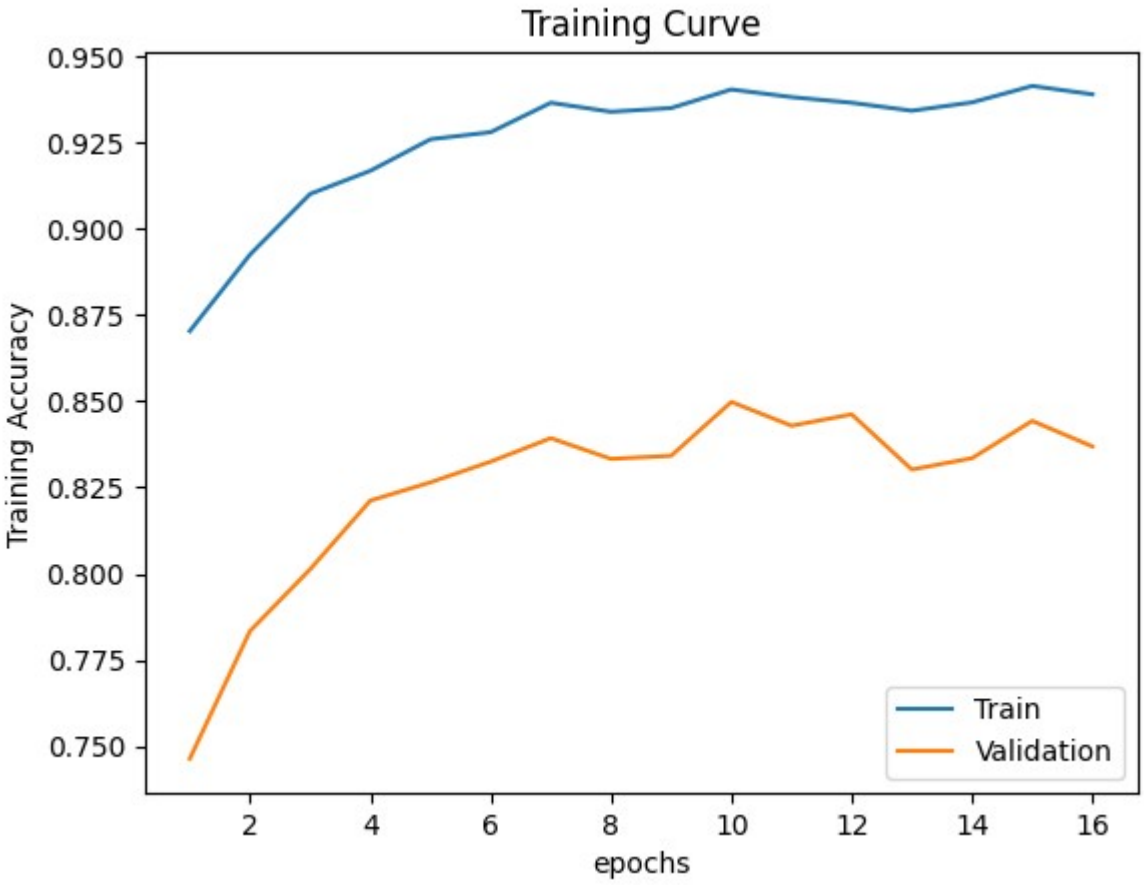






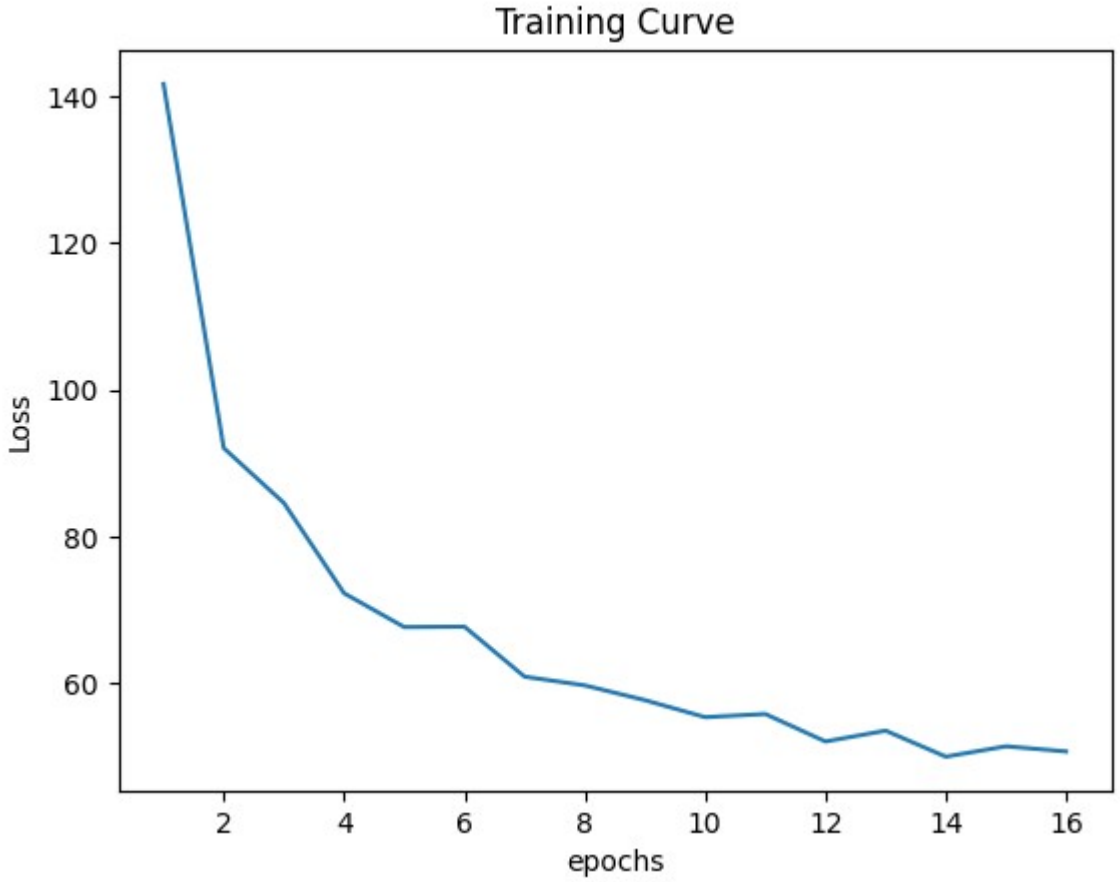

```
Final Training Accuracy: 0.9447666666666666
Final Validation Accuracy: 0.8516
rate: 0.01
Epoch 1/16, Training loss(ave. loss per minibatch): 0.75,Time: 1.43077s
Epoch 2/16, Training loss(ave. loss per minibatch): 0.48,Time: 2.89718s
Epoch 3/16, Training loss(ave. loss per minibatch): 0.39,Time: 4.35805s
Epoch 4/16, Training loss(ave. loss per minibatch): 0.34,Time: 5.78709s
Epoch 5/16, Training loss(ave. loss per minibatch): 0.31,Time: 7.60615s
Epoch 6/16, Training loss(ave. loss per minibatch): 0.29,Time: 9.03506s
Epoch 7/16, Training loss(ave. loss per minibatch): 0.27,Time: 10.44682s
Epoch 8/16, Training loss(ave. loss per minibatch): 0.25,Time: 11.83801s
Epoch 9/16, Training loss(ave. loss per minibatch): 0.22,Time: 13.21556s
Epoch 10/16, Training loss(ave. loss per minibatch): 0.22,Time: 14.6091s
Epoch 11/16, Training loss(ave. loss per minibatch): 0.21,Time: 15.97197s
Epoch 12/16, Training loss(ave. loss per minibatch): 0.19,Time: 17.50284s
Epoch 13/16, Training loss(ave. loss per minibatch): 0.19,Time: 19.17425s
Epoch 14/16, Training loss(ave. loss per minibatch): 0.18,Time: 20.56629s
Epoch 15/16, Training loss(ave. loss per minibatch): 0.18,Time: 21.98568s
Epoch 16/16, Training loss(ave. loss per minibatch): 0.18,Time: 23.36059s
```

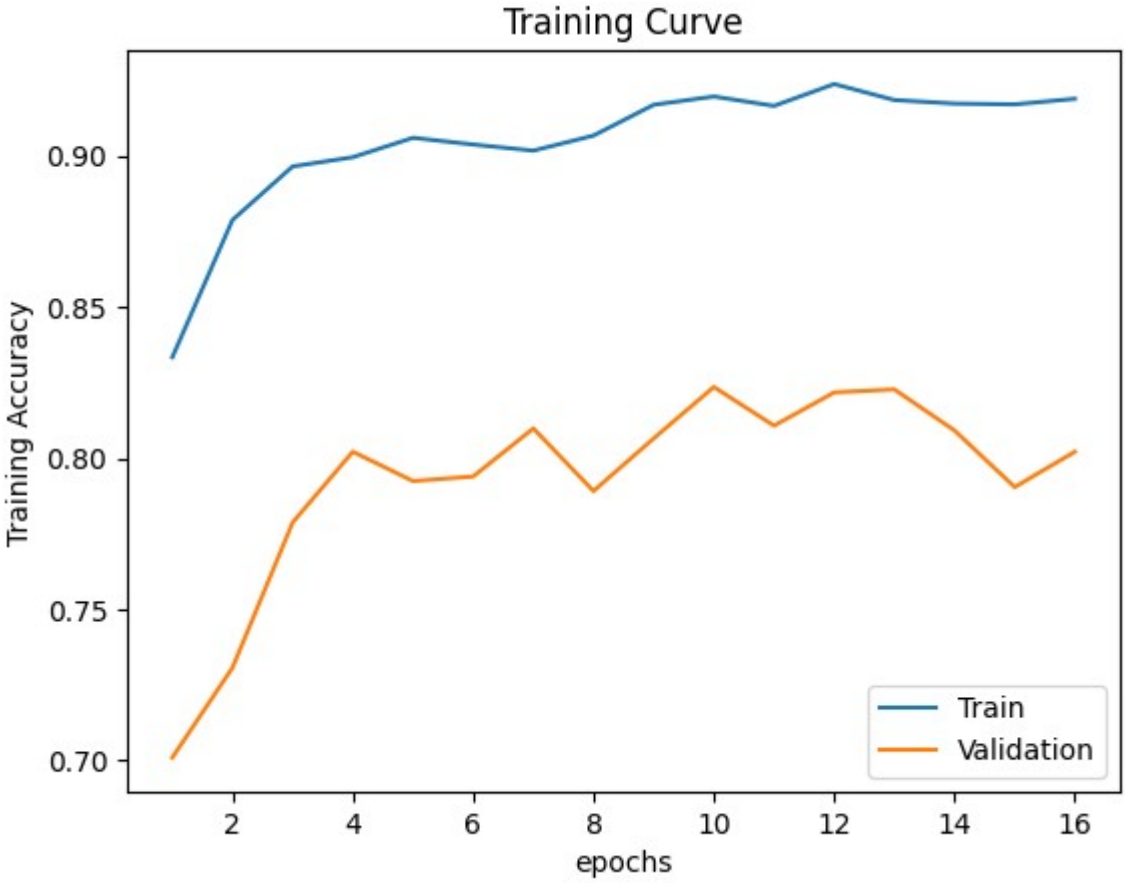





```
Final Training Accuracy: 0.9389666666666666
Final Validation Accuracy: 0.8369
rate: 0.05
Epoch 1/16, Training loss(ave. loss per minibatch): 0.9,Time: 1.39696s
Epoch 2/16, Training loss(ave. loss per minibatch): 0.59,Time: 2.77358s
Epoch 3/16, Training loss(ave. loss per minibatch): 0.54,Time: 4.19901s
Epoch 4/16, Training loss(ave. loss per minibatch): 0.46,Time: 5.92175s
Epoch 5/16, Training loss(ave. loss per minibatch): 0.43,Time: 7.34968s
Epoch 6/16, Training loss(ave. loss per minibatch): 0.43,Time: 8.78344s
Epoch 7/16, Training loss(ave. loss per minibatch): 0.39,Time: 10.202s
Epoch 8/16, Training loss(ave. loss per minibatch): 0.38,Time: 11.60538s
Epoch 9/16, Training loss(ave. loss per minibatch): 0.37,Time: 13.01903s
Epoch 10/16, Training loss(ave. loss per minibatch): 0.35,Time: 14.45608s
```

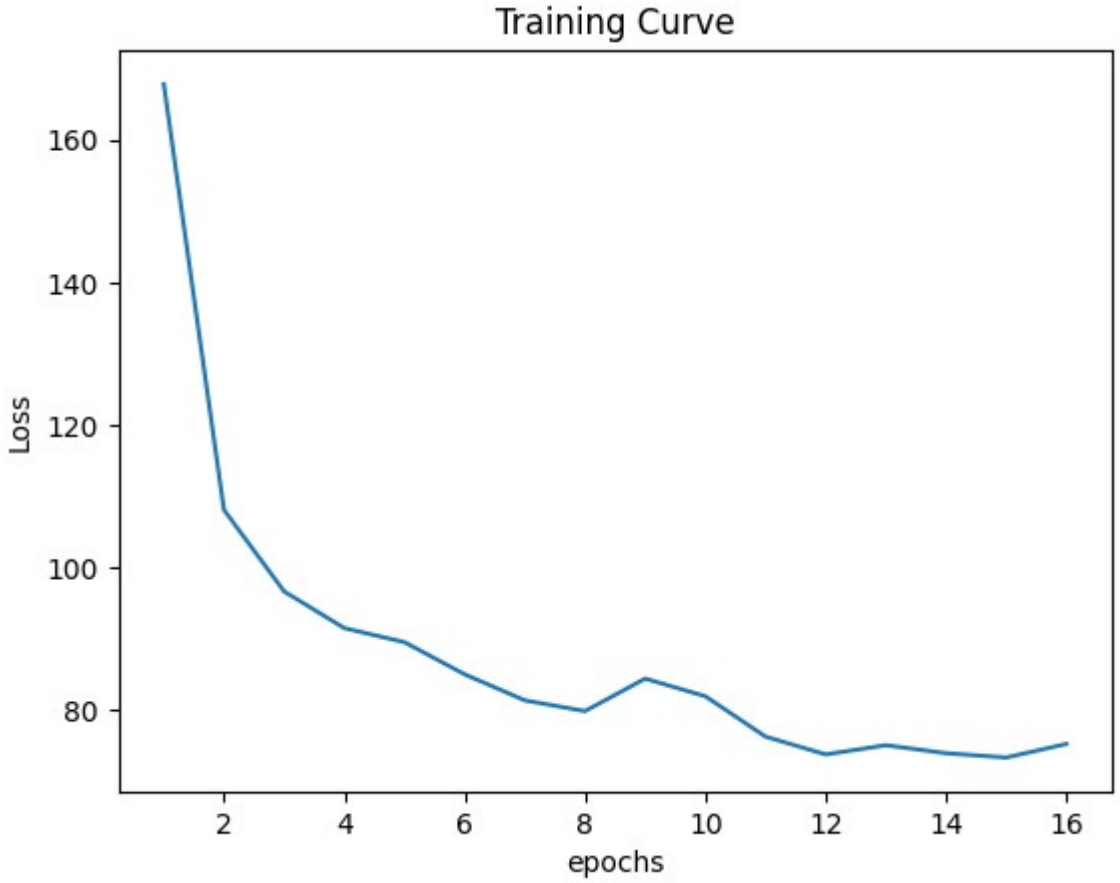
Epoch 11/16, Training loss(ave. loss per minibatch): 0.36,Time: 15.91702s
Epoch 12/16, Training loss(ave. loss per minibatch): 0.33,Time: 17.78162s
Epoch 13/16, Training loss(ave. loss per minibatch): 0.34,Time: 19.21693s
Epoch 14/16, Training loss(ave. loss per minibatch): 0.32,Time: 20.66146s
Epoch 15/16, Training loss(ave. loss per minibatch): 0.33,Time: 22.04932s
Epoch 16/16, Training loss(ave. loss per minibatch): 0.32,Time: 23.38058s

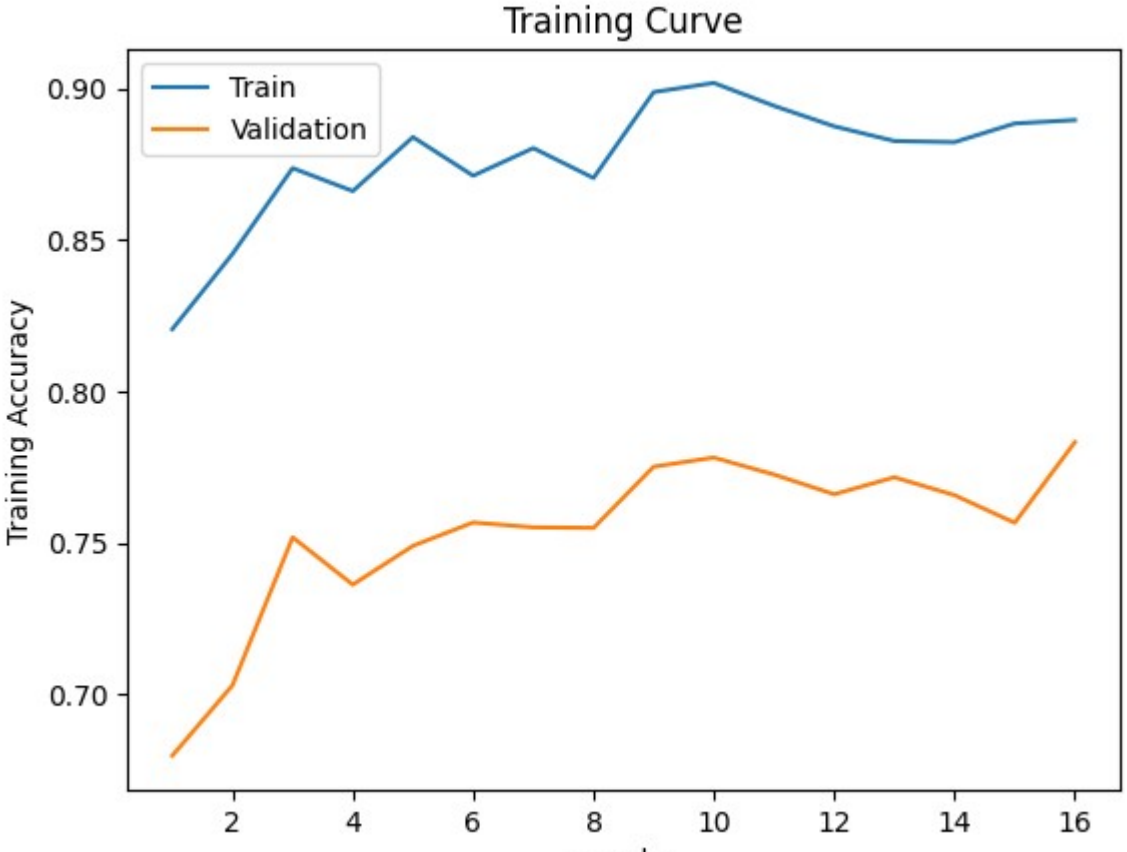




Final Training Accuracy: 0.9187666666666666
Final Validation Accuracy: 0.8021
rate: 0.1
Epoch 1/16, Training loss(ave. loss per minibatch): 1.07,Time: 1.35312s
Epoch 2/16, Training loss(ave. loss per minibatch): 0.69,Time: 2.77298s
Epoch 3/16, Training loss(ave. loss per minibatch): 0.62,Time: 4.35598s

```
Epoch 4/16, Training loss(ave. loss per minibatch): 0.58,Time: 5.98863s
Epoch 5/16, Training loss(ave. loss per minibatch): 0.57,Time: 7.41448s
Epoch 6/16, Training loss(ave. loss per minibatch): 0.54,Time: 8.80728s
Epoch 7/16, Training loss(ave. loss per minibatch): 0.52,Time: 10.19425s
Epoch 8/16, Training loss(ave. loss per minibatch): 0.51,Time: 11.53332s
Epoch 9/16, Training loss(ave. loss per minibatch): 0.54,Time: 12.9242s
Epoch 10/16, Training loss(ave. loss per minibatch): 0.52,Time: 14.34994s
Epoch 11/16, Training loss(ave. loss per minibatch): 0.49,Time: 15.91641s
Epoch 12/16, Training loss(ave. loss per minibatch): 0.47,Time: 17.48628s
Epoch 13/16, Training loss(ave. loss per minibatch): 0.48,Time: 18.86173s
Epoch 14/16, Training loss(ave. loss per minibatch): 0.47,Time: 20.28619s
Epoch 15/16, Training loss(ave. loss per minibatch): 0.47,Time: 21.64216s
Epoch 16/16, Training loss(ave. loss per minibatch): 0.48,Time: 23.03855s
```



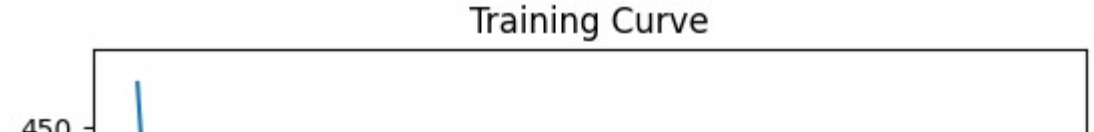


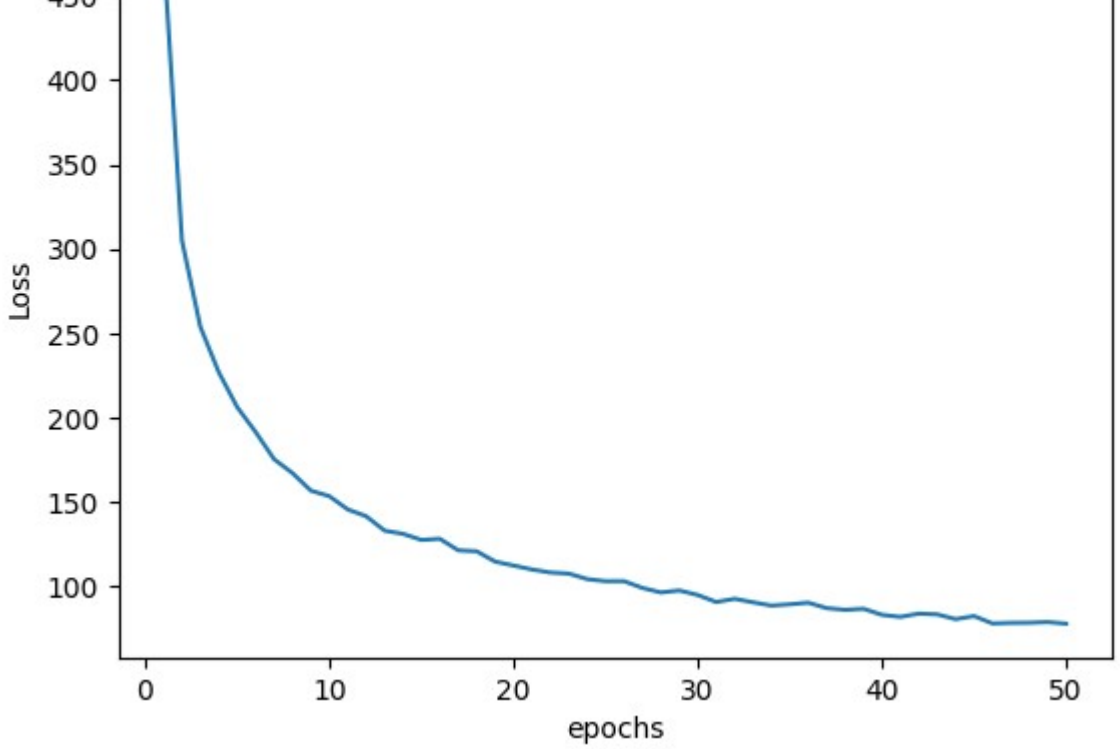
epochs

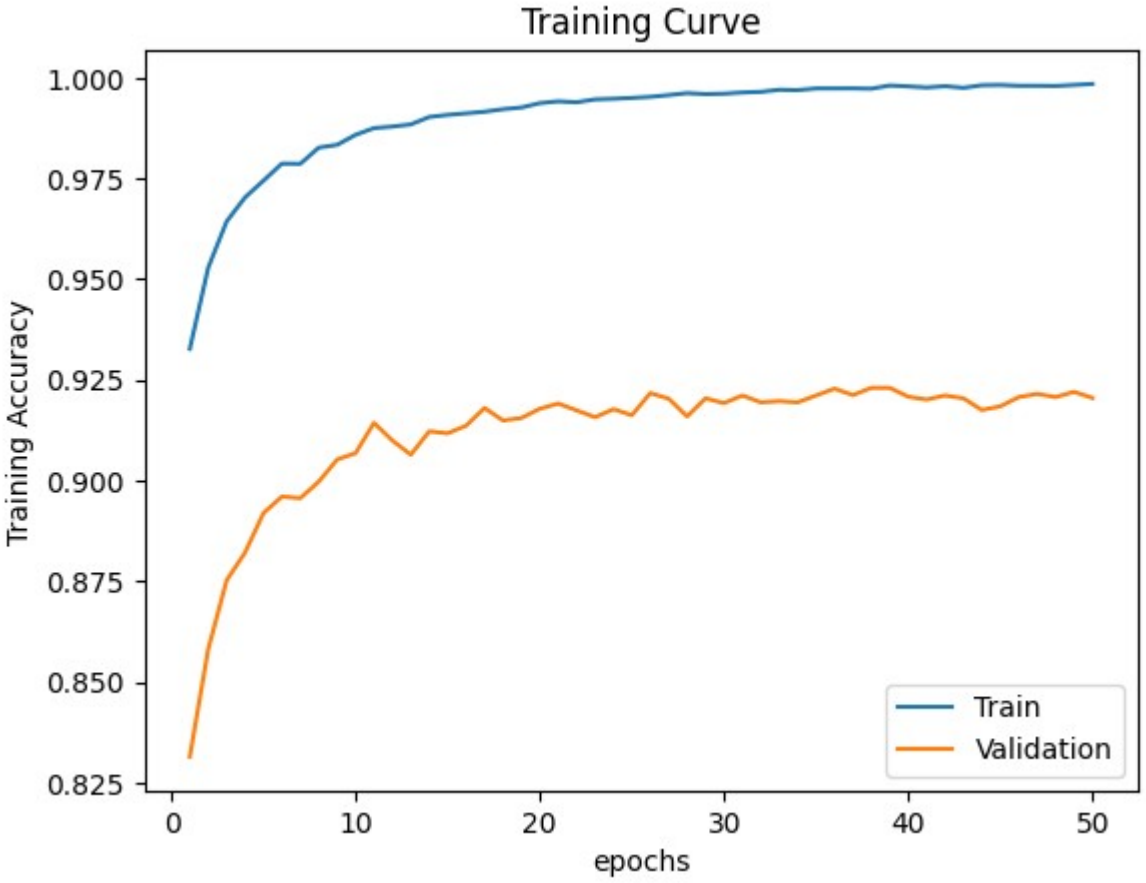
```
Final Training Accuracy: 0.8895
Final Validation Accuracy: 0.7833
```

```
train(model_both, mnist_train, num_epochs=50, lr_choice=0.001)   # best validation accuracy at eith
reset_mod_params(model_both)
```

```
Epoch 1/50, Training loss(ave. loss per minibatch): 0.51,Time: 4.60674s
Epoch 2/50, Training loss(ave. loss per minibatch): 0.33,Time: 9.14412s
Epoch 3/50, Training loss(ave. loss per minibatch): 0.27,Time: 13.47193s
Epoch 4/50, Training loss(ave. loss per minibatch): 0.24,Time: 18.21347s
Epoch 5/50, Training loss(ave. loss per minibatch): 0.22,Time: 22.60693s
Epoch 6/50, Training loss(ave. loss per minibatch): 0.2,Time: 27.40879s
Epoch 7/50, Training loss(ave. loss per minibatch): 0.19,Time: 31.79289s
Epoch 8/50, Training loss(ave. loss per minibatch): 0.18,Time: 36.24366s
Epoch 9/50, Training loss(ave. loss per minibatch): 0.17,Time: 40.9138s
Epoch 10/50, Training loss(ave. loss per minibatch): 0.16,Time: 45.42902s
Epoch 11/50, Training loss(ave. loss per minibatch): 0.16,Time: 50.13153s
Epoch 12/50, Training loss(ave. loss per minibatch): 0.15,Time: 54.54713s
Epoch 13/50, Training loss(ave. loss per minibatch): 0.14,Time: 58.96339s
Epoch 14/50, Training loss(ave. loss per minibatch): 0.14,Time: 63.57405s
Epoch 15/50, Training loss(ave. loss per minibatch): 0.14,Time: 67.81406s
Epoch 16/50, Training loss(ave. loss per minibatch): 0.14,Time: 72.53415s
Epoch 17/50, Training loss(ave. loss per minibatch): 0.13,Time: 76.81927s
Epoch 18/50, Training loss(ave. loss per minibatch): 0.13,Time: 81.54149s
Epoch 19/50, Training loss(ave. loss per minibatch): 0.12,Time: 86.61189s
Epoch 20/50, Training loss(ave. loss per minibatch): 0.12,Time: 90.93482s
Epoch 21/50, Training loss(ave. loss per minibatch): 0.12,Time: 95.57864s
Epoch 22/50, Training loss(ave. loss per minibatch): 0.12,Time: 99.94679s
Epoch 23/50, Training loss(ave. loss per minibatch): 0.11,Time: 104.20902s
Epoch 24/50, Training loss(ave. loss per minibatch): 0.11,Time: 108.86478s
Epoch 25/50, Training loss(ave. loss per minibatch): 0.11,Time: 113.24106s
Epoch 26/50, Training loss(ave. loss per minibatch): 0.11,Time: 117.91656s
Epoch 27/50, Training loss(ave. loss per minibatch): 0.11,Time: 122.5585s
Epoch 28/50, Training loss(ave. loss per minibatch): 0.1,Time: 126.96923s
Epoch 29/50, Training loss(ave. loss per minibatch): 0.1,Time: 131.66053s
Epoch 30/50, Training loss(ave. loss per minibatch): 0.1,Time: 135.96938s
Epoch 31/50, Training loss(ave. loss per minibatch): 0.1,Time: 140.65908s
Epoch 32/50, Training loss(ave. loss per minibatch): 0.1,Time: 144.86634s
Epoch 33/50, Training loss(ave. loss per minibatch): 0.1,Time: 149.15487s
Epoch 34/50, Training loss(ave. loss per minibatch): 0.09,Time: 153.87664s
Epoch 35/50, Training loss(ave. loss per minibatch): 0.1,Time: 158.16568s
Epoch 36/50, Training loss(ave. loss per minibatch): 0.1,Time: 162.96786s
Epoch 37/50, Training loss(ave. loss per minibatch): 0.09,Time: 167.19401s
Epoch 38/50, Training loss(ave. loss per minibatch): 0.09,Time: 171.37553s
Epoch 39/50, Training loss(ave. loss per minibatch): 0.09,Time: 176.05826s
Epoch 40/50, Training loss(ave. loss per minibatch): 0.09,Time: 180.22541s
Epoch 41/50, Training loss(ave. loss per minibatch): 0.09,Time: 184.66393s
Epoch 42/50, Training loss(ave. loss per minibatch): 0.09,Time: 189.12856s
Epoch 43/50, Training loss(ave. loss per minibatch): 0.09,Time: 193.50776s
Epoch 44/50, Training loss(ave. loss per minibatch): 0.09,Time: 198.16215s
Epoch 45/50, Training loss(ave. loss per minibatch): 0.09,Time: 202.48185s
Epoch 46/50, Training loss(ave. loss per minibatch): 0.08,Time: 206.93703s
Epoch 47/50, Training loss(ave. loss per minibatch): 0.08,Time: 211.445s
Epoch 48/50, Training loss(ave. loss per minibatch): 0.08,Time: 215.70458s
Epoch 49/50, Training loss(ave. loss per minibatch): 0.08,Time: 220.66383s
Epoch 50/50, Training loss(ave. loss per minibatch): 0.08,Time: 224.92901s
```





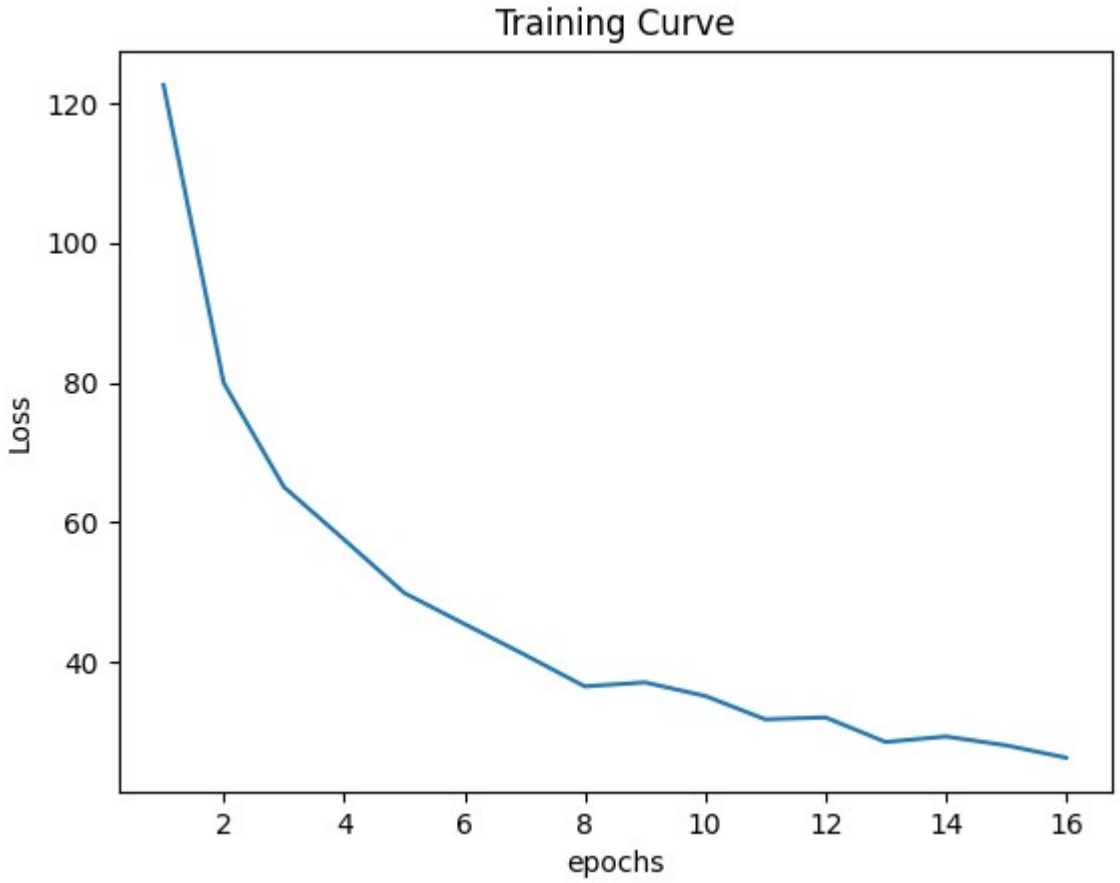


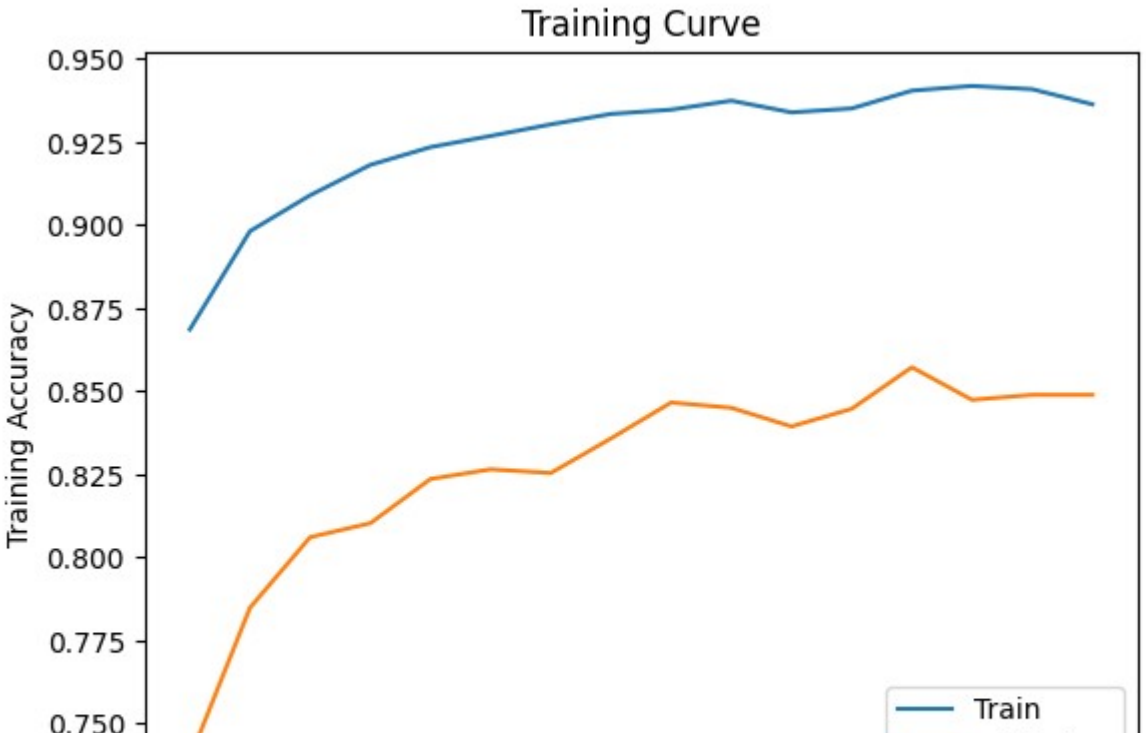

```
Final Training Accuracy: 0.9984666666666666
Final Validation Accuracy: 0.9205
```

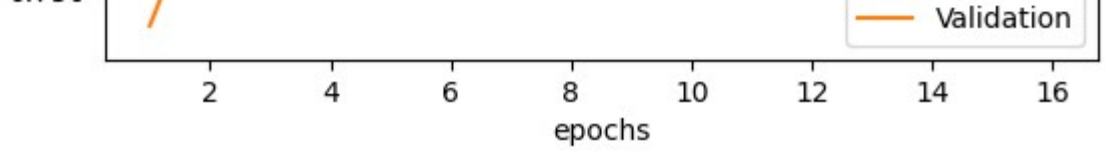
gradient clipping value

```
clips = [0.1, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0]   # 1.5 gives the best validation accuracy
reset_mod_params(model_both)

for grad in clips:
  print('max gradient: ' + str(grad))
  train(model_both, mnist_train[:10000], num_epochs=16, grad_clip=grad)
  reset_mod_params(model_both)
```

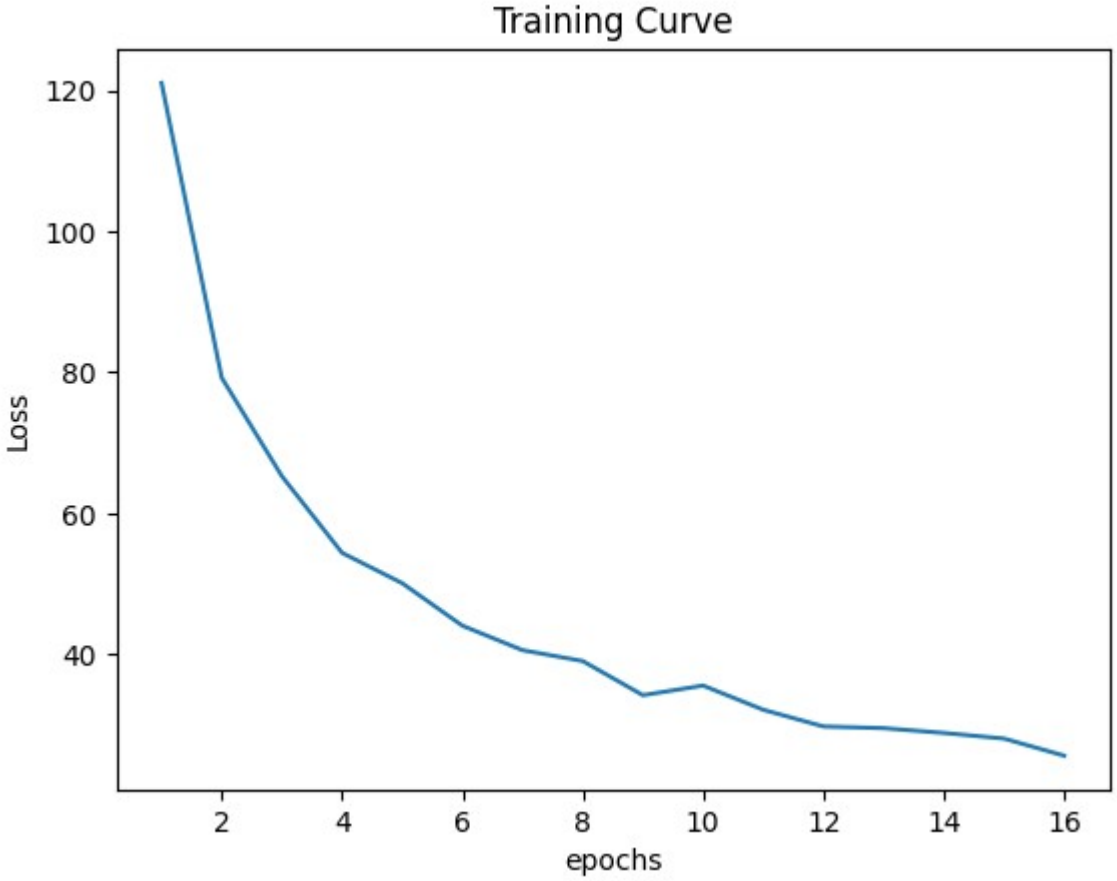
```
max gradient: 0.1
Epoch 1/16, Training loss(ave. loss per minibatch): 0.78,Time: 1.38181s
Epoch 2/16, Training loss(ave. loss per minibatch): 0.51,Time: 2.77611s
Epoch 3/16, Training loss(ave. loss per minibatch): 0.41,Time: 4.31514s
Epoch 4/16, Training loss(ave. loss per minibatch): 0.37,Time: 5.96349s
Epoch 5/16, Training loss(ave. loss per minibatch): 0.32,Time: 7.35786s
Epoch 6/16, Training loss(ave. loss per minibatch): 0.29,Time: 8.75985s
Epoch 7/16, Training loss(ave. loss per minibatch): 0.26,Time: 10.16421s
Epoch 8/16, Training loss(ave. loss per minibatch): 0.23,Time: 11.56372s
Epoch 9/16, Training loss(ave. loss per minibatch): 0.24,Time: 12.93023s
Epoch 10/16, Training loss(ave. loss per minibatch): 0.22,Time: 14.34547s
Epoch 11/16, Training loss(ave. loss per minibatch): 0.2,Time: 15.95364s
Epoch 12/16, Training loss(ave. loss per minibatch): 0.2,Time: 17.53198s
Epoch 13/16, Training loss(ave. loss per minibatch): 0.18,Time: 18.89665s
Epoch 14/16, Training loss(ave. loss per minibatch): 0.19,Time: 20.34846s
Epoch 15/16, Training loss(ave. loss per minibatch): 0.18,Time: 21.75754s
Epoch 16/16, Training loss(ave. loss per minibatch): 0.17,Time: 23.13905s
```

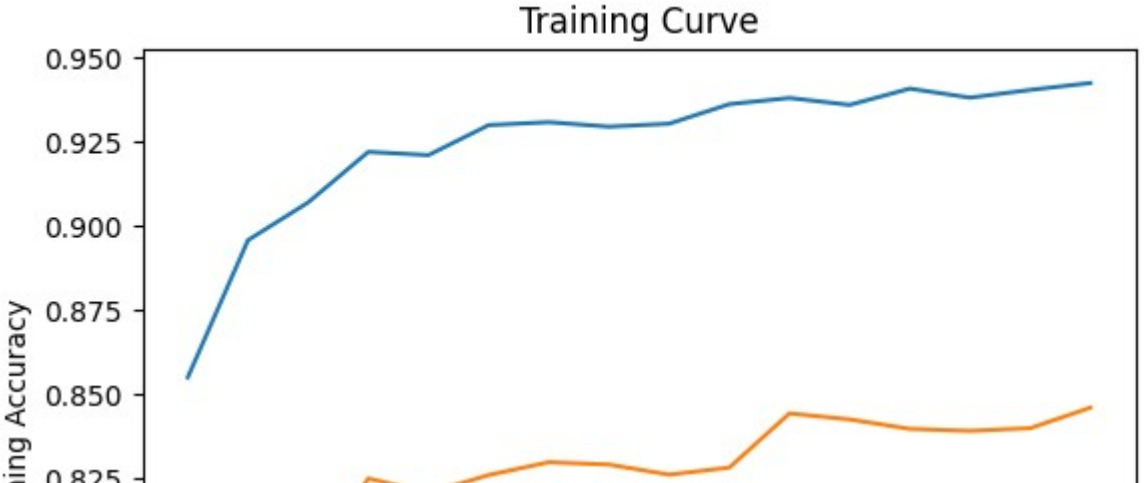


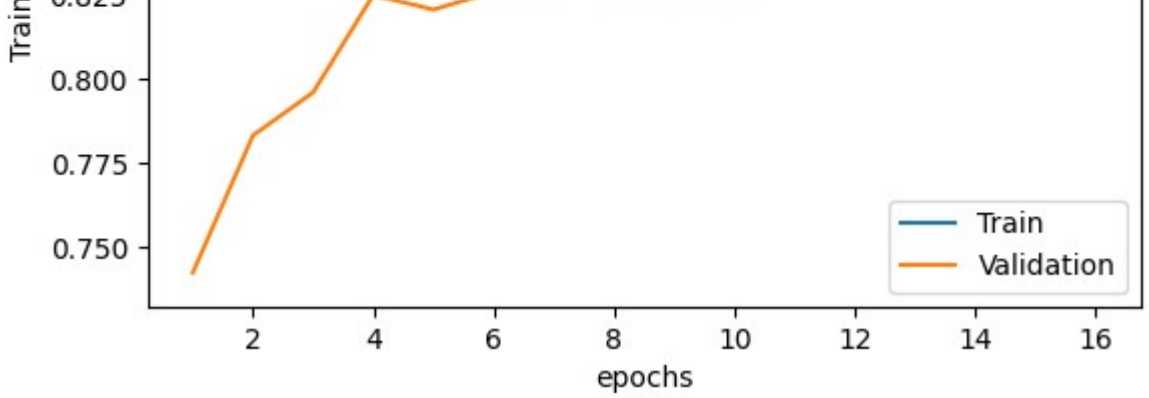




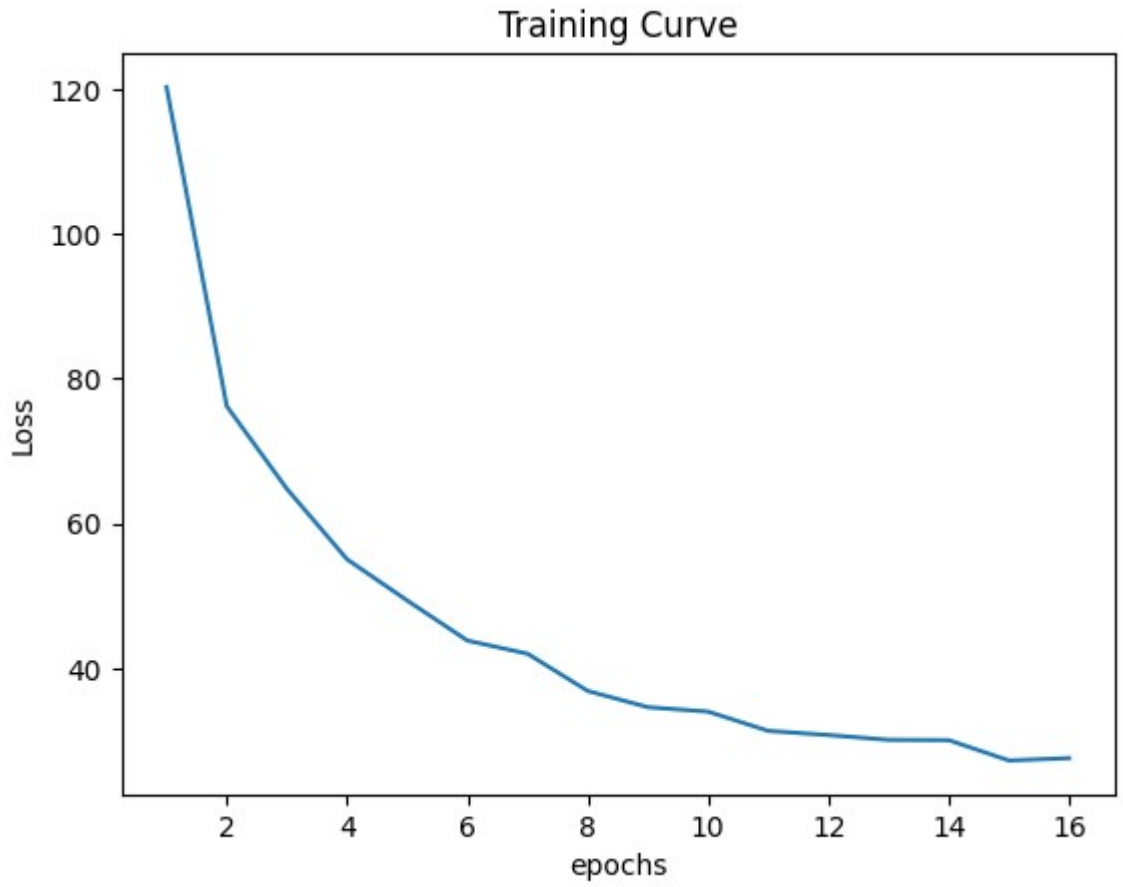

```
Final Training Accuracy: 0.9362666666666667
Final Validation Accuracy: 0.8489
max gradient: 0.5
Epoch 1/16, Training loss(ave. loss per minibatch): 0.77,Time: 1.38786s
Epoch 2/16, Training loss(ave. loss per minibatch): 0.5,Time: 2.82454s
Epoch 3/16, Training loss(ave. loss per minibatch): 0.42,Time: 4.64938s
Epoch 4/16, Training loss(ave. loss per minibatch): 0.35,Time: 6.08437s
Epoch 5/16, Training loss(ave. loss per minibatch): 0.32,Time: 7.53717s
Epoch 6/16, Training loss(ave. loss per minibatch): 0.28,Time: 8.94999s
Epoch 7/16, Training loss(ave. loss per minibatch): 0.26,Time: 10.35957s
Epoch 8/16, Training loss(ave. loss per minibatch): 0.25,Time: 11.7544s
Epoch 9/16, Training loss(ave. loss per minibatch): 0.22,Time: 13.19354s
Epoch 10/16, Training loss(ave. loss per minibatch): 0.23,Time: 14.63581s
Epoch 11/16, Training loss(ave. loss per minibatch): 0.2,Time: 16.33357s
Epoch 12/16, Training loss(ave. loss per minibatch): 0.19,Time: 17.73545s
Epoch 13/16, Training loss(ave. loss per minibatch): 0.19,Time: 19.15532s
Epoch 14/16, Training loss(ave. loss per minibatch): 0.18,Time: 20.60044s
Epoch 15/16, Training loss(ave. loss per minibatch): 0.18,Time: 22.03969s
Epoch 16/16, Training loss(ave. loss per minibatch): 0.16,Time: 23.4344s
```

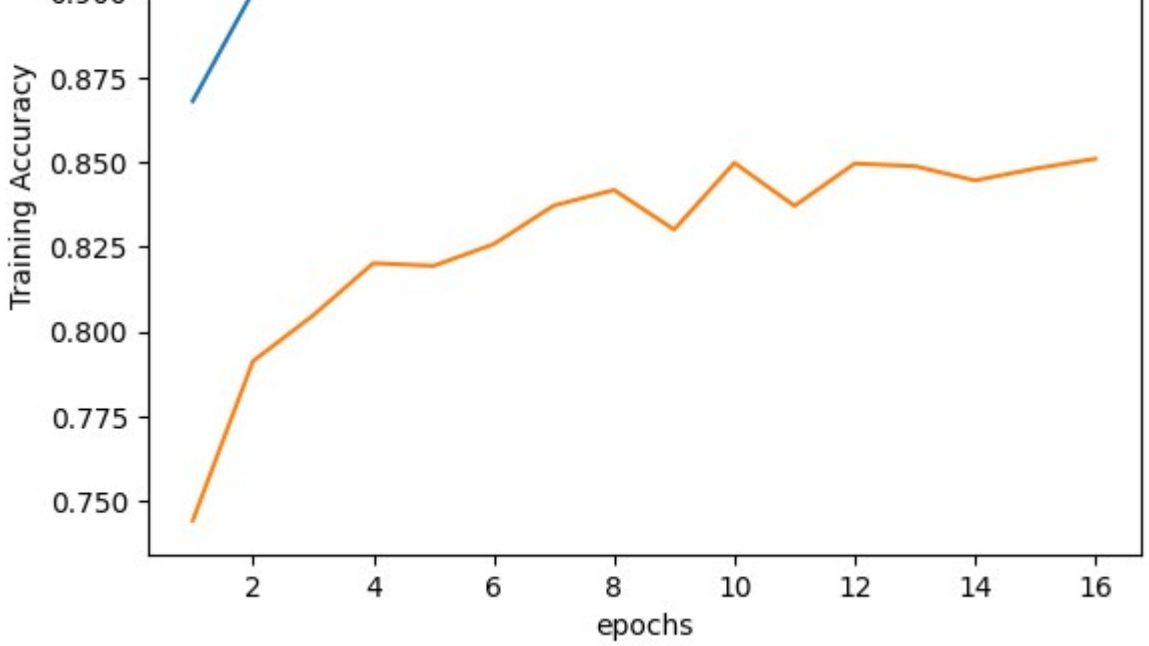




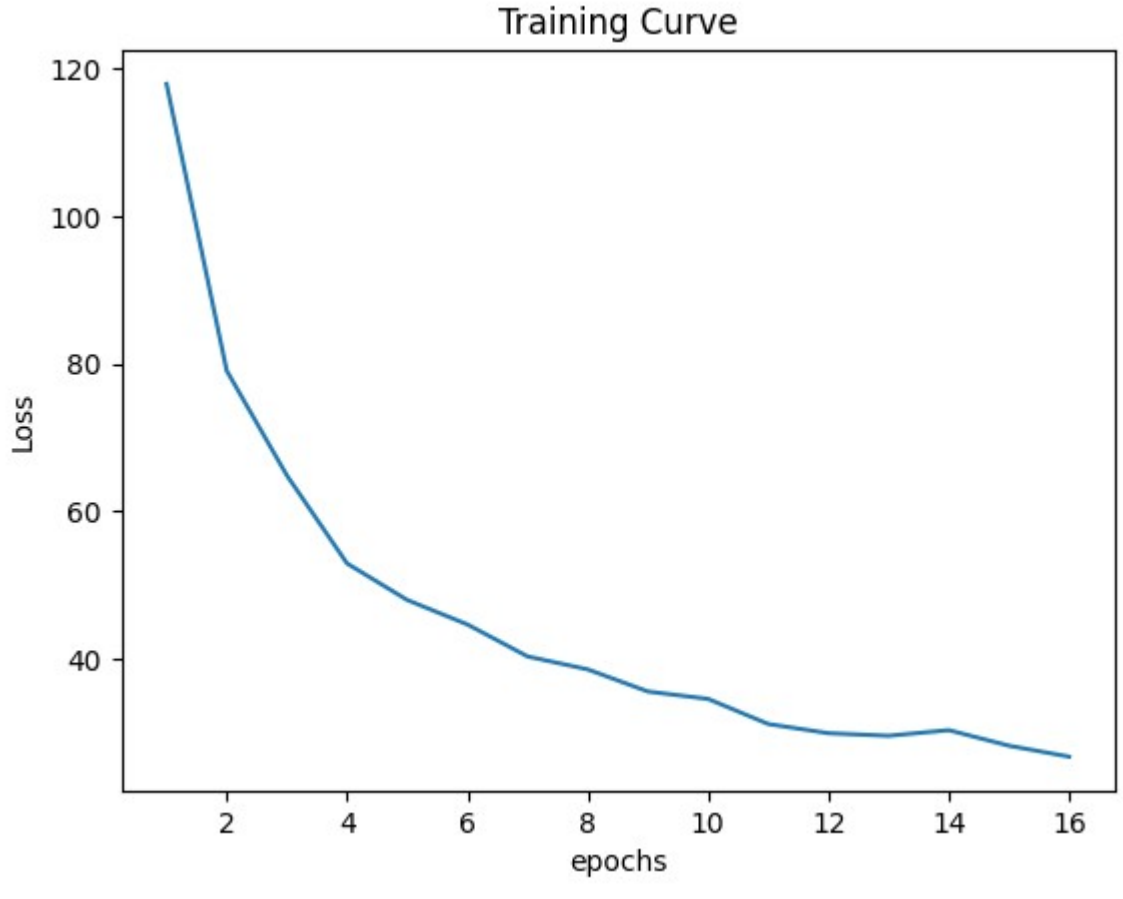


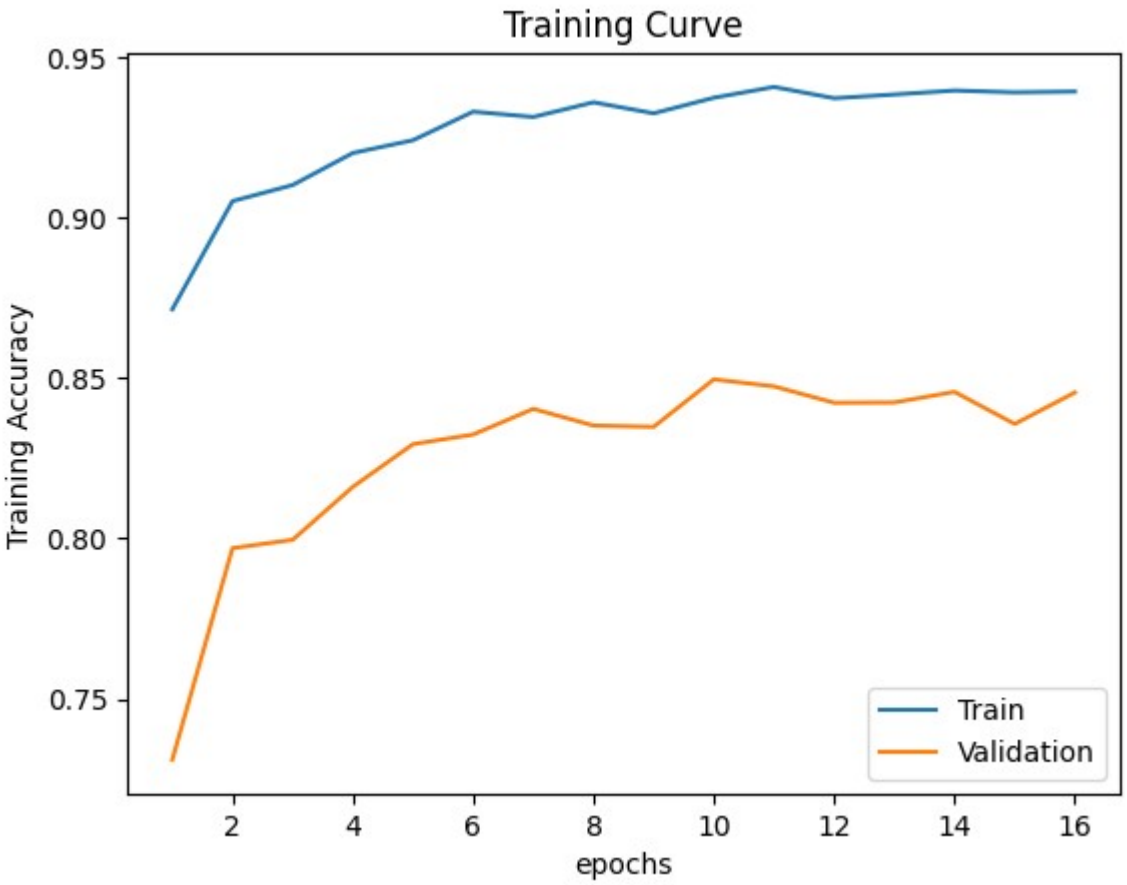

```
Final Training Accuracy: 0.9424
Final Validation Accuracy: 0.846
max gradient: 1.0
Epoch 1/16, Training loss(ave. loss per minibatch): 0.77,Time: 1.44488s
Epoch 2/16, Training loss(ave. loss per minibatch): 0.49,Time: 3.10567s
Epoch 3/16, Training loss(ave. loss per minibatch): 0.41,Time: 4.59046s
Epoch 4/16, Training loss(ave. loss per minibatch): 0.35,Time: 6.03707s
Epoch 5/16, Training loss(ave. loss per minibatch): 0.31,Time: 7.45851s
Epoch 6/16, Training loss(ave. loss per minibatch): 0.28,Time: 8.86919s
Epoch 7/16, Training loss(ave. loss per minibatch): 0.27,Time: 10.31284s
Epoch 8/16, Training loss(ave. loss per minibatch): 0.23,Time: 11.73131s
Epoch 9/16, Training loss(ave. loss per minibatch): 0.22,Time: 13.1756s
Epoch 10/16, Training loss(ave. loss per minibatch): 0.22,Time: 14.98565s
Epoch 11/16, Training loss(ave. loss per minibatch): 0.2,Time: 16.47668s
Epoch 12/16, Training loss(ave. loss per minibatch): 0.2,Time: 17.95671s
Epoch 13/16, Training loss(ave. loss per minibatch): 0.19,Time: 19.41798s
Epoch 14/16, Training loss(ave. loss per minibatch): 0.19,Time: 20.90212s
Epoch 15/16, Training loss(ave. loss per minibatch): 0.17,Time: 22.33272s
Epoch 16/16, Training loss(ave. loss per minibatch): 0.18,Time: 23.77872s
```



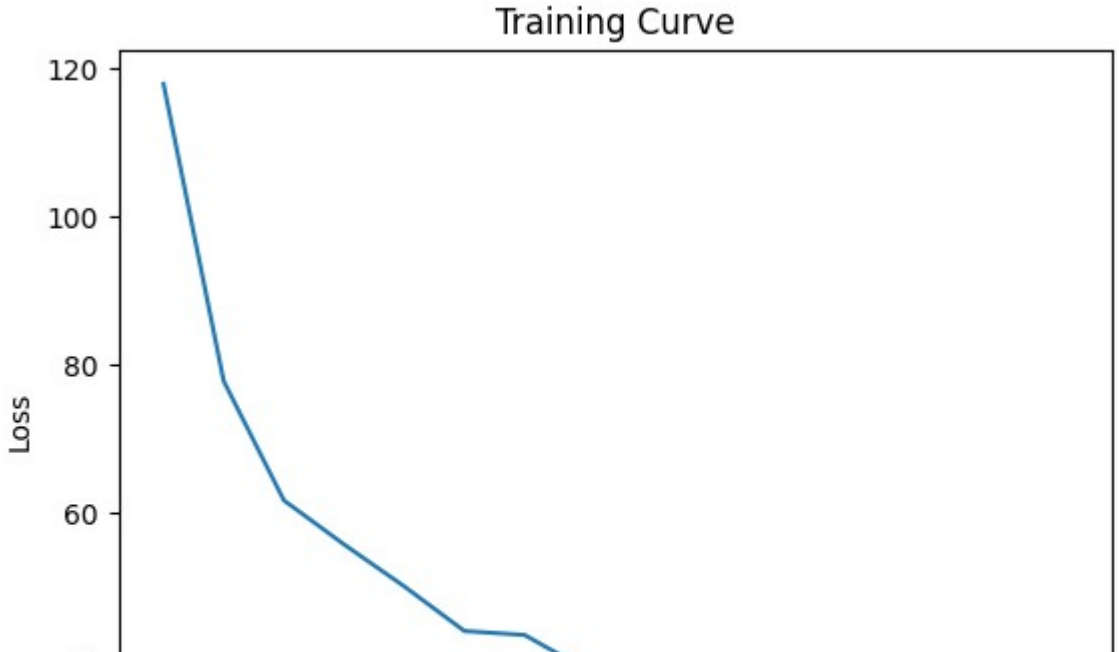


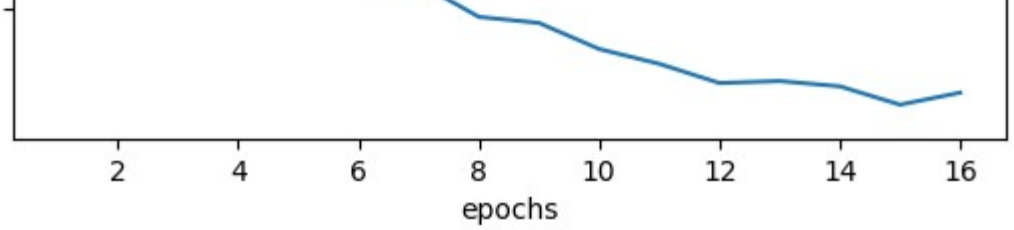

```
Final Training Accuracy: 0.9424
Final Validation Accuracy: 0.851
max gradient: 1.5
Epoch 1/16, Training loss(ave. loss per minibatch): 0.75,Time: 1.74134s
Epoch 2/16, Training loss(ave. loss per minibatch): 0.5,Time: 3.23016s
Epoch 3/16, Training loss(ave. loss per minibatch): 0.41,Time: 4.68035s
Epoch 4/16, Training loss(ave. loss per minibatch): 0.34,Time: 6.10416s
Epoch 5/16, Training loss(ave. loss per minibatch): 0.31,Time: 7.5595s
Epoch 6/16, Training loss(ave. loss per minibatch): 0.28,Time: 9.00574s
Epoch 7/16, Training loss(ave. loss per minibatch): 0.26,Time: 10.49701s
Epoch 8/16, Training loss(ave. loss per minibatch): 0.25,Time: 11.96298s
Epoch 9/16, Training loss(ave. loss per minibatch): 0.23,Time: 13.79461s
Epoch 10/16, Training loss(ave. loss per minibatch): 0.22,Time: 15.25393s
Epoch 11/16, Training loss(ave. loss per minibatch): 0.2,Time: 16.66872s
Epoch 12/16, Training loss(ave. loss per minibatch): 0.19,Time: 18.09882s
Epoch 13/16, Training loss(ave. loss per minibatch): 0.19,Time: 19.51402s
Epoch 14/16, Training loss(ave. loss per minibatch): 0.19,Time: 20.94042s
Epoch 15/16, Training loss(ave. loss per minibatch): 0.18,Time: 22.37597s
Epoch 16/16, Training loss(ave. loss per minibatch): 0.17,Time: 24.02212s
```

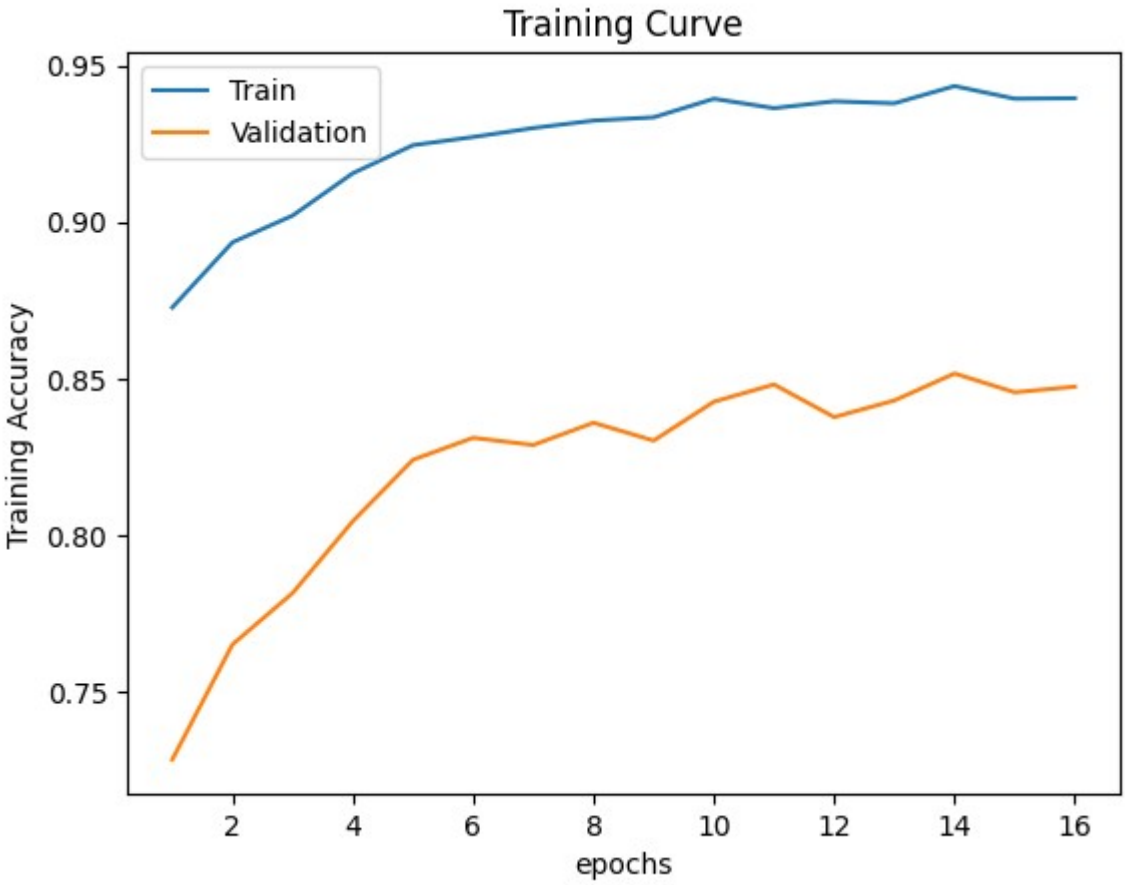




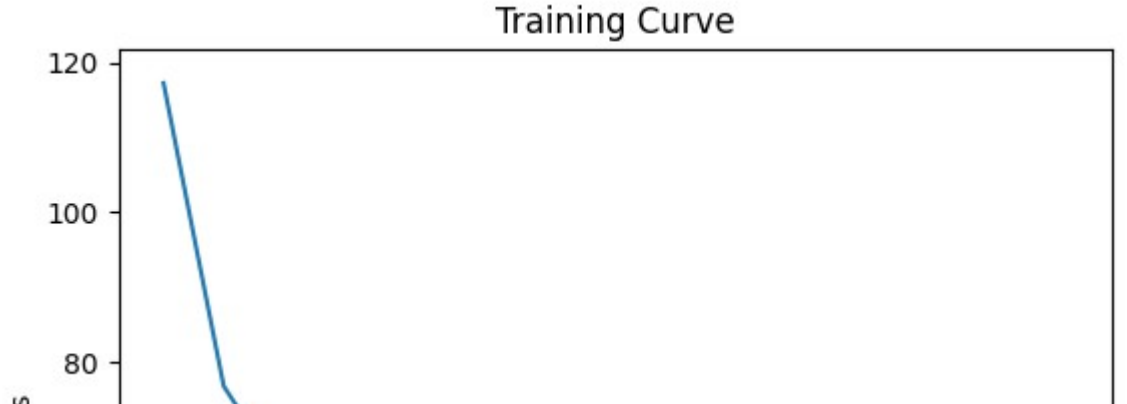

```
Final Training Accuracy: 0.9392666666666667
Final Validation Accuracy: 0.8455
max gradient: 2.0
Epoch 1/16, Training loss(ave. loss per minibatch): 0.75,Time: 1.51032s
Epoch 2/16, Training loss(ave. loss per minibatch): 0.5,Time: 2.9398s
Epoch 3/16, Training loss(ave. loss per minibatch): 0.39,Time: 4.43525s
Epoch 4/16, Training loss(ave. loss per minibatch): 0.35,Time: 5.94197s
Epoch 5/16, Training loss(ave. loss per minibatch): 0.32,Time: 7.66345s
Epoch 6/16, Training loss(ave. loss per minibatch): 0.28,Time: 9.78202s
Epoch 7/16, Training loss(ave. loss per minibatch): 0.28,Time: 12.75028s
Epoch 8/16, Training loss(ave. loss per minibatch): 0.25,Time: 14.30346s
Epoch 9/16, Training loss(ave. loss per minibatch): 0.24,Time: 15.79853s
Epoch 10/16, Training loss(ave. loss per minibatch): 0.22,Time: 17.30562s
Epoch 11/16, Training loss(ave. loss per minibatch): 0.21,Time: 18.80474s
Epoch 12/16, Training loss(ave. loss per minibatch): 0.19,Time: 20.37918s
Epoch 13/16, Training loss(ave. loss per minibatch): 0.19,Time: 21.85927s
Epoch 14/16, Training loss(ave. loss per minibatch): 0.19,Time: 23.51229s
Epoch 15/16, Training loss(ave. loss per minibatch): 0.17,Time: 25.20061s
Epoch 16/16, Training loss(ave. loss per minibatch): 0.18,Time: 26.67042s
```

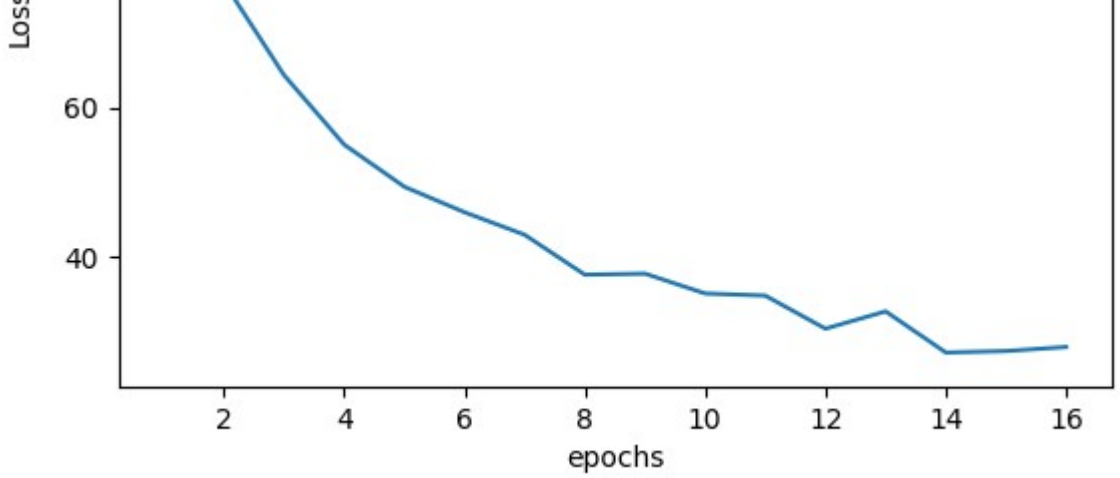


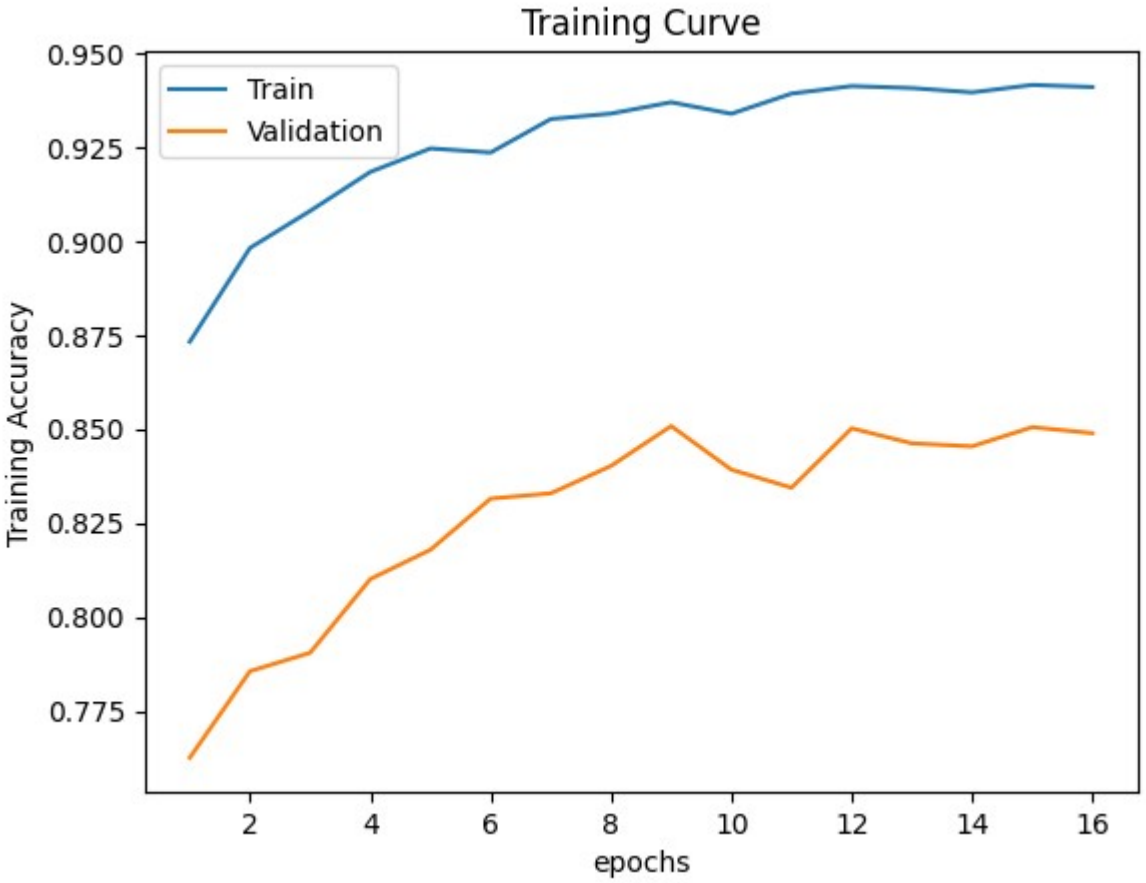




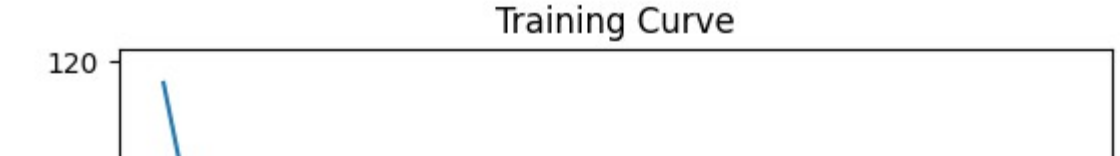

```
Final Training Accuracy: 0.9394833333333333
Final Validation Accuracy: 0.8475
max gradient: 2.5
Epoch 1/16, Training loss(ave. loss per minibatch): 0.75,Time: 1.48914s
Epoch 2/16, Training loss(ave. loss per minibatch): 0.49,Time: 3.00977s
Epoch 3/16, Training loss(ave. loss per minibatch): 0.41,Time: 4.53122s
Epoch 4/16, Training loss(ave. loss per minibatch): 0.35,Time: 5.99814s
Epoch 5/16, Training loss(ave. loss per minibatch): 0.31,Time: 7.52944s
Epoch 6/16, Training loss(ave. loss per minibatch): 0.29,Time: 9.34287s
Epoch 7/16, Training loss(ave. loss per minibatch): 0.27,Time: 10.78877s
Epoch 8/16, Training loss(ave. loss per minibatch): 0.24,Time: 12.27437s
Epoch 9/16, Training loss(ave. loss per minibatch): 0.24,Time: 13.73754s
Epoch 10/16, Training loss(ave. loss per minibatch): 0.22,Time: 15.2676s
Epoch 11/16, Training loss(ave. loss per minibatch): 0.22,Time: 16.7832s
Epoch 12/16, Training loss(ave. loss per minibatch): 0.19,Time: 18.26238s
Epoch 13/16, Training loss(ave. loss per minibatch): 0.21,Time: 20.1669s
Epoch 14/16, Training loss(ave. loss per minibatch): 0.17,Time: 21.57727s
Epoch 15/16, Training loss(ave. loss per minibatch): 0.17,Time: 23.05883s
Epoch 16/16, Training loss(ave. loss per minibatch): 0.18,Time: 24.52057s
```

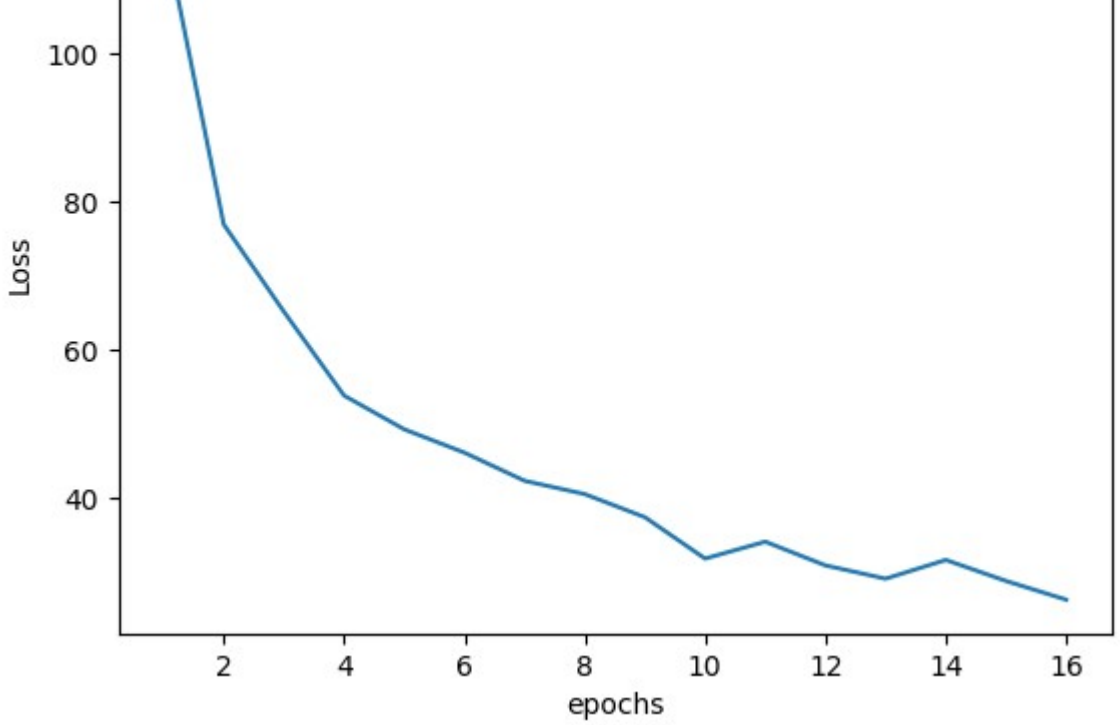


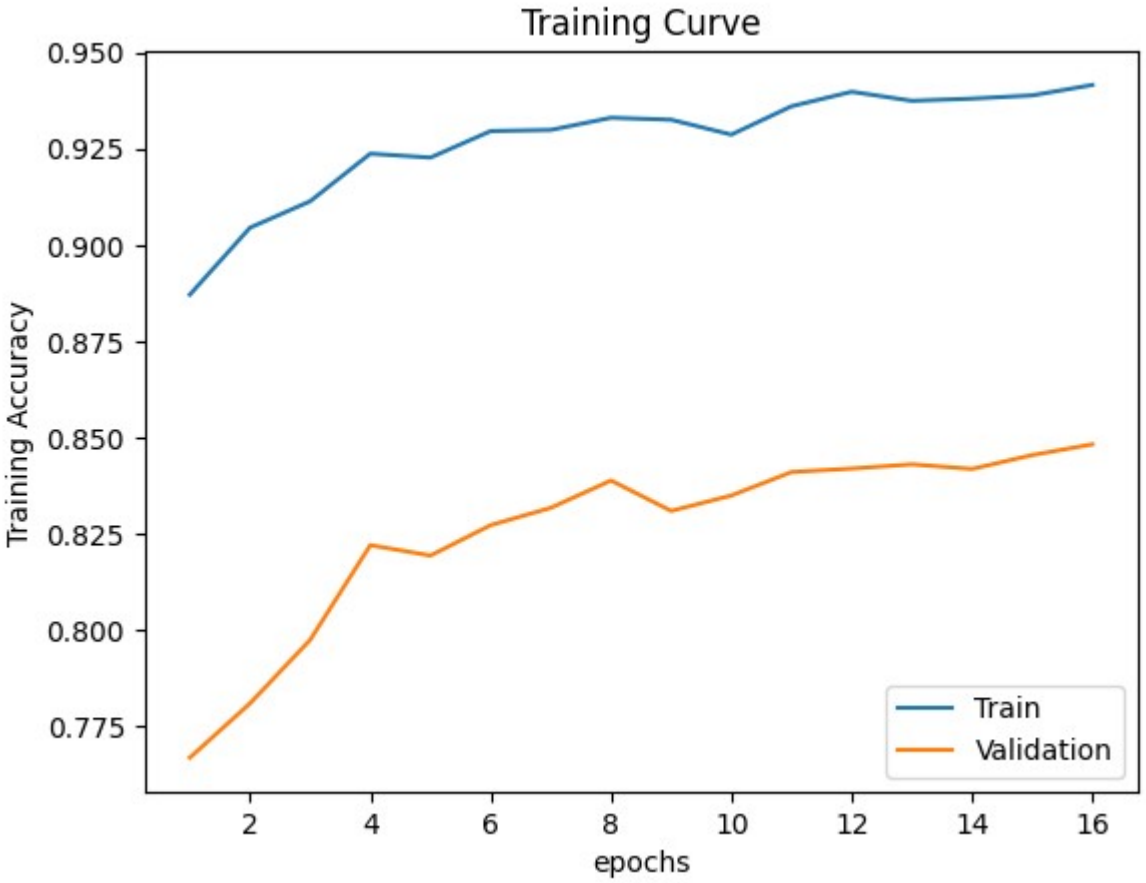





```
Final Training Accuracy: 0.9409666666666666
Final Validation Accuracy: 0.8489
max gradient: 3.0
Epoch 1/16, Training loss(ave. loss per minibatch): 0.75,Time: 1.49071s
Epoch 2/16, Training loss(ave. loss per minibatch): 0.49,Time: 2.93084s
Epoch 3/16, Training loss(ave. loss per minibatch): 0.41,Time: 4.42117s
Epoch 4/16, Training loss(ave. loss per minibatch): 0.34,Time: 6.19117s
Epoch 5/16, Training loss(ave. loss per minibatch): 0.31,Time: 7.84563s
Epoch 6/16, Training loss(ave. loss per minibatch): 0.29,Time: 9.31822s
Epoch 7/16, Training loss(ave. loss per minibatch): 0.27,Time: 10.77412s
Epoch 8/16, Training loss(ave. loss per minibatch): 0.26,Time: 12.33976s
Epoch 9/16, Training loss(ave. loss per minibatch): 0.24,Time: 13.82555s
Epoch 10/16, Training loss(ave. loss per minibatch): 0.2,Time: 15.315s
Epoch 11/16, Training loss(ave. loss per minibatch): 0.22,Time: 16.80729s
Epoch 12/16, Training loss(ave. loss per minibatch): 0.2,Time: 18.63854s
Epoch 13/16, Training loss(ave. loss per minibatch): 0.19,Time: 20.10957s
Epoch 14/16, Training loss(ave. loss per minibatch): 0.2,Time: 21.6101s
Epoch 15/16, Training loss(ave. loss per minibatch): 0.18,Time: 23.03806s
Epoch 16/16, Training loss(ave. loss per minibatch): 0.17,Time: 24.62552s
```







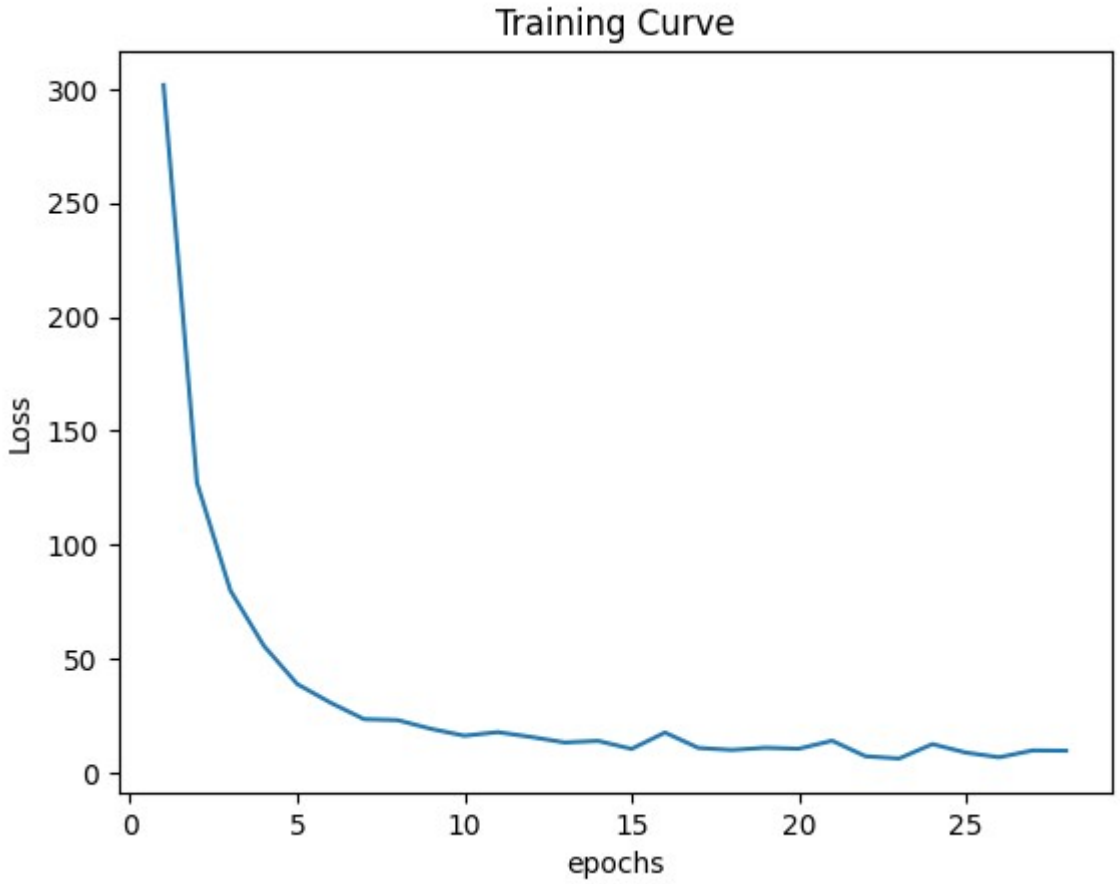

```
Final Training Accuracy: 0.94155
Final Validation Accuracy: 0.8482
```

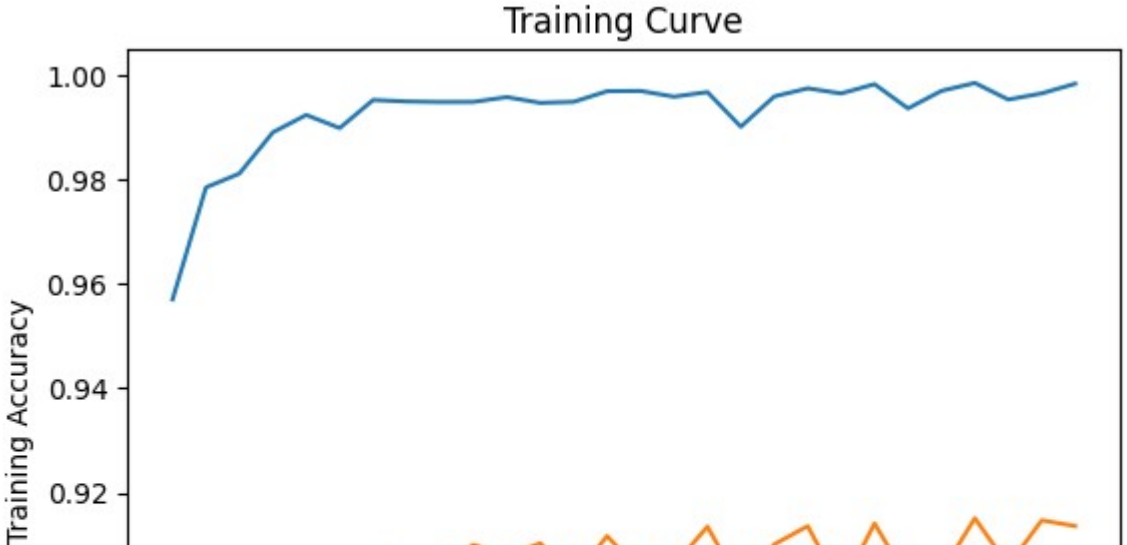
Training on the full training dataset

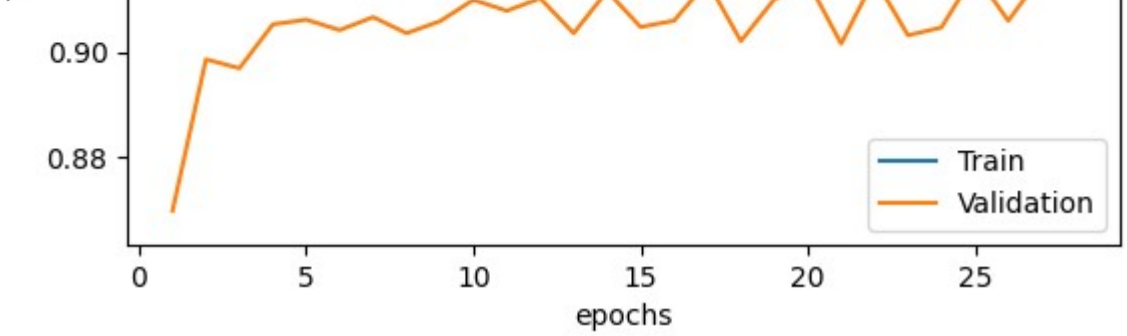
```
train(model, mnist_train, lr_choice=0.001, num_epochs=28, grad_clip=1.5)
```

```
Epoch 1/28, Training loss(ave. loss per minibatch): 0.32,Time: 4.50151s
Epoch 2/28, Training loss(ave. loss per minibatch): 0.14,Time: 8.53627s
Epoch 3/28, Training loss(ave. loss per minibatch): 0.09,Time: 12.26765s
Epoch 4/28, Training loss(ave. loss per minibatch): 0.06,Time: 16.57383s
Epoch 5/28, Training loss(ave. loss per minibatch): 0.04,Time: 20.43186s
Epoch 6/28, Training loss(ave. loss per minibatch): 0.03,Time: 24.30658s
```

```
Epoch 7/28, Training loss(ave. loss per minibatch): 0.02,Time: 28.70751s
Epoch 8/28, Training loss(ave. loss per minibatch): 0.02,Time: 32.68039s
Epoch 9/28, Training loss(ave. loss per minibatch): 0.02,Time: 36.79875s
Epoch 10/28, Training loss(ave. loss per minibatch): 0.02,Time: 41.3006s
Epoch 11/28, Training loss(ave. loss per minibatch): 0.02,Time: 45.21382s
Epoch 12/28, Training loss(ave. loss per minibatch): 0.02,Time: 49.58004s
Epoch 13/28, Training loss(ave. loss per minibatch): 0.01,Time: 53.50195s
Epoch 14/28, Training loss(ave. loss per minibatch): 0.01,Time: 57.44882s
Epoch 15/28, Training loss(ave. loss per minibatch): 0.01,Time: 61.86534s
Epoch 16/28, Training loss(ave. loss per minibatch): 0.02,Time: 65.74626s
Epoch 17/28, Training loss(ave. loss per minibatch): 0.01,Time: 69.70436s
Epoch 18/28, Training loss(ave. loss per minibatch): 0.01,Time: 74.12596s
Epoch 19/28, Training loss(ave. loss per minibatch): 0.01,Time: 78.03296s
Epoch 20/28, Training loss(ave. loss per minibatch): 0.01,Time: 81.99744s
Epoch 21/28, Training loss(ave. loss per minibatch): 0.01,Time: 86.29422s
Epoch 22/28, Training loss(ave. loss per minibatch): 0.01,Time: 90.16831s
Epoch 23/28, Training loss(ave. loss per minibatch): 0.01,Time: 94.25993s
Epoch 24/28, Training loss(ave. loss per minibatch): 0.01,Time: 98.42636s
Epoch 25/28, Training loss(ave. loss per minibatch): 0.01,Time: 102.28357s
Epoch 26/28, Training loss(ave. loss per minibatch): 0.01,Time: 106.60248s
Epoch 27/28, Training loss(ave. loss per minibatch): 0.01,Time: 110.66159s
Epoch 28/28, Training loss(ave. loss per minibatch): 0.01,Time: 114.62303s
```



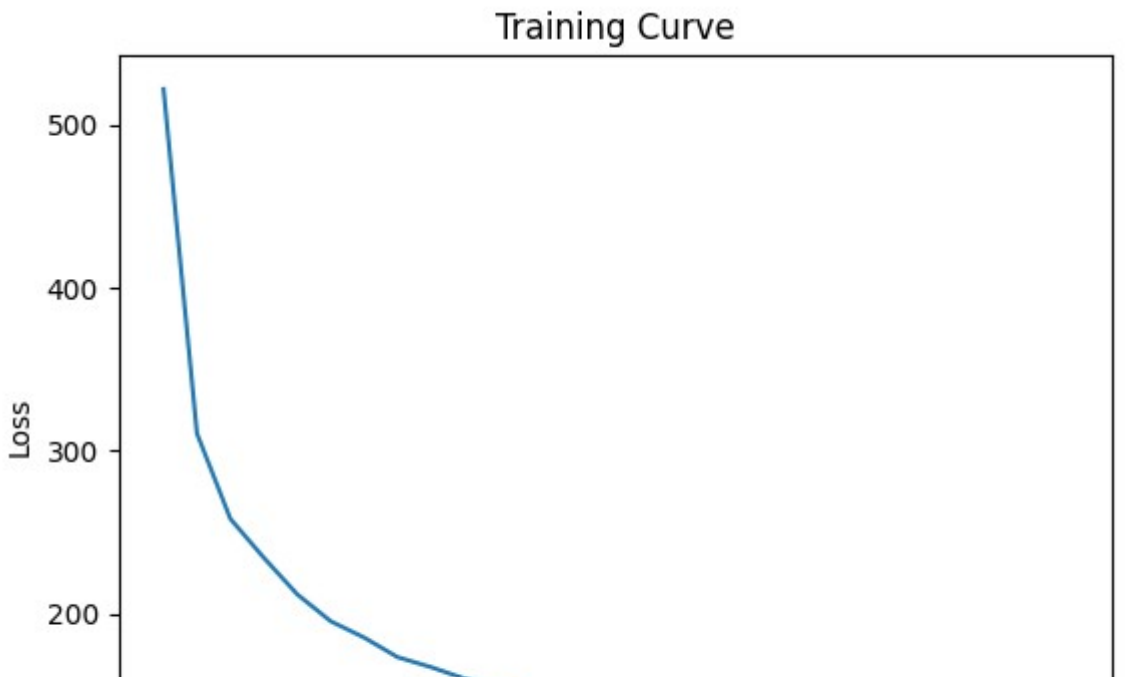


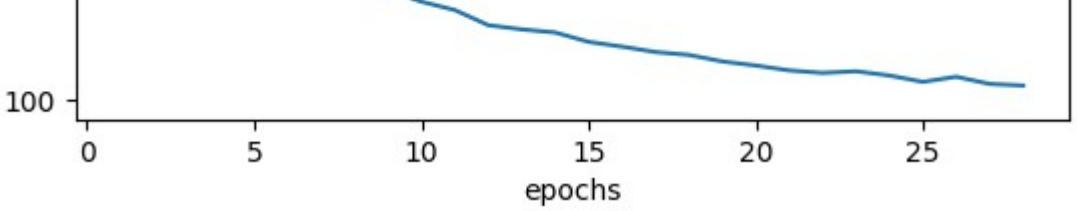


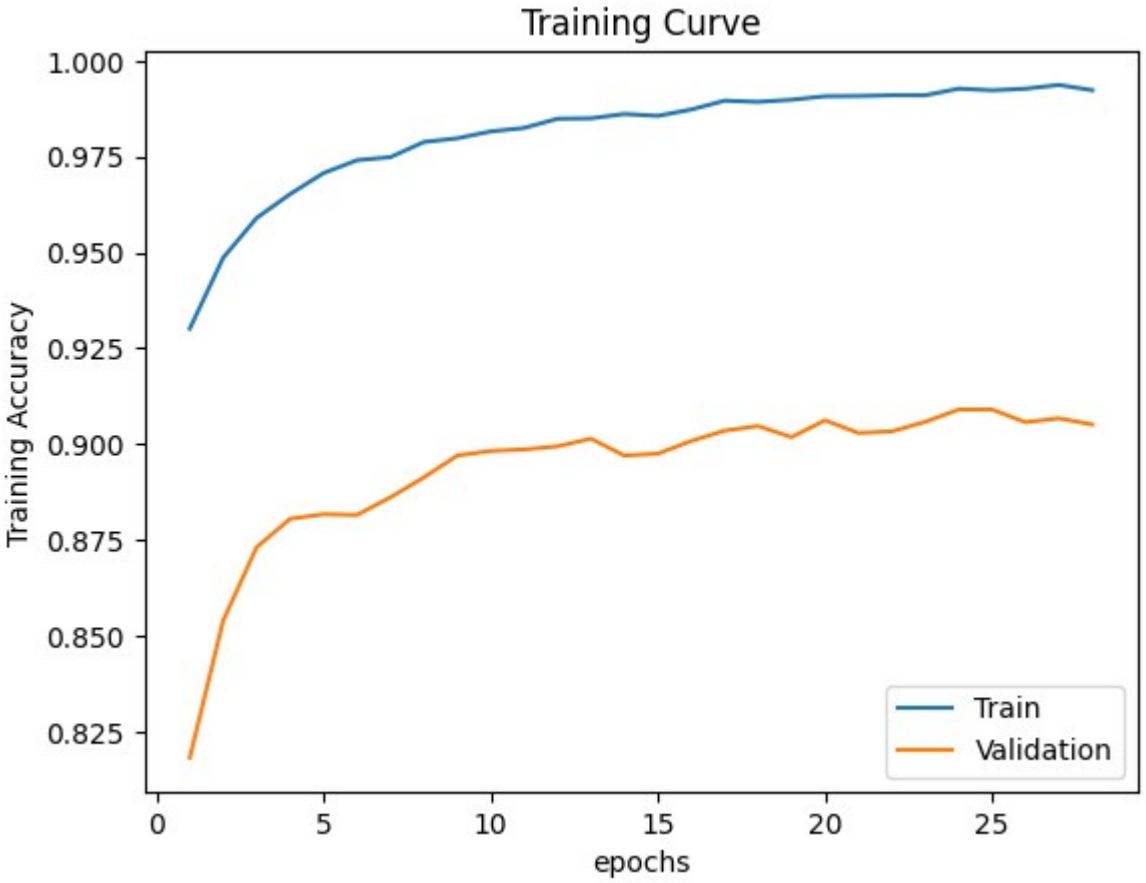

```
Final Training Accuracy: 0.9983666666666666
Final Validation Accuracy: 0.9136
```

```
train(model_drop, mnist_train, lr_choice=0.001, num_epochs=28, grad_clip=1.5)
```

```
Epoch 1/28, Training loss(ave. loss per minibatch): 0.56,Time: 4.63764s
Epoch 2/28, Training loss(ave. loss per minibatch): 0.33,Time: 8.78233s
Epoch 3/28, Training loss(ave. loss per minibatch): 0.28,Time: 13.10235s
Epoch 4/28, Training loss(ave. loss per minibatch): 0.25,Time: 17.44915s
Epoch 5/28, Training loss(ave. loss per minibatch): 0.23,Time: 21.61247s
Epoch 6/28, Training loss(ave. loss per minibatch): 0.21,Time: 26.16808s
Epoch 7/28, Training loss(ave. loss per minibatch): 0.2,Time: 30.52556s
Epoch 8/28, Training loss(ave. loss per minibatch): 0.18,Time: 34.8368s
Epoch 9/28, Training loss(ave. loss per minibatch): 0.18,Time: 39.41491s
Epoch 10/28, Training loss(ave. loss per minibatch): 0.17,Time: 43.67116s
Epoch 11/28, Training loss(ave. loss per minibatch): 0.17,Time: 48.15706s
Epoch 12/28, Training loss(ave. loss per minibatch): 0.16,Time: 52.38622s
Epoch 13/28, Training loss(ave. loss per minibatch): 0.15,Time: 56.55088s
Epoch 14/28, Training loss(ave. loss per minibatch): 0.15,Time: 60.95914s
Epoch 15/28, Training loss(ave. loss per minibatch): 0.14,Time: 65.18783s
Epoch 16/28, Training loss(ave. loss per minibatch): 0.14,Time: 69.78642s
Epoch 17/28, Training loss(ave. loss per minibatch): 0.14,Time: 74.11115s
Epoch 18/28, Training loss(ave. loss per minibatch): 0.14,Time: 78.3355s
Epoch 19/28, Training loss(ave. loss per minibatch): 0.13,Time: 82.8955s
Epoch 20/28, Training loss(ave. loss per minibatch): 0.13,Time: 87.10627s
Epoch 21/28, Training loss(ave. loss per minibatch): 0.13,Time: 91.34893s
Epoch 22/28, Training loss(ave. loss per minibatch): 0.12,Time: 95.98053s
Epoch 23/28, Training loss(ave. loss per minibatch): 0.13,Time: 100.22226s
Epoch 24/28, Training loss(ave. loss per minibatch): 0.12,Time: 105.05111s
Epoch 25/28, Training loss(ave. loss per minibatch): 0.12,Time: 109.3619s
Epoch 26/28, Training loss(ave. loss per minibatch): 0.12,Time: 113.57096s
Epoch 27/28, Training loss(ave. loss per minibatch): 0.12,Time: 118.11445s
Epoch 28/28, Training loss(ave. loss per minibatch): 0.12,Time: 122.24811s
```





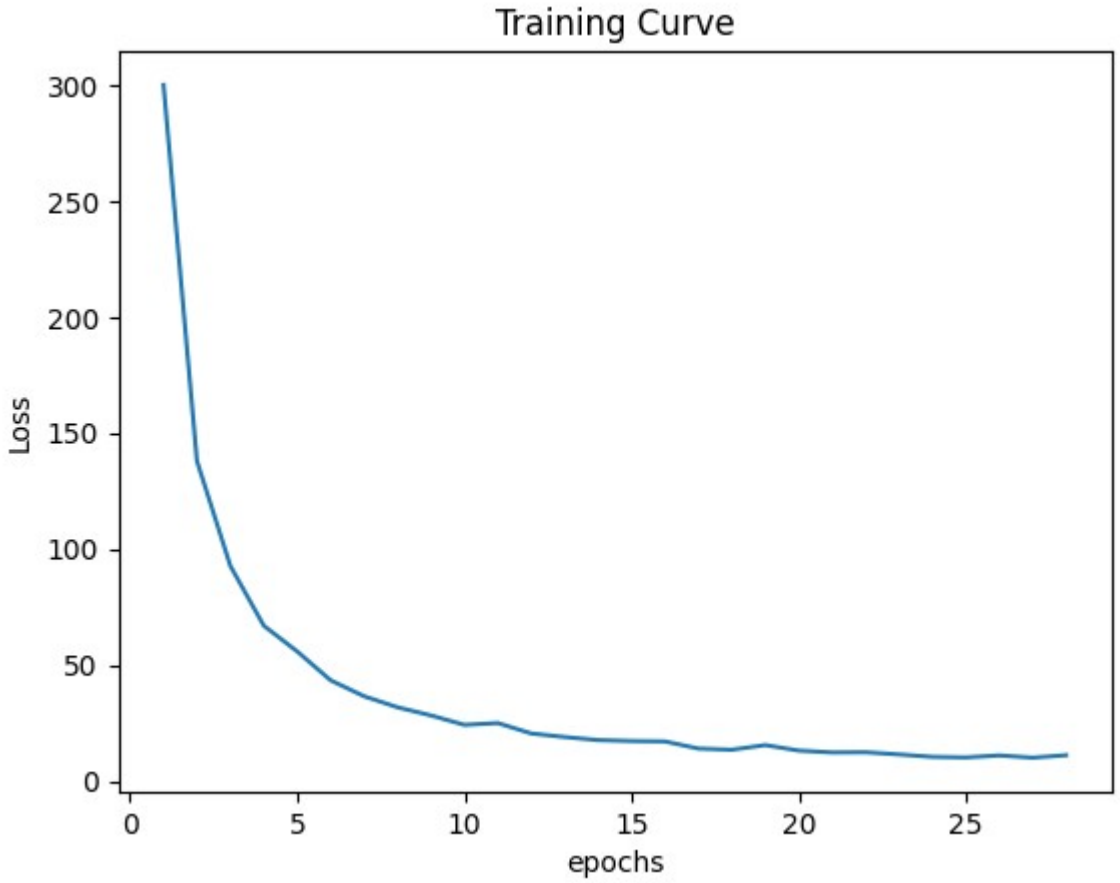


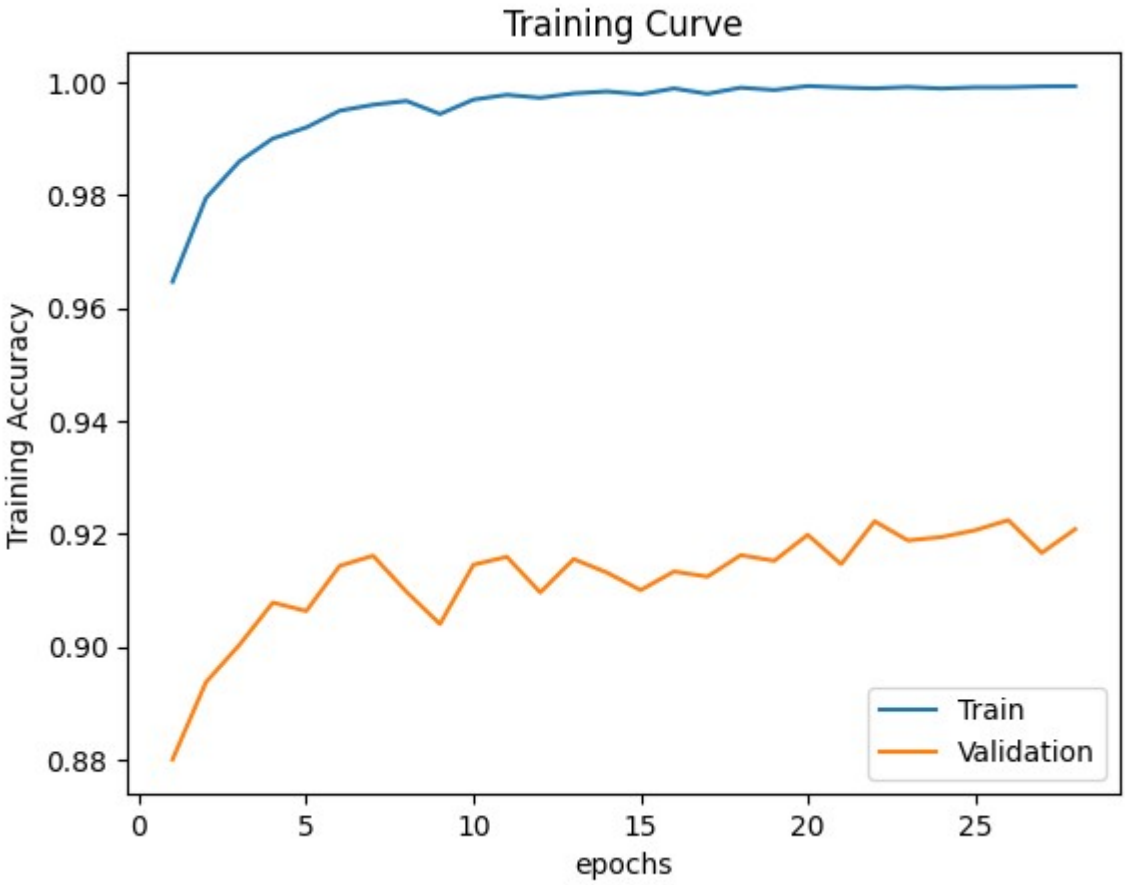

```
Final Training Accuracy: 0.9923
Final Validation Accuracy: 0.9051
```

```
train(model_bn, mnist_train, lr_choice=0.001, num_epochs=28, grad_clip=1.5)
```

```
Epoch 1/28, Training loss(ave. loss per minibatch): 0.32,Time: 4.7764s
Epoch 2/28, Training loss(ave. loss per minibatch): 0.15,Time: 9.06145s
Epoch 3/28, Training loss(ave. loss per minibatch): 0.1,Time: 13.49442s
Epoch 4/28, Training loss(ave. loss per minibatch): 0.07,Time: 18.52781s
Epoch 5/28, Training loss(ave. loss per minibatch): 0.06,Time: 23.29467s
Epoch 6/28, Training loss(ave. loss per minibatch): 0.05,Time: 28.19371s
Epoch 7/28, Training loss(ave. loss per minibatch): 0.04,Time: 32.67989s
Epoch 8/28, Training loss(ave. loss per minibatch): 0.03,Time: 37.19207s
Epoch 9/28, Training loss(ave. loss per minibatch): 0.03,Time: 41.79867s
Epoch 10/28, Training loss(ave. loss per minibatch): 0.03,Time: 46.18244s
Epoch 11/28, Training loss(ave. loss per minibatch): 0.03,Time: 51.20674s
Epoch 12/28, Training loss(ave. loss per minibatch): 0.02,Time: 55.75255s
Epoch 13/28, Training loss(ave. loss per minibatch): 0.02,Time: 60.5609s
Epoch 14/28, Training loss(ave. loss per minibatch): 0.02,Time: 64.99895s
Epoch 15/28, Training loss(ave. loss per minibatch): 0.02,Time: 69.33207s
Epoch 16/28, Training loss(ave. loss per minibatch): 0.02,Time: 73.87101s
Epoch 17/28, Training loss(ave. loss per minibatch): 0.01,Time: 78.26936s
Epoch 18/28, Training loss(ave. loss per minibatch): 0.01,Time: 82.90815s
Epoch 19/28, Training loss(ave. loss per minibatch): 0.02,Time: 87.47281s
Epoch 20/28, Training loss(ave. loss per minibatch): 0.01,Time: 91.9118s
Epoch 21/28, Training loss(ave. loss per minibatch): 0.01,Time: 96.75357s
Epoch 22/28, Training loss(ave. loss per minibatch): 0.01,Time: 101.11686s
Epoch 23/28, Training loss(ave. loss per minibatch): 0.01,Time: 105.56125s
Epoch 24/28, Training loss(ave. loss per minibatch): 0.01,Time: 110.09144s
Epoch 25/28, Training loss(ave. loss per minibatch): 0.01,Time: 114.5169s
```

```
Epoch 26/28, Training loss(ave. loss per minibatch): 0.01,Time: 119.29627s
Epoch 27/28, Training loss(ave. loss per minibatch): 0.01,Time: 123.58447s
Epoch 28/28, Training loss(ave. loss per minibatch): 0.01,Time: 128.44768s
```



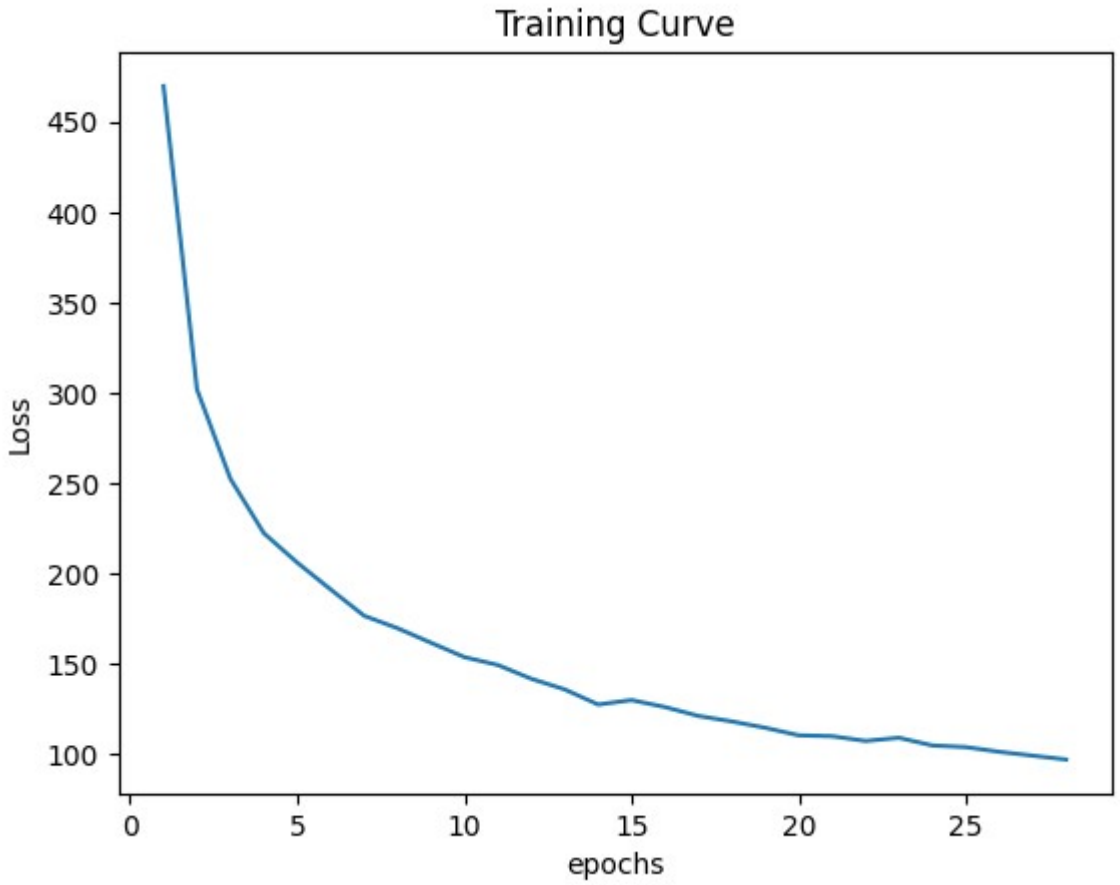


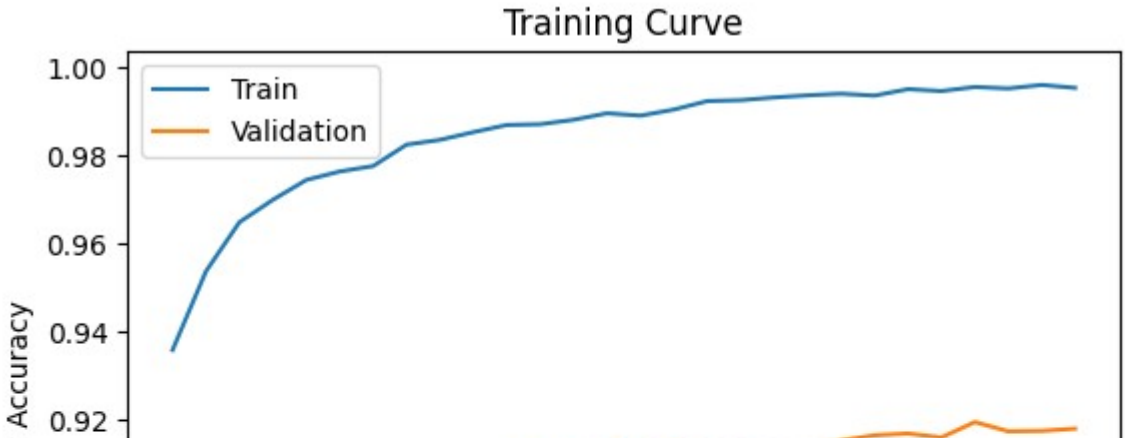

```
Final Training Accuracy: 0.99925
Final Validation Accuracy: 0.9208
```

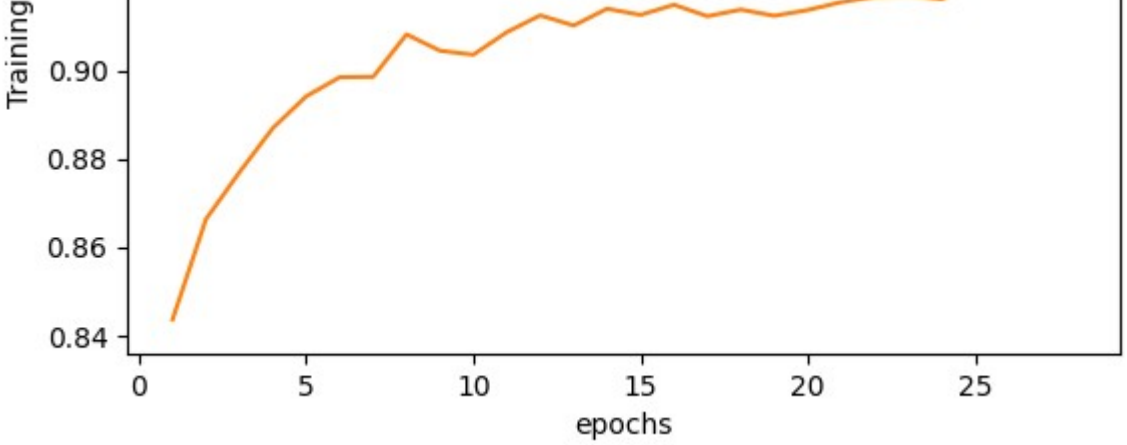
```
train(model_both, mnist_train, lr_choice=0.001, num_epochs=28, grad_clip=1.5)
```

```
Epoch 1/28, Training loss(ave. loss per minibatch): 0.5,Time: 4.57838s
Epoch 2/28, Training loss(ave. loss per minibatch): 0.32,Time: 9.0271s
Epoch 3/28, Training loss(ave. loss per minibatch): 0.27,Time: 13.8764s
```

```
Epoch 4/28, Training loss(ave. loss per minibatch): 0.24,Time: 18.34594s
Epoch 5/28, Training loss(ave. loss per minibatch): 0.22,Time: 23.27402s
Epoch 6/28, Training loss(ave. loss per minibatch): 0.2,Time: 27.70068s
Epoch 7/28, Training loss(ave. loss per minibatch): 0.19,Time: 32.09356s
Epoch 8/28, Training loss(ave. loss per minibatch): 0.18,Time: 37.1967s
Epoch 9/28, Training loss(ave. loss per minibatch): 0.17,Time: 41.76503s
Epoch 10/28, Training loss(ave. loss per minibatch): 0.16,Time: 46.61043s
Epoch 11/28, Training loss(ave. loss per minibatch): 0.16,Time: 51.03332s
Epoch 12/28, Training loss(ave. loss per minibatch): 0.15,Time: 55.77917s
Epoch 13/28, Training loss(ave. loss per minibatch): 0.14,Time: 60.65256s
Epoch 14/28, Training loss(ave. loss per minibatch): 0.14,Time: 65.16086s
Epoch 15/28, Training loss(ave. loss per minibatch): 0.14,Time: 70.21728s
Epoch 16/28, Training loss(ave. loss per minibatch): 0.13,Time: 74.9737s
Epoch 17/28, Training loss(ave. loss per minibatch): 0.13,Time: 79.89613s
Epoch 18/28, Training loss(ave. loss per minibatch): 0.13,Time: 84.47187s
Epoch 19/28, Training loss(ave. loss per minibatch): 0.12,Time: 88.88638s
Epoch 20/28, Training loss(ave. loss per minibatch): 0.12,Time: 93.85122s
Epoch 21/28, Training loss(ave. loss per minibatch): 0.12,Time: 98.27126s
Epoch 22/28, Training loss(ave. loss per minibatch): 0.11,Time: 103.24928s
Epoch 23/28, Training loss(ave. loss per minibatch): 0.12,Time: 107.97103s
Epoch 24/28, Training loss(ave. loss per minibatch): 0.11,Time: 112.86208s
Epoch 25/28, Training loss(ave. loss per minibatch): 0.11,Time: 117.27067s
Epoch 26/28, Training loss(ave. loss per minibatch): 0.11,Time: 121.84453s
Epoch 27/28, Training loss(ave. loss per minibatch): 0.11,Time: 126.74596s
Epoch 28/28, Training loss(ave. loss per minibatch): 0.1,Time: 131.48978s
```



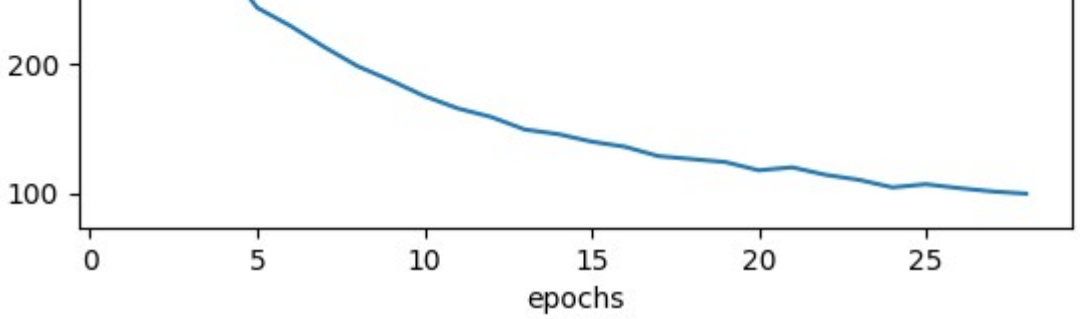


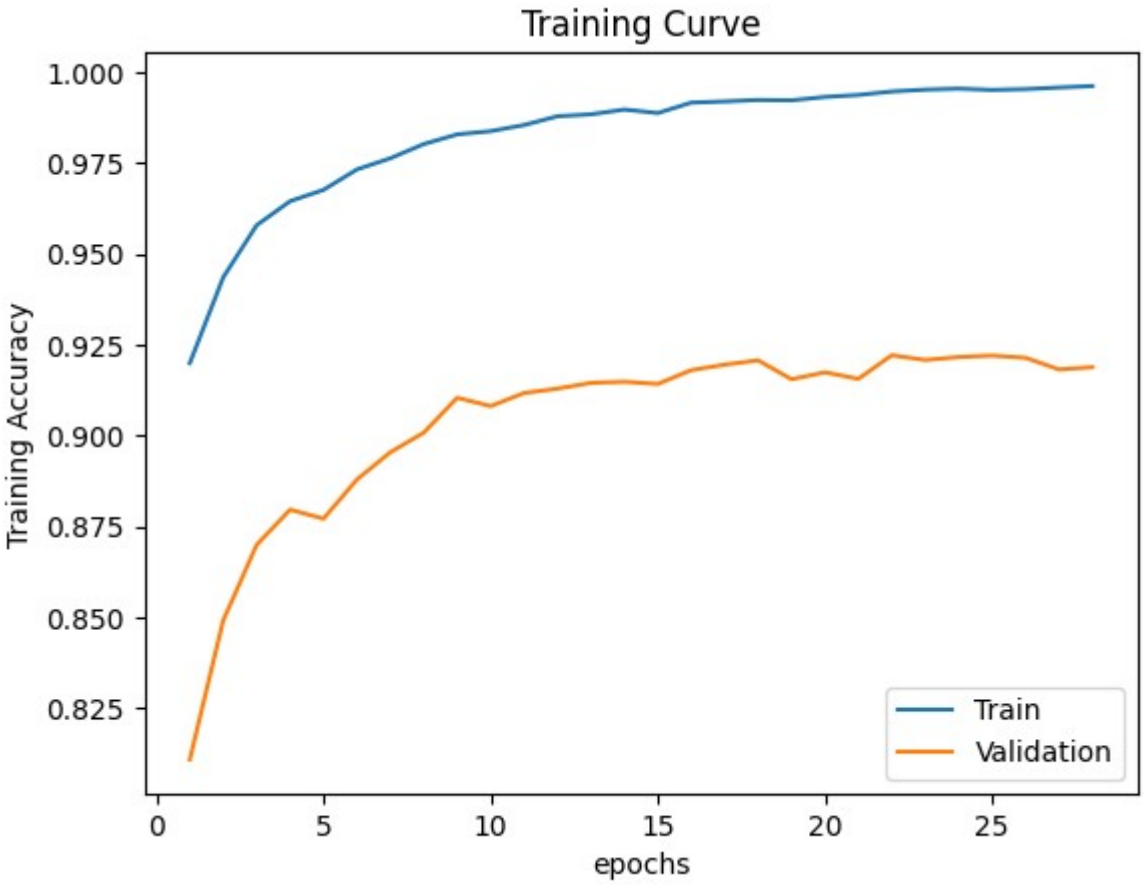



```
Final Training Accuracy: 0.9952666666666666
Final Validation Accuracy: 0.9181
```

```
train(model_five_layer, mnist_train, lr_choice=0.001, num_epochs=28, grad_clip=1.5)
```

```
Epoch 1/28, Training loss(ave. loss per minibatch): 0.66,Time: 13.06186s
Epoch 2/28, Training loss(ave. loss per minibatch): 0.4,Time: 26.62034s
Epoch 3/28, Training loss(ave. loss per minibatch): 0.34,Time: 40.83995s
Epoch 4/28, Training loss(ave. loss per minibatch): 0.29,Time: 54.3591s
Epoch 5/28, Training loss(ave. loss per minibatch): 0.26,Time: 67.3423s
Epoch 6/28, Training loss(ave. loss per minibatch): 0.24,Time: 80.34226s
Epoch 7/28, Training loss(ave. loss per minibatch): 0.23,Time: 93.47383s
Epoch 8/28, Training loss(ave. loss per minibatch): 0.21,Time: 106.85021s
Epoch 9/28, Training loss(ave. loss per minibatch): 0.2,Time: 119.61672s
Epoch 10/28, Training loss(ave. loss per minibatch): 0.19,Time: 132.45001s
Epoch 11/28, Training loss(ave. loss per minibatch): 0.18,Time: 145.70622s
Epoch 12/28, Training loss(ave. loss per minibatch): 0.17,Time: 159.70314s
Epoch 13/28, Training loss(ave. loss per minibatch): 0.16,Time: 174.46999s
Epoch 14/28, Training loss(ave. loss per minibatch): 0.16,Time: 187.89642s
Epoch 15/28, Training loss(ave. loss per minibatch): 0.15,Time: 200.65387s
Epoch 16/28, Training loss(ave. loss per minibatch): 0.14,Time: 213.58375s
Epoch 17/28, Training loss(ave. loss per minibatch): 0.14,Time: 226.61279s
Epoch 18/28, Training loss(ave. loss per minibatch): 0.13,Time: 239.55475s
Epoch 19/28, Training loss(ave. loss per minibatch): 0.13,Time: 252.61111s
Epoch 20/28, Training loss(ave. loss per minibatch): 0.13,Time: 266.18543s
Epoch 21/28, Training loss(ave. loss per minibatch): 0.13,Time: 278.87663s
Epoch 22/28, Training loss(ave. loss per minibatch): 0.12,Time: 291.70545s
Epoch 23/28, Training loss(ave. loss per minibatch): 0.12,Time: 304.49479s
Epoch 24/28, Training loss(ave. loss per minibatch): 0.11,Time: 317.43271s
Epoch 25/28, Training loss(ave. loss per minibatch): 0.11,Time: 330.12644s
Epoch 26/28, Training loss(ave. loss per minibatch): 0.11,Time: 343.12149s
Epoch 27/28, Training loss(ave. loss per minibatch): 0.11,Time: 357.52326s
Epoch 28/28, Training loss(ave. loss per minibatch): 0.11,Time: 370.82447s
```






```
Final Training Accuracy: 0.9962166666666666
Final Validation Accuracy: 0.9188
```

model_five_layer has the best validation accuracy but is not substantially better than the 3-layer version (model_both). This is probably because the extra layers are not reveal any new or complex patterns. The five layer model takes almost three times as long to complete the 28 epochs of learning and so while it gives a better result, it is not as efficient (in terms of time spent) as the 3-layer model_both.

Both `nn.Dropout()` and `nn.Batchnorm1d()` were used in an attempt to reduce overfitting but only 'Batchnorm' appeared to improve the validation accuracy. Also, only Batchnorm appeared to reduce overfitting whne both were used alone but using both techniques resulted in a better result than using either of them alone (using *Overfitting* ≈ *Validation Accuracy / Training Accuracy* to give a metric of a change in overfitting, where a number closer to 1 corresponds to a less overfit model):

Model	Overfitting	Technique
model	0.916	Neither
model_drop	0.912	Dropout
model_bn	0.922	Batchnorm
model_both	0.923	Both

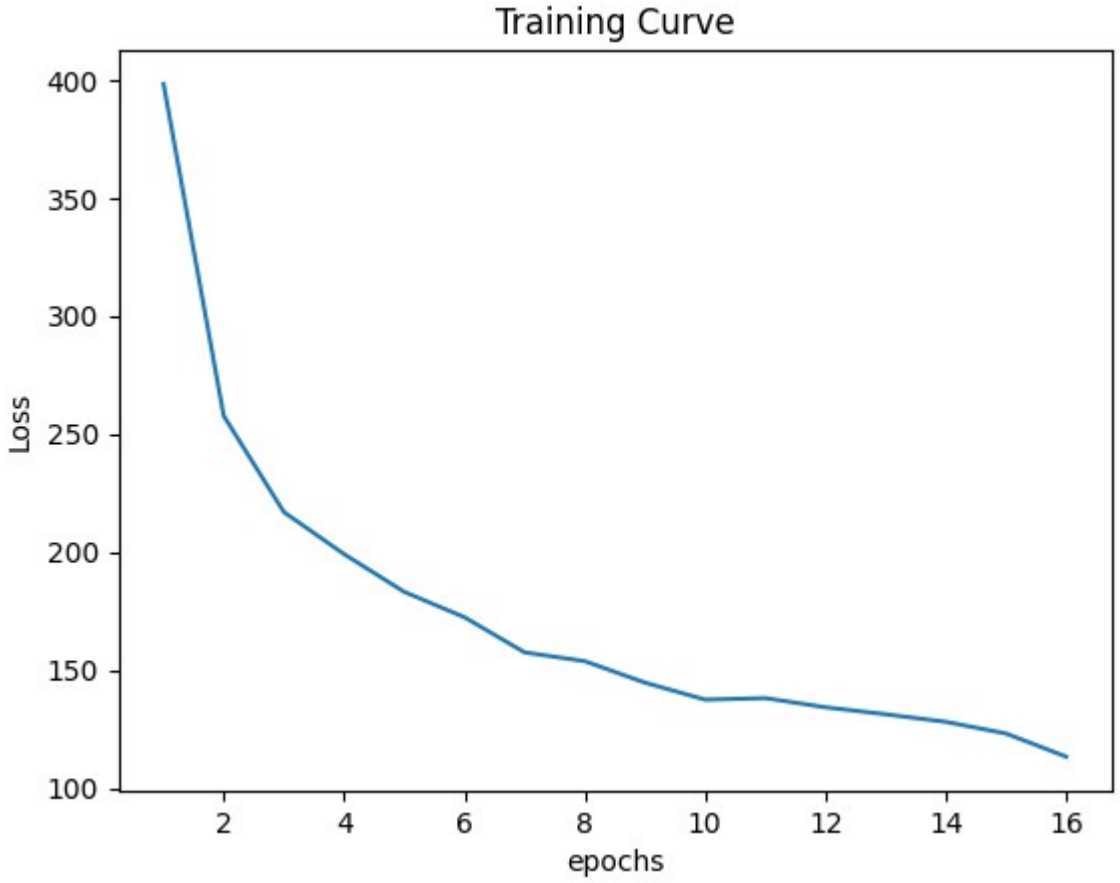
Investigate the relationship between learning rate (lr) and gradient

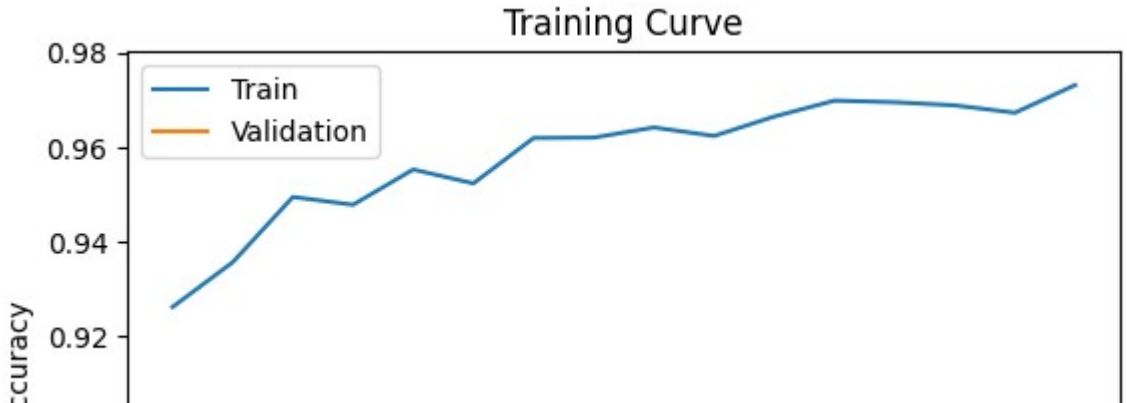
clipping value (grad_clip)

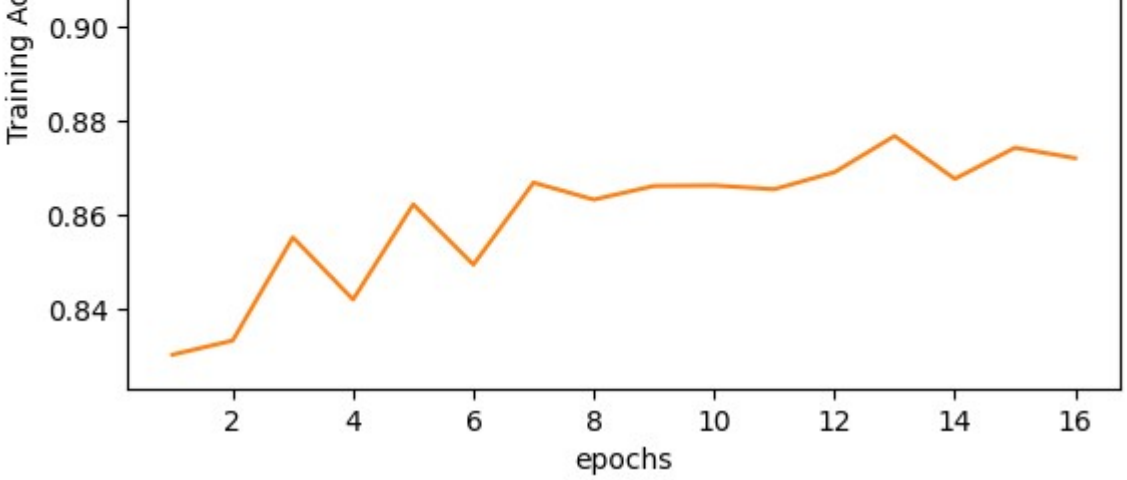
Base Model

```
reset_mod_params(model)
train(model, mnist_train, num_epochs=16)   # lr = 0.01 and grad_clip = 1.0
```

```
Epoch 1/16, Training loss(ave. loss per minibatch): 0.42,Time: 4.29285s
Epoch 2/16, Training loss(ave. loss per minibatch): 0.27,Time: 8.18713s
Epoch 3/16, Training loss(ave. loss per minibatch): 0.23,Time: 12.00361s
Epoch 4/16, Training loss(ave. loss per minibatch): 0.21,Time: 16.20558s
Epoch 5/16, Training loss(ave. loss per minibatch): 0.2,Time: 20.07421s
Epoch 6/16, Training loss(ave. loss per minibatch): 0.18,Time: 23.90921s
Epoch 7/16, Training loss(ave. loss per minibatch): 0.17,Time: 28.04192s
Epoch 8/16, Training loss(ave. loss per minibatch): 0.16,Time: 31.82568s
Epoch 9/16, Training loss(ave. loss per minibatch): 0.15,Time: 35.943s
Epoch 10/16, Training loss(ave. loss per minibatch): 0.15,Time: 40.24691s
Epoch 11/16, Training loss(ave. loss per minibatch): 0.15,Time: 44.56765s
Epoch 12/16, Training loss(ave. loss per minibatch): 0.14,Time: 49.73657s
Epoch 13/16, Training loss(ave. loss per minibatch): 0.14,Time: 53.58065s
Epoch 14/16, Training loss(ave. loss per minibatch): 0.14,Time: 57.46901s
Epoch 15/16, Training loss(ave. loss per minibatch): 0.13,Time: 61.65376s
Epoch 16/16, Training loss(ave. loss per minibatch): 0.12,Time: 65.49626s
```





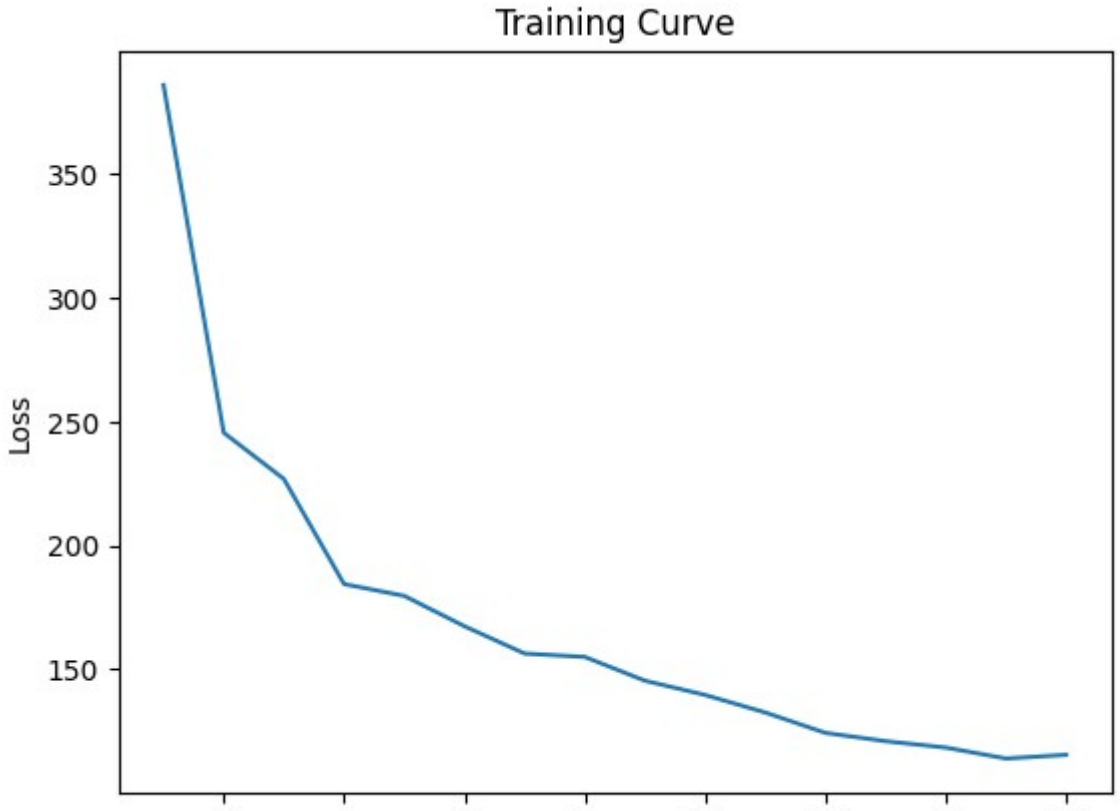


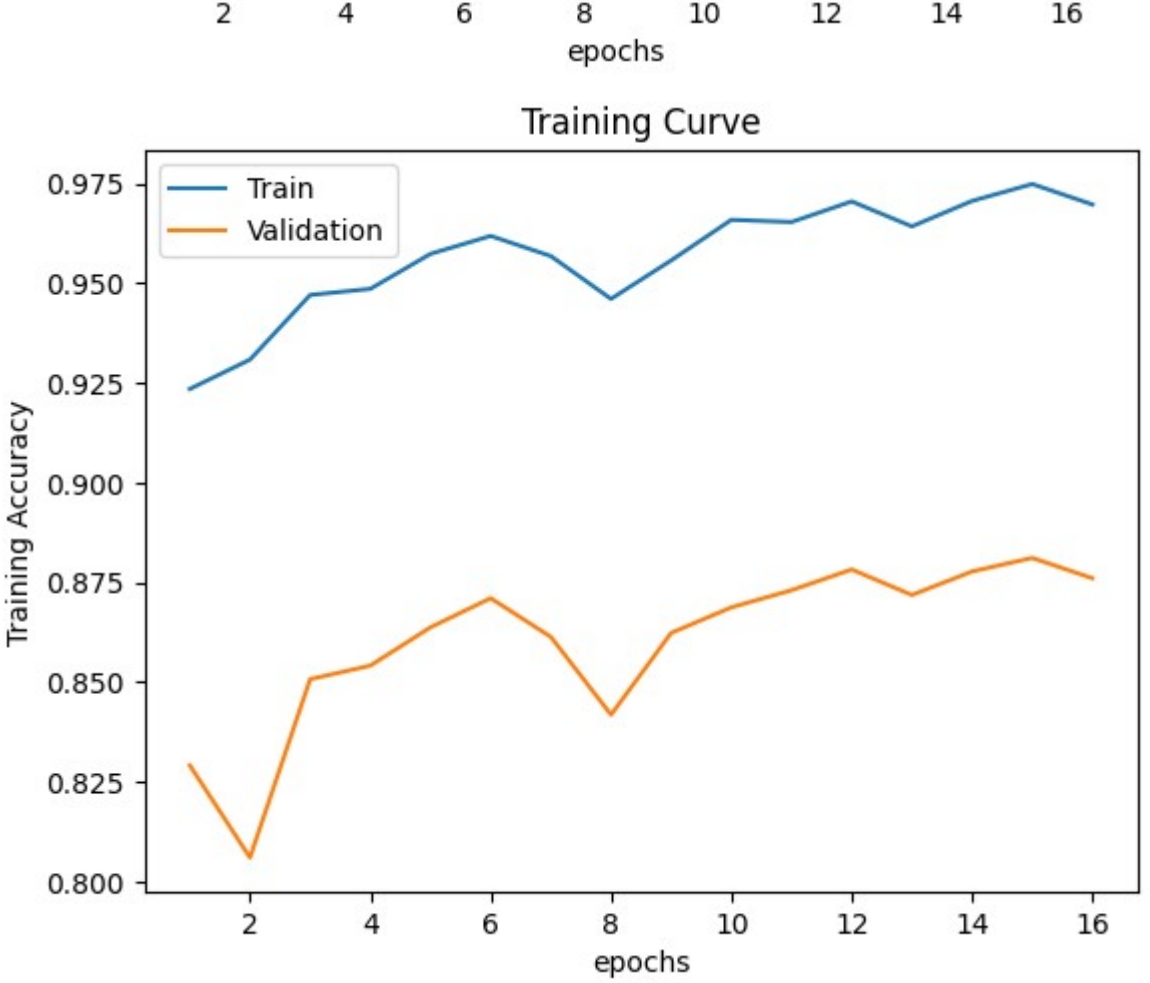

```
Final Training Accuracy: 0.9731166666666666
Final Validation Accuracy: 0.8721
```

Increasing grad_clip

```
reset_mod_params(model)
train(model, mnist_train, num_epochs=16, grad_clip=10)   # lr = 0.01 (kept constant)
```

```
Epoch 1/16, Training loss(ave. loss per minibatch): 0.41,Time: 4.32521s
Epoch 2/16, Training loss(ave. loss per minibatch): 0.26,Time: 9.01112s
Epoch 3/16, Training loss(ave. loss per minibatch): 0.24,Time: 13.30396s
Epoch 4/16, Training loss(ave. loss per minibatch): 0.2,Time: 17.55762s
Epoch 5/16, Training loss(ave. loss per minibatch): 0.19,Time: 22.40847s
Epoch 6/16, Training loss(ave. loss per minibatch): 0.18,Time: 26.9517s
Epoch 7/16, Training loss(ave. loss per minibatch): 0.17,Time: 31.42488s
Epoch 8/16, Training loss(ave. loss per minibatch): 0.17,Time: 36.10341s
Epoch 9/16, Training loss(ave. loss per minibatch): 0.16,Time: 40.70958s
Epoch 10/16, Training loss(ave. loss per minibatch): 0.15,Time: 45.6448s
Epoch 11/16, Training loss(ave. loss per minibatch): 0.14,Time: 50.36151s
Epoch 12/16, Training loss(ave. loss per minibatch): 0.13,Time: 55.87952s
Epoch 13/16, Training loss(ave. loss per minibatch): 0.13,Time: 60.23879s
Epoch 14/16, Training loss(ave. loss per minibatch): 0.13,Time: 64.50002s
Epoch 15/16, Training loss(ave. loss per minibatch): 0.12,Time: 69.17221s
Epoch 16/16, Training loss(ave. loss per minibatch): 0.12,Time: 73.37435s
```



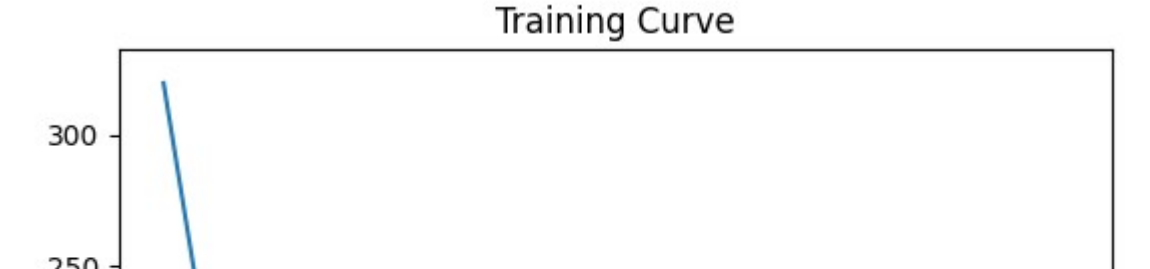


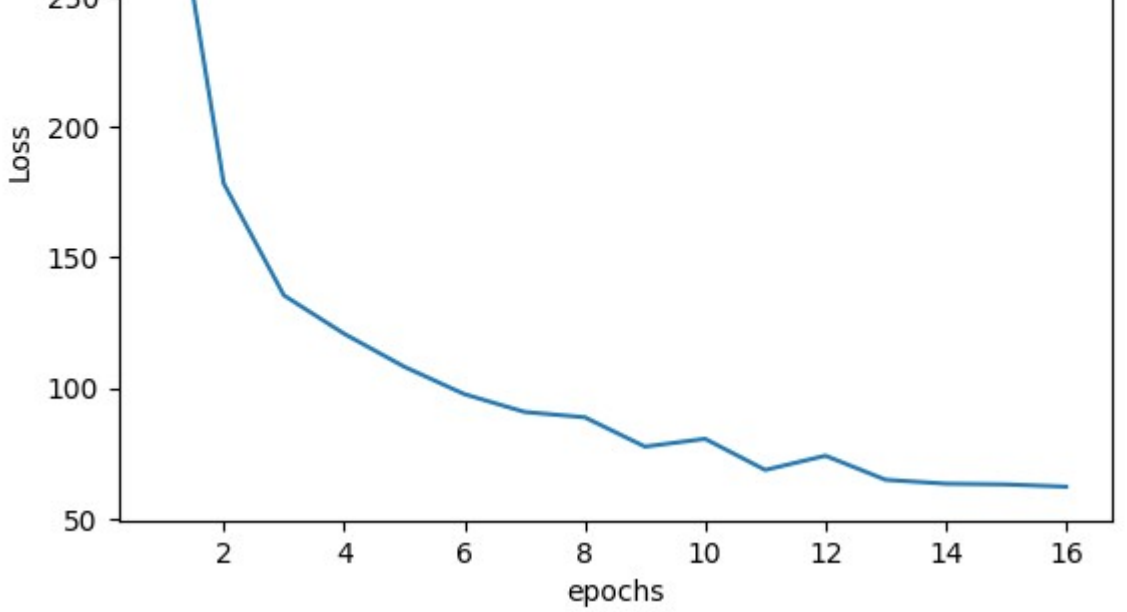

```
Final Training Accuracy: 0.9696833333333333
Final Validation Accuracy: 0.876
```

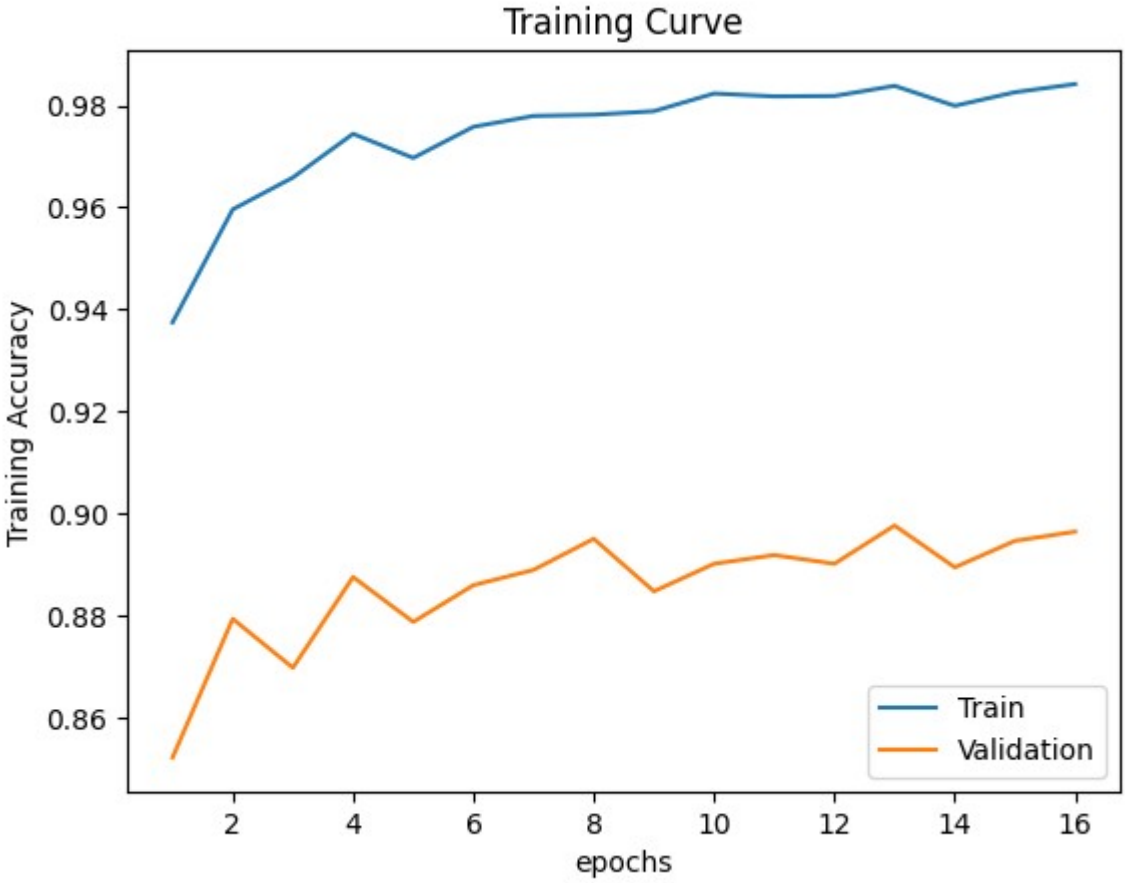
Lower validation accuracy, changing learning rate

```
# lowering learning rate
reset_mod_params(model)
train(model, mnist_train, num_epochs=16, lr_choice=0.005, grad_clip=10)
```

```
Epoch 1/16, Training loss(ave. loss per minibatch): 0.34,Time: 4.85887s
Epoch 2/16, Training loss(ave. loss per minibatch): 0.19,Time: 9.03366s
Epoch 3/16, Training loss(ave. loss per minibatch): 0.14,Time: 13.22676s
Epoch 4/16, Training loss(ave. loss per minibatch): 0.13,Time: 17.97989s
Epoch 5/16, Training loss(ave. loss per minibatch): 0.12,Time: 22.11942s
Epoch 6/16, Training loss(ave. loss per minibatch): 0.1,Time: 26.70376s
Epoch 7/16, Training loss(ave. loss per minibatch): 0.1,Time: 31.08924s
Epoch 8/16, Training loss(ave. loss per minibatch): 0.09,Time: 35.24628s
Epoch 9/16, Training loss(ave. loss per minibatch): 0.08,Time: 39.78309s
Epoch 10/16, Training loss(ave. loss per minibatch): 0.09,Time: 44.03097s
Epoch 11/16, Training loss(ave. loss per minibatch): 0.07,Time: 48.58682s
Epoch 12/16, Training loss(ave. loss per minibatch): 0.08,Time: 53.10451s
Epoch 13/16, Training loss(ave. loss per minibatch): 0.07,Time: 57.55506s
Epoch 14/16, Training loss(ave. loss per minibatch): 0.07,Time: 62.35181s
Epoch 15/16, Training loss(ave. loss per minibatch): 0.07,Time: 66.54953s
Epoch 16/16, Training loss(ave. loss per minibatch): 0.07,Time: 71.16695s
```





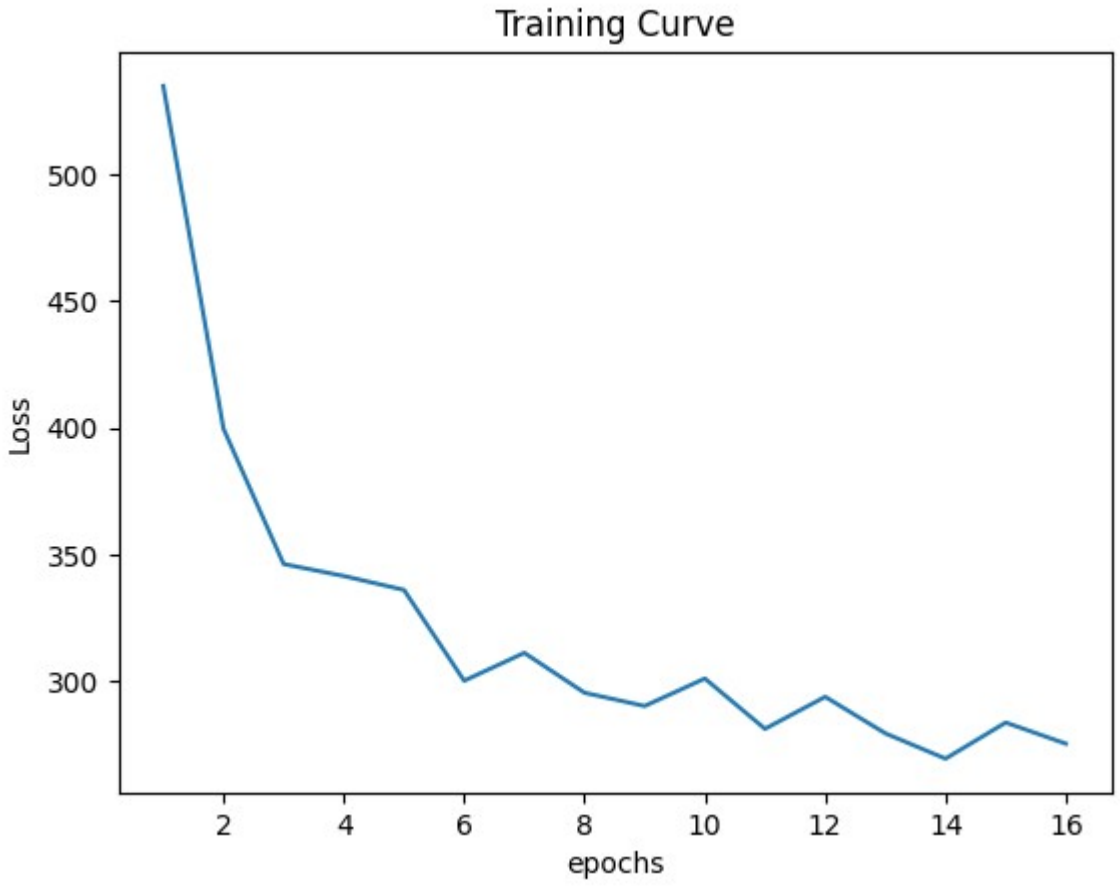


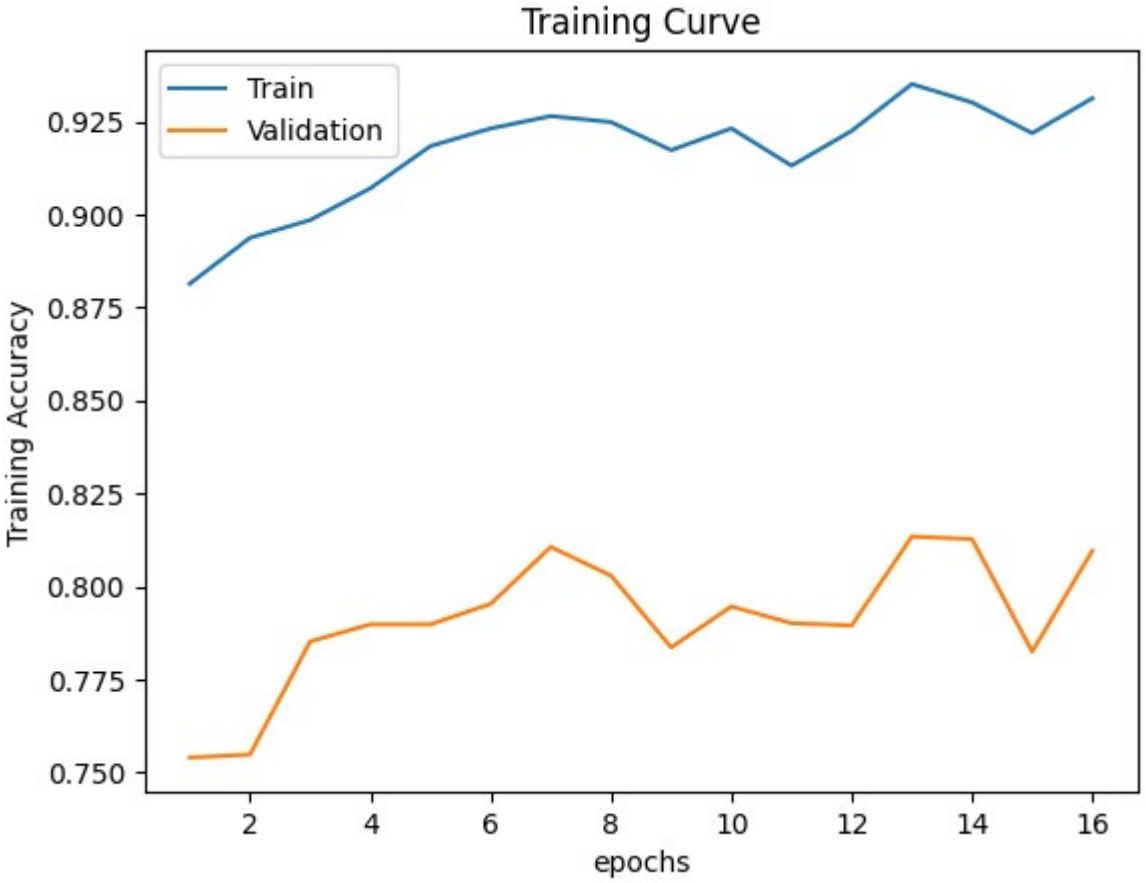

```
Final Training Accuracy: 0.9841833333333333
Final Validation Accuracy: 0.8964
```

```
# increasing learning rate
reset_mod_params(model)
train(model, mnist_train, num_epochs=16, lr_choice=0.02, grad_clip=10)
```

```
Epoch 1/16, Training loss(ave. loss per minibatch): 0.57,Time: 4.08536s
Epoch 2/16, Training loss(ave. loss per minibatch): 0.43,Time: 8.39936s
Epoch 3/16, Training loss(ave. loss per minibatch): 0.37,Time: 13.36249s
Epoch 4/16, Training loss(ave. loss per minibatch): 0.36,Time: 18.8402s
Epoch 5/16, Training loss(ave. loss per minibatch): 0.36,Time: 23.97146s
Epoch 6/16, Training loss(ave. loss per minibatch): 0.32,Time: 28.2706s
Epoch 7/16, Training loss(ave. loss per minibatch): 0.33,Time: 32.82232s
Epoch 8/16, Training loss(ave. loss per minibatch): 0.31,Time: 37.91542s
Epoch 9/16, Training loss(ave. loss per minibatch): 0.31,Time: 42.43811s
Epoch 10/16, Training loss(ave. loss per minibatch): 0.32,Time: 47.10482s
Epoch 11/16, Training loss(ave. loss per minibatch): 0.3,Time: 51.31655s
Epoch 12/16, Training loss(ave. loss per minibatch): 0.31,Time: 55.75425s
```

```
Epoch 13/16, Training loss(ave. loss per minibatch): 0.3,Time: 60.30502s
Epoch 14/16, Training loss(ave. loss per minibatch): 0.29,Time: 64.55233s
Epoch 15/16, Training loss(ave. loss per minibatch): 0.3,Time: 69.06788s
Epoch 16/16, Training loss(ave. loss per minibatch): 0.29,Time: 73.41531s
```






```
Final Training Accuracy: 0.9312
Final Validation Accuracy: 0.8096
```

Both learning rate `lr_choice` and gradient clip value `grad_clip` affect how much a model's weights are updated in training. Gradient clipping helps avoid the gradient changing the weights too drastically and the learning rate decides the size of the change in the direction of the gradient.

Here, it appears that an increase in `grad_clip` can result in an increase in voaltility in training, sometimes eventually resulting in large jumps and a decrease in accuracy.

Inrceasing the learning rate will accentuate the volatility and potentially worse accuracy caused by an increase in `grad_clip`, whereas decreasing learning rate can help reduce the volatility and imporve accuracy.

1.B

Single Hidden Layer FFN vs Multi Layer FFN

Creating Single Hidden Layer FFN

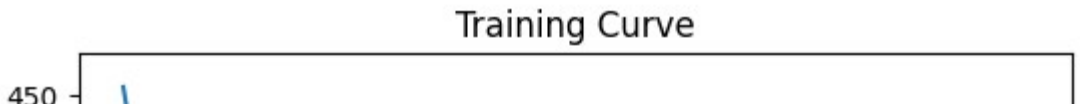
```
model_shl = nn.Sequential(nn.Flatten(),
                          nn.Linear(28*28, 256),
                          nn.BatchNorm1d(256),
                          nn.ReLU(),
                          nn.Dropout(),
                          nn.Linear(256, 10))
model_shl.to(device)
```

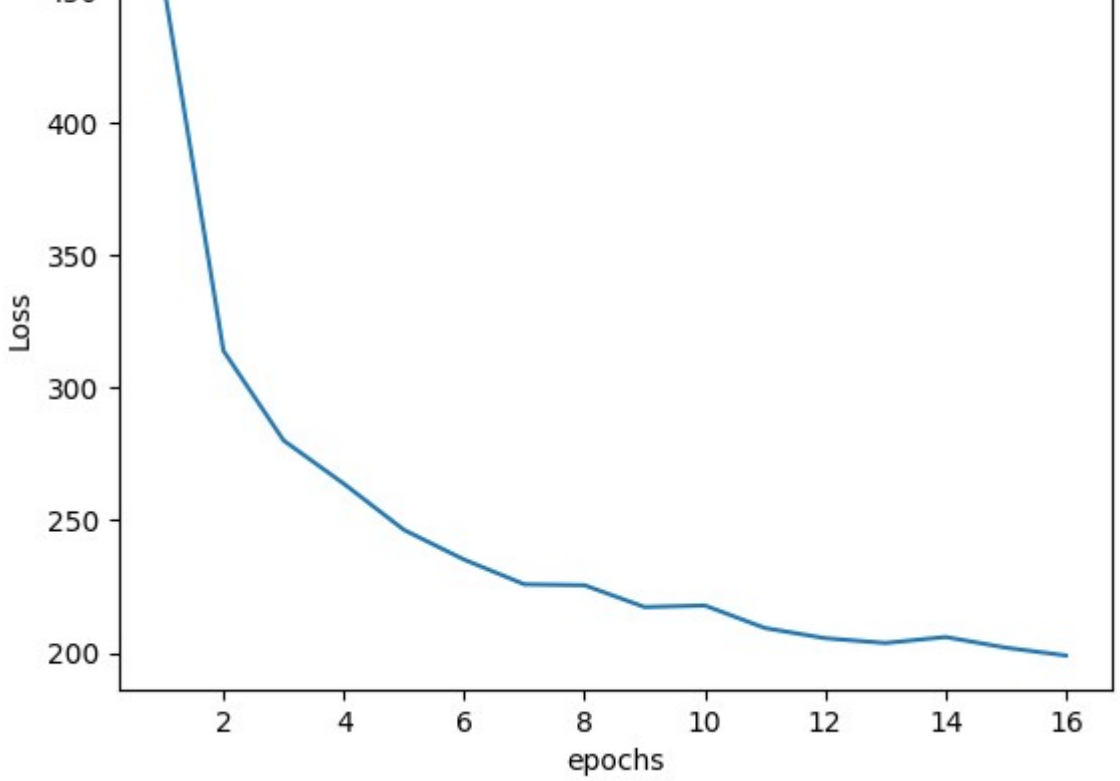
```
Sequential(
  (0): Flatten(start_dim=1, end_dim=-1)
  (1): Linear(in_features=784, out_features=256, bias=True)
  (2): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (3): ReLU()
  (4): Dropout(p=0.5, inplace=False)
  (5): Linear(in_features=256, out_features=10, bias=True)
)
```

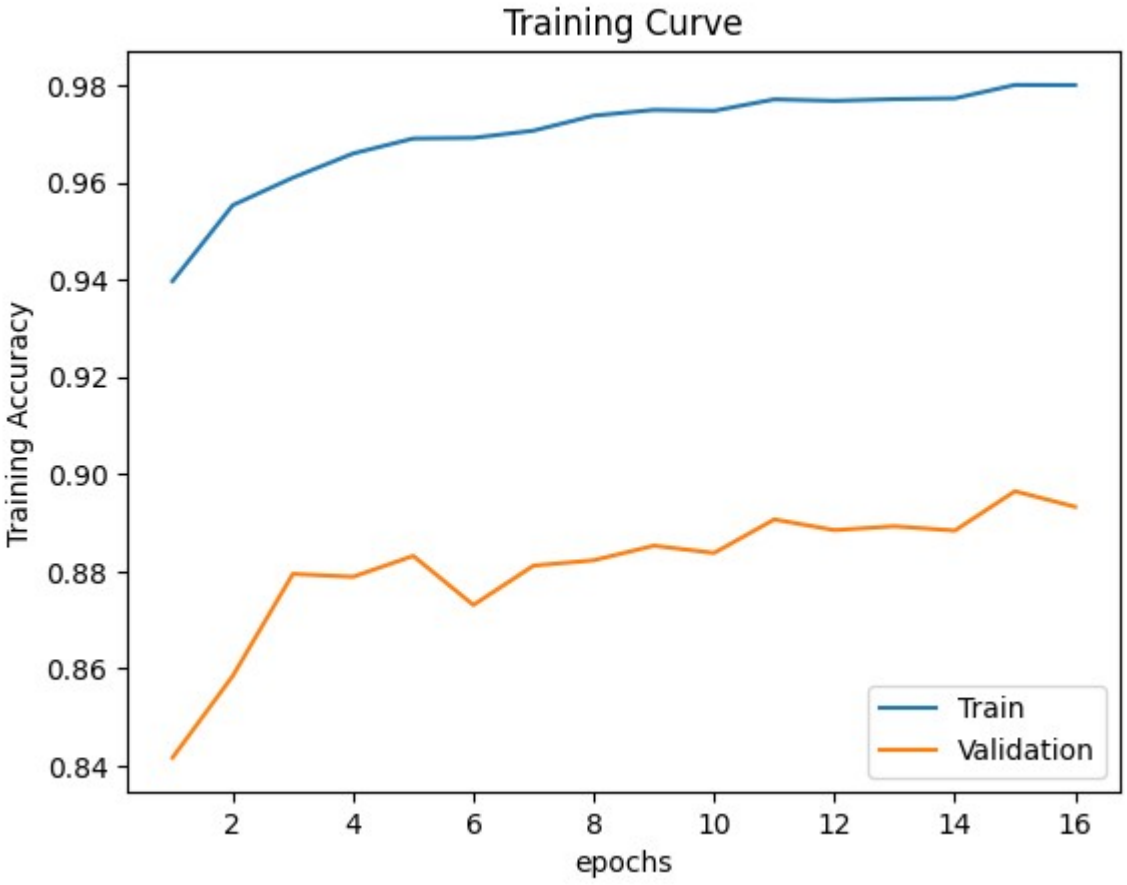
Training Single Hidden Layer FFN

```
train(model_shl, mnist_train, num_epochs=16)
```

```
Epoch 1/16, Training loss(ave. loss per minibatch): 0.48,Time: 3.72727s
Epoch 2/16, Training loss(ave. loss per minibatch): 0.33,Time: 7.29354s
Epoch 3/16, Training loss(ave. loss per minibatch): 0.3,Time: 11.17706s
Epoch 4/16, Training loss(ave. loss per minibatch): 0.28,Time: 14.65176s
Epoch 5/16, Training loss(ave. loss per minibatch): 0.26,Time: 18.2514s
Epoch 6/16, Training loss(ave. loss per minibatch): 0.25,Time: 22.16593s
Epoch 7/16, Training loss(ave. loss per minibatch): 0.24,Time: 25.58981s
Epoch 8/16, Training loss(ave. loss per minibatch): 0.24,Time: 29.12278s
Epoch 9/16, Training loss(ave. loss per minibatch): 0.23,Time: 32.91525s
Epoch 10/16, Training loss(ave. loss per minibatch): 0.23,Time: 37.30295s
Epoch 11/16, Training loss(ave. loss per minibatch): 0.22,Time: 41.30414s
Epoch 12/16, Training loss(ave. loss per minibatch): 0.22,Time: 45.20895s
Epoch 13/16, Training loss(ave. loss per minibatch): 0.22,Time: 48.65552s
Epoch 14/16, Training loss(ave. loss per minibatch): 0.22,Time: 52.02814s
Epoch 15/16, Training loss(ave. loss per minibatch): 0.22,Time: 55.86539s
Epoch 16/16, Training loss(ave. loss per minibatch): 0.21,Time: 59.27058s
```








```
Final Training Accuracy: 0.9800333333333333
Final Validation Accuracy: 0.8933
```

Comparing to deep FFN

Single Hiddden Layer FFN (model_shl):

> Final Training Accuracy: 0.980

> Final Validation Accuracy: 0.893

Deep FFN (model_five_layer):

Deep FFN (model_five_layer):

> Final Training Accuracy: 0.996

> Final Validation Accuracy: 0.919

The deep FFN is a clearly better model with a validation accuracy that is 0.26 better. Both models have the same number of parameters apart from in a deep FFN you can also vary the number of layers in the model.

A single layer model can perform as well as a multi layer model if what they FNNs are trying to model is quite a simple function/ relationship but for more complex data like image classification, adding layers can reveal more information about the data which a single model would struggle more with. However, there is no guarantee that adding more layers will always improve accuracy and in some cases, it will actually slow the computational efficiency without offering any substantial improvements in accuracy.