PA3_na21b033

October 16, 2023

##1.1. Installing necesarry packages

[1]:
```
!pip install -q torch_snippets
!pip install -q torchinfo
!pip install -q torchmetrics

from torchmetrics import ConfusionMatrix
from torch_snippets import*
from torchinfo import summary
import torch
from torchvision import datasets, transforms
from torch import nn
import matplotlib.pyplot as plt
import sklearn
from sklearn.metrics import classification_report
import pandas as pd
```

```
                                        65.4/65.4 kB
2.3 MB/s eta 0:00:00
                                        115.3/115.3

kB 7.4 MB/s eta 0:00:00
                                        62.5/62.5 kB
7.5 MB/s eta 0:00:00
                                        78.6/78.6 kB
7.8 MB/s eta 0:00:00
  Preparing metadata (setup.py) … done
                                        203.7/203.7

kB 9.4 MB/s eta 0:00:00
                                        4.3/4.3 MB
18.7 MB/s eta 0:00:00
                                        1.6/1.6 MB
31.8 MB/s eta 0:00:00
                                        98.9/98.9 kB
14.0 MB/s eta 0:00:00
                                        3.7/3.7 MB
34.1 MB/s eta 0:00:00
                                        30.6/30.6 MB
```

```
38.8 MB/s eta 0:00:00
                           169.4/169.4

kB 18.8 MB/s eta 0:00:00
                           3.2/3.2 MB
46.5 MB/s eta 0:00:00
                           468.9/468.9

kB 33.8 MB/s eta 0:00:00
  Building wheel for typing (setup.py) … done
                           805.2/805.2

kB 9.8 MB/s eta 0:00:00
```

[2]:
```
device="cuda" if torch.cuda.is_available() else "cpu"
device #device agniostic code
```

[2]:
```
'cuda'
```

##1.2. Getting the data

[3]:
```
transformer= transforms.Compose([transforms.ToTensor()]) #dataset contains imgs␣
 ↪in PIL format
```

[4]:
```
train_ds=datasets.MNIST(root="MNIST/", train=True, download=True, transform=␣
 ↪transformer)
#convert labels to one hot encoding
test_ds=datasets.MNIST(root="MNIST/", train=False, download=True, transform=␣
 ↪transformer)

train_dl= torch.utils.data.DataLoader(train_ds, batch_size=32, drop_last=True)
test_dl= torch.utils.data.DataLoader(test_ds, batch_size=32, drop_last=True)
```

```
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz to
MNIST/MNIST/raw/train-images-idx3-ubyte.gz

100%|      | 9912422/9912422 [00:00<00:00, 142083807.49it/s]

Extracting MNIST/MNIST/raw/train-images-idx3-ubyte.gz to MNIST/MNIST/raw



Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz to
MNIST/MNIST/raw/train-labels-idx1-ubyte.gz

100%|      | 28881/28881 [00:00<00:00, 115808502.70it/s]
```

```
Extracting MNIST/MNIST/raw/train-labels-idx1-ubyte.gz to MNIST/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz


Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz to
MNIST/MNIST/raw/t10k-images-idx3-ubyte.gz

100%|     | 1648877/1648877 [00:00<00:00, 39557806.99it/s]

Extracting MNIST/MNIST/raw/t10k-images-idx3-ubyte.gz to MNIST/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz to
MNIST/MNIST/raw/t10k-labels-idx1-ubyte.gz


100%|     | 4542/4542 [00:00<00:00, 20843029.29it/s]

Extracting MNIST/MNIST/raw/t10k-labels-idx1-ubyte.gz to MNIST/MNIST/raw
```

[5]:
```
im, label=next(iter(train_dl))
im.shape, label.shape #labels are not one hot encoded; img_shape-> [28*28]
```

[5]:
```
(torch.Size([32, 1, 28, 28]), torch.Size([32]))
```

##1.3. **Building naive_rnn**

[6]:
```
input_size=28
sequence_length= 28
hidden_size= 128
num_layers=2
num_classes= 10
```

[7]:
```
class naive_RNN(nn.Module):
  def __init__(self, input_size, hidden_size, num_layers, num_classes):
    super().__init__()
    self.input_size= input_size
    self.hidden_size= hidden_size
    self.num_layers= num_layers
    self.num_classes= num_classes

    self.rnn_block= nn.RNN(input_size, hidden_size, num_layers,␣
 ↪batch_first=True)

    self.classification_block= nn.Sequential(nn.Linear(hidden_size,␣
 ↪num_classes),
                                             nn.Softmax(dim=-1))
  def forward(self, x):
    h0, c0= torch.zeros(self.num_layers, x.size(0), self.hidden_size).
↪to(device), torch.zeros(self.num_layers, x.size(0), self.hidden_size).
↪to(device)
    x, h_out= self.rnn_block(x, h0)
    x= self.classification_block(x[:, -1,:])
    return x

  def loss_and_accuracy(self, y_true, y_p):
    loss_fn= nn.CrossEntropyLoss()
    loss= loss_fn(y_p, y_true)
    y_p= torch.argmax(y_p, dim=-1)
    acc= torch.sum(torch.eq(y_true, y_p))/len(y_true)
    return loss, acc
```

[8]:
```
naive_rnn= naive_RNN(input_size, hidden_size, num_layers, num_classes).
↪to(device)
summary(naive_rnn, input_size= (32, 28, 28), col_names= ["input_size",␣
↪"output_size", "num_params"])
```

[8]:
```
================================================================================
===================================
Layer (type:depth-idx)                   Input Shape               Output Shape
Param #
================================================================================
===================================
naive_RNN                                 [32, 28, 28]              [32, 10]
--
 RNN: 1-1                                [32, 28, 28]              [32, 28, 128]
53,248
 Sequential: 1-2                         [32, 128]                 [32, 10]
--
     Linear: 2-1                         [32, 128]                 [32, 10]
1,290
     Softmax: 2-2                        [32, 10]                  [32, 10]
--
================================================================================
===================================
Total params: 54,538
Trainable params: 54,538
Non-trainable params: 0
Total mult-adds (M): 47.75
================================================================================
===================================
Input size (MB): 0.10
Forward/backward pass size (MB): 0.92
```

```
Params size (MB): 0.22
Estimated Total Size (MB): 1.24
================================================================================
===================================
```

###**1.3.1. Training the model**

[9]:
```
def train_epoch(model, input, criterion, optimizer, sequence_length=28):
  x, y= input
  x, y= x.to(device).view(-1, sequence_length, input_size), y.to(device)
  model.train()
  output=model(x)
  loss, acc=criterion(y, output)
  optimizer.zero_grad()
  loss.backward()
  optimizer.step()
  return loss.item(), acc.item()

def val_epoch(model, input, criterion, sequence_length=28):
  x, y= input
  x, y= x.to(device).view(-1, sequence_length, input_size), y.to(device)
  model.eval()
  with torch.inference_mode():
    output=model(x)
    loss, acc=criterion(y, output)
    return loss.item(), acc.item()
```

[10]:
```
EPOCHS=5 #the val acc seems to saturate after 5th epoch
criterion= naive_rnn.loss_and_accuracy
optimizer=torch.optim.Adam(params= naive_rnn.parameters())
```

[11]:
```
log= Report(EPOCHS)
for epoch in range(EPOCHS):
  n=len(train_dl)
  for ix, input in enumerate(train_dl):
    train_loss, train_accuracy=train_epoch(naive_rnn, input, criterion,␣
 ↪optimizer)
    log.record(epoch+(ix+1)/n, train_loss=train_loss, train_acc=␣
 ↪train_accuracy, end="\r")
  n=len(test_dl)
  for ix, input in enumerate(test_dl):
    val_loss, val_accuracy=val_epoch(naive_rnn, input, criterion)
    log.record(epoch+(ix+1)/n, val_loss=val_loss, val_acc= val_accuracy,␣
 ↪end="\r")
  log.report_avgs(epoch+1)
```

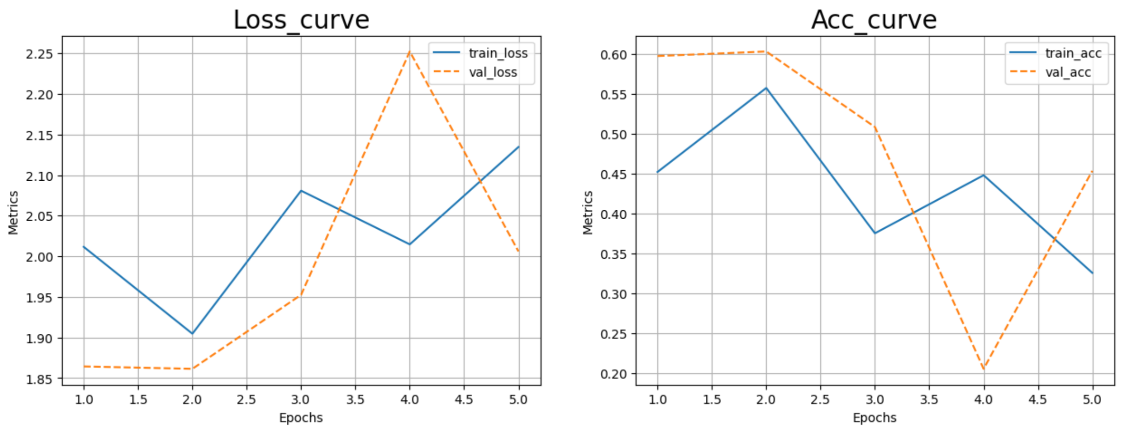
```
EPOCH: 1.000  train_loss: 2.012  val_loss: 1.864  train_acc: 0.452  val_acc:
```

```
0.598  (20.20s - 80.81s remaining)
EPOCH: 2.000  train_loss: 1.905  val_loss: 1.861  train_acc: 0.558  val_acc:
0.603  (34.29s - 51.43s remaining)
EPOCH: 3.000  train_loss: 2.081  val_loss: 1.953  train_acc: 0.375  val_acc:
0.509  (48.35s - 32.24s remaining)
EPOCH: 4.000  train_loss: 2.015  val_loss: 2.252  train_acc: 0.448  val_acc:
0.205  (62.59s - 15.65s remaining)
EPOCH: 5.000  train_loss: 2.135  val_loss: 2.006  train_acc: 0.326  val_acc:
0.454  (76.71s - 0.00s remaining)
```

[12]:
```
naive_rnn_results= {"train_acc": train_accuracy, "val_acc": val_accuracy,␣
 ↪"train_loss": train_loss, "val_loss": val_loss, "epochs": 5}
```

[13]:
```
fig, ax= plt.subplots(ncols=2, figsize=(15,5)) #w,h
log.plot_epochs(["train_loss", "val_loss"], ax=ax[0], title="Loss_curve");␣
 ↪#loss_curve
ax[0].legend(loc='upper right', fontsize=10)
log.plot_epochs(["train_acc", "val_acc"], ax=ax[1], title="Acc_curve");␣
 ↪#acc_curve
```

```
100%|      | 106/106 [00:00<00:00, 429.38it/s]
WARNING:matplotlib.legend:No artists with labels found to put in legend.  Note
that artists whose label start with an underscore are ignored when legend() is
called with no argument.
100%|      | 106/106 [00:00<00:00, 428.96it/s]
```



##1.4. Building LSTM

[14]:
```
class LSTM(nn.Module):
  def __init__(self, input_size, hidden_size, num_layers, num_classes):
    super().__init__()
    self.input_size= input_size
    self.hidden_size= hidden_size
    self.num_layers= num_layers
    self.num_classes= num_classes

    self.lstm_block= nn.LSTM(input_size, hidden_size, num_layers,␣
 ↪batch_first=True)

    self.classification_block= nn.Sequential(nn.Linear(hidden_size,␣
 ↪num_classes),
                                            nn.Softmax(dim=-1))

  def forward(self, x):
    h0, c0= torch.zeros(self.num_layers, x.size(0), self.hidden_size).
 ↪to(device), torch.zeros(self.num_layers, x.size(0), self.hidden_size).
 ↪to(device)
    x, h_out= self.lstm_block(x, (h0, c0))
    x= self.classification_block(x[:, -1,:])
    return x

  def loss_and_accuracy(self, y_true, y_p):
    loss_fn= nn.CrossEntropyLoss()
    loss= loss_fn(y_p, y_true)
    y_p= torch.argmax(y_p, dim=-1)
    acc= torch.sum(torch.eq(y_true, y_p))/len(y_true)
    return loss, acc
```

[15]:
```
lstm= LSTM(input_size, hidden_size, num_layers, num_classes).to(device)
summary(lstm, input_size= (32, 28, 28), col_names= ["input_size",␣
 ↪"output_size", "num_params"])
```

[15]:
```
================================================================================
===================================
Layer (type:depth-idx)                   Input Shape               Output Shape
Param #
================================================================================
===================================
LSTM                                     [32, 28, 28]              [32, 10]
--
 LSTM: 1-1                              [32, 28, 28]              [32, 28, 128]
212,992
 Sequential: 1-2                        [32, 128]                 [32, 10]
--
     Linear: 2-1                       [32, 128]                 [32, 10]
1,290
     Softmax: 2-2                      [32, 10]                  [32, 10]
--
================================================================================
===================================
Total params: 214,282
Trainable params: 214,282
Non-trainable params: 0
Total mult-adds (M): 190.88
================================================================================
===================================
Input size (MB): 0.10
Forward/backward pass size (MB): 0.92
Params size (MB): 0.86
Estimated Total Size (MB): 1.88
================================================================================
===================================
```

###1.4.1. **Training the model**

[16]:
```
EPOCHS=5 #the val acc seems to saturate after 5th epoch
criterion= lstm.loss_and_accuracy
optimizer=torch.optim.Adam(params= lstm.parameters())
```

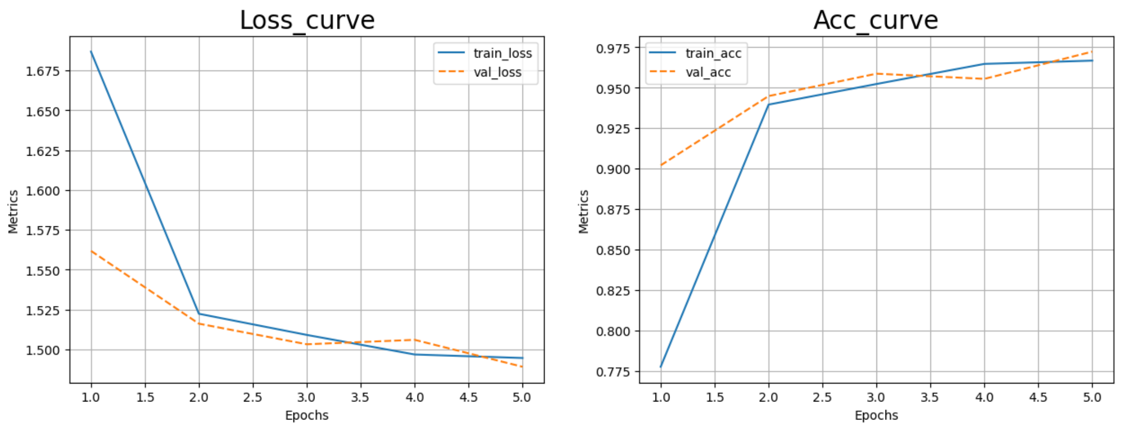
[17]:
```
log= Report(EPOCHS)
for epoch in range(EPOCHS):
  n=len(train_dl)
  for ix, input in enumerate(train_dl):
    train_loss, train_accuracy=train_epoch(lstm, input, criterion, optimizer)
    log.record(epoch+(ix+1)/n, train_loss=train_loss, train_acc=␣
 ↪train_accuracy, end="\r")
  n=len(test_dl)
  for ix, input in enumerate(test_dl):
    val_loss, val_accuracy=val_epoch(lstm, input, criterion)
    log.record(epoch+(ix+1)/n, val_loss=val_loss, val_acc= val_accuracy,␣
 ↪end="\r")
  log.report_avgs(epoch+1)
```

```
EPOCH: 1.000  train_loss: 1.687  val_loss: 1.562  train_acc: 0.778  val_acc:
0.902  (17.80s - 71.20s remaining)
EPOCH: 2.000  train_loss: 1.522  val_loss: 1.516  train_acc: 0.940  val_acc:
0.945  (33.62s - 50.43s remaining)
EPOCH: 3.000  train_loss: 1.509  val_loss: 1.503  train_acc: 0.952  val_acc:
0.959  (49.39s - 32.93s remaining)
EPOCH: 4.000  train_loss: 1.497  val_loss: 1.506  train_acc: 0.965  val_acc:
0.956  (65.18s - 16.29s remaining)
EPOCH: 5.000  train_loss: 1.495  val_loss: 1.489  train_acc: 0.967  val_acc:
0.972  (81.18s - 0.00s remaining)
```

[18]:
```
lstm_results= {"train_acc": train_accuracy, "val_acc": val_accuracy,␣
 ↪"train_loss": train_loss, "val_loss": val_loss, "epochs": 5}
```

```
[19]: fig, ax= plt.subplots(ncols=2, figsize=(15,5)) #w,h
      log.plot_epochs(["train_loss", "val_loss"], ax=ax[0], title="Loss_curve");␣
       ↪#loss_curve
      ax[0].legend(loc='upper right', fontsize=10)
      log.plot_epochs(["train_acc", "val_acc"], ax=ax[1], title="Acc_curve");␣
       ↪#acc_curve
```

```
100%|      | 106/106 [00:00<00:00, 205.34it/s]
WARNING:matplotlib.legend:No artists with labels found to put in legend.  Note
that artists whose label start with an underscore are ignored when legend() is
called with no argument.
100%|      | 106/106 [00:00<00:00, 220.85it/s]
```



##1.5. Building GRU

```
[20]: class GRU(nn.Module):
        def __init__(self, input_size, hidden_size, num_layers, num_classes):
          super().__init__()
          self.input_size= input_size
          self.hidden_size= hidden_size
          self.num_layers= num_layers
          self.num_classes= num_classes

          self.rnn_block= nn.GRU(input_size, hidden_size, num_layers,␣
       ↪batch_first=True)

          self.classification_block= nn.Sequential(nn.Linear(hidden_size,␣
       ↪num_classes),
                                                   nn.Softmax(dim=-1))

        def forward(self, x):
```

```
        h0, c0= torch.zeros(self.num_layers, x.size(0), self.hidden_size).
 ↪to(device), torch.zeros(self.num_layers, x.size(0), self.hidden_size).
 ↪to(device)
        x, h_out= self.rnn_block(x,h0)
        x= self.classification_block(x[:, -1,:])
        return x

    def loss_and_accuracy(self, y_true, y_p):
        loss_fn= nn.CrossEntropyLoss()
        loss= loss_fn(y_p, y_true)
        y_p= torch.argmax(y_p, dim=-1)
        acc= torch.sum(torch.eq(y_true, y_p))/len(y_true)
        return loss, acc
```

[21]:
```
gru= GRU(input_size, hidden_size, num_layers, num_classes).to(device)
summary(gru, input_size= (32, 28, 28), col_names= ["input_size", "output_size",␣
 ↪"num_params"])
```

[21]:
```
================================================================================
===================================
Layer (type:depth-idx)                   Input Shape               Output Shape
Param #
================================================================================
===================================
GRU                                      [32, 28, 28]              [32, 10]
--
 GRU: 1-1                               [32, 28, 28]              [32, 28, 128]
159,744
 Sequential: 1-2                        [32, 128]                 [32, 10]
--
     Linear: 2-1                       [32, 128]                 [32, 10]
1,290
     Softmax: 2-2                      [32, 10]                  [32, 10]
--
================================================================================
===================================
Total params: 161,034
Trainable params: 161,034
Non-trainable params: 0
Total mult-adds (M): 143.17
================================================================================
===================================
Input size (MB): 0.10
Forward/backward pass size (MB): 0.92
Params size (MB): 0.64
Estimated Total Size (MB): 1.66
================================================================================
===================================
```

###1.5.1. **Training the model**

```
[22]: EPOCHS=5 #the val acc seems to saturate after 5th epoch
      criterion= gru.loss_and_accuracy
      optimizer=torch.optim.Adam(params= gru.parameters())
```

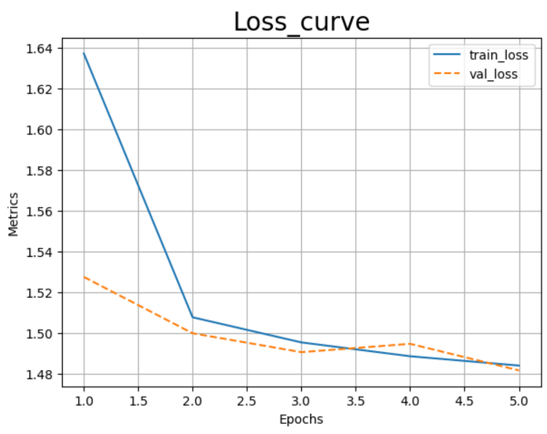
```
[23]: log= Report(EPOCHS)
      for epoch in range(EPOCHS):
        n=len(train_dl)
        for ix, input in enumerate(train_dl):
          loss, accuracy=train_epoch(gru, input, criterion, optimizer)
          log.record(epoch+(ix+1)/n, train_loss=loss, train_acc= accuracy, end="\r")
        n=len(test_dl)
        for ix, input in enumerate(test_dl):
          loss, accuracy=val_epoch(gru, input, criterion)
          log.record(epoch+(ix+1)/n, val_loss=loss, val_acc= accuracy, end="\r")
        log.report_avgs(epoch+1)
```

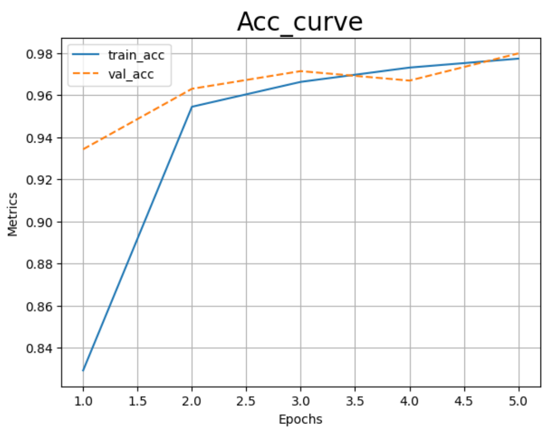
```
EPOCH: 1.000  train_loss: 1.637  val_loss: 1.528  train_acc: 0.829  val_acc:
0.934  (14.63s - 58.52s remaining)
EPOCH: 2.000  train_loss: 1.508  val_loss: 1.500  train_acc: 0.955  val_acc:
0.963  (29.07s - 43.60s remaining)
EPOCH: 3.000  train_loss: 1.495  val_loss: 1.491  train_acc: 0.966  val_acc:
0.971  (43.62s - 29.08s remaining)
EPOCH: 4.000  train_loss: 1.489  val_loss: 1.495  train_acc: 0.973  val_acc:
0.967  (58.24s - 14.56s remaining)
EPOCH: 5.000  train_loss: 1.484  val_loss: 1.482  train_acc: 0.977  val_acc:
0.980  (72.74s - 0.00s remaining)
```

```
[24]: gru_results= {"train_acc": train_accuracy, "val_acc": val_accuracy,␣
        ↪"train_loss": train_loss, "val_loss": val_loss, "epochs": 5}
```

```
[25]: fig, ax= plt.subplots(ncols=2, figsize=(15,5)) #w,h
      log.plot_epochs(["train_loss", "val_loss"], ax=ax[0], title="Loss_curve");␣
        ↪#loss_curve
      ax[0].legend(loc='upper right', fontsize=10)
      log.plot_epochs(["train_acc", "val_acc"], ax=ax[1], title="Acc_curve");␣
        ↪#acc_curve
```

```
100%|      | 106/106 [00:00<00:00, 404.98it/s]
WARNING:matplotlib.legend:No artists with labels found to put in legend.  Note
that artists whose label start with an underscore are ignored when legend() is
called with no argument.
100%|      | 106/106 [00:00<00:00, 385.47it/s]
```





##1.6. Building LSTM_bi

[26]:
```
class LSTM_bi(nn.Module):
  def __init__(self, input_size, hidden_size, num_layers, num_classes):
    super().__init__()
    self.input_size= input_size
    self.hidden_size= hidden_size
    self.num_layers= num_layers
    self.num_classes= num_classes

    self.lstm_block= nn.LSTM(input_size, hidden_size, num_layers,␣
↪batch_first=True, bidirectional= True)

    self.classification_block= nn.Sequential(nn.Linear(2*hidden_size,␣
↪num_classes),
                                             nn.Softmax(dim=-1))

  def forward(self, x):
    h0, c0= torch.zeros(2*self.num_layers, x.size(0), self.hidden_size).
↪to(device), torch.zeros(2*self.num_layers, x.size(0), self.hidden_size).
↪to(device)
    x, h_out= self.lstm_block(x, (h0, c0))
    x= self.classification_block(x[:, -1,:])
    return x

  def loss_and_accuracy(self, y_true, y_p):
    loss_fn= nn.CrossEntropyLoss()
    loss= loss_fn(y_p, y_true)
    y_p= torch.argmax(y_p, dim=-1)
    acc= torch.sum(torch.eq(y_true, y_p))/len(y_true)
    return loss, acc
```

```
[27]: lstm_bi= LSTM_bi(input_size, hidden_size, num_layers, num_classes).to(device)
      summary(lstm_bi, input_size= (32, 28, 28), col_names= ["input_size",␣
       ↪"output_size", "num_params"])
```

```
[27]: ================================================================================
      ===================================
      Layer (type:depth-idx)                   Input Shape               Output Shape
      Param #
      ================================================================================
      ===================================
      LSTM_bi                                  [32, 28, 28]              [32, 10]
      --
       LSTM: 1-1                               [32, 28, 28]              [32, 28, 256]
      557,056
       Sequential: 1-2                         [32, 256]                 [32, 10]
      --
           Linear: 2-1                         [32, 256]                 [32, 10]
      2,570
           Softmax: 2-2                        [32, 10]                  [32, 10]
      --
      ================================================================================
      ===================================
      Total params: 559,626
      Trainable params: 559,626
      Non-trainable params: 0
      Total mult-adds (M): 499.20
      ================================================================================
      ===================================
      Input size (MB): 0.10
      Forward/backward pass size (MB): 1.84
      Params size (MB): 2.24
      Estimated Total Size (MB): 4.18
      ================================================================================
      ===================================
```

###**1.6.1. Training the model**

```
[28]: EPOCHS=5 #the val acc seems to saturate after 5th epoch
      criterion= lstm_bi.loss_and_accuracy
      optimizer=torch.optim.Adam(params= lstm_bi.parameters())
```

```
[29]: log= Report(EPOCHS)
      for epoch in range(EPOCHS):
        n=len(train_dl)
        for ix, input in enumerate(train_dl):
          train_loss, train_accuracy=train_epoch(lstm_bi, input, criterion, optimizer)
```

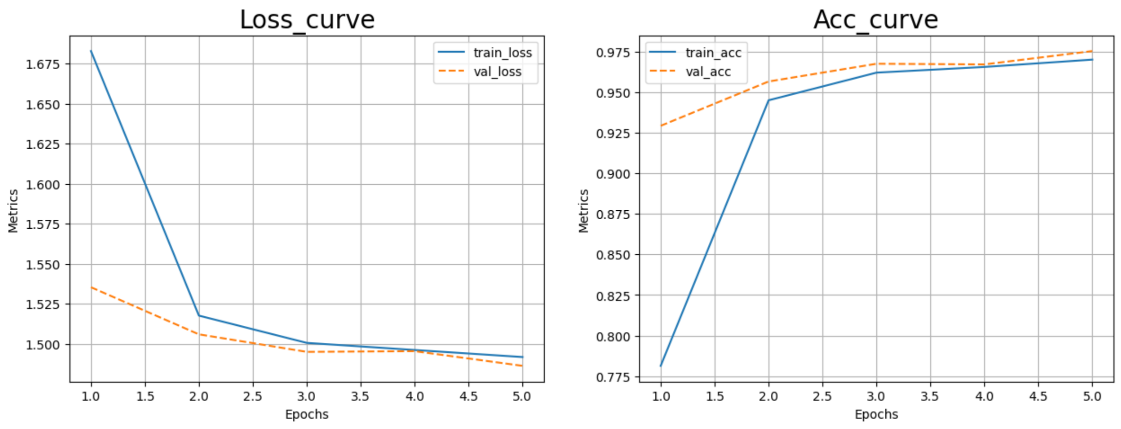
```
    log.record(epoch+(ix+1)/n, train_loss=train_loss, train_acc=␣
↪train_accuracy, end="\r")
  n=len(test_dl)
  for ix, input in enumerate(test_dl):
    val_loss, val_accuracy=val_epoch(lstm_bi, input, criterion)
    log.record(epoch+(ix+1)/n, val_loss=val_loss, val_acc= val_accuracy,␣
↪end="\r")
  log.report_avgs(epoch+1)
```

```
EPOCH: 1.000  train_loss: 1.683  val_loss: 1.535  train_acc: 0.781  val_acc:
0.929  (18.35s - 73.38s remaining)
EPOCH: 2.000  train_loss: 1.518  val_loss: 1.506  train_acc: 0.945  val_acc:
0.956  (35.55s - 53.33s remaining)
EPOCH: 3.000  train_loss: 1.500  val_loss: 1.495  train_acc: 0.962  val_acc:
0.967  (52.79s - 35.20s remaining)
EPOCH: 4.000  train_loss: 1.496  val_loss: 1.495  train_acc: 0.965  val_acc:
0.967  (70.88s - 17.72s remaining)
EPOCH: 5.000  train_loss: 1.492  val_loss: 1.486  train_acc: 0.970  val_acc:
0.975  (87.94s - 0.00s remaining)
```

[30]:
```
lstm_bi_results= {"train_acc": train_accuracy, "val_acc": val_accuracy,␣
↪"train_loss": train_loss, "val_loss": val_loss, "epochs": 5}
```

[31]:
```
fig, ax= plt.subplots(ncols=2, figsize=(15,5)) #w,h
log.plot_epochs(["train_loss", "val_loss"], ax=ax[0], title="Loss_curve");␣
↪#loss_curve
ax[0].legend(loc='upper right', fontsize=10)
log.plot_epochs(["train_acc", "val_acc"], ax=ax[1], title="Acc_curve");␣
↪#acc_curve
```

```
100%|      | 106/106 [00:00<00:00, 404.25it/s]
WARNING:matplotlib.legend:No artists with labels found to put in legend.  Note
that artists whose label start with an underscore are ignored when legend() is
called with no argument.
100%|      | 106/106 [00:00<00:00, 410.50it/s]
```



[32]:
```
import pandas as pd
data=[naive_rnn_results, lstm_results, gru_results, lstm_bi_results]
index=["naive_rnn", "lstm", "gru", "lstm_bi"]
result_df= pd.DataFrame(data)
result_df.set_index([index], inplace= True)
result_df
```

[32]:
```
           train_acc  val_acc  train_loss  val_loss  epochs
naive_rnn    0.50000  0.50000    1.970288  1.938931       5
lstm         1.00000  0.93750    1.461198  1.517358       5
gru          1.00000  0.93750    1.461198  1.517358       5
lstm_bi      0.96875  0.96875    1.498682  1.482065       5
```

Note that the LSTM model and the BiDirectional LSTM perform on par, there is no gain in additional bidirection. So for the best performing model we shall consider LSTM itself (for lower run time and complexity)

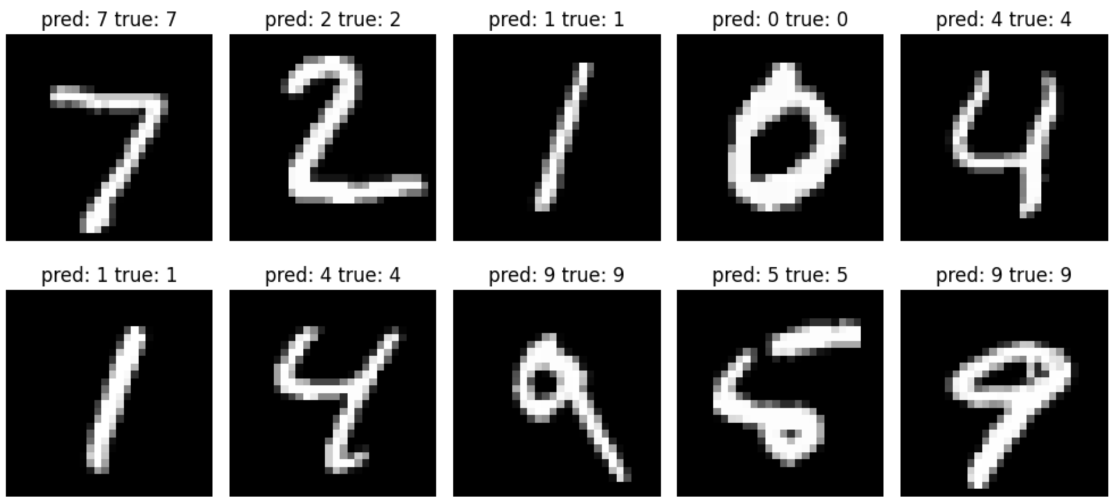
##1.7. **Tesing on val_images**

[33]:
```
ims, labels= next(iter(test_dl))
ims, labels= ims[:10], labels[:10]
ims.shape, labels.shape
```

[33]:
```
(torch.Size([10, 1, 28, 28]), torch.Size([10]))
```

[36]:
```
with torch.inference_mode():
  out= lstm(ims.view(-1, 28,28).to(device))
  out= torch.argmax(out, dim=-1)
```

[37]:
```
titles=[f"pred: {out[i].item()} true: {labels[i].item()}" for i in␣
 ↪range(len(out))]
subplots(ims.squeeze().numpy(), titles=titles, figsize= (10,5))
```



Our lstm model performs really well.

#2. Adding 2 binary string

##2.1. Custom dataset

[38]:
```
def gen_bin(n):
  gen=[]
  for i in range(2**n, 2**(n+1)):
    gen.append(bin(i))
  return gen

def break_bin(num):
  temp= list(num[3:])
  temp= [int(i) for i in temp]
  return temp

def input_prep(s1, s2):
  label, out= [], []
  num1, num2= break_bin(s1), break_bin(s2)
  num1, num2= num1[::-1], num2[::-1]
  sum = bin(int(s1[3:], 2) + int(s2[3:], 2))[2:]
  for idx in range(len(num1)):
    out.append(torch.Tensor([num1[idx], num2[idx]]))
  sum= list(sum)
  sum=sum[::-1]
  if len(sum)<len(num1)+1:
    sum.extend([0]*(len(num1)+1-len(sum)))
  sum= torch.LongTensor([int(i) for i in sum])
  out= torch.vstack(out)
  return out, sum
```

[39]:
```python
import random

class binary_string_dataset(Dataset):
  def __init__(self, samples, binary_string_len):
    self.samples= samples
    self.string_len= binary_string_len
    self.data, self.target=[], []

    for i in range(samples):
      x= gen_bin(binary_string_len)
      idx_1, idx_2= random.randint(0, len(x)-1), random.randint(0, len(x)-1)
      input, target= input_prep(x[idx_1], x[idx_2])
      self.data.extend([input])
      self.target.extend([target])

  def __len__(self):
    return self.samples

  def __getitem__(self, idx):
    return self.data[idx], self.target[idx]
```

[40]:
```python
N=10

train_ds= binary_string_dataset(32*10, N)
test_ds= binary_string_dataset(32*3, N)
train_ds[2]
```

[40]:
```
(tensor([[0., 1.],
         [1., 0.],
         [0., 1.],
         [1., 1.],
         [0., 0.],
         [0., 1.],
         [0., 1.],
         [1., 0.],
         [1., 0.],
         [1., 0.]]),
 tensor([1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0]))
```

[41]:
```python
len(train_ds)
```

[41]:
```
320
```

[42]:
```python
train_dl= torch.utils.data.DataLoader(train_ds, batch_size=32)
test_dl= torch.utils.data.DataLoader(test_ds, batch_size=32)
```

##**2.2. Setting up a LSTM model**

[43]:
```
input_size=2

#hypermarameters:
hidden_size=64
sequence_length= N
num_layers=2
num_classes= N
```

[44]:
```
class LSTM(nn.Module):
  def __init__(self, input_size, hidden_size, num_layers, num_classes):
    super().__init__()
    self.input_size= input_size
    self.hidden_size= hidden_size
    self.num_layers= num_layers
    self.num_classes= num_classes

    self.lstm_block= nn.LSTM(input_size, hidden_size, num_layers,␣
↪batch_first=True)

    self.classification_block= nn.Sequential(nn.Linear(hidden_size,␣
↪num_classes+1),
                                             nn.Sigmoid())

  def forward(self, x):
    h0, c0= torch.zeros(self.num_layers, x.size(0), self.hidden_size).
↪to(device), torch.zeros(self.num_layers, x.size(0), self.hidden_size).
↪to(device)
    x, h_out= self.lstm_block(x, (h0, c0))
    x= self.classification_block(x[:, -1,:])
    return x

  def loss_and_accuracy(self, y_true, y_p):
    loss_fn= nn.BCELoss()
    y_true= y_true.type(torch.float32)
    loss= loss_fn(y_p, y_true)
    y_p= torch.round(y_p)
    acc= torch.sum(torch.eq(y_true, y_p))/(len(y_true)*(N+1))
    return loss, acc
```

[45]:
```
lstm= LSTM(input_size, hidden_size, num_layers, num_classes).to(device)
summary(lstm, input_size= (32, 5, 2), col_names= ["input_size", "output_size",␣
↪"num_params"])
```

[45]:
```
================================================================================
===================================
Layer (type:depth-idx)                   Input Shape               Output Shape
Param #
```

```
================================================================================
===================================
LSTM                                      [32, 5, 2]                [32, 11]
--
 LSTM: 1-1                               [32, 5, 2]                [32, 5, 64]
50,688
 Sequential: 1-2                         [32, 64]                  [32, 11]
--
     Linear: 2-1                         [32, 64]                  [32, 11]
715
     Sigmoid: 2-2                        [32, 11]                  [32, 11]
--
================================================================================
===================================
Total params: 51,403
Trainable params: 51,403
Non-trainable params: 0
Total mult-adds (M): 8.13
================================================================================
===================================
Input size (MB): 0.00
Forward/backward pass size (MB): 0.08
Params size (MB): 0.21
Estimated Total Size (MB): 0.29
================================================================================
===================================
```

[46]:
```
EPOCHS=500
criterion= lstm.loss_and_accuracy
optimizer=torch.optim.Adam(params= lstm.parameters())
```

##2.3. **Running the model**

[47]:
```
log= Report(EPOCHS)
for epoch in range(EPOCHS):
  n=len(train_dl)
  for ix, input in enumerate(train_dl):
    train_loss, train_accuracy=train_epoch(lstm, input, criterion, optimizer,␣
↪sequence_length=sequence_length)
    log.record(epoch+(ix+1)/n, train_loss=train_loss, train_acc=␣
↪train_accuracy, end="\r")
  n=len(test_dl)
  for ix, input in enumerate(test_dl):
    val_loss, val_accuracy=val_epoch(lstm, input, criterion,␣
↪sequence_length=sequence_length)
    log.record(epoch+(ix+1)/n, val_loss=val_loss, val_acc= val_accuracy,␣
↪end="\r")
```

```
log.report_avgs(epoch+1)
```

```
EPOCH: 1.000  train_loss: 0.693  val_loss: 0.695  train_acc: 0.505  val_acc:
0.462  (0.05s - 27.06s remaining)
EPOCH: 2.000  train_loss: 0.692  val_loss: 0.694  train_acc: 0.520  val_acc:
0.514  (0.09s - 21.82s remaining)
EPOCH: 3.000  train_loss: 0.691  val_loss: 0.694  train_acc: 0.539  val_acc:
0.506  (0.12s - 19.98s remaining)
EPOCH: 4.000  train_loss: 0.691  val_loss: 0.695  train_acc: 0.530  val_acc:
0.507  (0.15s - 19.01s remaining)
EPOCH: 5.000  train_loss: 0.690  val_loss: 0.695  train_acc: 0.539  val_acc:
0.515  (0.19s - 18.44s remaining)
EPOCH: 6.000  train_loss: 0.690  val_loss: 0.694  train_acc: 0.545  val_acc:
0.514  (0.22s - 18.16s remaining)
EPOCH: 7.000  train_loss: 0.689  val_loss: 0.693  train_acc: 0.548  val_acc:
0.512  (0.25s - 17.94s remaining)
EPOCH: 8.000  train_loss: 0.687  val_loss: 0.693  train_acc: 0.545  val_acc:
0.512  (0.29s - 17.95s remaining)
EPOCH: 9.000  train_loss: 0.685  val_loss: 0.691  train_acc: 0.543  val_acc:
0.513  (0.32s - 17.67s remaining)
EPOCH: 10.000  train_loss: 0.682  val_loss: 0.687  train_acc: 0.544  val_acc:
0.516  (0.36s - 17.41s remaining)
EPOCH: 11.000  train_loss: 0.678  val_loss: 0.684  train_acc: 0.552  val_acc:
0.524  (0.39s - 17.17s remaining)
EPOCH: 12.000  train_loss: 0.675  val_loss: 0.684  train_acc: 0.558  val_acc:
0.520  (0.42s - 17.01s remaining)
EPOCH: 13.000  train_loss: 0.671  val_loss: 0.680  train_acc: 0.553  val_acc:
0.516  (0.45s - 17.01s remaining)
EPOCH: 14.000  train_loss: 0.666  val_loss: 0.671  train_acc: 0.564  val_acc:
0.546  (0.49s - 16.88s remaining)
EPOCH: 15.000  train_loss: 0.661  val_loss: 0.667  train_acc: 0.571  val_acc:
0.557  (0.52s - 16.78s remaining)
EPOCH: 16.000  train_loss: 0.656  val_loss: 0.666  train_acc: 0.575  val_acc:
0.546  (0.56s - 16.89s remaining)
EPOCH: 17.000  train_loss: 0.652  val_loss: 0.666  train_acc: 0.582  val_acc:
0.554  (0.60s - 17.01s remaining)
EPOCH: 18.000  train_loss: 0.649  val_loss: 0.668  train_acc: 0.591  val_acc:
0.545  (0.63s - 16.91s remaining)
EPOCH: 19.000  train_loss: 0.648  val_loss: 0.668  train_acc: 0.590  val_acc:
0.541  (0.67s - 16.94s remaining)
EPOCH: 20.000  train_loss: 0.646  val_loss: 0.665  train_acc: 0.589  val_acc:
0.547  (0.70s - 16.83s remaining)
EPOCH: 21.000  train_loss: 0.642  val_loss: 0.659  train_acc: 0.600  val_acc:
0.550  (0.73s - 16.73s remaining)
EPOCH: 22.000  train_loss: 0.638  val_loss: 0.657  train_acc: 0.606  val_acc:
0.552  (0.76s - 16.62s remaining)
EPOCH: 23.000  train_loss: 0.635  val_loss: 0.658  train_acc: 0.608  val_acc:
```

```
0.552  (0.80s - 16.56s remaining)
EPOCH: 24.000  train_loss: 0.634  val_loss: 0.658  train_acc: 0.609  val_acc:
0.561  (0.83s - 16.48s remaining)
EPOCH: 25.000  train_loss: 0.630  val_loss: 0.656  train_acc: 0.618  val_acc:
0.559  (0.87s - 16.60s remaining)
EPOCH: 26.000  train_loss: 0.627  val_loss: 0.657  train_acc: 0.622  val_acc:
0.566  (0.91s - 16.51s remaining)
EPOCH: 27.000  train_loss: 0.624  val_loss: 0.658  train_acc: 0.626  val_acc:
0.566  (0.94s - 16.42s remaining)
EPOCH: 28.000  train_loss: 0.622  val_loss: 0.658  train_acc: 0.629  val_acc:
0.562  (0.97s - 16.33s remaining)
EPOCH: 29.000  train_loss: 0.616  val_loss: 0.656  train_acc: 0.640  val_acc:
0.570  (1.00s - 16.26s remaining)
EPOCH: 30.000  train_loss: 0.611  val_loss: 0.655  train_acc: 0.646  val_acc:
0.576  (1.03s - 16.19s remaining)
EPOCH: 31.000  train_loss: 0.607  val_loss: 0.653  train_acc: 0.651  val_acc:
0.579  (1.07s - 16.17s remaining)
EPOCH: 32.000  train_loss: 0.603  val_loss: 0.652  train_acc: 0.660  val_acc:
0.586  (1.11s - 16.18s remaining)
EPOCH: 33.000  train_loss: 0.598  val_loss: 0.648  train_acc: 0.665  val_acc:
0.589  (1.14s - 16.12s remaining)
EPOCH: 34.000  train_loss: 0.592  val_loss: 0.642  train_acc: 0.674  val_acc:
0.600  (1.17s - 16.06s remaining)
EPOCH: 35.000  train_loss: 0.582  val_loss: 0.638  train_acc: 0.691  val_acc:
0.598  (1.20s - 15.99s remaining)
EPOCH: 36.000  train_loss: 0.572  val_loss: 0.633  train_acc: 0.704  val_acc:
0.613  (1.24s - 15.93s remaining)
EPOCH: 37.000  train_loss: 0.561  val_loss: 0.626  train_acc: 0.714  val_acc:
0.622  (1.28s - 15.99s remaining)
EPOCH: 38.000  train_loss: 0.549  val_loss: 0.613  train_acc: 0.728  val_acc:
0.633  (1.31s - 15.96s remaining)
EPOCH: 39.000  train_loss: 0.535  val_loss: 0.599  train_acc: 0.742  val_acc:
0.658  (1.35s - 15.94s remaining)
EPOCH: 40.000  train_loss: 0.522  val_loss: 0.584  train_acc: 0.748  val_acc:
0.669  (1.38s - 15.90s remaining)
EPOCH: 41.000  train_loss: 0.511  val_loss: 0.574  train_acc: 0.752  val_acc:
0.680  (1.42s - 15.86s remaining)
EPOCH: 42.000  train_loss: 0.502  val_loss: 0.565  train_acc: 0.761  val_acc:
0.683  (1.45s - 15.82s remaining)
EPOCH: 43.000  train_loss: 0.490  val_loss: 0.555  train_acc: 0.776  val_acc:
0.695  (1.49s - 15.82s remaining)
EPOCH: 44.000  train_loss: 0.479  val_loss: 0.544  train_acc: 0.786  val_acc:
0.705  (1.52s - 15.77s remaining)
EPOCH: 45.000  train_loss: 0.468  val_loss: 0.535  train_acc: 0.796  val_acc:
0.722  (1.55s - 15.71s remaining)
EPOCH: 46.000  train_loss: 0.456  val_loss: 0.523  train_acc: 0.801  val_acc:
0.731  (1.59s - 15.66s remaining)
EPOCH: 47.000  train_loss: 0.445  val_loss: 0.513  train_acc: 0.814  val_acc:
```

```
0.752  (1.62s - 15.66s remaining)
EPOCH: 48.000  train_loss: 0.434  val_loss: 0.503  train_acc: 0.821  val_acc:
0.759  (1.66s - 15.62s remaining)
EPOCH: 49.000  train_loss: 0.425  val_loss: 0.496  train_acc: 0.827  val_acc:
0.757  (1.70s - 15.61s remaining)
EPOCH: 50.000  train_loss: 0.414  val_loss: 0.488  train_acc: 0.834  val_acc:
0.765  (1.73s - 15.57s remaining)
EPOCH: 51.000  train_loss: 0.407  val_loss: 0.483  train_acc: 0.837  val_acc:
0.765  (1.76s - 15.50s remaining)
EPOCH: 52.000  train_loss: 0.399  val_loss: 0.480  train_acc: 0.840  val_acc:
0.759  (1.79s - 15.46s remaining)
EPOCH: 53.000  train_loss: 0.394  val_loss: 0.470  train_acc: 0.842  val_acc:
0.766  (1.83s - 15.41s remaining)
EPOCH: 54.000  train_loss: 0.386  val_loss: 0.466  train_acc: 0.843  val_acc:
0.775  (1.86s - 15.36s remaining)
EPOCH: 55.000  train_loss: 0.385  val_loss: 0.465  train_acc: 0.843  val_acc:
0.777  (1.90s - 15.34s remaining)
EPOCH: 56.000  train_loss: 0.378  val_loss: 0.453  train_acc: 0.851  val_acc:
0.791  (1.93s - 15.30s remaining)
EPOCH: 57.000  train_loss: 0.371  val_loss: 0.442  train_acc: 0.852  val_acc:
0.795  (1.96s - 15.25s remaining)
EPOCH: 58.000  train_loss: 0.364  val_loss: 0.427  train_acc: 0.859  val_acc:
0.799  (2.00s - 15.25s remaining)
EPOCH: 59.000  train_loss: 0.358  val_loss: 0.412  train_acc: 0.857  val_acc:
0.814  (2.03s - 15.20s remaining)
EPOCH: 60.000  train_loss: 0.352  val_loss: 0.406  train_acc: 0.861  val_acc:
0.818  (2.07s - 15.15s remaining)
EPOCH: 61.000  train_loss: 0.341  val_loss: 0.392  train_acc: 0.867  val_acc:
0.836  (2.10s - 15.13s remaining)
EPOCH: 62.000  train_loss: 0.323  val_loss: 0.385  train_acc: 0.879  val_acc:
0.838  (2.13s - 15.08s remaining)
EPOCH: 63.000  train_loss: 0.316  val_loss: 0.376  train_acc: 0.883  val_acc:
0.834  (2.17s - 15.03s remaining)
EPOCH: 64.000  train_loss: 0.309  val_loss: 0.367  train_acc: 0.887  val_acc:
0.843  (2.20s - 14.99s remaining)
EPOCH: 65.000  train_loss: 0.300  val_loss: 0.361  train_acc: 0.896  val_acc:
0.850  (2.24s - 14.98s remaining)
EPOCH: 66.000  train_loss: 0.293  val_loss: 0.355  train_acc: 0.900  val_acc:
0.848  (2.27s - 14.93s remaining)
EPOCH: 67.000  train_loss: 0.286  val_loss: 0.349  train_acc: 0.903  val_acc:
0.850  (2.31s - 14.91s remaining)
EPOCH: 68.000  train_loss: 0.278  val_loss: 0.343  train_acc: 0.909  val_acc:
0.854  (2.34s - 14.85s remaining)
EPOCH: 69.000  train_loss: 0.271  val_loss: 0.337  train_acc: 0.911  val_acc:
0.862  (2.37s - 14.80s remaining)
EPOCH: 70.000  train_loss: 0.264  val_loss: 0.332  train_acc: 0.916  val_acc:
0.868  (2.40s - 14.75s remaining)
EPOCH: 71.000  train_loss: 0.257  val_loss: 0.327  train_acc: 0.921  val_acc:
```

```
0.871  (2.43s - 14.70s remaining)
EPOCH: 72.000  train_loss: 0.250  val_loss: 0.323  train_acc: 0.922  val_acc:
0.875  (2.46s - 14.65s remaining)
EPOCH: 73.000  train_loss: 0.244  val_loss: 0.320  train_acc: 0.925  val_acc:
0.876  (2.50s - 14.62s remaining)
EPOCH: 74.000  train_loss: 0.238  val_loss: 0.316  train_acc: 0.929  val_acc:
0.876  (2.53s - 14.58s remaining)
EPOCH: 75.000  train_loss: 0.232  val_loss: 0.313  train_acc: 0.931  val_acc:
0.877  (2.56s - 14.53s remaining)
EPOCH: 76.000  train_loss: 0.227  val_loss: 0.307  train_acc: 0.934  val_acc:
0.881  (2.60s - 14.53s remaining)
EPOCH: 77.000  train_loss: 0.223  val_loss: 0.302  train_acc: 0.936  val_acc:
0.887  (2.65s - 14.55s remaining)
EPOCH: 78.000  train_loss: 0.219  val_loss: 0.301  train_acc: 0.938  val_acc:
0.883  (2.68s - 14.52s remaining)
EPOCH: 79.000  train_loss: 0.217  val_loss: 0.302  train_acc: 0.938  val_acc:
0.879  (2.72s - 14.50s remaining)
EPOCH: 80.000  train_loss: 0.216  val_loss: 0.296  train_acc: 0.941  val_acc:
0.885  (2.75s - 14.46s remaining)
EPOCH: 81.000  train_loss: 0.211  val_loss: 0.290  train_acc: 0.942  val_acc:
0.887  (2.79s - 14.41s remaining)
EPOCH: 82.000  train_loss: 0.204  val_loss: 0.278  train_acc: 0.947  val_acc:
0.896  (2.83s - 14.41s remaining)
EPOCH: 83.000  train_loss: 0.191  val_loss: 0.271  train_acc: 0.957  val_acc:
0.897  (2.86s - 14.37s remaining)
EPOCH: 84.000  train_loss: 0.184  val_loss: 0.265  train_acc: 0.963  val_acc:
0.908  (2.89s - 14.33s remaining)
EPOCH: 85.000  train_loss: 0.176  val_loss: 0.261  train_acc: 0.967  val_acc:
0.907  (2.93s - 14.32s remaining)
EPOCH: 86.000  train_loss: 0.168  val_loss: 0.256  train_acc: 0.971  val_acc:
0.914  (2.97s - 14.28s remaining)
EPOCH: 87.000  train_loss: 0.162  val_loss: 0.252  train_acc: 0.974  val_acc:
0.912  (3.01s - 14.29s remaining)
EPOCH: 88.000  train_loss: 0.157  val_loss: 0.249  train_acc: 0.975  val_acc:
0.911  (3.04s - 14.25s remaining)
EPOCH: 89.000  train_loss: 0.153  val_loss: 0.245  train_acc: 0.977  val_acc:
0.912  (3.08s - 14.20s remaining)
EPOCH: 90.000  train_loss: 0.149  val_loss: 0.241  train_acc: 0.978  val_acc:
0.915  (3.11s - 14.17s remaining)
EPOCH: 91.000  train_loss: 0.144  val_loss: 0.236  train_acc: 0.980  val_acc:
0.920  (3.14s - 14.13s remaining)
EPOCH: 92.000  train_loss: 0.140  val_loss: 0.232  train_acc: 0.981  val_acc:
0.925  (3.18s - 14.09s remaining)
EPOCH: 93.000  train_loss: 0.136  val_loss: 0.228  train_acc: 0.982  val_acc:
0.927  (3.22s - 14.07s remaining)
EPOCH: 94.000  train_loss: 0.132  val_loss: 0.226  train_acc: 0.983  val_acc:
0.929  (3.25s - 14.05s remaining)
EPOCH: 95.000  train_loss: 0.128  val_loss: 0.223  train_acc: 0.986  val_acc:
```

```
0.928  (3.29s - 14.01s remaining)
EPOCH: 96.000  train_loss: 0.124  val_loss: 0.221  train_acc: 0.989  val_acc:
0.927  (3.32s - 13.99s remaining)
EPOCH: 97.000  train_loss: 0.120  val_loss: 0.220  train_acc: 0.989  val_acc:
0.930  (3.36s - 13.94s remaining)
EPOCH: 98.000  train_loss: 0.117  val_loss: 0.220  train_acc: 0.990  val_acc:
0.928  (3.39s - 13.90s remaining)
EPOCH: 99.000  train_loss: 0.115  val_loss: 0.222  train_acc: 0.991  val_acc:
0.929  (3.42s - 13.86s remaining)
EPOCH: 100.000  train_loss: 0.115  val_loss: 0.221  train_acc: 0.990  val_acc:
0.929  (3.45s - 13.82s remaining)
EPOCH: 101.000  train_loss: 0.118  val_loss: 0.211  train_acc: 0.989  val_acc:
0.933  (3.49s - 13.78s remaining)
EPOCH: 102.000  train_loss: 0.116  val_loss: 0.212  train_acc: 0.989  val_acc:
0.933  (3.52s - 13.74s remaining)
EPOCH: 103.000  train_loss: 0.117  val_loss: 0.198  train_acc: 0.982  val_acc:
0.944  (3.56s - 13.71s remaining)
EPOCH: 104.000  train_loss: 0.107  val_loss: 0.190  train_acc: 0.986  val_acc:
0.952  (3.59s - 13.67s remaining)
EPOCH: 105.000  train_loss: 0.101  val_loss: 0.188  train_acc: 0.991  val_acc:
0.945  (3.62s - 13.63s remaining)
EPOCH: 106.000  train_loss: 0.097  val_loss: 0.181  train_acc: 0.994  val_acc:
0.947  (3.66s - 13.61s remaining)
EPOCH: 107.000  train_loss: 0.092  val_loss: 0.179  train_acc: 0.994  val_acc:
0.952  (3.69s - 13.57s remaining)
EPOCH: 108.000  train_loss: 0.089  val_loss: 0.176  train_acc: 0.996  val_acc:
0.950  (3.73s - 13.54s remaining)
EPOCH: 109.000  train_loss: 0.086  val_loss: 0.175  train_acc: 0.995  val_acc:
0.951  (3.76s - 13.50s remaining)
EPOCH: 110.000  train_loss: 0.083  val_loss: 0.174  train_acc: 0.997  val_acc:
0.950  (3.80s - 13.46s remaining)
EPOCH: 111.000  train_loss: 0.081  val_loss: 0.172  train_acc: 0.997  val_acc:
0.953  (3.83s - 13.42s remaining)
EPOCH: 112.000  train_loss: 0.079  val_loss: 0.170  train_acc: 0.997  val_acc:
0.954  (3.86s - 13.38s remaining)
EPOCH: 113.000  train_loss: 0.077  val_loss: 0.167  train_acc: 0.998  val_acc:
0.954  (3.90s - 13.34s remaining)
EPOCH: 114.000  train_loss: 0.075  val_loss: 0.164  train_acc: 0.999  val_acc:
0.955  (3.93s - 13.30s remaining)
EPOCH: 115.000  train_loss: 0.073  val_loss: 0.162  train_acc: 0.998  val_acc:
0.955  (3.97s - 13.29s remaining)
EPOCH: 116.000  train_loss: 0.071  val_loss: 0.161  train_acc: 0.999  val_acc:
0.955  (4.00s - 13.25s remaining)
EPOCH: 117.000  train_loss: 0.069  val_loss: 0.160  train_acc: 0.999  val_acc:
0.956  (4.04s - 13.22s remaining)
EPOCH: 118.000  train_loss: 0.067  val_loss: 0.159  train_acc: 1.000  val_acc:
0.957  (4.07s - 13.18s remaining)
EPOCH: 119.000  train_loss: 0.065  val_loss: 0.158  train_acc: 1.000  val_acc:
```

```
0.957  (4.11s - 13.15s remaining)
EPOCH: 120.000  train_loss: 0.063  val_loss: 0.157  train_acc: 1.000  val_acc:
0.955  (4.14s - 13.12s remaining)
EPOCH: 121.000  train_loss: 0.062  val_loss: 0.156  train_acc: 1.000  val_acc:
0.956  (4.18s - 13.09s remaining)
EPOCH: 122.000  train_loss: 0.060  val_loss: 0.154  train_acc: 1.000  val_acc:
0.957  (4.21s - 13.05s remaining)
EPOCH: 123.000  train_loss: 0.059  val_loss: 0.153  train_acc: 1.000  val_acc:
0.958  (4.24s - 13.01s remaining)
EPOCH: 124.000  train_loss: 0.057  val_loss: 0.151  train_acc: 1.000  val_acc:
0.958  (4.28s - 12.97s remaining)
EPOCH: 125.000  train_loss: 0.056  val_loss: 0.149  train_acc: 1.000  val_acc:
0.960  (4.31s - 12.93s remaining)
EPOCH: 126.000  train_loss: 0.055  val_loss: 0.148  train_acc: 1.000  val_acc:
0.960  (4.34s - 12.89s remaining)
EPOCH: 127.000  train_loss: 0.053  val_loss: 0.146  train_acc: 1.000  val_acc:
0.960  (4.38s - 12.86s remaining)
EPOCH: 128.000  train_loss: 0.052  val_loss: 0.145  train_acc: 1.000  val_acc:
0.960  (4.42s - 12.84s remaining)
EPOCH: 129.000  train_loss: 0.051  val_loss: 0.145  train_acc: 1.000  val_acc:
0.961  (4.45s - 12.81s remaining)
EPOCH: 130.000  train_loss: 0.050  val_loss: 0.144  train_acc: 1.000  val_acc:
0.960  (4.49s - 12.77s remaining)
EPOCH: 131.000  train_loss: 0.048  val_loss: 0.143  train_acc: 1.000  val_acc:
0.959  (4.52s - 12.73s remaining)
EPOCH: 132.000  train_loss: 0.047  val_loss: 0.143  train_acc: 1.000  val_acc:
0.961  (4.56s - 12.70s remaining)
EPOCH: 133.000  train_loss: 0.046  val_loss: 0.141  train_acc: 1.000  val_acc:
0.962  (4.59s - 12.67s remaining)
EPOCH: 134.000  train_loss: 0.045  val_loss: 0.140  train_acc: 1.000  val_acc:
0.963  (4.63s - 12.64s remaining)
EPOCH: 135.000  train_loss: 0.044  val_loss: 0.139  train_acc: 1.000  val_acc:
0.965  (4.67s - 12.61s remaining)
EPOCH: 136.000  train_loss: 0.043  val_loss: 0.137  train_acc: 1.000  val_acc:
0.965  (4.70s - 12.57s remaining)
EPOCH: 137.000  train_loss: 0.042  val_loss: 0.136  train_acc: 1.000  val_acc:
0.966  (4.73s - 12.53s remaining)
EPOCH: 138.000  train_loss: 0.041  val_loss: 0.134  train_acc: 1.000  val_acc:
0.967  (4.77s - 12.50s remaining)
EPOCH: 139.000  train_loss: 0.040  val_loss: 0.133  train_acc: 1.000  val_acc:
0.966  (4.80s - 12.46s remaining)
EPOCH: 140.000  train_loss: 0.039  val_loss: 0.132  train_acc: 1.000  val_acc:
0.966  (4.83s - 12.43s remaining)
EPOCH: 141.000  train_loss: 0.039  val_loss: 0.131  train_acc: 1.000  val_acc:
0.965  (4.87s - 12.39s remaining)
EPOCH: 142.000  train_loss: 0.038  val_loss: 0.130  train_acc: 1.000  val_acc:
0.967  (4.90s - 12.35s remaining)
EPOCH: 143.000  train_loss: 0.037  val_loss: 0.129  train_acc: 1.000  val_acc:
```

```
0.966  (4.93s - 12.31s remaining)
EPOCH: 144.000  train_loss: 0.036  val_loss: 0.128  train_acc: 1.000  val_acc:
0.967  (4.96s - 12.27s remaining)
EPOCH: 145.000  train_loss: 0.035  val_loss: 0.128  train_acc: 1.000  val_acc:
0.967  (5.00s - 12.25s remaining)
EPOCH: 146.000  train_loss: 0.035  val_loss: 0.127  train_acc: 1.000  val_acc:
0.970  (5.04s - 12.21s remaining)
EPOCH: 147.000  train_loss: 0.034  val_loss: 0.127  train_acc: 1.000  val_acc:
0.971  (5.07s - 12.18s remaining)
EPOCH: 148.000  train_loss: 0.033  val_loss: 0.126  train_acc: 1.000  val_acc:
0.972  (5.10s - 12.14s remaining)
EPOCH: 149.000  train_loss: 0.033  val_loss: 0.126  train_acc: 1.000  val_acc:
0.973  (5.14s - 12.11s remaining)
EPOCH: 150.000  train_loss: 0.032  val_loss: 0.125  train_acc: 1.000  val_acc:
0.972  (5.18s - 12.08s remaining)
EPOCH: 151.000  train_loss: 0.032  val_loss: 0.124  train_acc: 1.000  val_acc:
0.973  (5.21s - 12.05s remaining)
EPOCH: 152.000  train_loss: 0.031  val_loss: 0.122  train_acc: 1.000  val_acc:
0.973  (5.24s - 12.00s remaining)
EPOCH: 153.000  train_loss: 0.031  val_loss: 0.120  train_acc: 1.000  val_acc:
0.971  (5.28s - 11.97s remaining)
EPOCH: 154.000  train_loss: 0.031  val_loss: 0.118  train_acc: 1.000  val_acc:
0.967  (5.31s - 11.93s remaining)
EPOCH: 155.000  train_loss: 0.030  val_loss: 0.118  train_acc: 1.000  val_acc:
0.966  (5.34s - 11.90s remaining)
EPOCH: 156.000  train_loss: 0.030  val_loss: 0.117  train_acc: 1.000  val_acc:
0.967  (5.38s - 11.86s remaining)
EPOCH: 157.000  train_loss: 0.029  val_loss: 0.115  train_acc: 1.000  val_acc:
0.970  (5.42s - 11.84s remaining)
EPOCH: 158.000  train_loss: 0.029  val_loss: 0.113  train_acc: 1.000  val_acc:
0.971  (5.45s - 11.80s remaining)
EPOCH: 159.000  train_loss: 0.028  val_loss: 0.115  train_acc: 1.000  val_acc:
0.973  (5.49s - 11.77s remaining)
EPOCH: 160.000  train_loss: 0.028  val_loss: 0.118  train_acc: 1.000  val_acc:
0.973  (5.52s - 11.73s remaining)
EPOCH: 161.000  train_loss: 0.028  val_loss: 0.119  train_acc: 1.000  val_acc:
0.973  (5.56s - 11.70s remaining)
EPOCH: 162.000  train_loss: 0.027  val_loss: 0.115  train_acc: 1.000  val_acc:
0.973  (5.59s - 11.67s remaining)
EPOCH: 163.000  train_loss: 0.026  val_loss: 0.111  train_acc: 1.000  val_acc:
0.969  (5.63s - 11.64s remaining)
EPOCH: 164.000  train_loss: 0.026  val_loss: 0.107  train_acc: 1.000  val_acc:
0.968  (5.67s - 11.61s remaining)
EPOCH: 165.000  train_loss: 0.025  val_loss: 0.106  train_acc: 1.000  val_acc:
0.973  (5.71s - 11.59s remaining)
EPOCH: 166.000  train_loss: 0.024  val_loss: 0.110  train_acc: 1.000  val_acc:
0.973  (5.74s - 11.56s remaining)
EPOCH: 167.000  train_loss: 0.024  val_loss: 0.111  train_acc: 1.000  val_acc:
```

```
0.974  (5.78s - 11.52s remaining)
EPOCH: 168.000  train_loss: 0.023  val_loss: 0.108  train_acc: 1.000  val_acc:
0.975  (5.81s - 11.49s remaining)
EPOCH: 169.000  train_loss: 0.023  val_loss: 0.107  train_acc: 1.000  val_acc:
0.973  (5.85s - 11.45s remaining)
EPOCH: 170.000  train_loss: 0.023  val_loss: 0.105  train_acc: 1.000  val_acc:
0.973  (5.88s - 11.42s remaining)
EPOCH: 171.000  train_loss: 0.022  val_loss: 0.105  train_acc: 1.000  val_acc:
0.974  (5.91s - 11.38s remaining)
EPOCH: 172.000  train_loss: 0.022  val_loss: 0.106  train_acc: 1.000  val_acc:
0.975  (5.95s - 11.34s remaining)
EPOCH: 173.000  train_loss: 0.021  val_loss: 0.106  train_acc: 1.000  val_acc:
0.975  (5.98s - 11.31s remaining)
EPOCH: 174.000  train_loss: 0.021  val_loss: 0.105  train_acc: 1.000  val_acc:
0.975  (6.04s - 11.31s remaining)
EPOCH: 175.000  train_loss: 0.021  val_loss: 0.103  train_acc: 1.000  val_acc:
0.976  (6.07s - 11.27s remaining)
EPOCH: 176.000  train_loss: 0.020  val_loss: 0.103  train_acc: 1.000  val_acc:
0.975  (6.10s - 11.23s remaining)
EPOCH: 177.000  train_loss: 0.020  val_loss: 0.102  train_acc: 1.000  val_acc:
0.975  (6.14s - 11.20s remaining)
EPOCH: 178.000  train_loss: 0.020  val_loss: 0.102  train_acc: 1.000  val_acc:
0.974  (6.17s - 11.16s remaining)
EPOCH: 179.000  train_loss: 0.019  val_loss: 0.102  train_acc: 1.000  val_acc:
0.975  (6.20s - 11.12s remaining)
EPOCH: 180.000  train_loss: 0.019  val_loss: 0.102  train_acc: 1.000  val_acc:
0.975  (6.24s - 11.09s remaining)
EPOCH: 181.000  train_loss: 0.019  val_loss: 0.101  train_acc: 1.000  val_acc:
0.974  (6.27s - 11.06s remaining)
EPOCH: 182.000  train_loss: 0.019  val_loss: 0.100  train_acc: 1.000  val_acc:
0.974  (6.31s - 11.02s remaining)
EPOCH: 183.000  train_loss: 0.018  val_loss: 0.100  train_acc: 1.000  val_acc:
0.975  (6.34s - 10.98s remaining)
EPOCH: 184.000  train_loss: 0.018  val_loss: 0.100  train_acc: 1.000  val_acc:
0.974  (6.37s - 10.94s remaining)
EPOCH: 185.000  train_loss: 0.018  val_loss: 0.099  train_acc: 1.000  val_acc:
0.974  (6.40s - 10.90s remaining)
EPOCH: 186.000  train_loss: 0.017  val_loss: 0.099  train_acc: 1.000  val_acc:
0.975  (6.44s - 10.88s remaining)
EPOCH: 187.000  train_loss: 0.017  val_loss: 0.099  train_acc: 1.000  val_acc:
0.976  (6.47s - 10.84s remaining)
EPOCH: 188.000  train_loss: 0.017  val_loss: 0.099  train_acc: 1.000  val_acc:
0.974  (6.51s - 10.80s remaining)
EPOCH: 189.000  train_loss: 0.017  val_loss: 0.098  train_acc: 1.000  val_acc:
0.974  (6.54s - 10.76s remaining)
EPOCH: 190.000  train_loss: 0.017  val_loss: 0.097  train_acc: 1.000  val_acc:
0.974  (6.57s - 10.72s remaining)
EPOCH: 191.000  train_loss: 0.016  val_loss: 0.097  train_acc: 1.000  val_acc:
```

```
0.974  (6.60s - 10.68s remaining)
EPOCH: 192.000  train_loss: 0.016  val_loss: 0.097  train_acc: 1.000  val_acc:
0.974  (6.64s - 10.65s remaining)
EPOCH: 193.000  train_loss: 0.016  val_loss: 0.096  train_acc: 1.000  val_acc:
0.974  (6.67s - 10.61s remaining)
EPOCH: 194.000  train_loss: 0.016  val_loss: 0.096  train_acc: 1.000  val_acc:
0.975  (6.71s - 10.59s remaining)
EPOCH: 195.000  train_loss: 0.015  val_loss: 0.096  train_acc: 1.000  val_acc:
0.975  (6.74s - 10.55s remaining)
EPOCH: 196.000  train_loss: 0.015  val_loss: 0.096  train_acc: 1.000  val_acc:
0.976  (6.78s - 10.51s remaining)
EPOCH: 197.000  train_loss: 0.015  val_loss: 0.096  train_acc: 1.000  val_acc:
0.976  (6.81s - 10.47s remaining)
EPOCH: 198.000  train_loss: 0.015  val_loss: 0.095  train_acc: 1.000  val_acc:
0.976  (6.85s - 10.44s remaining)
EPOCH: 199.000  train_loss: 0.015  val_loss: 0.094  train_acc: 1.000  val_acc:
0.975  (6.88s - 10.41s remaining)
EPOCH: 200.000  train_loss: 0.014  val_loss: 0.094  train_acc: 1.000  val_acc:
0.975  (6.91s - 10.37s remaining)
EPOCH: 201.000  train_loss: 0.014  val_loss: 0.094  train_acc: 1.000  val_acc:
0.975  (6.95s - 10.33s remaining)
EPOCH: 202.000  train_loss: 0.014  val_loss: 0.094  train_acc: 1.000  val_acc:
0.976  (6.98s - 10.29s remaining)
EPOCH: 203.000  train_loss: 0.014  val_loss: 0.093  train_acc: 1.000  val_acc:
0.976  (7.01s - 10.26s remaining)
EPOCH: 204.000  train_loss: 0.014  val_loss: 0.093  train_acc: 1.000  val_acc:
0.976  (7.05s - 10.23s remaining)
EPOCH: 205.000  train_loss: 0.014  val_loss: 0.093  train_acc: 1.000  val_acc:
0.976  (7.09s - 10.20s remaining)
EPOCH: 206.000  train_loss: 0.013  val_loss: 0.093  train_acc: 1.000  val_acc:
0.976  (7.12s - 10.16s remaining)
EPOCH: 207.000  train_loss: 0.013  val_loss: 0.093  train_acc: 1.000  val_acc:
0.976  (7.15s - 10.13s remaining)
EPOCH: 208.000  train_loss: 0.013  val_loss: 0.092  train_acc: 1.000  val_acc:
0.976  (7.19s - 10.09s remaining)
EPOCH: 209.000  train_loss: 0.013  val_loss: 0.092  train_acc: 1.000  val_acc:
0.976  (7.22s - 10.05s remaining)
EPOCH: 210.000  train_loss: 0.013  val_loss: 0.091  train_acc: 1.000  val_acc:
0.976  (7.26s - 10.03s remaining)
EPOCH: 211.000  train_loss: 0.013  val_loss: 0.091  train_acc: 1.000  val_acc:
0.976  (7.29s - 9.99s remaining)
EPOCH: 212.000  train_loss: 0.012  val_loss: 0.091  train_acc: 1.000  val_acc:
0.976  (7.33s - 9.95s remaining)
EPOCH: 213.000  train_loss: 0.012  val_loss: 0.091  train_acc: 1.000  val_acc:
0.976  (7.36s - 9.91s remaining)
EPOCH: 214.000  train_loss: 0.012  val_loss: 0.090  train_acc: 1.000  val_acc:
0.976  (7.39s - 9.88s remaining)
EPOCH: 215.000  train_loss: 0.012  val_loss: 0.091  train_acc: 1.000  val_acc:
```

```
0.976  (7.42s - 9.84s remaining)
EPOCH: 216.000  train_loss: 0.012  val_loss: 0.091  train_acc: 1.000  val_acc:
0.976  (7.46s - 9.81s remaining)
EPOCH: 217.000  train_loss: 0.012  val_loss: 0.091  train_acc: 1.000  val_acc:
0.976  (7.50s - 9.78s remaining)
EPOCH: 218.000  train_loss: 0.012  val_loss: 0.090  train_acc: 1.000  val_acc:
0.977  (7.54s - 9.75s remaining)
EPOCH: 219.000  train_loss: 0.011  val_loss: 0.090  train_acc: 1.000  val_acc:
0.977  (7.57s - 9.71s remaining)
EPOCH: 220.000  train_loss: 0.011  val_loss: 0.089  train_acc: 1.000  val_acc:
0.977  (7.60s - 9.67s remaining)
EPOCH: 221.000  train_loss: 0.011  val_loss: 0.089  train_acc: 1.000  val_acc:
0.977  (7.63s - 9.63s remaining)
EPOCH: 222.000  train_loss: 0.011  val_loss: 0.088  train_acc: 1.000  val_acc:
0.976  (7.67s - 9.60s remaining)
EPOCH: 223.000  train_loss: 0.011  val_loss: 0.088  train_acc: 1.000  val_acc:
0.977  (7.70s - 9.57s remaining)
EPOCH: 224.000  train_loss: 0.011  val_loss: 0.088  train_acc: 1.000  val_acc:
0.977  (7.74s - 9.54s remaining)
EPOCH: 225.000  train_loss: 0.011  val_loss: 0.087  train_acc: 1.000  val_acc:
0.977  (7.77s - 9.50s remaining)
EPOCH: 226.000  train_loss: 0.011  val_loss: 0.087  train_acc: 1.000  val_acc:
0.976  (7.81s - 9.46s remaining)
EPOCH: 227.000  train_loss: 0.011  val_loss: 0.087  train_acc: 1.000  val_acc:
0.976  (7.84s - 9.43s remaining)
EPOCH: 228.000  train_loss: 0.010  val_loss: 0.088  train_acc: 1.000  val_acc:
0.976  (7.88s - 9.40s remaining)
EPOCH: 229.000  train_loss: 0.010  val_loss: 0.088  train_acc: 1.000  val_acc:
0.976  (7.91s - 9.36s remaining)
EPOCH: 230.000  train_loss: 0.010  val_loss: 0.088  train_acc: 1.000  val_acc:
0.976  (7.94s - 9.32s remaining)
EPOCH: 231.000  train_loss: 0.010  val_loss: 0.088  train_acc: 1.000  val_acc:
0.977  (7.98s - 9.30s remaining)
EPOCH: 232.000  train_loss: 0.010  val_loss: 0.088  train_acc: 1.000  val_acc:
0.977  (8.02s - 9.26s remaining)
EPOCH: 233.000  train_loss: 0.010  val_loss: 0.088  train_acc: 1.000  val_acc:
0.977  (8.05s - 9.23s remaining)
EPOCH: 234.000  train_loss: 0.010  val_loss: 0.087  train_acc: 1.000  val_acc:
0.977  (8.09s - 9.20s remaining)
EPOCH: 235.000  train_loss: 0.010  val_loss: 0.086  train_acc: 1.000  val_acc:
0.978  (8.12s - 9.16s remaining)
EPOCH: 236.000  train_loss: 0.010  val_loss: 0.085  train_acc: 1.000  val_acc:
0.978  (8.16s - 9.12s remaining)
EPOCH: 237.000  train_loss: 0.010  val_loss: 0.084  train_acc: 1.000  val_acc:
0.977  (8.19s - 9.09s remaining)
EPOCH: 238.000  train_loss: 0.009  val_loss: 0.084  train_acc: 1.000  val_acc:
0.977  (8.22s - 9.05s remaining)
EPOCH: 239.000  train_loss: 0.009  val_loss: 0.084  train_acc: 1.000  val_acc:
```

```
0.977  (8.25s - 9.01s remaining)
EPOCH: 240.000  train_loss: 0.009  val_loss: 0.084  train_acc: 1.000  val_acc:
0.978  (8.29s - 8.98s remaining)
EPOCH: 241.000  train_loss: 0.009  val_loss: 0.084  train_acc: 1.000  val_acc:
0.977  (8.32s - 8.95s remaining)
EPOCH: 242.000  train_loss: 0.009  val_loss: 0.085  train_acc: 1.000  val_acc:
0.977  (8.36s - 8.91s remaining)
EPOCH: 243.000  train_loss: 0.009  val_loss: 0.086  train_acc: 1.000  val_acc:
0.977  (8.39s - 8.87s remaining)
EPOCH: 244.000  train_loss: 0.009  val_loss: 0.087  train_acc: 1.000  val_acc:
0.979  (8.42s - 8.84s remaining)
EPOCH: 245.000  train_loss: 0.009  val_loss: 0.088  train_acc: 1.000  val_acc:
0.979  (8.46s - 8.81s remaining)
EPOCH: 246.000  train_loss: 0.009  val_loss: 0.088  train_acc: 1.000  val_acc:
0.978  (8.50s - 8.77s remaining)
EPOCH: 247.000  train_loss: 0.009  val_loss: 0.087  train_acc: 1.000  val_acc:
0.978  (8.53s - 8.74s remaining)
EPOCH: 248.000  train_loss: 0.009  val_loss: 0.085  train_acc: 1.000  val_acc:
0.979  (8.56s - 8.70s remaining)
EPOCH: 249.000  train_loss: 0.009  val_loss: 0.082  train_acc: 1.000  val_acc:
0.979  (8.60s - 8.67s remaining)
EPOCH: 250.000  train_loss: 0.009  val_loss: 0.080  train_acc: 1.000  val_acc:
0.976  (8.63s - 8.63s remaining)
EPOCH: 251.000  train_loss: 0.009  val_loss: 0.080  train_acc: 1.000  val_acc:
0.976  (8.66s - 8.59s remaining)
EPOCH: 252.000  train_loss: 0.008  val_loss: 0.080  train_acc: 1.000  val_acc:
0.978  (8.72s - 8.58s remaining)
EPOCH: 253.000  train_loss: 0.008  val_loss: 0.081  train_acc: 1.000  val_acc:
0.978  (8.76s - 8.55s remaining)
EPOCH: 254.000  train_loss: 0.008  val_loss: 0.084  train_acc: 1.000  val_acc:
0.978  (8.79s - 8.51s remaining)
EPOCH: 255.000  train_loss: 0.008  val_loss: 0.087  train_acc: 1.000  val_acc:
0.980  (8.82s - 8.48s remaining)
EPOCH: 256.000  train_loss: 0.008  val_loss: 0.086  train_acc: 1.000  val_acc:
0.979  (8.85s - 8.44s remaining)
EPOCH: 257.000  train_loss: 0.008  val_loss: 0.081  train_acc: 1.000  val_acc:
0.979  (8.89s - 8.41s remaining)
EPOCH: 258.000  train_loss: 0.008  val_loss: 0.078  train_acc: 1.000  val_acc:
0.977  (8.93s - 8.38s remaining)
EPOCH: 259.000  train_loss: 0.008  val_loss: 0.077  train_acc: 1.000  val_acc:
0.978  (8.97s - 8.34s remaining)
EPOCH: 260.000  train_loss: 0.008  val_loss: 0.079  train_acc: 1.000  val_acc:
0.978  (9.00s - 8.31s remaining)
EPOCH: 261.000  train_loss: 0.008  val_loss: 0.084  train_acc: 1.000  val_acc:
0.978  (9.03s - 8.27s remaining)
EPOCH: 262.000  train_loss: 0.007  val_loss: 0.083  train_acc: 1.000  val_acc:
0.979  (9.07s - 8.24s remaining)
EPOCH: 263.000  train_loss: 0.007  val_loss: 0.080  train_acc: 1.000  val_acc:
```

```
0.979  (9.12s - 8.22s remaining)
EPOCH: 264.000  train_loss: 0.007  val_loss: 0.077  train_acc: 1.000  val_acc:
0.978  (9.17s - 8.20s remaining)
EPOCH: 265.000  train_loss: 0.007  val_loss: 0.077  train_acc: 1.000  val_acc:
0.978  (9.21s - 8.17s remaining)
EPOCH: 266.000  train_loss: 0.007  val_loss: 0.080  train_acc: 1.000  val_acc:
0.977  (9.26s - 8.15s remaining)
EPOCH: 267.000  train_loss: 0.007  val_loss: 0.082  train_acc: 1.000  val_acc:
0.978  (9.32s - 8.13s remaining)
EPOCH: 268.000  train_loss: 0.007  val_loss: 0.080  train_acc: 1.000  val_acc:
0.979  (9.36s - 8.10s remaining)
EPOCH: 269.000  train_loss: 0.007  val_loss: 0.078  train_acc: 1.000  val_acc:
0.980  (9.40s - 8.08s remaining)
EPOCH: 270.000  train_loss: 0.007  val_loss: 0.077  train_acc: 1.000  val_acc:
0.979  (9.45s - 8.05s remaining)
EPOCH: 271.000  train_loss: 0.007  val_loss: 0.078  train_acc: 1.000  val_acc:
0.977  (9.49s - 8.02s remaining)
EPOCH: 272.000  train_loss: 0.007  val_loss: 0.080  train_acc: 1.000  val_acc:
0.977  (9.54s - 7.99s remaining)
EPOCH: 273.000  train_loss: 0.007  val_loss: 0.080  train_acc: 1.000  val_acc:
0.979  (9.58s - 7.96s remaining)
EPOCH: 274.000  train_loss: 0.007  val_loss: 0.079  train_acc: 1.000  val_acc:
0.979  (9.62s - 7.94s remaining)
EPOCH: 275.000  train_loss: 0.006  val_loss: 0.078  train_acc: 1.000  val_acc:
0.980  (9.66s - 7.91s remaining)
EPOCH: 276.000  train_loss: 0.006  val_loss: 0.077  train_acc: 1.000  val_acc:
0.979  (9.71s - 7.88s remaining)
EPOCH: 277.000  train_loss: 0.006  val_loss: 0.078  train_acc: 1.000  val_acc:
0.977  (9.76s - 7.86s remaining)
EPOCH: 278.000  train_loss: 0.006  val_loss: 0.079  train_acc: 1.000  val_acc:
0.977  (9.81s - 7.83s remaining)
EPOCH: 279.000  train_loss: 0.006  val_loss: 0.079  train_acc: 1.000  val_acc:
0.979  (9.86s - 7.81s remaining)
EPOCH: 280.000  train_loss: 0.006  val_loss: 0.079  train_acc: 1.000  val_acc:
0.979  (9.90s - 7.78s remaining)
EPOCH: 281.000  train_loss: 0.006  val_loss: 0.078  train_acc: 1.000  val_acc:
0.980  (9.96s - 7.76s remaining)
EPOCH: 282.000  train_loss: 0.006  val_loss: 0.077  train_acc: 1.000  val_acc:
0.979  (10.00s - 7.73s remaining)
EPOCH: 283.000  train_loss: 0.006  val_loss: 0.077  train_acc: 1.000  val_acc:
0.978  (10.04s - 7.70s remaining)
EPOCH: 284.000  train_loss: 0.006  val_loss: 0.078  train_acc: 1.000  val_acc:
0.977  (10.08s - 7.67s remaining)
EPOCH: 285.000  train_loss: 0.006  val_loss: 0.078  train_acc: 1.000  val_acc:
0.978  (10.14s - 7.65s remaining)
EPOCH: 286.000  train_loss: 0.006  val_loss: 0.078  train_acc: 1.000  val_acc:
0.979  (10.18s - 7.62s remaining)
EPOCH: 287.000  train_loss: 0.006  val_loss: 0.078  train_acc: 1.000  val_acc:
```

```
0.979  (10.23s - 7.59s remaining)
EPOCH: 288.000  train_loss: 0.006  val_loss: 0.077  train_acc: 1.000  val_acc:
0.980  (10.27s - 7.56s remaining)
EPOCH: 289.000  train_loss: 0.006  val_loss: 0.076  train_acc: 1.000  val_acc:
0.980  (10.31s - 7.53s remaining)
EPOCH: 290.000  train_loss: 0.006  val_loss: 0.077  train_acc: 1.000  val_acc:
0.978  (10.37s - 7.51s remaining)
EPOCH: 291.000  train_loss: 0.006  val_loss: 0.077  train_acc: 1.000  val_acc:
0.977  (10.41s - 7.48s remaining)
EPOCH: 292.000  train_loss: 0.006  val_loss: 0.078  train_acc: 1.000  val_acc:
0.979  (10.46s - 7.45s remaining)
EPOCH: 293.000  train_loss: 0.005  val_loss: 0.078  train_acc: 1.000  val_acc:
0.979  (10.51s - 7.42s remaining)
EPOCH: 294.000  train_loss: 0.005  val_loss: 0.077  train_acc: 1.000  val_acc:
0.979  (10.56s - 7.40s remaining)
EPOCH: 295.000  train_loss: 0.005  val_loss: 0.077  train_acc: 1.000  val_acc:
0.980  (10.60s - 7.37s remaining)
EPOCH: 296.000  train_loss: 0.005  val_loss: 0.076  train_acc: 1.000  val_acc:
0.980  (10.64s - 7.33s remaining)
EPOCH: 297.000  train_loss: 0.005  val_loss: 0.076  train_acc: 1.000  val_acc:
0.979  (10.69s - 7.30s remaining)
EPOCH: 298.000  train_loss: 0.005  val_loss: 0.076  train_acc: 1.000  val_acc:
0.978  (10.73s - 7.27s remaining)
EPOCH: 299.000  train_loss: 0.005  val_loss: 0.077  train_acc: 1.000  val_acc:
0.979  (10.78s - 7.25s remaining)
EPOCH: 300.000  train_loss: 0.005  val_loss: 0.077  train_acc: 1.000  val_acc:
0.979  (10.84s - 7.22s remaining)
EPOCH: 301.000  train_loss: 0.005  val_loss: 0.077  train_acc: 1.000  val_acc:
0.979  (10.88s - 7.20s remaining)
EPOCH: 302.000  train_loss: 0.005  val_loss: 0.077  train_acc: 1.000  val_acc:
0.980  (10.93s - 7.16s remaining)
EPOCH: 303.000  train_loss: 0.005  val_loss: 0.076  train_acc: 1.000  val_acc:
0.980  (10.98s - 7.14s remaining)
EPOCH: 304.000  train_loss: 0.005  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (11.02s - 7.11s remaining)
EPOCH: 305.000  train_loss: 0.005  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (11.07s - 7.08s remaining)
EPOCH: 306.000  train_loss: 0.005  val_loss: 0.076  train_acc: 1.000  val_acc:
0.979  (11.12s - 7.05s remaining)
EPOCH: 307.000  train_loss: 0.005  val_loss: 0.076  train_acc: 1.000  val_acc:
0.979  (11.18s - 7.03s remaining)
EPOCH: 308.000  train_loss: 0.005  val_loss: 0.077  train_acc: 1.000  val_acc:
0.979  (11.23s - 7.00s remaining)
EPOCH: 309.000  train_loss: 0.005  val_loss: 0.077  train_acc: 1.000  val_acc:
0.979  (11.28s - 6.97s remaining)
EPOCH: 310.000  train_loss: 0.005  val_loss: 0.076  train_acc: 1.000  val_acc:
0.979  (11.33s - 6.94s remaining)
EPOCH: 311.000  train_loss: 0.005  val_loss: 0.076  train_acc: 1.000  val_acc:
```

```
0.980  (11.38s - 6.92s remaining)
EPOCH: 312.000  train_loss: 0.005  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (11.43s - 6.89s remaining)
EPOCH: 313.000  train_loss: 0.005  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (11.47s - 6.85s remaining)
EPOCH: 314.000  train_loss: 0.005  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (11.52s - 6.82s remaining)
EPOCH: 315.000  train_loss: 0.005  val_loss: 0.075  train_acc: 1.000  val_acc:
0.979  (11.56s - 6.79s remaining)
EPOCH: 316.000  train_loss: 0.004  val_loss: 0.076  train_acc: 1.000  val_acc:
0.979  (11.61s - 6.76s remaining)
EPOCH: 317.000  train_loss: 0.004  val_loss: 0.076  train_acc: 1.000  val_acc:
0.979  (11.66s - 6.73s remaining)
EPOCH: 318.000  train_loss: 0.004  val_loss: 0.076  train_acc: 1.000  val_acc:
0.979  (11.70s - 6.70s remaining)
EPOCH: 319.000  train_loss: 0.004  val_loss: 0.076  train_acc: 1.000  val_acc:
0.979  (11.75s - 6.67s remaining)
EPOCH: 320.000  train_loss: 0.004  val_loss: 0.076  train_acc: 1.000  val_acc:
0.980  (11.80s - 6.64s remaining)
EPOCH: 321.000  train_loss: 0.004  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (11.85s - 6.61s remaining)
EPOCH: 322.000  train_loss: 0.004  val_loss: 0.074  train_acc: 1.000  val_acc:
0.980  (11.90s - 6.58s remaining)
EPOCH: 323.000  train_loss: 0.004  val_loss: 0.074  train_acc: 1.000  val_acc:
0.980  (11.95s - 6.55s remaining)
EPOCH: 324.000  train_loss: 0.004  val_loss: 0.074  train_acc: 1.000  val_acc:
0.979  (11.99s - 6.52s remaining)
EPOCH: 325.000  train_loss: 0.004  val_loss: 0.075  train_acc: 1.000  val_acc:
0.979  (12.03s - 6.48s remaining)
EPOCH: 326.000  train_loss: 0.004  val_loss: 0.075  train_acc: 1.000  val_acc:
0.979  (12.06s - 6.44s remaining)
EPOCH: 327.000  train_loss: 0.004  val_loss: 0.076  train_acc: 1.000  val_acc:
0.979  (12.10s - 6.40s remaining)
EPOCH: 328.000  train_loss: 0.004  val_loss: 0.076  train_acc: 1.000  val_acc:
0.979  (12.13s - 6.36s remaining)
EPOCH: 329.000  train_loss: 0.004  val_loss: 0.076  train_acc: 1.000  val_acc:
0.980  (12.17s - 6.33s remaining)
EPOCH: 330.000  train_loss: 0.004  val_loss: 0.076  train_acc: 1.000  val_acc:
0.980  (12.21s - 6.29s remaining)
EPOCH: 331.000  train_loss: 0.004  val_loss: 0.074  train_acc: 1.000  val_acc:
0.980  (12.24s - 6.25s remaining)
EPOCH: 332.000  train_loss: 0.004  val_loss: 0.073  train_acc: 1.000  val_acc:
0.978  (12.28s - 6.21s remaining)
EPOCH: 333.000  train_loss: 0.004  val_loss: 0.073  train_acc: 1.000  val_acc:
0.978  (12.31s - 6.17s remaining)
EPOCH: 334.000  train_loss: 0.004  val_loss: 0.073  train_acc: 1.000  val_acc:
0.977  (12.34s - 6.13s remaining)
EPOCH: 335.000  train_loss: 0.004  val_loss: 0.074  train_acc: 1.000  val_acc:
```

```
0.979  (12.38s - 6.10s remaining)
EPOCH: 336.000  train_loss: 0.004  val_loss: 0.075  train_acc: 1.000  val_acc:
0.978  (12.42s - 6.06s remaining)
EPOCH: 337.000  train_loss: 0.004  val_loss: 0.076  train_acc: 1.000  val_acc:
0.979  (12.45s - 6.02s remaining)
EPOCH: 338.000  train_loss: 0.004  val_loss: 0.077  train_acc: 1.000  val_acc:
0.979  (12.48s - 5.98s remaining)
EPOCH: 339.000  train_loss: 0.004  val_loss: 0.077  train_acc: 1.000  val_acc:
0.980  (12.51s - 5.94s remaining)
EPOCH: 340.000  train_loss: 0.004  val_loss: 0.076  train_acc: 1.000  val_acc:
0.981  (12.55s - 5.90s remaining)
EPOCH: 341.000  train_loss: 0.004  val_loss: 0.075  train_acc: 1.000  val_acc:
0.981  (12.59s - 5.87s remaining)
EPOCH: 342.000  train_loss: 0.004  val_loss: 0.073  train_acc: 1.000  val_acc:
0.978  (12.63s - 5.83s remaining)
EPOCH: 343.000  train_loss: 0.004  val_loss: 0.072  train_acc: 1.000  val_acc:
0.977  (12.66s - 5.80s remaining)
EPOCH: 344.000  train_loss: 0.004  val_loss: 0.071  train_acc: 1.000  val_acc:
0.976  (12.69s - 5.76s remaining)
EPOCH: 345.000  train_loss: 0.004  val_loss: 0.072  train_acc: 1.000  val_acc:
0.978  (12.73s - 5.72s remaining)
EPOCH: 346.000  train_loss: 0.004  val_loss: 0.074  train_acc: 1.000  val_acc:
0.978  (12.76s - 5.68s remaining)
EPOCH: 347.000  train_loss: 0.004  val_loss: 0.077  train_acc: 1.000  val_acc:
0.979  (12.79s - 5.64s remaining)
EPOCH: 348.000  train_loss: 0.004  val_loss: 0.078  train_acc: 1.000  val_acc:
0.981  (12.84s - 5.61s remaining)
EPOCH: 349.000  train_loss: 0.004  val_loss: 0.078  train_acc: 1.000  val_acc:
0.980  (12.87s - 5.57s remaining)
EPOCH: 350.000  train_loss: 0.004  val_loss: 0.076  train_acc: 1.000  val_acc:
0.981  (12.91s - 5.53s remaining)
EPOCH: 351.000  train_loss: 0.004  val_loss: 0.073  train_acc: 1.000  val_acc:
0.979  (12.94s - 5.49s remaining)
EPOCH: 352.000  train_loss: 0.003  val_loss: 0.071  train_acc: 1.000  val_acc:
0.978  (12.98s - 5.46s remaining)
EPOCH: 353.000  train_loss: 0.003  val_loss: 0.069  train_acc: 1.000  val_acc:
0.979  (13.02s - 5.42s remaining)
EPOCH: 354.000  train_loss: 0.003  val_loss: 0.071  train_acc: 1.000  val_acc:
0.979  (13.05s - 5.38s remaining)
EPOCH: 355.000  train_loss: 0.003  val_loss: 0.074  train_acc: 1.000  val_acc:
0.977  (13.09s - 5.34s remaining)
EPOCH: 356.000  train_loss: 0.003  val_loss: 0.078  train_acc: 1.000  val_acc:
0.980  (13.13s - 5.31s remaining)
EPOCH: 357.000  train_loss: 0.003  val_loss: 0.078  train_acc: 1.000  val_acc:
0.980  (13.16s - 5.27s remaining)
EPOCH: 358.000  train_loss: 0.003  val_loss: 0.075  train_acc: 1.000  val_acc:
0.981  (13.20s - 5.24s remaining)
EPOCH: 359.000  train_loss: 0.003  val_loss: 0.072  train_acc: 1.000  val_acc:
```

```
0.978  (13.24s - 5.20s remaining)
EPOCH: 360.000  train_loss: 0.003  val_loss: 0.069  train_acc: 1.000  val_acc:
0.980  (13.28s - 5.16s remaining)
EPOCH: 361.000  train_loss: 0.003  val_loss: 0.070  train_acc: 1.000  val_acc:
0.979  (13.31s - 5.13s remaining)
EPOCH: 362.000  train_loss: 0.003  val_loss: 0.075  train_acc: 1.000  val_acc:
0.977  (13.35s - 5.09s remaining)
EPOCH: 363.000  train_loss: 0.003  val_loss: 0.077  train_acc: 1.000  val_acc:
0.981  (13.38s - 5.05s remaining)
EPOCH: 364.000  train_loss: 0.003  val_loss: 0.075  train_acc: 1.000  val_acc:
0.981  (13.41s - 5.01s remaining)
EPOCH: 365.000  train_loss: 0.003  val_loss: 0.072  train_acc: 1.000  val_acc:
0.979  (13.45s - 4.98s remaining)
EPOCH: 366.000  train_loss: 0.003  val_loss: 0.070  train_acc: 1.000  val_acc:
0.979  (13.49s - 4.94s remaining)
EPOCH: 367.000  train_loss: 0.003  val_loss: 0.071  train_acc: 1.000  val_acc:
0.981  (13.54s - 4.91s remaining)
EPOCH: 368.000  train_loss: 0.003  val_loss: 0.074  train_acc: 1.000  val_acc:
0.979  (13.59s - 4.87s remaining)
EPOCH: 369.000  train_loss: 0.003  val_loss: 0.076  train_acc: 1.000  val_acc:
0.980  (13.64s - 4.84s remaining)
EPOCH: 370.000  train_loss: 0.003  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (13.69s - 4.81s remaining)
EPOCH: 371.000  train_loss: 0.003  val_loss: 0.073  train_acc: 1.000  val_acc:
0.979  (13.73s - 4.78s remaining)
EPOCH: 372.000  train_loss: 0.003  val_loss: 0.071  train_acc: 1.000  val_acc:
0.978  (13.79s - 4.74s remaining)
EPOCH: 373.000  train_loss: 0.003  val_loss: 0.071  train_acc: 1.000  val_acc:
0.980  (13.86s - 4.72s remaining)
EPOCH: 374.000  train_loss: 0.003  val_loss: 0.073  train_acc: 1.000  val_acc:
0.979  (13.92s - 4.69s remaining)
EPOCH: 375.000  train_loss: 0.003  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (13.97s - 4.66s remaining)
EPOCH: 376.000  train_loss: 0.003  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (14.00s - 4.62s remaining)
EPOCH: 377.000  train_loss: 0.003  val_loss: 0.073  train_acc: 1.000  val_acc:
0.980  (14.03s - 4.58s remaining)
EPOCH: 378.000  train_loss: 0.003  val_loss: 0.072  train_acc: 1.000  val_acc:
0.978  (14.07s - 4.54s remaining)
EPOCH: 379.000  train_loss: 0.003  val_loss: 0.072  train_acc: 1.000  val_acc:
0.979  (14.11s - 4.50s remaining)
EPOCH: 380.000  train_loss: 0.003  val_loss: 0.073  train_acc: 1.000  val_acc:
0.979  (14.14s - 4.47s remaining)
EPOCH: 381.000  train_loss: 0.003  val_loss: 0.074  train_acc: 1.000  val_acc:
0.979  (14.18s - 4.43s remaining)
EPOCH: 382.000  train_loss: 0.003  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (14.21s - 4.39s remaining)
EPOCH: 383.000  train_loss: 0.003  val_loss: 0.074  train_acc: 1.000  val_acc:
```

```
0.981  (14.25s - 4.35s remaining)
EPOCH: 384.000  train_loss: 0.003  val_loss: 0.073  train_acc: 1.000  val_acc:
0.980  (14.29s - 4.32s remaining)
EPOCH: 385.000  train_loss: 0.003  val_loss: 0.072  train_acc: 1.000  val_acc:
0.978  (14.32s - 4.28s remaining)
EPOCH: 386.000  train_loss: 0.003  val_loss: 0.072  train_acc: 1.000  val_acc:
0.979  (14.36s - 4.24s remaining)
EPOCH: 387.000  train_loss: 0.003  val_loss: 0.073  train_acc: 1.000  val_acc:
0.979  (14.40s - 4.20s remaining)
EPOCH: 388.000  train_loss: 0.003  val_loss: 0.074  train_acc: 1.000  val_acc:
0.980  (14.43s - 4.17s remaining)
EPOCH: 389.000  train_loss: 0.003  val_loss: 0.074  train_acc: 1.000  val_acc:
0.980  (14.47s - 4.13s remaining)
EPOCH: 390.000  train_loss: 0.003  val_loss: 0.074  train_acc: 1.000  val_acc:
0.981  (14.51s - 4.09s remaining)
EPOCH: 391.000  train_loss: 0.003  val_loss: 0.073  train_acc: 1.000  val_acc:
0.980  (14.54s - 4.05s remaining)
EPOCH: 392.000  train_loss: 0.003  val_loss: 0.072  train_acc: 1.000  val_acc:
0.979  (14.57s - 4.02s remaining)
EPOCH: 393.000  train_loss: 0.002  val_loss: 0.072  train_acc: 1.000  val_acc:
0.979  (14.61s - 3.98s remaining)
EPOCH: 394.000  train_loss: 0.002  val_loss: 0.073  train_acc: 1.000  val_acc:
0.980  (14.64s - 3.94s remaining)
EPOCH: 395.000  train_loss: 0.002  val_loss: 0.074  train_acc: 1.000  val_acc:
0.979  (14.67s - 3.90s remaining)
EPOCH: 396.000  train_loss: 0.002  val_loss: 0.074  train_acc: 1.000  val_acc:
0.980  (14.71s - 3.86s remaining)
EPOCH: 397.000  train_loss: 0.002  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (14.74s - 3.83s remaining)
EPOCH: 398.000  train_loss: 0.002  val_loss: 0.074  train_acc: 1.000  val_acc:
0.981  (14.78s - 3.79s remaining)
EPOCH: 399.000  train_loss: 0.002  val_loss: 0.073  train_acc: 1.000  val_acc:
0.980  (14.82s - 3.75s remaining)
EPOCH: 400.000  train_loss: 0.002  val_loss: 0.072  train_acc: 1.000  val_acc:
0.979  (14.85s - 3.71s remaining)
EPOCH: 401.000  train_loss: 0.002  val_loss: 0.072  train_acc: 1.000  val_acc:
0.979  (14.89s - 3.67s remaining)
EPOCH: 402.000  train_loss: 0.002  val_loss: 0.073  train_acc: 1.000  val_acc:
0.980  (14.92s - 3.64s remaining)
EPOCH: 403.000  train_loss: 0.002  val_loss: 0.074  train_acc: 1.000  val_acc:
0.980  (14.96s - 3.60s remaining)
EPOCH: 404.000  train_loss: 0.002  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (15.00s - 3.56s remaining)
EPOCH: 405.000  train_loss: 0.002  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (15.03s - 3.53s remaining)
EPOCH: 406.000  train_loss: 0.002  val_loss: 0.074  train_acc: 1.000  val_acc:
0.981  (15.07s - 3.49s remaining)
EPOCH: 407.000  train_loss: 0.002  val_loss: 0.073  train_acc: 1.000  val_acc:
```

```
0.980  (15.11s - 3.45s remaining)
EPOCH: 408.000  train_loss: 0.002  val_loss: 0.072  train_acc: 1.000  val_acc:
0.979  (15.14s - 3.41s remaining)
EPOCH: 409.000  train_loss: 0.002  val_loss: 0.072  train_acc: 1.000  val_acc:
0.979  (15.18s - 3.38s remaining)
EPOCH: 410.000  train_loss: 0.002  val_loss: 0.073  train_acc: 1.000  val_acc:
0.980  (15.21s - 3.34s remaining)
EPOCH: 411.000  train_loss: 0.002  val_loss: 0.074  train_acc: 1.000  val_acc:
0.979  (15.24s - 3.30s remaining)
EPOCH: 412.000  train_loss: 0.002  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (15.28s - 3.26s remaining)
EPOCH: 413.000  train_loss: 0.002  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (15.33s - 3.23s remaining)
EPOCH: 414.000  train_loss: 0.002  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (15.36s - 3.19s remaining)
EPOCH: 415.000  train_loss: 0.002  val_loss: 0.074  train_acc: 1.000  val_acc:
0.980  (15.39s - 3.15s remaining)
EPOCH: 416.000  train_loss: 0.002  val_loss: 0.073  train_acc: 1.000  val_acc:
0.979  (15.43s - 3.11s remaining)
EPOCH: 417.000  train_loss: 0.002  val_loss: 0.072  train_acc: 1.000  val_acc:
0.978  (15.46s - 3.08s remaining)
EPOCH: 418.000  train_loss: 0.002  val_loss: 0.072  train_acc: 1.000  val_acc:
0.979  (15.49s - 3.04s remaining)
EPOCH: 419.000  train_loss: 0.002  val_loss: 0.073  train_acc: 1.000  val_acc:
0.979  (15.54s - 3.00s remaining)
EPOCH: 420.000  train_loss: 0.002  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (15.58s - 2.97s remaining)
EPOCH: 421.000  train_loss: 0.002  val_loss: 0.076  train_acc: 1.000  val_acc:
0.980  (15.61s - 2.93s remaining)
EPOCH: 422.000  train_loss: 0.002  val_loss: 0.076  train_acc: 1.000  val_acc:
0.980  (15.65s - 2.89s remaining)
EPOCH: 423.000  train_loss: 0.002  val_loss: 0.075  train_acc: 1.000  val_acc:
0.981  (15.68s - 2.85s remaining)
EPOCH: 424.000  train_loss: 0.002  val_loss: 0.074  train_acc: 1.000  val_acc:
0.980  (15.71s - 2.82s remaining)
EPOCH: 425.000  train_loss: 0.002  val_loss: 0.072  train_acc: 1.000  val_acc:
0.980  (15.75s - 2.78s remaining)
EPOCH: 426.000  train_loss: 0.002  val_loss: 0.071  train_acc: 1.000  val_acc:
0.980  (15.79s - 2.74s remaining)
EPOCH: 427.000  train_loss: 0.002  val_loss: 0.072  train_acc: 1.000  val_acc:
0.982  (15.82s - 2.70s remaining)
EPOCH: 428.000  train_loss: 0.002  val_loss: 0.073  train_acc: 1.000  val_acc:
0.980  (15.86s - 2.67s remaining)
EPOCH: 429.000  train_loss: 0.002  val_loss: 0.076  train_acc: 1.000  val_acc:
0.980  (15.90s - 2.63s remaining)
EPOCH: 430.000  train_loss: 0.002  val_loss: 0.077  train_acc: 1.000  val_acc:
0.979  (15.94s - 2.59s remaining)
EPOCH: 431.000  train_loss: 0.002  val_loss: 0.077  train_acc: 1.000  val_acc:
```

```
0.979  (15.98s - 2.56s remaining)
EPOCH: 432.000  train_loss: 0.002  val_loss: 0.075  train_acc: 1.000  val_acc:
0.981  (16.01s - 2.52s remaining)
EPOCH: 433.000  train_loss: 0.002  val_loss: 0.073  train_acc: 1.000  val_acc:
0.979  (16.04s - 2.48s remaining)
EPOCH: 434.000  train_loss: 0.002  val_loss: 0.071  train_acc: 1.000  val_acc:
0.980  (16.08s - 2.44s remaining)
EPOCH: 435.000  train_loss: 0.002  val_loss: 0.070  train_acc: 1.000  val_acc:
0.980  (16.11s - 2.41s remaining)
EPOCH: 436.000  train_loss: 0.002  val_loss: 0.072  train_acc: 1.000  val_acc:
0.981  (16.15s - 2.37s remaining)
EPOCH: 437.000  train_loss: 0.002  val_loss: 0.075  train_acc: 1.000  val_acc:
0.979  (16.19s - 2.33s remaining)
EPOCH: 438.000  train_loss: 0.002  val_loss: 0.078  train_acc: 1.000  val_acc:
0.979  (16.22s - 2.30s remaining)
EPOCH: 439.000  train_loss: 0.002  val_loss: 0.078  train_acc: 1.000  val_acc:
0.979  (16.25s - 2.26s remaining)
EPOCH: 440.000  train_loss: 0.002  val_loss: 0.076  train_acc: 1.000  val_acc:
0.980  (16.29s - 2.22s remaining)
EPOCH: 441.000  train_loss: 0.002  val_loss: 0.073  train_acc: 1.000  val_acc:
0.980  (16.32s - 2.18s remaining)
EPOCH: 442.000  train_loss: 0.002  val_loss: 0.071  train_acc: 1.000  val_acc:
0.980  (16.37s - 2.15s remaining)
EPOCH: 443.000  train_loss: 0.002  val_loss: 0.070  train_acc: 1.000  val_acc:
0.981  (16.40s - 2.11s remaining)
EPOCH: 444.000  train_loss: 0.002  val_loss: 0.073  train_acc: 1.000  val_acc:
0.981  (16.44s - 2.07s remaining)
EPOCH: 445.000  train_loss: 0.002  val_loss: 0.077  train_acc: 1.000  val_acc:
0.980  (16.47s - 2.04s remaining)
EPOCH: 446.000  train_loss: 0.002  val_loss: 0.078  train_acc: 1.000  val_acc:
0.979  (16.51s - 2.00s remaining)
EPOCH: 447.000  train_loss: 0.002  val_loss: 0.076  train_acc: 1.000  val_acc:
0.980  (16.55s - 1.96s remaining)
EPOCH: 448.000  train_loss: 0.002  val_loss: 0.073  train_acc: 1.000  val_acc:
0.979  (16.58s - 1.92s remaining)
EPOCH: 449.000  train_loss: 0.002  val_loss: 0.070  train_acc: 1.000  val_acc:
0.980  (16.61s - 1.89s remaining)
EPOCH: 450.000  train_loss: 0.002  val_loss: 0.070  train_acc: 1.000  val_acc:
0.981  (16.65s - 1.85s remaining)
EPOCH: 451.000  train_loss: 0.002  val_loss: 0.074  train_acc: 1.000  val_acc:
0.981  (16.68s - 1.81s remaining)
EPOCH: 452.000  train_loss: 0.002  val_loss: 0.077  train_acc: 1.000  val_acc:
0.979  (16.71s - 1.77s remaining)
EPOCH: 453.000  train_loss: 0.002  val_loss: 0.076  train_acc: 1.000  val_acc:
0.980  (16.75s - 1.74s remaining)
EPOCH: 454.000  train_loss: 0.002  val_loss: 0.074  train_acc: 1.000  val_acc:
0.980  (16.79s - 1.70s remaining)
EPOCH: 455.000  train_loss: 0.002  val_loss: 0.071  train_acc: 1.000  val_acc:
```

```
0.980  (16.82s - 1.66s remaining)
EPOCH: 456.000  train_loss: 0.002  val_loss: 0.071  train_acc: 1.000  val_acc:
0.981  (16.86s - 1.63s remaining)
EPOCH: 457.000  train_loss: 0.002  val_loss: 0.073  train_acc: 1.000  val_acc:
0.982  (16.90s - 1.59s remaining)
EPOCH: 458.000  train_loss: 0.002  val_loss: 0.076  train_acc: 1.000  val_acc:
0.980  (16.93s - 1.55s remaining)
EPOCH: 459.000  train_loss: 0.002  val_loss: 0.076  train_acc: 1.000  val_acc:
0.980  (16.98s - 1.52s remaining)
EPOCH: 460.000  train_loss: 0.002  val_loss: 0.074  train_acc: 1.000  val_acc:
0.981  (17.02s - 1.48s remaining)
EPOCH: 461.000  train_loss: 0.002  val_loss: 0.072  train_acc: 1.000  val_acc:
0.982  (17.05s - 1.44s remaining)
EPOCH: 462.000  train_loss: 0.002  val_loss: 0.072  train_acc: 1.000  val_acc:
0.983  (17.09s - 1.41s remaining)
EPOCH: 463.000  train_loss: 0.002  val_loss: 0.073  train_acc: 1.000  val_acc:
0.982  (17.12s - 1.37s remaining)
EPOCH: 464.000  train_loss: 0.002  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (17.16s - 1.33s remaining)
EPOCH: 465.000  train_loss: 0.002  val_loss: 0.076  train_acc: 1.000  val_acc:
0.980  (17.20s - 1.29s remaining)
EPOCH: 466.000  train_loss: 0.002  val_loss: 0.074  train_acc: 1.000  val_acc:
0.982  (17.23s - 1.26s remaining)
EPOCH: 467.000  train_loss: 0.002  val_loss: 0.073  train_acc: 1.000  val_acc:
0.981  (17.27s - 1.22s remaining)
EPOCH: 468.000  train_loss: 0.002  val_loss: 0.072  train_acc: 1.000  val_acc:
0.983  (17.30s - 1.18s remaining)
EPOCH: 469.000  train_loss: 0.002  val_loss: 0.073  train_acc: 1.000  val_acc:
0.982  (17.34s - 1.15s remaining)
EPOCH: 470.000  train_loss: 0.001  val_loss: 0.074  train_acc: 1.000  val_acc:
0.980  (17.38s - 1.11s remaining)
EPOCH: 471.000  train_loss: 0.001  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (17.42s - 1.07s remaining)
EPOCH: 472.000  train_loss: 0.001  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (17.45s - 1.04s remaining)
EPOCH: 473.000  train_loss: 0.001  val_loss: 0.074  train_acc: 1.000  val_acc:
0.982  (17.48s - 1.00s remaining)
EPOCH: 474.000  train_loss: 0.001  val_loss: 0.073  train_acc: 1.000  val_acc:
0.984  (17.52s - 0.96s remaining)
EPOCH: 475.000  train_loss: 0.001  val_loss: 0.072  train_acc: 1.000  val_acc:
0.984  (17.55s - 0.92s remaining)
EPOCH: 476.000  train_loss: 0.001  val_loss: 0.074  train_acc: 1.000  val_acc:
0.982  (17.59s - 0.89s remaining)
EPOCH: 477.000  train_loss: 0.001  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (17.63s - 0.85s remaining)
EPOCH: 478.000  train_loss: 0.001  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (17.66s - 0.81s remaining)
EPOCH: 479.000  train_loss: 0.001  val_loss: 0.075  train_acc: 1.000  val_acc:
```

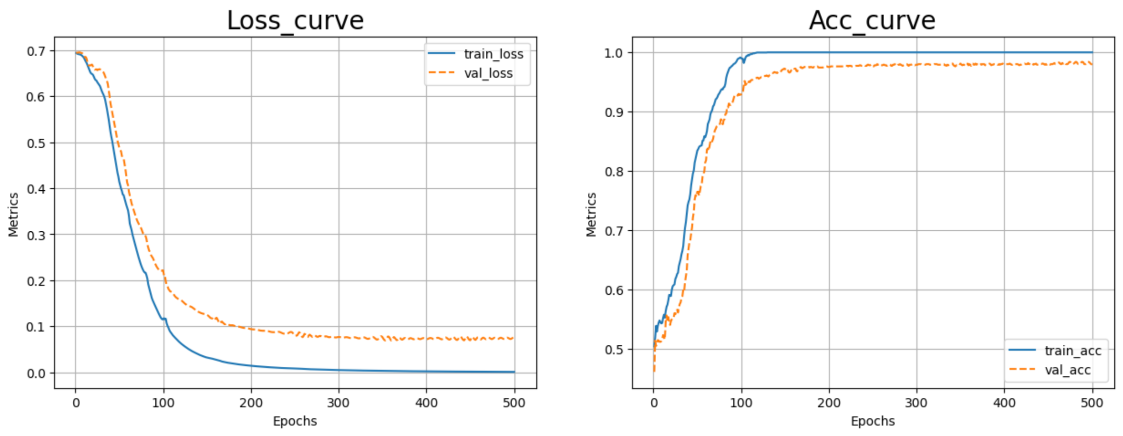
```
0.981  (17.69s - 0.78s remaining)
EPOCH: 480.000  train_loss: 0.001  val_loss: 0.073  train_acc: 1.000  val_acc:
0.982  (17.72s - 0.74s remaining)
EPOCH: 481.000  train_loss: 0.001  val_loss: 0.073  train_acc: 1.000  val_acc:
0.984  (17.77s - 0.70s remaining)
EPOCH: 482.000  train_loss: 0.001  val_loss: 0.073  train_acc: 1.000  val_acc:
0.984  (17.81s - 0.67s remaining)
EPOCH: 483.000  train_loss: 0.001  val_loss: 0.074  train_acc: 1.000  val_acc:
0.982  (17.84s - 0.63s remaining)
EPOCH: 484.000  train_loss: 0.001  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (17.88s - 0.59s remaining)
EPOCH: 485.000  train_loss: 0.001  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (17.91s - 0.55s remaining)
EPOCH: 486.000  train_loss: 0.001  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (17.94s - 0.52s remaining)
EPOCH: 487.000  train_loss: 0.001  val_loss: 0.074  train_acc: 1.000  val_acc:
0.982  (17.98s - 0.48s remaining)
EPOCH: 488.000  train_loss: 0.001  val_loss: 0.073  train_acc: 1.000  val_acc:
0.984  (18.03s - 0.44s remaining)
EPOCH: 489.000  train_loss: 0.001  val_loss: 0.073  train_acc: 1.000  val_acc:
0.984  (18.06s - 0.41s remaining)
EPOCH: 490.000  train_loss: 0.001  val_loss: 0.074  train_acc: 1.000  val_acc:
0.983  (18.10s - 0.37s remaining)
EPOCH: 491.000  train_loss: 0.001  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (18.14s - 0.33s remaining)
EPOCH: 492.000  train_loss: 0.001  val_loss: 0.076  train_acc: 1.000  val_acc:
0.980  (18.17s - 0.30s remaining)
EPOCH: 493.000  train_loss: 0.001  val_loss: 0.075  train_acc: 1.000  val_acc:
0.980  (18.22s - 0.26s remaining)
EPOCH: 494.000  train_loss: 0.001  val_loss: 0.074  train_acc: 1.000  val_acc:
0.981  (18.26s - 0.22s remaining)
EPOCH: 495.000  train_loss: 0.001  val_loss: 0.073  train_acc: 1.000  val_acc:
0.984  (18.29s - 0.18s remaining)
EPOCH: 496.000  train_loss: 0.001  val_loss: 0.073  train_acc: 1.000  val_acc:
0.984  (18.32s - 0.15s remaining)
EPOCH: 497.000  train_loss: 0.001  val_loss: 0.073  train_acc: 1.000  val_acc:
0.983  (18.36s - 0.11s remaining)
EPOCH: 498.000  train_loss: 0.001  val_loss: 0.075  train_acc: 1.000  val_acc:
0.981  (18.39s - 0.07s remaining)
EPOCH: 499.000  train_loss: 0.001  val_loss: 0.076  train_acc: 1.000  val_acc:
0.980  (18.43s - 0.04s remaining)
EPOCH: 500.000  train_loss: 0.001  val_loss: 0.076  train_acc: 1.000  val_acc:
0.979  (18.47s - 0.00s remaining)
```

[48]:
```
lstm_results= {"train_acc": train_accuracy, "val_acc": val_accuracy,␣
 ↪"train_loss": train_loss, "val_loss": val_loss, "epochs": 5}
```

##**2.4. Plotting the loss and acc curves**

```
[49]: fig, ax= plt.subplots(ncols=2, figsize=(15,5)) #w,h
      log.plot_epochs(["train_loss", "val_loss"], ax=ax[0], title="Loss_curve");␣
       ↪#loss_curve
      ax[0].legend(loc='upper right', fontsize=10)
      log.plot_epochs(["train_acc", "val_acc"], ax=ax[1], title="Acc_curve");␣
       ↪#acc_curve
```

```
100%|      | 601/601 [00:00<00:00, 1450.76it/s]
WARNING:matplotlib.legend:No artists with labels found to put in legend.  Note
that artists whose label start with an underscore are ignored when legend() is
called with no argument.
100%|      | 601/601 [00:00<00:00, 1257.36it/s]
```



##2.5. Testing on a new datapoint

```
[50]: x=gen_bin(10)
      inp, tar= input_prep(x[45], x[64])
      x[45], x[64], inp, tar
```

```
[50]: ('0b10000101101',
       '0b10001000000',
       tensor([[1., 0.],
               [0., 0.],
               [1., 0.],
               [1., 0.],
               [0., 0.],
               [1., 0.],
               [0., 1.],
               [0., 0.],
               [0., 0.],
               [0., 0.]]),
       tensor([1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0]))
```

[51]:
```
inp=torch.unsqueeze(inp, dim=0)
inp.shape
```

[51]:
```
torch.Size([1, 10, 2])
```

[54]:
```
torch.round(lstm(inp.to(device)))
```

[54]:
```
tensor([[1., 0., 1., 1., 1., 0., 1., 0., 0., 0., 0.]], device='cuda:0',
       grad_fn=<RoundBackward0>)
```

We can see that the rnn network performs very well on the test data.

##**2.6. Plotting the acc as a function of N**

[55]:
```
def experiments(lst_len_of_numbers, epochs):
  train_accs, test_accs=[], []
  input_size=2
  hidden_size=64
  num_layers=2
  for N in lst_len_of_numbers:
    train_ds= binary_string_dataset(32*10, N)
    test_ds= binary_string_dataset(32*3, N)
    train_dl= torch.utils.data.DataLoader(train_ds, batch_size=32)
    test_dl= torch.utils.data.DataLoader(test_ds, batch_size=32)

    sequence_length= N
    num_classes= N

    lstm= LSTM(input_size, hidden_size, num_layers, num_classes).to(device)
    EPOCHS=epochs
    criterion= lstm.loss_and_accuracy
    optimizer=torch.optim.Adam(params= lstm.parameters())
    log= Report(EPOCHS)
    for epoch in range(EPOCHS):
      n=len(train_dl)
      for ix, input in enumerate(train_dl):
        train_loss, train_accuracy=train_epoch(lstm, input, criterion,␣
↪optimizer, sequence_length=sequence_length)
        log.record(epoch+(ix+1)/n, train_loss=train_loss, train_acc=␣
↪train_accuracy, end="\r")
      n=len(test_dl)
      for ix, input in enumerate(test_dl):
        val_loss, val_accuracy=val_epoch(lstm, input, criterion,␣
↪sequence_length=sequence_length)
        log.record(epoch+(ix+1)/n, val_loss=val_loss, val_acc= val_accuracy,␣
↪end="\r")
    train_accs.append(train_accuracy)
    test_accs.append(val_accuracy)
    return train_accs, test_accs
```

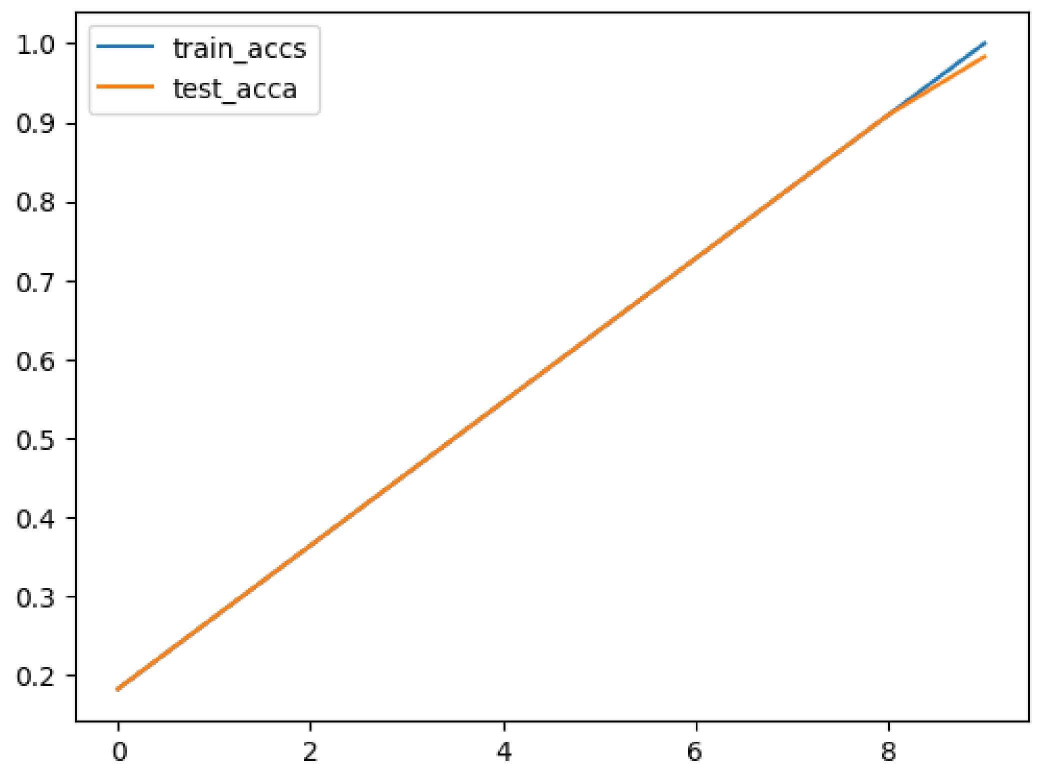
[56]:
```
lst_len_of_numbers=np.arange(1, 11)
epochs=500
train_accs, test_accs= experiments(lst_len_of_numbers, epochs)
```

```
EPOCH: 500.000  val_loss: 0.098  val_acc: 0.983  (17.48s - 0.00s remaining)
```

[57]:
```
train_accs, test_accs
```

[57]:
```
([0.1818181872367859,
  0.27272728085517883,
  0.3636363744735718,
  0.4545454680919647,
  0.5454545617103577,
  0.6363636255264282,
  0.7272727489471436,
  0.8181818723678589,
  0.9090909361839294,
  1.0],
 [0.1818181872367859,
  0.27272728085517883,
  0.3636363744735718,
  0.4545454680919647,
  0.5454545617103577,
  0.6363636255264282,
  0.7272727489471436,
  0.8181818723678589,
  0.9090909361839294,
  0.9829545617103577])
```

[58]:
```
plt.plot(train_accs, label="train_accs")
plt.plot(test_accs, label="test_acca")
plt.legend()
plt.show();
```



For the given hyperparameters we see a linearly increasing graph, it can also be seen that the test_accuracy starts lagging neard the tip end.