yagna_kaasaragadda_deep_learning_assignment_4_problem1

October 17, 2023

Transfer Learning:

Transfer learning is a technique used in deep learning by using the pretrained model for training on a new dataset instead of starting from scratch.

[ ]:

```
import torch
import os
from PIL import Image
from torchvision import transforms
from torchvision.datasets import DatasetFolder
import cv2
import numpy as np
```

[ ]:

```
from torchvision import transforms

transform = transforms.Compose([
  transforms.Resize(256),
  transforms.CenterCrop(224),
  transforms.ToTensor(),
  transforms.Normalize(
      mean=[0.485, 0.456, 0.406],
      std=[0.229, 0.224, 0.225])])

def load_image(img_path:str):
        np_img = cv2.imread(img_path) #CV2 to open and convert BMP mages into␣
 ↪NUMPY
        #np_img_gray = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        return Image.fromarray(np_img)  #we need Image for the transforms to␣
 ↪work correctly


dset = DatasetFolder(root='RowanDLclassNEA/NEUdata', loader = load_image,␣
 ↪extensions = ('.bmp',), transform = transform)
```

[ ]:

```
from torch.utils.data import random_split

train_set, val_set = random_split(
                      dset,
                      [1200, 600])

trainloader = torch.utils.data.DataLoader(
                    train_set,
                    batch_size=16,
                    shuffle=True)

valloader = torch.utils.data.DataLoader(
                    val_set,
                    batch_size=16,
                    shuffle=True)
```

[ ]:
```
import torchvision.models as models
import torch.nn as nn
from torchinfo import summary

resnet_no_weights = models.resnet18() # from neaclass1
resnet_weights = models.resnet18(weights = models.ResNet18_Weights.
 ↪IMAGENET1K_V1) # Used Resnet weights trained on ImageNet1k from neaclass2
```

[ ]:
```
layers = list(resnet_no_weights.children())[:-1]  #get all the layers except␣
 ↪the last one
layers.append(nn.Flatten())
layers.append(nn.Linear(512,6))
no_weights_resnet = nn.Sequential(*layers)

layers = list(resnet_weights.children())[:-1]  #get all the layers except the␣
 ↪last one
layers.append(nn.Flatten())
layers.append(nn.Linear(512,6))
weights_resnet = nn.Sequential(*layers)
```

TRAINING RESNET MODEL without Weights of IMAGENET

[ ]:
```
from torch import optim
from torch import nn
import torch.optim.lr_scheduler as lr_scheduler

device = "cuda" if torch.cuda.is_available() else "cpu"
no_weights_resnet = no_weights_resnet.to(device)
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(no_weights_resnet.parameters(),
                      lr=0.001,
                      momentum=0.9)

N_EPOCHS = 20
tr_loss_hist_1 = []
```

```python
val_loss_hist_1 = []
for epoch in range(N_EPOCHS):

    # Training
    train_loss = 0.0
    no_weights_resnet.train() # <1>
    for inputs, labels in trainloader:
        inputs = inputs.to(device)
        labels = labels.to(device)

        optimizer.zero_grad()

        outputs = no_weights_resnet(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        train_loss += loss.item()

    # Validation
    val_loss = 0.0
    no_weights_resnet.eval() # <2>
    for inputs, labels in valloader:
        inputs = inputs.to(device)
        labels = labels.to(device)

        outputs = no_weights_resnet(inputs)
        loss = criterion(outputs, labels)

        val_loss += loss.item()

    print("Epoch: {} Train Loss: {} Val Loss: {}".format(
                  epoch,
                  train_loss/len(trainloader),
                  val_loss/len(valloader)))
    tr_loss_hist_1.append(train_loss/len(trainloader))
    val_loss_hist_1.append(val_loss/len(valloader))
```

```
Epoch: 0 Train Loss: 1.1450279867649078 Val Loss: 1.2718891815135354
Epoch: 1 Train Loss: 0.7377950231234233 Val Loss: 5.439171345610368
Epoch: 2 Train Loss: 0.6393546883265178 Val Loss: 1.7065737968996952
Epoch: 3 Train Loss: 0.5233198421200117 Val Loss: 0.7633673725159544
Epoch: 4 Train Loss: 0.42880301386117936 Val Loss: 0.511101458025606
Epoch: 5 Train Loss: 0.3407818141579628 Val Loss: 0.4911849812457436
Epoch: 6 Train Loss: 0.30346518129110334 Val Loss: 0.23730455064459852
Epoch: 7 Train Loss: 0.34213852542142076 Val Loss: 2.935172291178452
Epoch: 8 Train Loss: 0.3180979681015015 Val Loss: 1.0992569323433072
Epoch: 9 Train Loss: 0.2663927067319552 Val Loss: 0.25383009938033
```

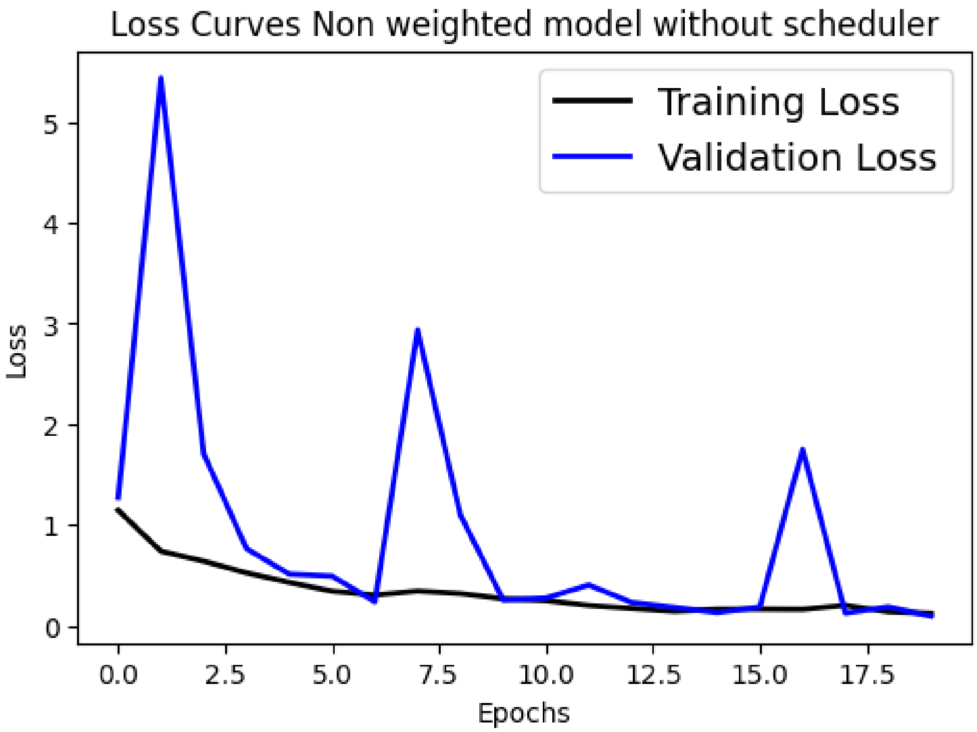
```
Epoch: 10 Train Loss: 0.2493830555677414 Val Loss: 0.27357308497946514
Epoch: 11 Train Loss: 0.19987160275379817 Val Loss: 0.402629581598663
Epoch: 12 Train Loss: 0.17116413176059722 Val Loss: 0.23078455550498084
Epoch: 13 Train Loss: 0.14316563231249652 Val Loss: 0.1819640198036244
Epoch: 14 Train Loss: 0.16198318932205438 Val Loss: 0.12959248584842212
Epoch: 15 Train Loss: 0.16376959941039482 Val Loss: 0.18539345215417838
Epoch: 16 Train Loss: 0.1616867220401764 Val Loss: 1.7496556904363005
Epoch: 17 Train Loss: 0.2017402907460928 Val Loss: 0.12423909559698873
Epoch: 18 Train Loss: 0.13849182942261298 Val Loss: 0.1871733620554503
Epoch: 19 Train Loss: 0.12074925287316243 Val Loss: 0.09341507108489934
```

[ ]:

```
tset = DatasetFolder(root='RowanDLclassNEA/NEUdata_split/Test', loader =␣
 ↪load_image, extensions = ('.bmp',), transform = transform)

testloader = torch.utils.data.DataLoader(
                    tset,
                    batch_size=16,
                    shuffle=True)

num_correct = 0.0

for x_test_batch, y_test_batch in testloader:

    no_weights_resnet.eval()

    y_test_batch = y_test_batch.to(device)

    x_test_batch = x_test_batch.to(device)

    y_pred_batch = no_weights_resnet(x_test_batch)

    _, predicted = torch.max(y_pred_batch, 1)

    num_correct += (predicted == y_test_batch).float().sum()

accuracy_1 = num_correct/(len(testloader)*testloader.batch_size)

print("Test Accuracy of non weighted Resnet: {}".format(accuracy_1))
import matplotlib.pyplot as plt

# Plotting the loss curve
plt.figure(figsize=[6,4])
plt.plot(tr_loss_hist_1, 'black', linewidth=2.0)
plt.plot(val_loss_hist_1, 'blue', linewidth=2.0)
plt.legend(['Training Loss', 'Validation Loss'], fontsize=14)
plt.xlabel('Epochs', fontsize=10)
plt.ylabel('Loss', fontsize=10)
plt.title('Loss Curves Non weighted model without scheduler', fontsize=12)
```

```
Test Accuracy of non weighted Resnet: 0.9270833730697632
```

```
[ ]: Text(0.5, 1.0, 'Loss Curves Non weighted model without scheduler')
```



```
[ ]: from torch import optim
     from torch import nn
     import torch.optim.lr_scheduler as lr_scheduler

     device = "cuda" if torch.cuda.is_available() else "cpu"
     no_weights_resnet_scheduler = no_weights_resnet.to(device)
     criterion = nn.CrossEntropyLoss()
     optimizer = optim.SGD(no_weights_resnet.parameters(),
                           lr=0.001,
                           momentum=0.9)

     scheduler = lr_scheduler.LinearLR(optimizer, start_factor=1, end_factor=0.5,␣
       ↪total_iters=20)

     N_EPOCHS = 20
     tr_loss_hist_3 = []
     val_loss_hist_3 = []
```

```python
for epoch in range(N_EPOCHS):

    # Training
    train_loss = 0.0
    no_weights_resnet.train() # <1>
    for inputs, labels in trainloader:
        inputs = inputs.to(device)
        labels = labels.to(device)

        optimizer.zero_grad()

        outputs = no_weights_resnet_scheduler(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        scheduler.step()

        train_loss += loss.item()

    # Validation
    val_loss = 0.0
    no_weights_resnet_scheduler.eval() # <2>
    for inputs, labels in valloader:
        inputs = inputs.to(device)
        labels = labels.to(device)

        outputs = no_weights_resnet_scheduler(inputs)
        loss = criterion(outputs, labels)

        val_loss += loss.item()

    print("Epoch: {} Train Loss: {} Val Loss: {}".format(
                  epoch,
                  train_loss/len(trainloader),
                  val_loss/len(valloader)))
    tr_loss_hist_3.append(train_loss/len(trainloader))
    val_loss_hist_3.append(val_loss/len(valloader))
```

```
Epoch: 0 Train Loss: 0.09012710683047771 Val Loss: 0.08948544912824505
Epoch: 1 Train Loss: 0.07459103158985575 Val Loss: 0.08601135078405557
Epoch: 2 Train Loss: 0.07867303734024365 Val Loss: 0.062315873740437
Epoch: 3 Train Loss: 0.0730819141678512 Val Loss: 0.113883566050055
Epoch: 4 Train Loss: 0.08103152529646952 Val Loss: 0.0693984224865409
Epoch: 5 Train Loss: 0.08973647153005004 Val Loss: 0.07817137677614626
Epoch: 6 Train Loss: 0.07378084929970403 Val Loss: 0.06537758820003976
Epoch: 7 Train Loss: 0.06534526179234187 Val Loss: 0.11696336175756235
Epoch: 8 Train Loss: 0.07691852075668673 Val Loss: 0.6706761132533613
Epoch: 9 Train Loss: 0.06203098317918678 Val Loss: 0.06055454738854773б
```

```
Epoch: 10 Train Loss: 0.05880419522834321 Val Loss: 0.07144632312089302
Epoch: 11 Train Loss: 0.08232608660434683 Val Loss: 0.07050191935789037
Epoch: 12 Train Loss: 0.05929422864690423 Val Loss: 0.06726676460637368
Epoch: 13 Train Loss: 0.06430536095363398 Val Loss: 0.05377845867463436
Epoch: 14 Train Loss: 0.07602392934883634 Val Loss: 0.21376114220076584
Epoch: 15 Train Loss: 0.059928310252726075 Val Loss: 0.09448349747904822
Epoch: 16 Train Loss: 0.053606167131414015 Val Loss: 0.12373773405622494
Epoch: 17 Train Loss: 0.044881873767202096 Val Loss: 0.15454503212516246
Epoch: 18 Train Loss: 0.0496141190888981 Val Loss: 0.03976297178029965
Epoch: 19 Train Loss: 0.04363506817414115 Val Loss: 0.05308745230401033
```

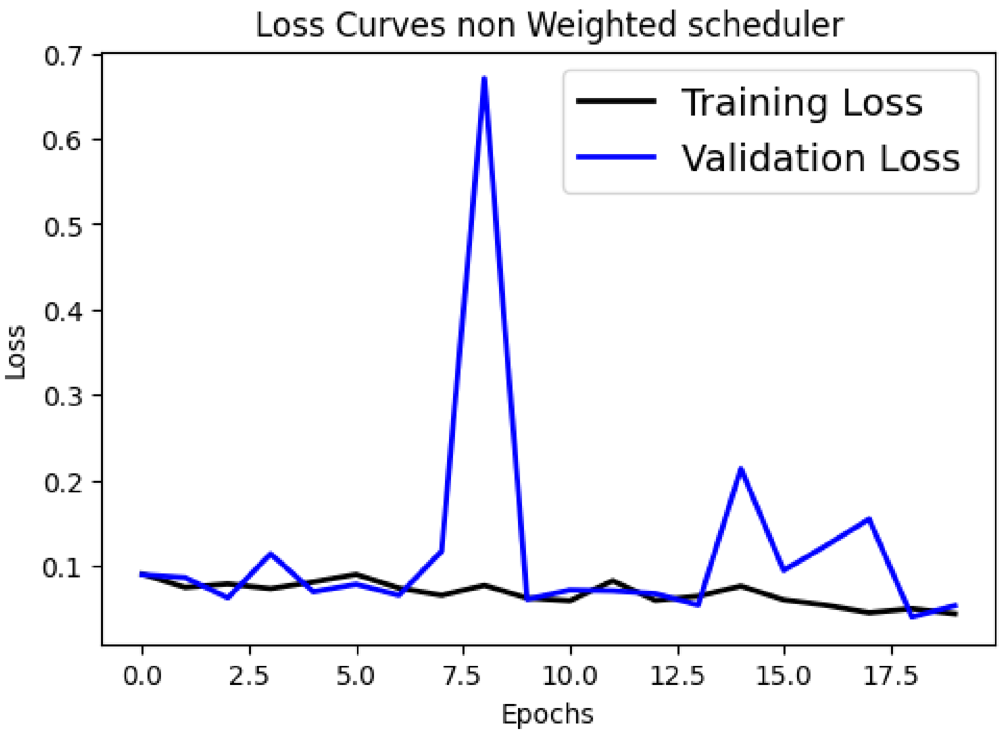
[ ]:

```python
num_correct = 0.0

for x_test_batch, y_test_batch in testloader:

    no_weights_resnet_scheduler.eval()

    y_test_batch = y_test_batch.to(device)

    x_test_batch = x_test_batch.to(device)

    y_pred_batch = no_weights_resnet_scheduler(x_test_batch)

    _, predicted = torch.max(y_pred_batch, 1)

    num_correct += (predicted == y_test_batch).float().sum()

accuracy_3 = num_correct/(len(testloader)*testloader.batch_size)

print("Test Accuracy of non weighted resnet with scheduler: {}".
 ↪format(accuracy_3))
import matplotlib.pyplot as plt

# Plotting the loss curve
plt.figure(figsize=[6,4])
plt.plot(tr_loss_hist_3, 'black', linewidth=2.0)
plt.plot(val_loss_hist_3, 'blue', linewidth=2.0)
plt.legend(['Training Loss', 'Validation Loss'], fontsize=14)
plt.xlabel('Epochs', fontsize=10)
plt.ylabel('Loss', fontsize=10)
plt.title('Loss Curves non Weighted scheduler', fontsize=12)
```

```
Test Accuracy of non weighted resnet with scheduler: 0.9322916865348816
```

[ ]:

```
Text(0.5, 1.0, 'Loss Curves non Weighted scheduler')
```



[ ]:

```
from torch import optim
from torch import nn
import torch.optim.lr_scheduler as lr_scheduler

criterion = nn.CrossEntropyLoss()
device = "cuda" if torch.cuda.is_available() else "cpu"
weights_resnet = weights_resnet.to(device)

optimizer = optim.SGD(weights_resnet.parameters(),
                      lr=0.001,
                      momentum=0.9)

scheduler = lr_scheduler.LinearLR(optimizer, start_factor=1.0, end_factor=0.25,␣
 ↪total_iters=10)


N_EPOCHS = 20
tr_loss_hist_2 = []
val_loss_hist_2 = []
for epoch in range(N_EPOCHS):

    # Training
    train_loss = 0.0
```

```python
    weights_resnet.train() # <1>
    for inputs, labels in trainloader:
        inputs = inputs.to(device)
        labels = labels.to(device)

        optimizer.zero_grad()

        outputs = weights_resnet(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        scheduler.step()

        train_loss += loss.item()

    # Validation
    val_loss = 0.0
    weights_resnet.eval() # <2>
    for inputs, labels in valloader:
        inputs = inputs.to(device)
        labels = labels.to(device)

        outputs = weights_resnet(inputs)
        loss = criterion(outputs, labels)

        val_loss += loss.item()

    print("Epoch: {} Train Loss: {} Val Loss: {}".format(
                  epoch,
                  train_loss/len(trainloader),
                  val_loss/len(valloader)))
    tr_loss_hist_2.append(train_loss/len(trainloader))
    val_loss_hist_2.append(val_loss/len(valloader))
```

```
Epoch: 0 Train Loss: 0.9141104662418366 Val Loss: 0.2694621674324337
Epoch: 1 Train Loss: 0.277562323119647 Val Loss: 0.12261670033790563
Epoch: 2 Train Loss: 0.1720369727909565 Val Loss: 0.07192007437544434
Epoch: 3 Train Loss: 0.12107513311008612 Val Loss: 0.043498319132547626
Epoch: 4 Train Loss: 0.0874715423087279 Val Loss: 0.03595669393574721
Epoch: 5 Train Loss: 0.061345686527589954 Val Loss: 0.02889556258818821
Epoch: 6 Train Loss: 0.06083359488596519 Val Loss: 0.02828169111652594
Epoch: 7 Train Loss: 0.04648646489406626 Val Loss: 0.020105291807435845
Epoch: 8 Train Loss: 0.04827002794171373 Val Loss: 0.027458066182014972
Epoch: 9 Train Loss: 0.033531837649643424 Val Loss: 0.015414760129428223
Epoch: 10 Train Loss: 0.04881436980329454 Val Loss: 0.015582965294781485
Epoch: 11 Train Loss: 0.03962558256462216 Val Loss: 0.013231928965787551
Epoch: 12 Train Loss: 0.03914815075074633 Val Loss: 0.012736092318511126
Epoch: 13 Train Loss: 0.026289602754016716 Val Loss: 0.01133230513598966
```

```
Epoch: 14 Train Loss: 0.02570605160978933 Val Loss: 0.012006353501132444
Epoch: 15 Train Loss: 0.024784315079450607 Val Loss: 0.010936209077236095
Epoch: 16 Train Loss: 0.0166550086128215 Val Loss: 0.011110659596804334
Epoch: 17 Train Loss: 0.0209674815988789 Val Loss: 0.01054140578152759
Epoch: 18 Train Loss: 0.013900610599666834 Val Loss: 0.009714781588531639
Epoch: 19 Train Loss: 0.019143979735672473 Val Loss: 0.008002031946824374
```

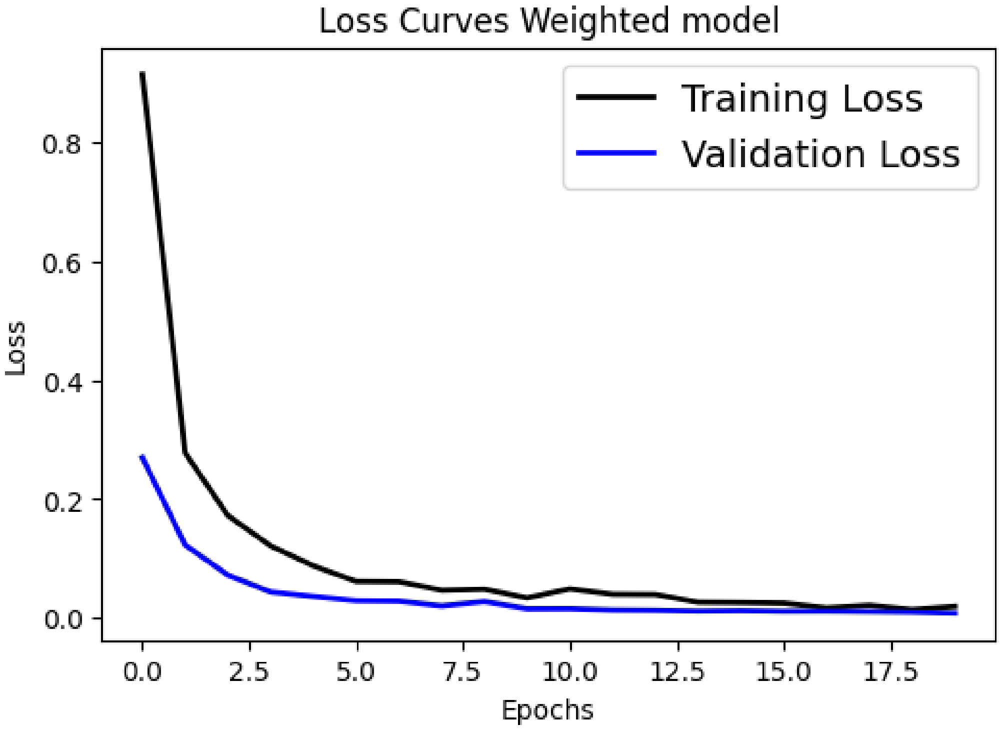
[ ]:

```
num_correct = 0.0

for x_test_batch, y_test_batch in testloader:

    weights_resnet.eval()

    y_test_batch = y_test_batch.to(device)

    x_test_batch = x_test_batch.to(device)

    y_pred_batch = weights_resnet(x_test_batch)

    _, predicted = torch.max(y_pred_batch, 1)

    num_correct += (predicted == y_test_batch).float().sum()

accuracy_2 = num_correct/(len(testloader)*testloader.batch_size)

print("Test Accuracy of weighted resnet: {}".format(accuracy_2))
import matplotlib.pyplot as plt

# Plotting the loss curve
plt.figure(figsize=[6,4])
plt.plot(tr_loss_hist_2, 'black', linewidth=2.0)
plt.plot(val_loss_hist_2, 'blue', linewidth=2.0)
plt.legend(['Training Loss', 'Validation Loss'], fontsize=14)
plt.xlabel('Epochs', fontsize=10)
plt.ylabel('Loss', fontsize=10)
plt.title('Loss Curves Weighted model', fontsize=12)
```

```
Test Accuracy of weighted resnet: 0.9375
```

[ ]:

```
Text(0.5, 1.0, 'Loss Curves Weighted model')
```


Loss Curves Weighted model
Training Loss
Validation Loss
Loss
0.0
0.2
0.4
0.6
0.8
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs