Assignment 5

Shivani Panchiwala

UTA ID: 1001982478

```
# Professor allowed me to use other small dataset because my laptop is getting stuck while implement Food101 dataset which large.
# Dataset - https://www.kaggle.com/datasets/prasunroy/natural-images
#https://www.kaggle.com/code/androbomb/using-cnn-to-classify-images-w-pytorch/notebook
```

Basic CNN Architecture

```
Net(
  (conv1): Conv2d(3, 12, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv2): Conv2d(12, 24, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (drop): Dropout2d(p=0.2, inplace=False)
  (fc): Linear(in_features=24576, out_features=8, bias=True)
)
```

As a optimaizer, I decided to use the **ADAM (ADAptive Moment estimation)** optimization algorithm, that is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing. For more on optimizers in PyTorch.

The traning function we need to define needs the following steps:

1. Set the model to training mode;
2. Process the images in batches; we will iterate over images in batches. Inside each batches, we have to
   A. Import labels and features;
   B. Reset the optimizer
   C. Push the data forward through the layers of the model
   D. compute the loss
   E. Backpropagate
3. Compute the Average Loss of the Model during the Epoch

We will thus calling eath once per epoch.

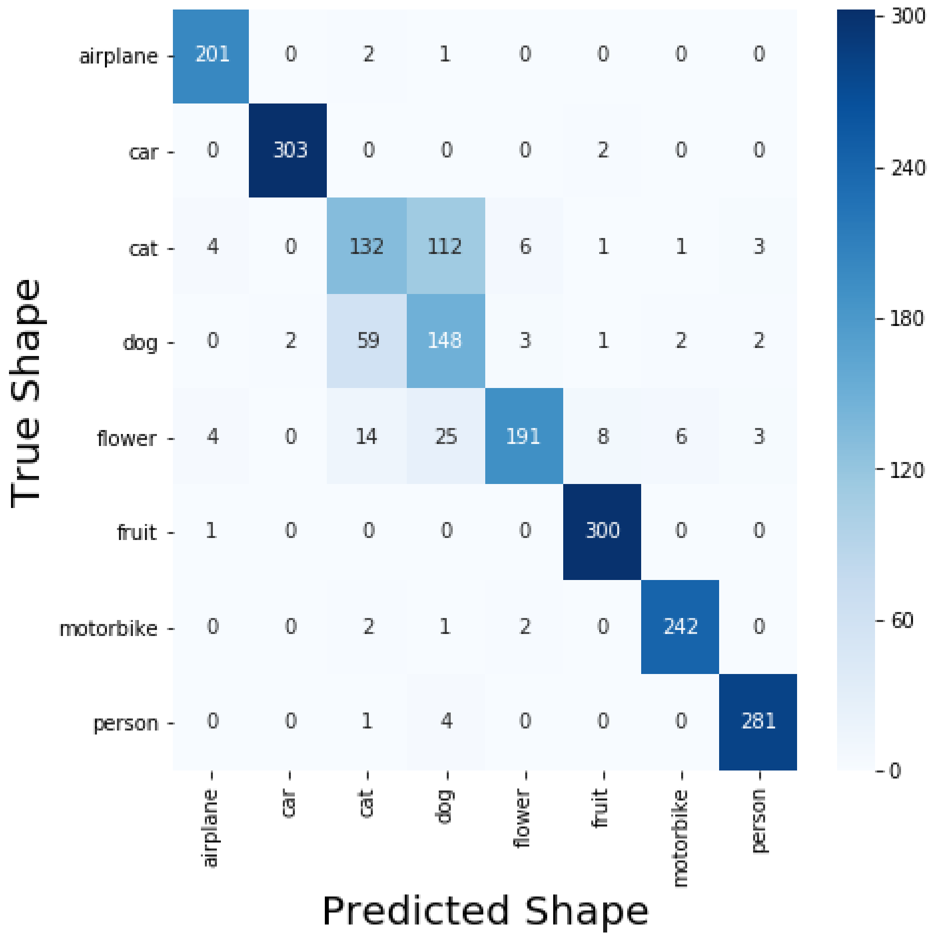
I took epoch =10

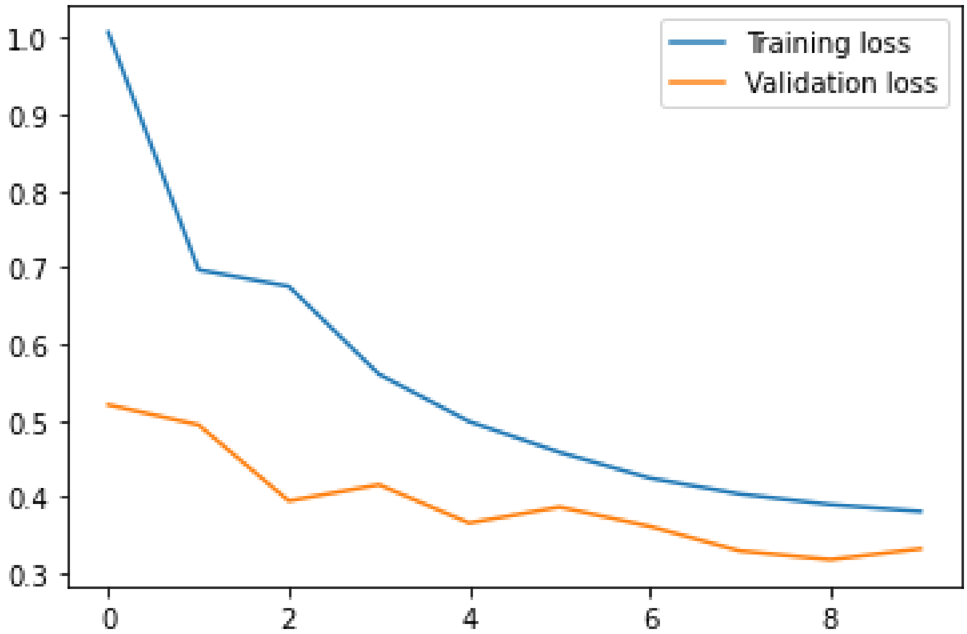
Epoch: 10

Training batch 1 Loss: 0.215580
Training batch 2 Loss: 0.400570
Training batch 3 Loss: 0.431857
Training batch 4 Loss: 0.457938
Training batch 5 Loss: 0.393525
Training batch 6 Loss: 0.371742
Training batch 7 Loss: 0.244514
Training batch 8 Loss: 0.454677
Training batch 9 Loss: 0.197725
Training batch 10 Loss: 0.324622
Training batch 11 Loss: 0.438288
Training batch 12 Loss: 0.288834
Training batch 13 Loss: 0.297090
Training batch 14 Loss: 0.244109
Training batch 15 Loss: 0.429189
Training batch 16 Loss: 0.376998
Training batch 17 Loss: 0.506088
Training batch 18 Loss: 0.253847
Training batch 19 Loss: 0.438082
Training batch 20 Loss: 0.878997
Training batch 21 Loss: 0.137446
Training batch 22 Loss: 0.544574
Training batch 23 Loss: 0.506861
Training batch 24 Loss: 0.316770
Training batch 25 Loss: 0.412908
Training batch 26 Loss: 0.513011
Training batch 27 Loss: 0.392239
Training batch 28 Loss: 0.492679
Training batch 29 Loss: 0.127278
Training batch 30 Loss: 0.724068
Training batch 31 Loss: 0.575240
Training batch 32 Loss: 0.350005
Training batch 33 Loss: 0.330589
Training batch 34 Loss: 0.306604
Training batch 35 Loss: 0.307523
Training batch 36 Loss: 0.918360
Training batch 37 Loss: 0.270447
Training batch 38 Loss: 0.090224
Training batch 39 Loss: 0.477621
Training batch 40 Loss: 0.277689
Training batch 41 Loss: 0.325278
Training batch 42 Loss: 0.275837
Training batch 43 Loss: 0.253396
Training batch 44 Loss: 0.292126
Training batch 45 Loss: 0.354066
Training batch 46 Loss: 0.300094
Training batch 47 Loss: 0.556521
Training batch 48 Loss: 0.332499
Training batch 49 Loss: 0.488544
Training batch 50 Loss: 0.533675
Training batch 51 Loss: 0.399599
Training batch 52 Loss: 0.426033
Training batch 53 Loss: 0.563066

```
    Training batch 54 Loss: 0.299225
    Training batch 55 Loss: 0.596873
    Training batch 56 Loss: 0.575532
    Training batch 57 Loss: 0.283631
    Training batch 58 Loss: 0.285560
    Training batch 59 Loss: 0.483531
    Training batch 60 Loss: 0.423469
    Training batch 61 Loss: 0.479524
    Training batch 62 Loss: 0.242024
    Training batch 63 Loss: 0.340023
    Training batch 64 Loss: 0.303990
    Training batch 65 Loss: 0.273993
    Training batch 66 Loss: 0.368834
    Training batch 67 Loss: 0.582135
    Training batch 68 Loss: 0.323422
    Training batch 69 Loss: 0.252664
    Training batch 70 Loss: 0.242119
    Training batch 71 Loss: 0.367343
    Training batch 72 Loss: 0.460398
    Training batch 73 Loss: 0.670769
    Training batch 74 Loss: 0.251815
    Training batch 75 Loss: 0.220071
    Training batch 76 Loss: 0.251227
    Training batch 77 Loss: 0.320876
    Training batch 78 Loss: 0.402557
    Training batch 79 Loss: 0.371436
    Training batch 80 Loss: 0.368226
    Training batch 81 Loss: 0.486412
    Training batch 82 Loss: 0.267487
    Training batch 83 Loss: 0.209389
    Training batch 84 Loss: 0.282732
    Training batch 85 Loss: 0.451458
    Training batch 86 Loss: 0.276844
    Training batch 87 Loss: 0.272532
    Training batch 88 Loss: 0.542886
    Training batch 89 Loss: 0.355977
    Training batch 90 Loss: 0.370555
    Training batch 91 Loss: 0.375482
    Training batch 92 Loss: 0.196394
    Training batch 93 Loss: 0.518548
    Training batch 94 Loss: 0.567814
    Training batch 95 Loss: 0.235751
    Training batch 96 Loss: 0.338469
    Training batch 97 Loss: 0.328206
Training set: Average loss: 0.380777
Validation set: Average loss: 0.331784, Accuracy: 1798/2070 (87%)
```

```
Getting predictions from test set...
```





All Convolutional Network

Network used:

```
Allconvo(
  (conv1): Conv2d(3, 16, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1))
  (conv2): Conv2d(16, 32, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1))
  (conv3): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1))
  (conv4): Conv2d(64, 64, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  (dropout): Dropout2d(p=0.4, inplace=False)
  (batchnorm1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (batchnorm2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (batchnorm3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (fc1): Linear(in_features=1600, out_features=512, bias=True)
  (fc2): Linear(in_features=512, out_features=256, bias=True)
  (fc3): Linear(in_features=256, out_features=4, bias=True)
)
```

```
Epoch: 10
      Training batch 1 Loss: 0.256552
      Training batch 2 Loss: 0.277665
      Training batch 3 Loss: 0.371196
      Training batch 4 Loss: 0.259750
      Training batch 5 Loss: 0.177695
      Training batch 6 Loss: 0.243364
      Training batch 7 Loss: 0.221827
      Training batch 8 Loss: 0.362315
      Training batch 9 Loss: 0.191053
      Training batch 10 Loss: 0.198967
      Training batch 11 Loss: 0.244452
      Training batch 12 Loss: 0.339923
      Training batch 13 Loss: 0.241270
      Training batch 14 Loss: 0.213669
      Training batch 15 Loss: 0.314441
      Training batch 16 Loss: 0.271592
      Training batch 17 Loss: 0.311054
      Training batch 18 Loss: 0.250080
      Training batch 19 Loss: 0.337135
      Training batch 20 Loss: 0.244453
      Training batch 21 Loss: 0.214220
      Training batch 22 Loss: 0.457922
      Training batch 23 Loss: 0.407007
      Training batch 24 Loss: 0.334117
      Training batch 25 Loss: 0.275229
      Training batch 26 Loss: 0.283150
      Training batch 27 Loss: 0.222227
      Training batch 28 Loss: 0.333721
      Training batch 29 Loss: 0.251266
      Training batch 30 Loss: 0.343995
      Training batch 31 Loss: 0.240108
      Training batch 32 Loss: 0.151909
      Training batch 33 Loss: 0.284326
```

```
        Training batch 34 Loss: 0.348609
        Training batch 35 Loss: 0.295026
        Training batch 36 Loss: 0.381754
        Training batch 37 Loss: 0.287323
        Training batch 38 Loss: 0.258291
        Training batch 39 Loss: 0.287313
        Training batch 40 Loss: 0.250249
        Training batch 41 Loss: 0.384035
        Training batch 42 Loss: 0.285144
        Training batch 43 Loss: 0.173163
        Training batch 44 Loss: 0.210336
        Training batch 45 Loss: 0.319180
        Training batch 46 Loss: 0.167396
        Training batch 47 Loss: 0.346009
        Training batch 48 Loss: 0.316399
        Training batch 49 Loss: 0.254362
        Training batch 50 Loss: 0.267434
        Training batch 51 Loss: 0.313971
        Training batch 52 Loss: 0.292695
        Training batch 53 Loss: 0.521185
        Training batch 54 Loss: 0.375270
        Training batch 55 Loss: 0.377329
        Training batch 56 Loss: 0.404574
        Training batch 57 Loss: 0.156820
        Training batch 58 Loss: 0.140445
        Training batch 59 Loss: 0.371543
        Training batch 60 Loss: 0.257344
        Training batch 61 Loss: 0.439362
        Training batch 62 Loss: 0.177520
        Training batch 63 Loss: 0.338472
        Training batch 64 Loss: 0.344273
        Training batch 65 Loss: 0.273731
        Training batch 66 Loss: 0.254769
        Training batch 67 Loss: 0.194062
        Training batch 68 Loss: 0.290815
        Training batch 69 Loss: 0.287370
        Training batch 70 Loss: 0.330671
        Training batch 71 Loss: 0.414598
        Training batch 72 Loss: 0.422282
        Training batch 73 Loss: 0.327267
        Training batch 74 Loss: 0.151157
        Training batch 75 Loss: 0.225146
        Training batch 76 Loss: 0.242818
        Training batch 77 Loss: 0.279342
        Training batch 78 Loss: 0.291669
        Training batch 79 Loss: 0.196575
        Training batch 80 Loss: 0.275824
        Training batch 81 Loss: 0.314042
        Training batch 82 Loss: 0.144787
        Training batch 83 Loss: 0.327011
        Training batch 84 Loss: 0.233003
        Training batch 85 Loss: 0.147416
        Training batch 86 Loss: 0.241256
        Training batch 87 Loss: 0.250847
        Training batch 88 Loss: 0.156115
        Training batch 89 Loss: 0.394556
        Training batch 90 Loss: 0.294581
        Training batch 91 Loss: 0.292210
```

```
     Training batch 92 Loss: 0.105365
     Training batch 93 Loss: 0.265292
     Training batch 94 Loss: 0.115586
     Training batch 95 Loss: 0.178074
     Training batch 96 Loss: 0.165241
     Training batch 97 Loss: 0.312843
Training set: Average loss: 0.276977
Validation set: Average loss: 0.274173, Accuracy: 1837/2070 (89%)
```

In All convolutional is more accurate then Basic CNN.

L1 Regularization

I took only 5 epoch

```
Starting epoch 5
Loss after mini-batch   500: 2.94215 (of which 0.63957 L1 loss)
Loss after mini-batch  1000: 2.93445 (of which 0.63186 L1 loss)
Loss after mini-batch  1500: 2.93992 (of which 0.63733 L1 loss)
Loss after mini-batch  2000: 2.94098 (of which 0.63839 L1 loss)
Loss after mini-batch  2500: 2.93725 (of which 0.63468 L1 loss)
Loss after mini-batch  3000: 2.93692 (of which 0.63433 L1 loss)
Loss after mini-batch  3500: 2.94259 (of which 0.63999 L1 loss)
Loss after mini-batch  4000: 2.93637 (of which 0.63378 L1 loss)
Loss after mini-batch  4500: 2.93862 (of which 0.63604 L1 loss)
Loss after mini-batch  5000: 2.93913 (of which 0.63653 L1 loss)
Loss after mini-batch  5500: 2.93972 (of which 0.63714 L1 loss)
Loss after mini-batch  6000: 2.94161 (of which 0.63903 L1 loss)
Training process has finished.
```

L2 Regularization

```
Starting epoch 5
Loss after mini-batch   500: 2.30324 (of which 0.00011 l2 loss)
Loss after mini-batch  1000: 2.30488 (of which 0.00011 l2 loss)
Loss after mini-batch  1500: 2.30274 (of which 0.00004 l2 loss)
Loss after mini-batch  2000: 2.30191 (of which 0.00004 l2 loss)
Loss after mini-batch  2500: 2.30102 (of which 0.00007 l2 loss)
Loss after mini-batch  3000: 2.30194 (of which 0.00008 l2 loss)
Loss after mini-batch  3500: 2.30228 (of which 0.00007 l2 loss)
Loss after mini-batch  4000: 2.30383 (of which 0.00009 l2 loss)
Loss after mini-batch  4500: 2.30140 (of which 0.00006 l2 loss)
Loss after mini-batch  5000: 2.30330 (of which 0.00009 l2 loss)
Loss after mini-batch  5500: 2.30312 (of which 0.00007 l2 loss)
Loss after mini-batch  6000: 2.30295 (of which 0.00011 l2 loss)
Training process has finished.
```

Transfer Learning

```
#I did this whole assignmet using transfer learning....
```

```
!unzip archive\ \(1\).zip
# The images are in a folder named 'input/natural-
images/natural_images'
training_folder_name = 'data/natural_images'

# All images are 128x128 pixels
img_size = (128,128)

# The folder contains a subfolder for each class of shape
classes = sorted(os.listdir(training_folder_name))
print(classes)
```

```
['airplane', 'car', 'cat', 'dog', 'flower', 'fruit', 'motorbike',
'person']
```

```
from PIL import Image

# function to resize image
def resize_image(src_image, size=(128,128), bg_color="white"):
    from PIL import Image, ImageOps

    # resize the image so the longest dimension matches our target size
    src_image.thumbnail(size, Image.ANTIALIAS)

    # Create a new square background image
    new_image = Image.new("RGB", size, bg_color)

    # Paste the resized image into the center of the square background
    new_image.paste(src_image, (int((size[0] - src_image.size[0]) / 2),
 int((size[1] - src_image.size[1]) / 2)))

    # return the resized image
    return new_image
```

```
training_folder_name = 'data/natural_images'

# New location for the resized images
train_folder = '../working/data/natural_images'


# Create resized copies of all of the source images
size = (128,128)

# Create the output folder if it doesn't already exist
```

```python
if os.path.exists(train_folder):
    shutil.rmtree(train_folder)

# Loop through each subfolder in the input folder
print('Transforming images...')
for root, folders, files in os.walk(training_folder_name):
    for sub_folder in folders:
        print('processing folder ' + sub_folder)
        # Create a matching subfolder in the output dir
        saveFolder = os.path.join(train_folder,sub_folder)
        if not os.path.exists(saveFolder):
            os.makedirs(saveFolder)
        # Loop through the files in the subfolder
        file_names = os.listdir(os.path.join(root,sub_folder))
        for file_name in file_names:
            # Open the file
            file_path = os.path.join(root,sub_folder, file_name)
            #print("reading " + file_path)
            image = Image.open(file_path)
            # Create a resized version and save it
            resized_image = resize_image(image, size)
            saveAs = os.path.join(saveFolder, file_name)
            #print("writing " + saveAs)
            resized_image.save(saveAs)

print('Done.')
```

```
Transforming images...
processing folder fruit
processing folder person
processing folder dog
processing folder airplane
processing folder car
processing folder flower
processing folder cat
processing folder motorbike
Done.
```

After that I took same cnn model as I mentioned above..