Assignment 2
Neural Network

Due Date: 6th May, 2022

Guidelines

- Submit all of your code, saved models and results in a single zip file with name FirstName_RollNumber_02.zip
- Submit single zip file containing:
    a) Code (task01.ipynb, task02.ipynb, task03.ipynb)
    b) Saved Models
    c) Report
- There should be a report.pdf detailing your experience and highlighting any interesting result.
- Email instructor or TAs if there are any questions.
- **Don't resubmit the dataset provided in your submission.**

Overview

In this assignment, you will write the code to train the Neural Networks for face classification, face verification and multi-label classification. The goals of this assignment are as follows:

- Understand how neural networks can be used for classification of images
- Understand how a model can be created, trained, validated, and saved in PyTorch
- Understand how to read/write and process images in python

You can use sklearn, matplotlib and pytorch for this assignment. If your system has a GPU, turn it on to improve performance. **You can also use Free GPU on Google Colab.**

Note: This assignment will take time so start early.

CelebA Dataset

Download the celebA.zip file. **CelebFaces Attributes Dataset (CelebA)** is a large-scale face attributes dataset with more than **200K** celebrity images, each with **40** attribute annotations. You will be using an aligned and cropped version of celebA. CelebA has large diversities, and rich annotations, including:

- **10,177** number of **identities**,
- **202,599** number of **face images**, and
- **5 landmark locations**, **40 binary attributes** annotations per image

Figure 1: Sample Images from CelebA dataset

Task 01: Celebrity Face Classification using Fully Connected Neural (30 Marks)

In this task you will perform simple multi-class classification, in which an instance/image always belongs to one and only one class. *In multi-class classification we use a single softmax layer as the last layer, and learn a single probability distribution that ranges over all classes***.** You need to build a neural network that accepts an image of a person and tells who this person is, i.e. assign a class label to it.

1. **Data Pre-processing:** Extract 120 unique identities (of your own choice) from the CelebA dataset **('CelebA/Anno/identity_CelebA.txt' contains the identity information against each image)** . You can do it manually but we would recommend using a more systematic approach (e.g. pandas or any other library of your choice). **Please make sure that the number of images for all selected identities are almost the same.**

2. **Data Loading:** Write a function load_dataset() which accepts dataset path, size of train data, validation data, test data, batch size and returns the train, test and validation data. **Hint:** Create 120 directories one for each identity and use imagefolder method as we did in the later part of the tutorial.

3. **Initialize Network:** Write a function init_net(no_of_layers, input_dim, neuron_per_layer) to initialize the network. You can use pytorch builtin functions to create and initialize the network. No_of_layers and input_dim should be an integer whereas neuron_per_layer

will be a list having elements describing the number of neurons in each hidden layer last element should be the dimension of the output layer (i.e. for this task 120)

4. **Training Network:** Write a function train(epochs, train_x, train_y, val_x, val_y, loss_func, optimizer, learning rate) to train the initialized network. For both training and validation data record accuracy and loss at each epoch to plot accuracy and loss curves later.

5. **Save Network:** Save the trained network using either pickle or pytorch for later use.

6. **Test Network:** Now write a function test() that uses a trained network to make predictions on a test set. Function should return loss and accuracy on test data.

7. **Visualize Results:** Plot the **"epoch vs accuracy"** and **"epoch vs loss"** , confusion matrix. Also show your best and worst predictions on the test dataset.

Note: You may need to perform data augmentation for task 01. Random crop, random horizontal and vertical flips, random rotation, gaussian blur are some of the common image data augmentation techniques. You need to think which of the augmentation techniques will work best for this task.

Task 2: Celebrity Face Verification using Fully Connected Neural Networks (30 Marks)

In this part you need to design a neural network that accepts 2 face images simultaneously as input and output the probability whether these two images belong to the same person or not.

1. **Data Preprocessing:** Write a script to generate the data. Suppose there are 3 unique persons in the dataset X, Y, Z and each has 2 faces. Then your label should look like this:

Image 1	**Image 2**	**Label**
X_1	X_2	1
X_1	Y_1	0
Y_2	Z_1	0
Z_1	Z_2	1

In the above table numbers in subscripts show the image number and Label=1 if and only if both the images belong to the same person (row 1 and 4 in above table). If the images belong to different person Label = 0 (row 2 and 3 in above table) images belong to different persons. **Please make sure that you have an almost similar representation of both the labels in your generated dataset.** One way to ensure this could be to write two separate scripts to generate label 1 and label 0 dataset and then merge them. Once you are done creating the dataset, feed this dataset to the

load_dataset function. Creating all possible combinations of **202,599** images won't be a good approach for this assignment. Similarly, having too few images will be an issue too. You need to think of how many training examples would work just fine for this task.

2. **Data Loading:** Write a function load_dataset() which accepts dataset path, size of train data, validation data, test data, batch size and returns the train, test and validation data. **Hint:** In the getitem() function of custom dataset class you will be reading two images simultaneously.

3. **Initialize Network:** Write a function init_net(no_of_layers, input_dim, neuron_per_layer) to initialize the network. You can use pytorch builtin functions to create and initialize the network. No_of_layers and input_dim should be an integer whereas neuron_per_layer will be a list having elements describing the number of neurons in each hidden layer last element should be the dimension of the output layer.

4. **Training Network:** Write a function train(epochs, train_x, train_y, val_x, val_y, loss_func, optimizer, learning rate) to train the initialized network. For both training and validation data record accuracy and loss at each epoch to plot accuracy and loss curves later.

5. **Save Network:** Save the trained network using either pickle or pytorch for later use.

6. **Test Network:** Now write a function test() that uses a trained network to make predictions on a test set. Function should return loss and accuracy on test data.

7. **Visualize Results:** Plot the **"epoch vs accuracy"** and **"epoch vs loss"** , confusion matrix. Also show your best and worst predictions on the test dataset.

Task 3: Multi-Label Classification using Fully Connected Neural Networks (40 Marks)

In this task you will be performing multi-label classification. In Multi-label classification an instance/image can be one for multiple classes/labels. *In multi-label classification we use multiple sigmoids on the last layer and learn a separate distribution for each class.*

1. **Data Preprocessing:** Extract a subset of data having almost equal representation of both the classes on the basis of below mentioned attributes i.e. have same image count for male and female or wearing earrings and not wearing earrings class. **('CelebA/Anno/list_attr_celeba.txt' contains the attribute information).**

    - Is Male / Is Female
    - Is have Black hair / have not black hair
    - Is wearing earrings / not wearing earrings
    - Is wearing eyeglasses/ not wearing eyeglasses
    - Is have straight hairs / have not straight hairs
    - Is smiling / not smiling
    - Is wearing necktie / not wearing necktie

2. **Data Loading:** Write a function load_dataset() which accepts dataset path, size of train data, validation data, test data, batch size and returns the train, test and validation data. **Hint:** You will have to write custom dataset class for this task.

3. **Initialize Network:** Write a function init_net(no_of_layers, input_dim, neuron_per_layer) to initialize the network. You can use pytorch builtin functions to create and initialize the network. No_of_layers and input_dim should be an integer whereas neuron_per_layer will be a list having elements describing the number of neurons in each hidden layer last element should be the dimension of the output layer (i.e. for this task 7 one per attribute). For example, an output of 1000111 will show a Male person, have not black hair, not wearing earrings, not wearing eyeglasses, have straight hairs, is smiling, and is wearing necktie. Wearing Glasses, Smiling, Wearing hat, Old. **Similarly the ground truth label for each image will contain 7 values. The total loss will be the sum of losses for each attribute.**

4. **Training Network:** Write a function train(epochs, train_x, train_y, val_x, val_y, loss_func, optimizer, learning rate) to train the initialized network. For both training and validation data record accuracy and loss at each epoch to plot accuracy and loss curves later.

5. **Save Network:** Save the trained network using either pickle or pytorch for later use.

6. **Test Network:** Now write a function test() that uses a trained network to make predictions on a test set. Function should return loss and accuracy on test data.

7. **Visualize Results:** Plot the **"epoch vs accuracy"** and **"epoch vs loss"**, confusion matrix. Also show your best and worst predictions on the test dataset.

Note: You are required to train a separate neural network for each task.

Report (10 Marks)

For all the task show the "epoch vs accuracy" and "epoch vs loss" graph for your best model. Report the train/validation/test accuracy and loss for all the experiments you did along with the model parameters.