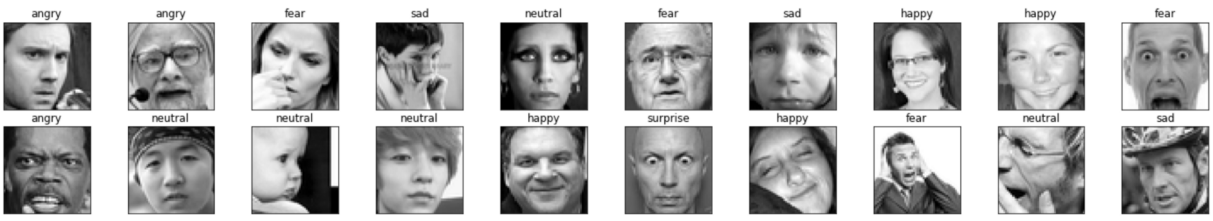
Assignment 1: Facial Expression Emotion Classification

CS886 - Fall 2021

Out: 12 Oct 2021
Due: 29 Oct 2021



In this assignment, you will complete a hands-on exercise of facial emotion recognition using a facial images dataset in raw pixel form, called 'FER2013', which classifies a facial image into 7 categories of emotion ['Angry', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprise', 'Neutral']. The idea is to use a CNN to classify unseen images into these seven categories according to the emotion being presented.

In problem 1, your aim is to learn how to load a pre-trained model, evaluate the model on test data to reproduce the results, plot a confusion matrix and show the data distribution. In problem 2, your aim is to train a model and plot the training curves. Finally, a short report on the observations and further thoughts will summarize your understanding.

Submission Material: Submit Google Colaboratory notebook showing code and outputs and a report pdf. Refer to 'What to hand in' after you go over both problem 1 and 2 sections. **Submit all materials on LEARN to the A1 dropbox**.

Problem 1 *(10 points)*

Goal: Reproduce one of the the state of art result for FER2013 recorded on [leaderboard], with the VGGNet architecture and achieving accuracy of 73.28 %, using this repo

Steps:

a. Download the dataset file `fer2013.csv` from this Kaggle link and the pretrained model file `VGGNet` from the same repo. You may need to create a free account. link.

b. Upload both the `fer2013.csv` and `VGGNet` files to your Google drive.

c. Open Google Colab using https://colab.research.google.com/. Create Click on `New Notebook`, then click on `Runtime`, `Change Runtime` and select `GPU` for hardware acceleration. Rename the notebook as:
{`Studentid`}_`CS886_A1.ipynb`

d. Put the code in the first cell to mount your google drive:

```
from google.colab import drive
drive.mount('/content/drive')
```

Run this cell and when it asks to input in a box, please **provide authentication** to allow access to your drive

e. Refer to the `Evaluation.ipynb` notebook here. Copy each code cell into your notebook and run each cell. Wherever `fer2013.csv` is mentioned, replace with `'drive/MyDrive/fer2013.csv'`. Do the same with `VGGNet` to point to the correct file in drive.

f. Run each cell by `Crtl + Enter` or click on little black arrow on the left of each cell and understand the function of each cell.

g. **Evaluate test data and report Top 1 and Top 2 accuracy**

h. **Plot a confusion matrix** of the test data using the seaborn library. This should give you a visualization with colors showing the accuracy of each classification, with rows being the actual emotion labels, and columns being the predicted emotion labels. The color of each cell gives the number of images classified in that way. You can use this article as a reference. Don't forget the color legend!

i. **Extract the first image from the training set for each class and plot it as an image. Also plot the distribution of the number of examples in 'Training', 'PrivateTest' and 'PublicTest' data sets** in `fer2013.csv` as three bar charts, one for each data set showing how many images are in each emotion class
The pseudocode below can be used as a reference.

```
import matplotlib.pyplot as plt
#load data
fer2013, emotion_mapping = load_data('drive/MyDrive/fer2013.csv')
xtrain, ytrain = prepare_data(fer2013[fer2013['Usage']=='Training'])

# 7 emotion classes is fixed and known
n_classes = 7
# array to collect number of samples in each set
num_of_samples=[]
# set up a plot of the correct size for images
# do the following for each data set
plt.figure(figsize=(12, 16.5))
for i in range(0, n_classes):
    #plot image using plt.imshow with cmap='gray'
    #count number of images
plt.show()

#Plot number of images per class
plt.figure(figsize=(12, 4))
#plot your data for each data set
plt.show()
```

j. Goto File, download as ipynb to keep a copy and continue with next Problem 2 on the same colab.

BONUS: take a picture of your own face, and attempt to classify your facial expression using this pre-trained model. Show your picture and explain what happened.

Problem 2 *(10 points)*

Goal: To train a classifier model for FER2013 from scratch

Steps:

a. Continue with same notebook as in Problem 1. We shall first get some methods from the repo, that will be useful for training. Again, this is all in this repo - in the `utils` folder.

b. Get code from `logger.py` and copy in a new cell

c. Get code from `hparams.py`, copy in a new cell. Make the following changes at the top

```
#possible_nets = set(filename.split(".")[0] for filename in os.
                                        listdir('models'))
nets = {
    'vgg': Vgg
}
possible_nets = 'vgg'
```

d. get the function `setup_network` from `setup_network.py` and put in a new cell.

e. get code for `save` method from `checkpoint.py` and `train` and `evaluate` methods from `loops.py` and copy in a new cell.

f. Get code from `train.py` and copy in a new cell. You will need to comment the imports for below, as you have already got the appropriate methods from the respective files in the repo

```
# from data.fer2013 import get_dataloaders
# from utils.checkpoint import save
# from utils.hparams import setup_hparams
# from utils.loops import train, evaluate
# from utils.setup_network import setup_network
```

and replace last part of train cell with the folllowing:

```
if __name__ == "__main__":
    # Important parameters
    # hps = setup_hparams(sys.argv[1:])
    hps = setup_hparams('')
    hps['network'] = 'vgg'
    hps['name'] = 'my_vgg'
    hps['bs'] = 64
    logger, net = setup_network(hps)

    run(net, logger, hps)
```

g. Run all the new cells added and train the CNN classifier for at least 100 epochs if possible on Google colab (related param is `n_epochs`) and **record the train and val accuracy for each epoch.** This should take about 4 hours, but this can vary a lot, so if you can't get to 100 epochs, just report how long (clock time) you ran it for.

h. **Plot the accuracy curve for training and validation accuracy**

i. Goto File, download as ipynb

BONUS: do the same thing with another dataset related to emotion. It could be a facial expression or other dataset.

What to hand in: {`Studentid`}`_CS886_A1.ipynb`
This will be graded as follows. You get 2.5 marks for each of the following, up to a maximum of 10 marks.

- get all the code running for problem 1
- get all the code running for problem 2
- plot the confusion matrix properly
- plot train and validation accuracy for each epoch

Then, there are two bonus questions which can get you an extra 2 marks each. **These are more challenging! Keep in mind that the two BONUS questions together will be worth less than 1% of your final grade!**

- classify an image of your own face.
- train the same network on a different dataset.

Problem 3 *(10 points)*

Goal: Discuss your results and observations, clarify your understanding and form an opinion

Steps:

Provide short (50-100 words) answers to the following questions:

a. Did you achieve the expected test results?
b. What did you learn from the confusion matrix? Which expressions are harder and which are easier and why?
c. Write down one idea to improve the model accuracy
d. Write down one limitation or concern for use of this model in real world settings. Your concern may be ethical or technical.

What to hand in: {Studentid}_CS886_A1_Report.pdf
This will be graded 50% on completeness (are all questions answered?) and 50% on clarity (can I understand your answers?). See further definition of completeness and clarity here: https://cs.uwaterloo.ca/~jhoey/teaching/cs886-affect/project.html