Instructions

This document is a template, and you are not required to follow it exactly. However, the kinds of questions we ask here are the kinds of questions we want you to focus on. While you might have answered similar questions to these in your project presentations, we want you to go into a lot more detail in this write-up; you can refer to the Lab homeworks for ideas on how to present your data or results.

You don't have to answer every question in this template, but you should answer roughly this many questions. Your answers to such questions should be paragraph-length, not just a bullet point. You likely still have questions of your own -- that's okay! We want you to convey what you've learned, how you've learned it, and demonstrate that the content from the course has influenced how you've thought about this project.

CS 475/675 Real-Time Facial Recognition with Sentiment Analysis

Project mentor: Gordon Sun

Owen Bianchi obianch1@jh.edu, Robert Seney rseney1@jh.edu

Link_to_git_repo: https://github.com/oween12/ML_Final_Project

Outline and Deliverables

List the deliverables from your project proposal. For each uncompleted deliverable, please include a sentence or two on why you weren't able to complete it (e.g. "decided to use an existing implementation instead" or "ran out of time"). For each completed deliverable, indicate which section of this notebook covers what you did.

If you spent substantial time on any aspects that weren't deliverables in your proposal, please list those under "Additional Work" and indicate where in the notebook you discuss them.

Uncompleted Deliverables

1. "The group expects to have an additional protection added to prevent racial differences and potential issues therein": due to time limitations, we were unable to look in depth at racial differences and any learned behavior within our model that would rely specifically on race.

2. "The group would like to present a facial recognition and emotion detection algorithm which has no issues detecting faces and emotions of faces of different races" : again due to a similar reason as the first, we were unable to look specificallly at this concern.
3. "The group would like to present an emotion detection algorithm which is unhindered by the covering of certain body parts, such as the mouth or the eyes" : due to time constraints, excessive testing beyond the implementation of mock-scaring discussed below was not possible.

Completed Deliverables

1. "The group must be reliably able to establish facial recognition over the data being analyzed": We discuss our model and its ability to recognive faces in "Models and Evaluations" below.
2. "The group must be able to reliably predict the emotion of a given sample image through the use of a Convolutional Neural Network": We discuss the training of our model in "Methods" below.
3. "The group must be able to implement this facial recognition real time, using the camera feed from an attached camera": We discuss the implementation of facial recognition real time in "Methods" below.
4. "The group expects to present an emotion detection algorithm which is not hindered by the presence of facial hair or scars which may deform typical facial features" : We believe that we have begun to approach this goal, although we are not entirely sure of the success of our approach. The approach is discussed in detail within "Preprocessing" below
5. "The group expects to present an algorithm which can effectively and timely predict the emotion that a live-feed is providing, real time" : We were able to implement our model real-time and give real-time emotion predictions, detailed in "Results" below

Preliminaries

What problem were you trying to solve or understand?

In this machine learning task, we are attempting to create software which is able to recieve real time input from a camera, detect human faces, and accurately predict the emotion which the face is displaying. This task utilizes real people's faces and can be utilized in realistic scenarios, providing informaiton on the feelings of people from just a video feed. This problem is similar to many problems that we have observed in class and homeworks, especially with regards to neural networks. Our initial approach to this problem follows the processes used in Homework 3 and the use of convolutional neural networks. This problem is made unique by an additional task that we have added to the preprocessing of our training data. This task involves changing a small patch of the image to be a black box. The idea behind this is to encourage the model to learn and perform in an abscence of complete data. This would theoretically be helpful in the case of scaring or a partial covering of the face. There are clear ethical implications with regard to this problem, especially in terms of racial accuracy. Our model could potentially perform extremely well with regards to light-skinned individuals but perform extremely poorly for those with darker skin. This would make our model unethical to utilize in any real-world scenario, as this discrepency could have implications past just emotion detection.

Dataset(s)

The data set that was utilized for this project comes from a kaggle competition, which can be found here:

https://www.kaggle.com/competitions/challenges-in-representation-learning-facial-expression-recognition-challenge/data

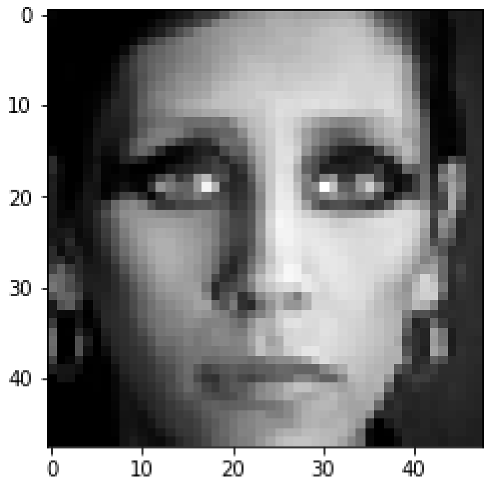
This dataset is composed of 34,034 images, each of which is a grayscale 48 x 48 image. These images are of faces displaying one of seven emotions: anger, disgust, fear, happiness, sadness, suprise, or neutral. Our original thoughts involved utilizing the Yale face database, a smaller data base of 165 images portraying 11 different emotions. When we started our project we did not fully understand how limiting the size of this dataset would be, lead us to find a bigger dataset with the same image information, which we found in this new dataset. The numbers for the emotions correspond to the following:

- 0 - Anger
- 1 - Disgust
- 2 - Fear
- 3 - Happy
- 4 - Sad
- 5 - Suprise
- 6 - Neutral

```
# Load your data and print 2-3 examples
from google.colab import files

uploaded = files.upload()
```

```
from PIL import Image, ImageOps
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os

raw_data = (pd.read_csv('raw_images.csv')).to_numpy()
data_labels = raw_data[:,0]
data = np.empty((raw_data.shape[0],2304))
for i in range(data.shape[0]):
    data[i] = np.fromstring(raw_data[i][2], dtype=int, sep=" ")
print("INDEX:  4  EMOTION:", data_labels[4])
plt.imshow(data[4].reshape(48,48))
plt.set_cmap("gray")
plt.show()
print("INDEX:  5  EMOTION:", data_labels[5])
plt.imshow(data[5].reshape(48,48))
plt.set_cmap("gray")
plt.show()
print("INDEX:  6  EMOTION:", data_labels[6])
plt.imshow(data[6].reshape(48,48))
plt.set_cmap("gray")
plt.show()
```

```
INDEX:  4  EMOTION: 6
```



```
INDEX:  5  EMOTION: 2
```

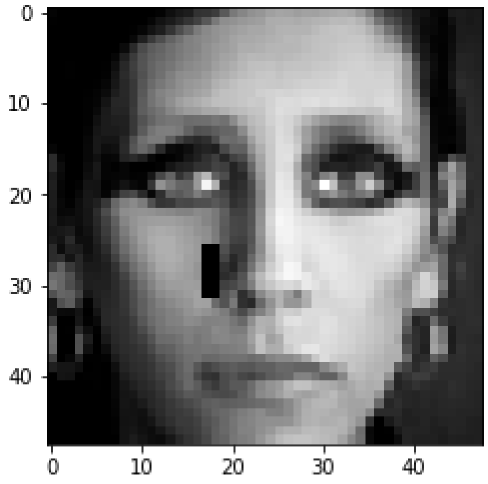
Pre-processing

In preprocessing our data, it was important to include multiple different factors. Because all of our data points were images, a single image data would include 2304 points of data. This was true of our training data, but not necessarily true of our experimental data, which came from live camera feed. There are multiple processes that are performed in order to acheive this ideal pre-processing point. We would first re-size the image to be a 48 x 48 array, like the size of the training data. We would then ensure that the image was grayscaled to greatly simplify the data being observed. This would lead into a process of normalization, which was used to eliminate and lighting effects that may be present within the image. Finally, in order to achieve one of our deliverables, we added small black boxes, with a maximum size of 6 x 2, to the image in random locations of the image. The box could also be randomly assigned a width or height of 0, so this would mean that no black box would be present. This would be used to represent scaring, as the intent was to ensure that the model did not become too dependent on certain features on the face that may be disfigured by a potential scar.

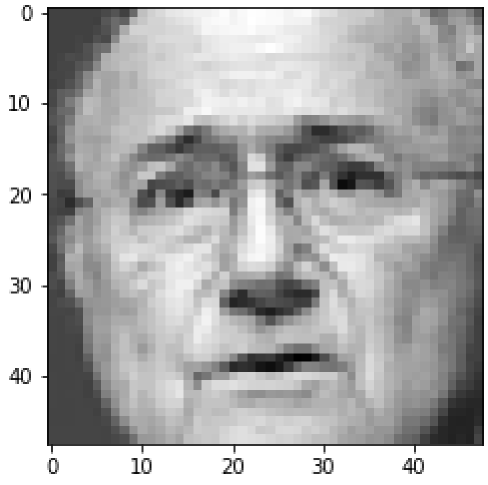
```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt

#Delete all instances of black boxes
raw_filtered = np.copy(raw_data)
num_black_boxes = 0
for i in range(raw_data.shape[0]):
      if np.std(np.fromstring(raw_data[i][2], dtype=int, sep=" ")) == 0:
          raw_filtered = np.delete(raw_filtered, obj=(i - num_black_boxes), axis=0)
          num_black_boxes += 1
```

```
#Create arrays and separate labels from images
labels = raw_filtered[:,0]
images = np.empty((raw_filtered.shape[0], 2304))
images_scar = np.empty((raw_filtered.shape[0], 2304))

#Store images in arrays
for i in range(images.shape[0]):
    images[i] = np.fromstring(raw_filtered[i][2], dtype=int, sep=" ")
    images_scar[i] = np.fromstring(raw_filtered[i][2], dtype=int, sep=" ")

#Add Random Scars
for i in range(images_scar.shape[0]):
    shape = np.random.randint(2) + 1
    x = np.random.randint(24) + 12
    y = 48 * (np.random.randint(24) + 12)
    l = np.random.randint(7)
    w = np.random.randint(3)

    if shape == 1: #vertical rectangle
        for j in range(l):
            for k in range(w):
                images_scar[i][x + y + (48*j) + k] = 0
    elif shape == 2: #horizontal rectangle
        for j in range(l):
            for k in range(w):
                images_scar[i][x + y + (48*k) + j] = 0

#Normalize the images
for i in range(images.shape[0]):
    images[i] = (images[i] - np.mean(images[i]))/(np.std(images[i]))
    images_scar[i] = (images_scar[i] - np.mean(images_scar[i]))/(np.std(images_scar[i

#Printing Images
print("INDEX:  4  EMOTION:", data_labels[4])
plt.imshow(images_scar[4].reshape(48,48))
plt.set_cmap("gray")
plt.show()
print("INDEX:  5  EMOTION:", data_labels[5])
plt.imshow(images_scar[5].reshape(48,48))
plt.set_cmap("gray")
plt.show()
print("INDEX:  6  EMOTION:", data_labels[6])
plt.imshow(images_scar[6].reshape(48,48))
plt.set_cmap("gray")
plt.show()
```

```
INDEX:  4  EMOTION: 6
```



```
INDEX:  5  EMOTION: 2
```



```
INDEX:  6  EMOTION: 4
```

Models and Evaluation

Experimental Setup

When observing our task and potential neural networks, we would look primarily at the dev accuracy to ensure that our model was performing properly. We evaluated in this manner because for this task, overall accuracy is the model important thing in correctly identifying emotion. When a happy face is displayed, calling that face anything but happy is not ideal, so accuracy itself becomes the most important for this task. For our loss function we utilized cross entropy to train our model. We did not look in depth at other loss functions, as cross entropy looks directly at the difference between two variables and returns a value based on how similar they are. In order to split our data into train and test sets, we took our 34,034 images and split them 80/10/10 to ensure a properly sized training, dev, and test split. This ensured that we would have about 27,227 training images, and 3,403 images in both the dev and test sets.

```
https://github.com/oween12/ML_Final_Project
```

Baselines

In order to establish a baseline for the neural network that we were creating, we observed published machine learning algorithms and papers on the same / similar topic. Emotion detection in faces is not a new subject and there is ample data on this topic. From our data collection we found that many papers were able to achieve accuracies of 80-90% so in seeing this, we set our sights about hitting the 80% mark. We believed that this would be a reasonable standard to achieve, providing an accurate model without going into as much depth as others, who may have had computers that were stronger and able to run deeper networks more rapidly. We would utilize this paper to get an understanding of where our model should begin and adjustments we could make therein to improve our models performance.

Methods

When going about this project, we chose to create a neural network for emotion classification. This decision involved adjusting the depth and width of the network continuously to achieve better results. Our process involved the use of five primary functions.

- torch.nn.Conv2d - this was used to create our layers of the neural network. This would require specifying a number for input channel, output channels, kernel size and padding. Throughout our neural network, the value for kernel size was help at 3, and the padding value was held at 1. The input and output channels changed between specific values, with layers of widths 32, 64, and 128.
- torch.nn.MaxPool2d - this function was used to pool the data that was being observed from the various convolutional layers. In this the kernel and stride was specified, indicating the size over which the max pool was occuring and the number of spaces skipped before the next max pool. In general, the max pool would achieve higher dev accuracies than an average pool function in the same places.

- torch.nn.BatchNorm2d - this function was used to normalize data before utilizing this data in the relu activation function. This would help to simplify the process of going through the relu function, decreasing necessary time to run, but this also had the effect of increasing the overall accuracy of the model.
- torch.nn.Dropout - this function was used to include a dropout layer, where, with a specified percent (such as 0.25), a random subset of the data is removed when training. This helps to prevent overfitting and dependencies on certain features or images to make certain classifications.
- F.relu - this was the activation function that was applied to our normalized convolutional neural network layers. We wont go into detail about how this function works, as it was covered in depth in class, but we wanted to mention that this was the primary activation function used in our model.

Outside of the creation of our network, we had various hyperparameters that needed to be chosen, including the learning rate, number of epochs, and batch size. The learning rate we settled upon was 0.05, which provided a good balance of finding a good solution in a reasonable amount of time. Lower learning rates, such as 0.01 could take 1-2 hours to train, and for our application this would be unreasonable. The number of epochs that we utilized was 1000, as we typcially stopped seeing any beneficial increase after this point, so further training would only likely lead to overfitting. The batch size that we utilized was a suggestion from one of the papers on deep learning for this application, 256.

In order to do real-time emotion classification, we had to perform facial recognition and work with a video stream. In order to perform facial recognition, we used a well known algorithm called Haar-Classifiers. That, paired with the python package OpenCV allowed us to perform real-time emotion classification. The video stream was parsed by its respective frames. For each frame, Haar-Classifiers was used to look for faces. These faces were then extracted, resized into the appropriate 48x48 dimensions, grayscaled, and passed into our trained network. The output was then the live video stream with a box around the face and the emotion detected on the screen as well.

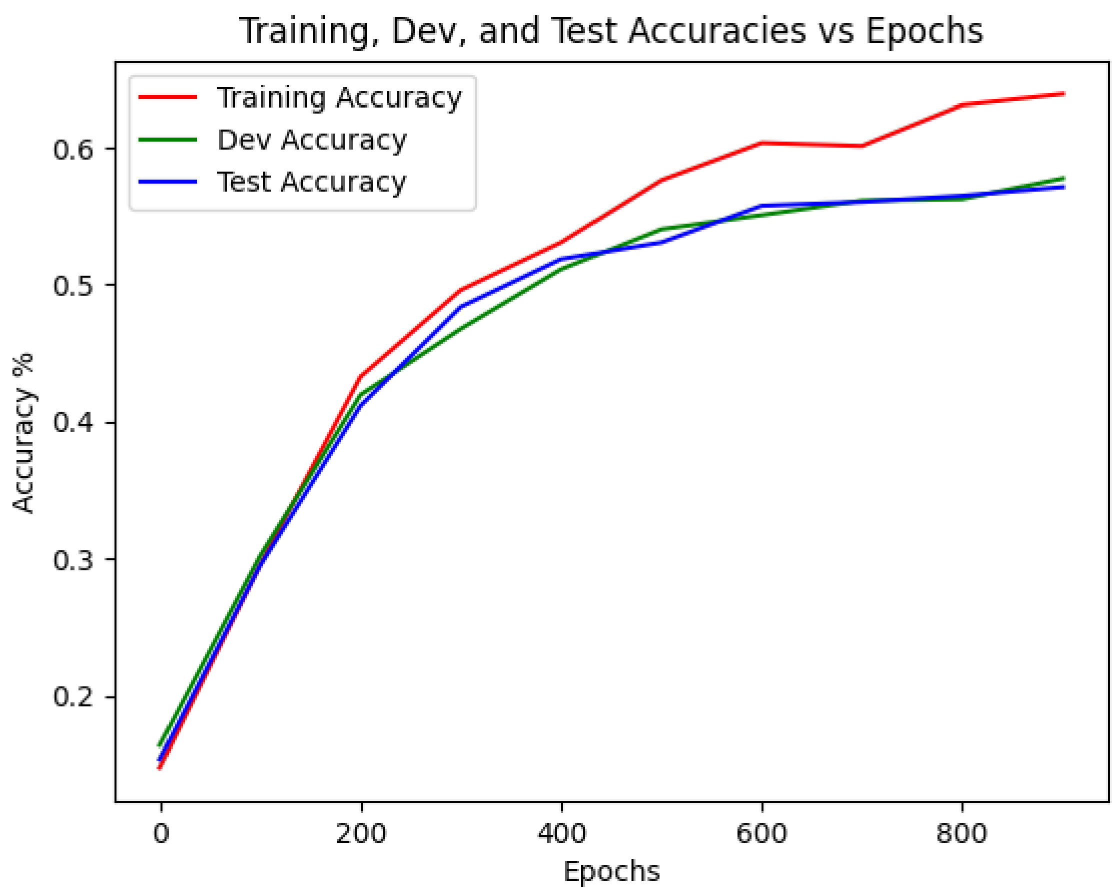
```
https://github.com/oween12/ML_Final_Project
```

```
from google.colab import files
uploaded = files.upload()
```

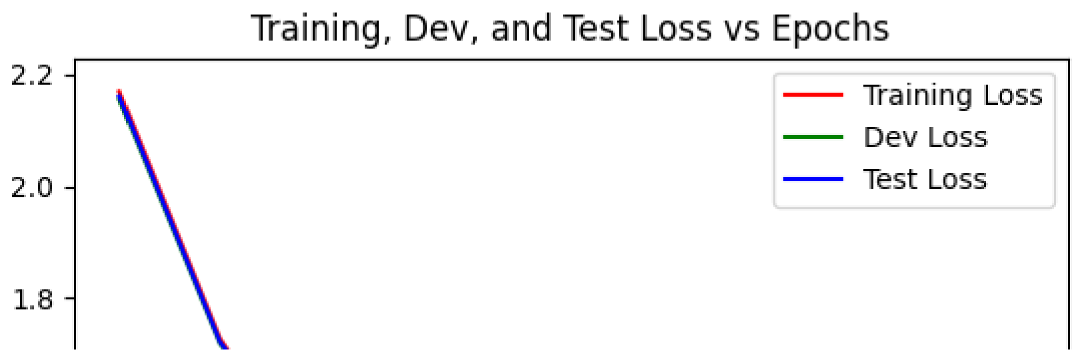
Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving best_acc_graph.png to best_acc_graph (1).png
Saving best_loss_graph.png to best_loss_graph.png
```

```
from PIL import Image, ImageOps
import matplotlib.pyplot as plt

acc_graph = Image.open('best_acc_graph.png')
acc_graph
```



```
loss_graph = Image.open('best_loss_graph.png')
loss_graph
```



Results

After observing how our model performs, we see some key detractions from our baseline. We had originally wanted to achieve a validation accuracy of approximately 80%, but when observing how our classifier performed on the data, our best model was only able to achieve approximately 55% test and dev accuracy, a significant drop from our expected value. This came as somewhat of a shock, as both time and hardware limitations became large issues. In terms of hardware limitations, for me personally, running on a likely suboptimal PC, increasing the width or depth of the network too much would cause severe performance issues and would most times end with my computer crashing.

Looking at our model performance, we do not believe that our model overfit or underfit, we made careful adjustments to the learning rate, and number of epochs. We paid close attention to the number of epochs and attempted to avoid overfitting by having too many epochs, or underfitting by having too few. In addition, including the random placement of scars and including a dropout layer help to prevent the model from overfitting to certain features that may have been present within the data.

Discussion

What you've learned

This project overall was quite eye opening. It was amazing to see the power of machine learning and work through solving a problem from the ground up. What came as a shock to us was the importance of data. Starting with the small Yale database, there was not much we could do. Data augmentation can only take us so far and we believe that the model would overfit the data no matter what we did. The massive dataset from Kaggle was much more useful. With it, we could take hundreds of different batches, all with different emotions and people expressing those emotions.

This allowed our model to be able to perform at a higher level and be used for much broader applications.

Given two more weeks to work on this project, there is definitely more that we would like to accomplish. First and foremost, we would like to see if there is a way to, without deepening or widening our network very much, to increase the validation accuracy to the baseline 80% that we had wanted to achieve. In addition, some of the missing deliverables with respect to racial bias within our model could be more clearly and expressely evaluated. In addition to this, we would like to approach a method to remove these biases. We would also like to investigate how the removal of certain facial features would affect model performance, such as completely covering an eye, or the mouth. It would be interesting to see if the model can still accurately predict emotion given such limitations.