Final Project

Project Topic
Face classification and detection.

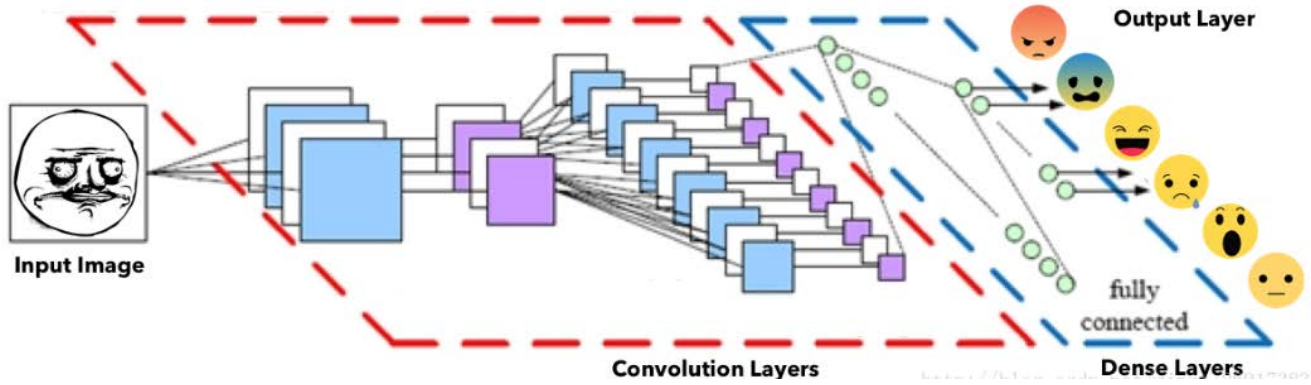
Group Members
106060012 陳品諼
106061139 邱郁閔

Project Description
我們希望可以藉由照片或影片來預測人的心情，因為人的情緒有很多種類，但我們把它總體歸為 7 類（Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral），使用的資料來源 kaggle 的一個比賽，這個資料集中，有較難辨識的圖片，也有不是正臉的圖片，因此較為高難度。

Dataset
The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image.

train.csv contains two columns, "emotion" and "pixels". The "emotion" column contains a numeric code ranging from 0 to 6（0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral.）, inclusive, for the emotion that is present in the image. The "pixels" column contains a string surrounded in quotes for each image. The contents of this string a space-separated pixel values in row major order. test.csv contains only the "pixels" column and your task is to predict the emotion column.
https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data

Strategies for the Project



-數據預處理

下載下來的 fer2013.csv 每一 row 包含 48*48 pixel grayscale 和他所屬的 emotion，把 pixel 列換轉換成 48row48cloumn 的 array

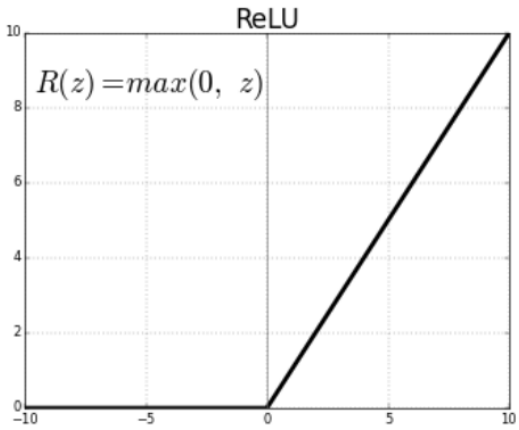
把全部資料的 90%分成 train data 另外 10%為 test data

-使用 Keras 來搭建 Convolutional Neural Network（CNN）,CNN 包含:

- Convolution Operation
- Pooling
- Fully Connected Networks

- Convolution Operation & Pooling

- num_features 輸出空間的維度(即卷積中濾波器的輸出數量)，設為 64
- input_shape 讓輸入的影像有一致的格式，設為 48*48，1 表示灰階
- activation function 設定為 relu



- data_format 輸入的各個維度順序。"channels_last" 對應輸入尺寸為 (batch, steps, channels)
- padding='same'會用 zero-padding 的手法，讓輸入的圖不會受到 kernel map 的大小影響
- BatchNormalization 標準化數據
- max pooling 作用是降躁跟減少運算資源，pool_size 使用 2x2 的濾波器，strides 作為縮小比例的因數設為(2, 2)
- Dropout 是一種對抗過擬合的正則化方法，在訓練時每一次的迭代 (epoch)皆以一定的機率丟棄隱藏層神經元，而被丟棄的神經元不會傳遞訊息，每一層以 0.5 的機率丟棄神經元，所以再向前傳播時紅色被打叉的神經元不會傳遞訊息。
- Convolution Operation 搭配一個 Pooling

```
#desinging the CNN
model = Sequential()

model.add(Conv2D(num_features, kernel_size=(3, 3), activation='relu', input_shape=(width, height, 1), data_format='channels_last', kernel_regul
model.add(Conv2D(num_features, kernel_size=(3, 3), activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.5))
```

同上，搭建地 2,3,4 層的 Convolutional Layer

- Flatten

將 feature maps 攤平放入一個向量中

```
model.add(Flatten())
```

- Fully Connected Networks

activation='softmax' Softmax 回歸是使用 Softmax 運算使得最後一層輸出的機率分佈總和為 1

```
model.add(Dense(2*2*2*num_features, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(2*2*num_features, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(2*num_features, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(num_labels, activation='softmax'))
```

- Compiling the CNN

- optimizer 是選擇優化梯度下降的演算法，使用 `adam`
- loss 是選擇 loss function，這邊使用 `binary_crossentropy`
- 最後一個是選擇 performance metric

```
#Compliling the model with adam optimixer and categorical crossentropy loss
model.compile(loss=categorical_crossentropy,
              optimizer=Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-7),
              metrics=['accuracy'])
```

-train model

把剛剛資料放進做好的 CNN 演算法，訓練參數

Train on 29068 samples， validate on 3230 samples，epochs=100

```
#training the model
model.fit(np.array(X_train), np.array(y_train),
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(np.array(X_valid), np.array(y_valid)),
          shuffle=True)
```

-save model

使用 h5py 來儲存訓練好的參數

```
#saving the  model to be used later
fer_json = model.to_json()
with open("fer.json", "w") as json_file:
    json_file.write(fer_json)
model.save_weights("fer.h5")
```

-Accuracy

把 test 丟進 model 裡，看他的結果為 64%

我們所下載的資料為 fer2013 因為這個資料裡有很多側臉或較難辨識的表情，因此正確率不高，會使用這個 dataset 是因為在這個資料有用來作比賽，而當時最高的正確率為 71%，也並不高，因此這個正確率還算可以。

```
Accuracy on test set :64.7812761214823%
```

-測試

・接下來我們在找了一些圖片測試這個 model，包括'Angry', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprise', 'Neutral'七種。

・使用 OpenCV 來讀取圖片和徵測臉部

・以 cv2.imread 讀進來的資料，會儲存成一個 NumPy 的陣列，此 NumPy 陣列的前兩個維度分別是圖片的高度與寬度，第三個維度則是圖片的 channel（RGB 彩色圖片的 channel 是 3，灰階圖片則為 1），並用 cvtColor 轉成灰階圖片

```
full_size_image = cv2.imread("happy.jpg")
print("Image Loaded")
gray=cv2.cvtColor(full_size_image,cv2.COLOR_RGB2GRAY)
```

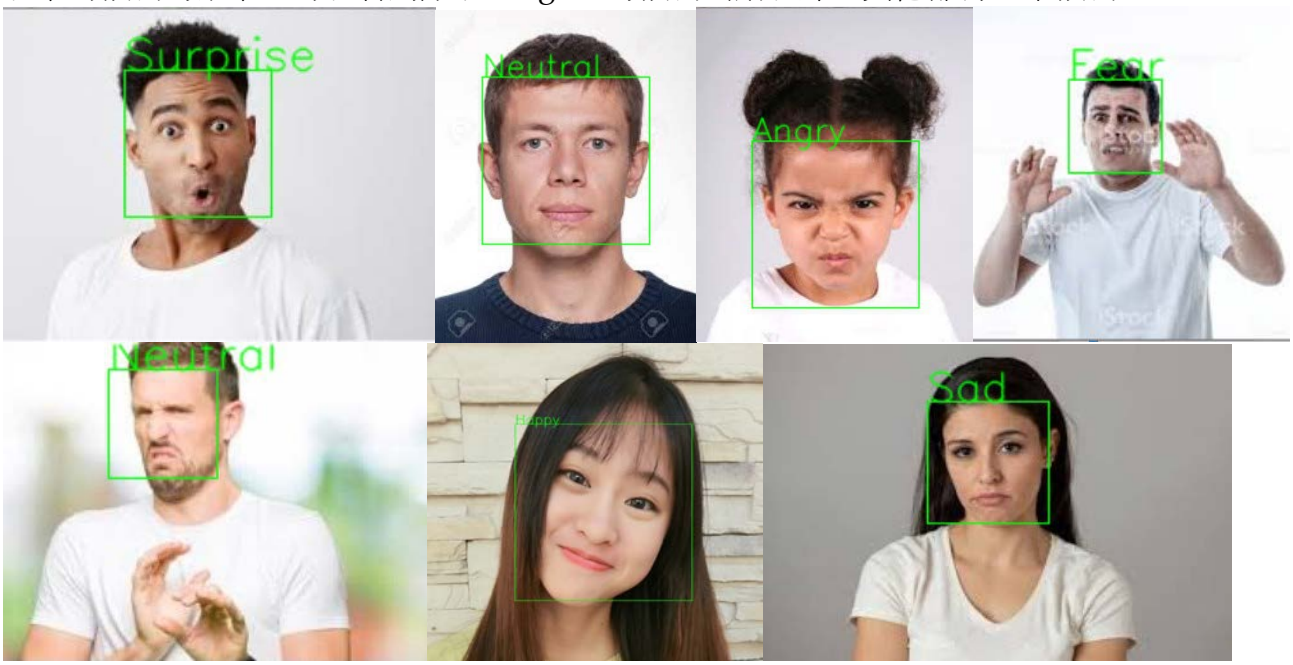
使用 OpenCV 來做臉部辨識，然後把圖片數據丟到 model 裡做預測

```
face = cv2.CascadeClassifier(cv2.data.haarcascades+"haarcascade_frontalface_default.xml")
faces = face.detectMultiScale(gray, 1.3 , 10)

#detecting faces
for (x, y, w, h) in faces:
        roi_gray = gray[y:y + h, x:x + w]
        cropped_img = np.expand_dims(np.expand_dims(cv2.resize(roi_gray, (48, 48)), -1), 0)
        cv2.normalize(cropped_img, cropped_img, alpha=0, beta=1, norm_type=cv2.NORM_L2, dtype=cv2.CV_32F)
        cv2.rectangle(full_size_image, (x, y), (x + w, y + h), (0, 255, 0), 1)
        #predicting the emotion
        yhat= loaded_model.predict(cropped_img)
        cv2.putText(full_size_image, labels[int(np.argmax(yhat))], (x, y), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 1, cv2.LINE_AA)
        print("Emotion: "+labels[int(np.argmax(yhat))])

cv2.imshow('Emotion', full_size_image)
cv2.waitKey()
```

以下為預測的結果，可以看到除了'Disgust'的預測出錯以外，其他都有正確預測。



及時影片辨識

Conclusion

在這次的報告中仔細了解了 CNN 的構造，並且將 CNN 在 python 上實作。雖然最後正確率只有 64.7%，但是依照這個 dataset 來論，已經算是蠻滿意的結果，在途中也本來打算要放棄這個題目，所以我們也有另外做一個股票值的未來分析，還好最後 train 出來的結果是不錯的。這次的 final 讓我們學到了很多，包括了 CNN，也了解了圖像、影像的處理，最後成功地做出了圖片和隨時攝像頭的人臉情緒辨識。