COMP 3710

APPLIED ARTIFICIAL INTELLIGENCE

AI INDIVIDUAL PROJECT

FACIAL EMOTION RECOGNITION APPROACH BASED ON CONVOLUTIONAL NEURAL NETWORK USING TENSORFLOW AND KERAS

HEMANTH ANIL
T00564146

TABLE OF CONTENTS

Introduction

In this research paper, the author presents a Facial emotion recognition approach based on Convolutional Neural Network using Tensorflow and Keras as backend. The author used a dataset from kaggle and this provides an easy and secure way of login to a virtual world. The work achieved an accuracy of 99%.

Implementation

- Pycharm 2019.2.3
- Tensorflow 2.0.0
- Numpy: 1.17.4 - The mathematical library used for computing multi-dimensional arrays and matrices very fast and easy.
- Keras : 2.3.1
- Keras-preprocessing: 1.1.0
- Scikit-learn : 0.21.3
- Pandas : 0.25.3 - Pandas are used for data manipulation and analysis.
- Matplotlib: 3.1.1 - Plotting library used to show charts, plotting lines and even images.
- Os - inbuilt library used to access files from the computer. Displays folders and subfolders.
- Cv2 - also called as OpenCv is an image and video processing library.
- macOS High Sierra - 4GB Memory, Processor 1.6 GHZ Intel Core i5.

Background

Facial expression recognition has a wide range of applications in human-machine interaction, pattern recognition, image understanding, machine vision, and other fields. It has gradually become hot research. Moreover, the author found that it will be a great multi-factor authentication to integrate into other IoT projects. The author is a fourth-year Computing Science Student and has good background knowledge in deep learning concepts and have created many projects using DL4J.

Project details

Data

The author used a public dataset from the Kaggle website for training and test sets. The dataset consists of 7 emotions: Anger (136), Fear (76),

Sadness (85), Happy (208), Disgust (178), Contempt (55) and Surprise (250). The project starts by importing the images from the Kaggle website and unzipping the file.
The images are converted into a numpy array and then normalized by dividing by 255. The shape of the image array is very important in Keras Model. It takes an input array of (height, width, channel). The author used Scikit to learn to split arrays into test and train subsets.

Model

Keras has a different layers module that contains different types of layers used. For this research, the author will be importing the Convolutional layer. Keras has Sequential and Functional API models. Sequential is a widely used model for image recognition and classification. Therefore, the Sequential model is used for this research. In terms of the optimizer, Keras offers many optimizers such as rmsprop, adam, sgd(stochastic gradient descent). Adam optimizer is used for this research.

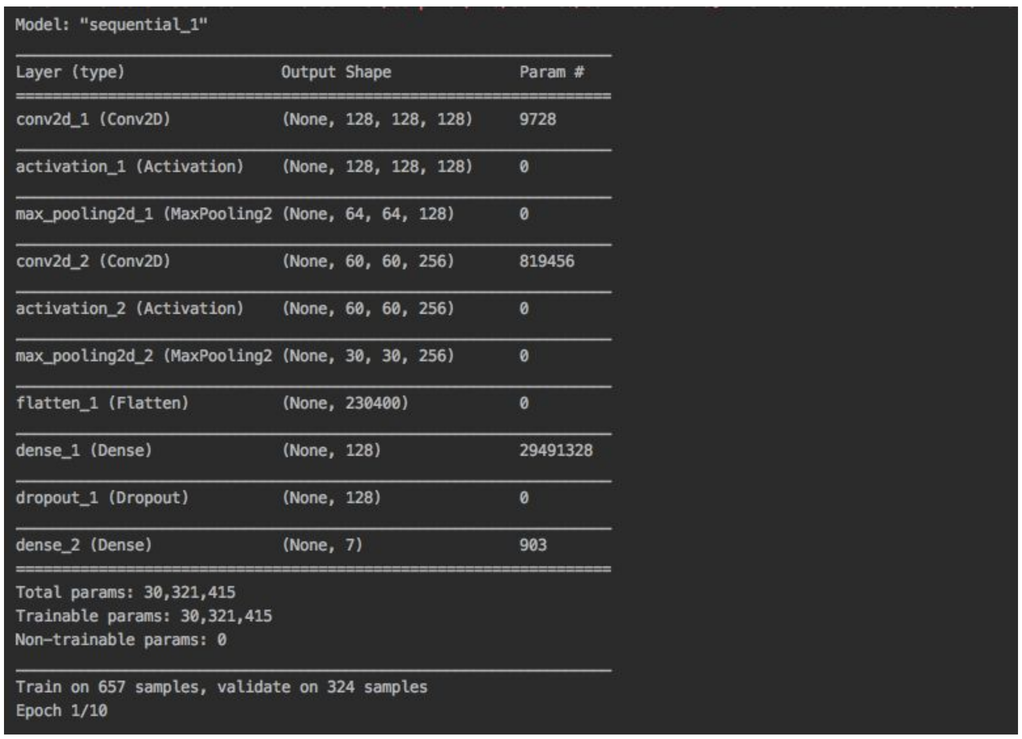
```
input_shape = (128, 128, 3)

model = Sequential()


model.add(Conv2D(128, (5, 5), input_shape=input_shape, padding='same'))
#model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(256, (5, 5)))
#model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(7, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=["accuracy"])
model.summary()
```

A sequential model was created by telling Keras to stack all layers sequentially. Using the add function we add the first layer and the type of layer as a Conv2D. This first layer is called the input layer and consists of different parameters (Filter size: 128; Kernel size: [5,5]). Filter size is the output dimension and kernel size is the height and weight of the 2D convolutional window. Next, the activation function is defined for the neural network. used ‘relu’ which is commonly used due to its strong representation of non-linearity. Relu stands for the Rectified Linear Unit. In the beginning, the input shape is defined as (128, 128, 3).

A MaxPool2D layer is defined to reduce the spatial size of incoming features and thus reduce the number of parameters and computation of the network thereby reducing overfitting. To avoid the network to memorize we use other layers such as Flatten, Dense.
The Conv2D takes and learn the spatial features, passed into a dense layer where it is flattened. We add a Dropout layer with a value of 0.2 which drops randomly some layers in a neural network. For activation ‘Softmax’ is used. The softmax function is used for the multi-classification model and it returns the probabilities of each class and the target class will have a high probability.

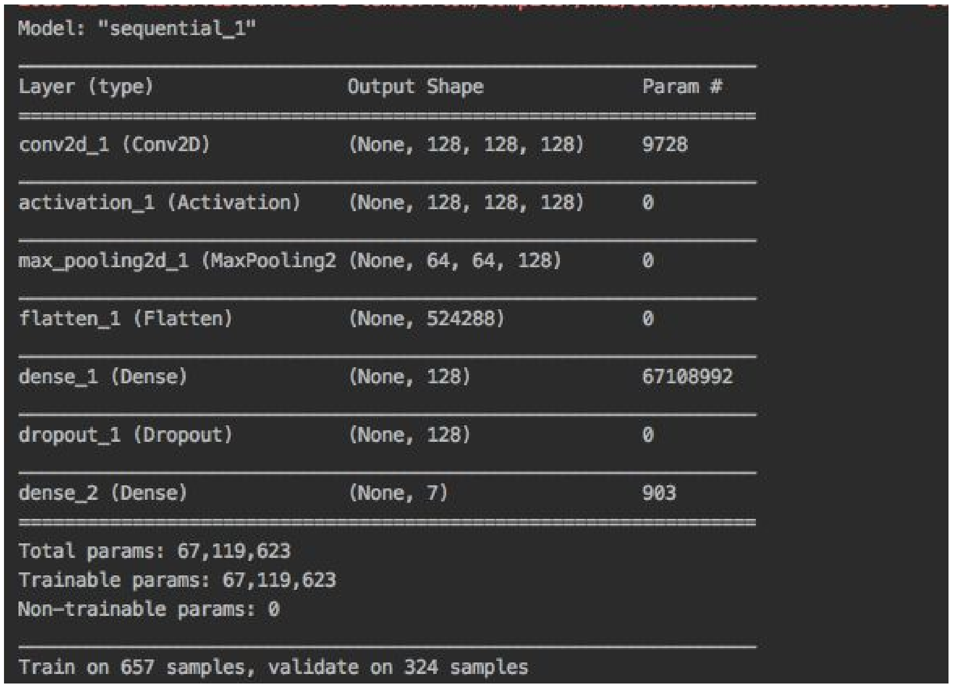
Using Keras function ‘.summary()’ we can preview the model.

Model:1 Total parameters:30,321,415



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 128, 128, 128)     9728
_________________________________________________________________
activation_1 (Activation)    (None, 128, 128, 128)     0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 60, 60, 256)       819456
_________________________________________________________________
activation_2 (Activation)    (None, 60, 60, 256)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 30, 30, 256)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 230400)            0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               29491328
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 903
=================================================================
Total params: 30,321,415
Trainable params: 30,321,415
Non-trainable params: 0
_________________________________________________________________
Train on 657 samples, validate on 324 samples
Epoch 1/10
```

Model:2 Total parameters: 67,119,623



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 128, 128, 128)     9728
_________________________________________________________________
activation_1 (Activation)    (None, 128, 128, 128)     0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 64, 64, 128)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 524288)            0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               67108992
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 903
=================================================================
Total params: 67,119,623
Trainable params: 67,119,623
Non-trainable params: 0
_________________________________________________________________
Train on 657 samples, validate on 324 samples
```

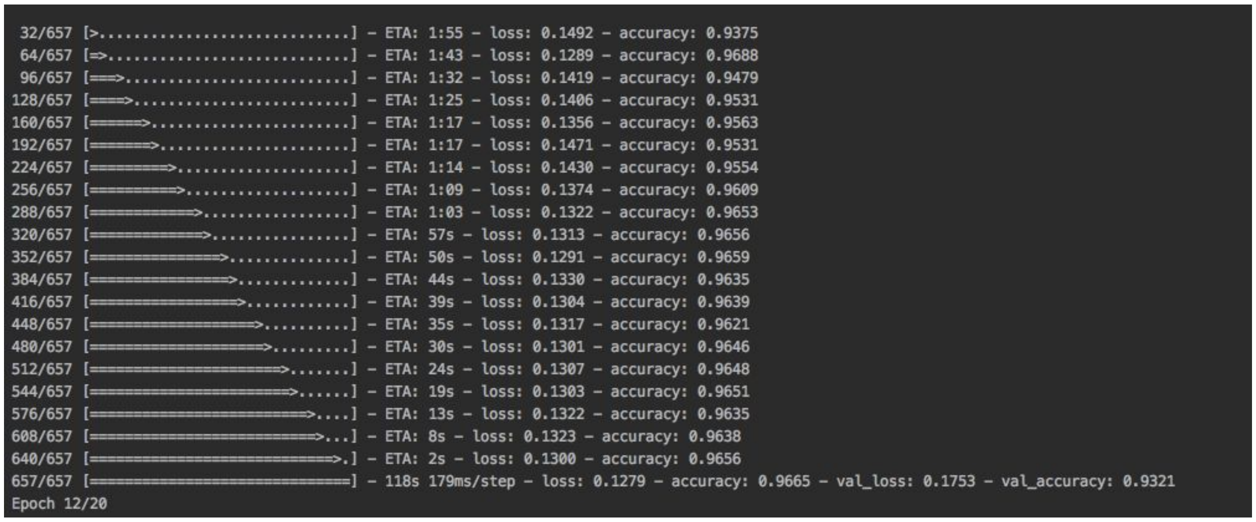
Compile model

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=["accuracy"])
model.summary()
```

There are three parameters. The loss: ‘categorical_crossentropy’ - loss function that the optimizer uses to minimize. The optimizer used ‘adam’ and metrics is measuring the model’s performance after training. In this case, the author is using ‘accuracy’ since it is a classification problem.

Using fit function the author trains the network with some parameters. Epochs mean we want to go over the training data 20 times.

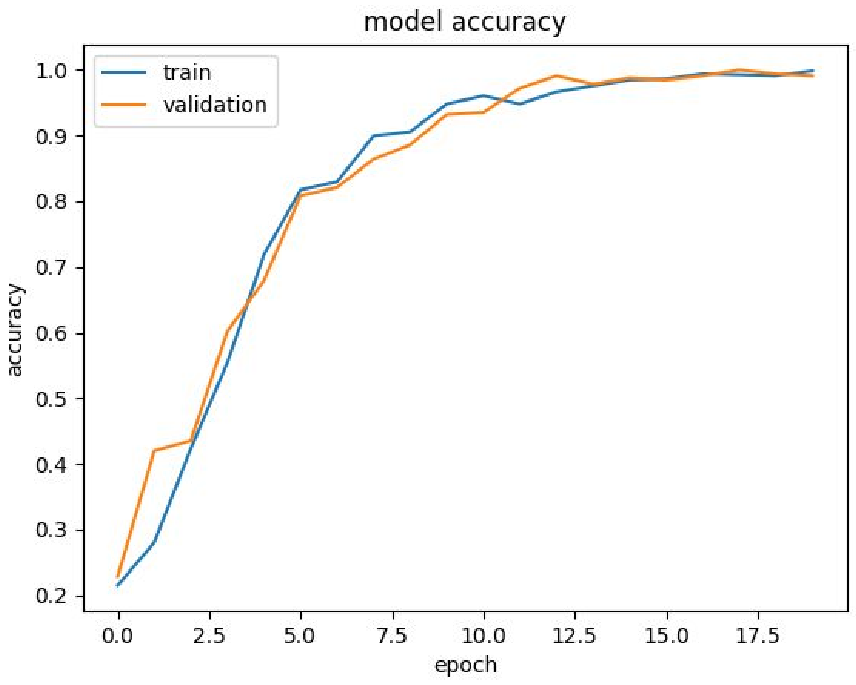
```
hist = model.fit(X_train, y_train, batch_size=64, epochs=20, verbose=1,
validation_data=(X_test, y_test),
                callbacks=callbacks_list)
model.save_weights('model_weights.h5')
model.save('model_keras.h5')
```

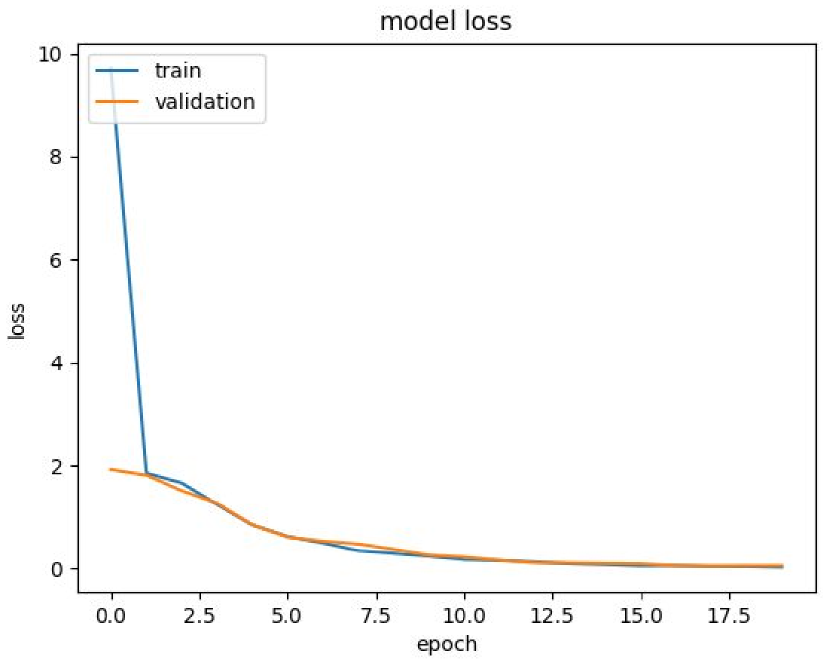


```
 32/657 [>.............................] - ETA: 1:55 - loss: 0.1492 - accuracy: 0.9375
 64/657 [=>............................] - ETA: 1:43 - loss: 0.1289 - accuracy: 0.9688
 96/657 [===>..........................] - ETA: 1:32 - loss: 0.1419 - accuracy: 0.9479
128/657 [====>.........................] - ETA: 1:25 - loss: 0.1406 - accuracy: 0.9531
160/657 [======>.......................] - ETA: 1:17 - loss: 0.1356 - accuracy: 0.9563
192/657 [=======>......................] - ETA: 1:17 - loss: 0.1471 - accuracy: 0.9531
224/657 [=========>....................] - ETA: 1:14 - loss: 0.1430 - accuracy: 0.9554
256/657 [==========>...................] - ETA: 1:09 - loss: 0.1374 - accuracy: 0.9609
288/657 [============>.................] - ETA: 1:03 - loss: 0.1322 - accuracy: 0.9653
320/657 [=============>................] - ETA: 57s - loss: 0.1313 - accuracy: 0.9656
352/657 [===============>..............] - ETA: 50s - loss: 0.1291 - accuracy: 0.9659
384/657 [================>.............] - ETA: 44s - loss: 0.1330 - accuracy: 0.9635
416/657 [=================>............] - ETA: 39s - loss: 0.1304 - accuracy: 0.9639
448/657 [===================>..........] - ETA: 35s - loss: 0.1317 - accuracy: 0.9621
480/657 [====================>.........] - ETA: 30s - loss: 0.1301 - accuracy: 0.9646
512/657 [======================>.......] - ETA: 24s - loss: 0.1307 - accuracy: 0.9648
544/657 [=======================>......] - ETA: 19s - loss: 0.1303 - accuracy: 0.9651
576/657 [=========================>....] - ETA: 13s - loss: 0.1322 - accuracy: 0.9635
608/657 [==========================>...] - ETA: 8s - loss: 0.1323 - accuracy: 0.9638
640/657 [============================>.] - ETA: 2s - loss: 0.1300 - accuracy: 0.9656
657/657 [==============================] - 118s 179ms/step - loss: 0.1279 - accuracy: 0.9665 - val_loss: 0.1753 - val_accuracy: 0.9321
Epoch 12/20
```

Results

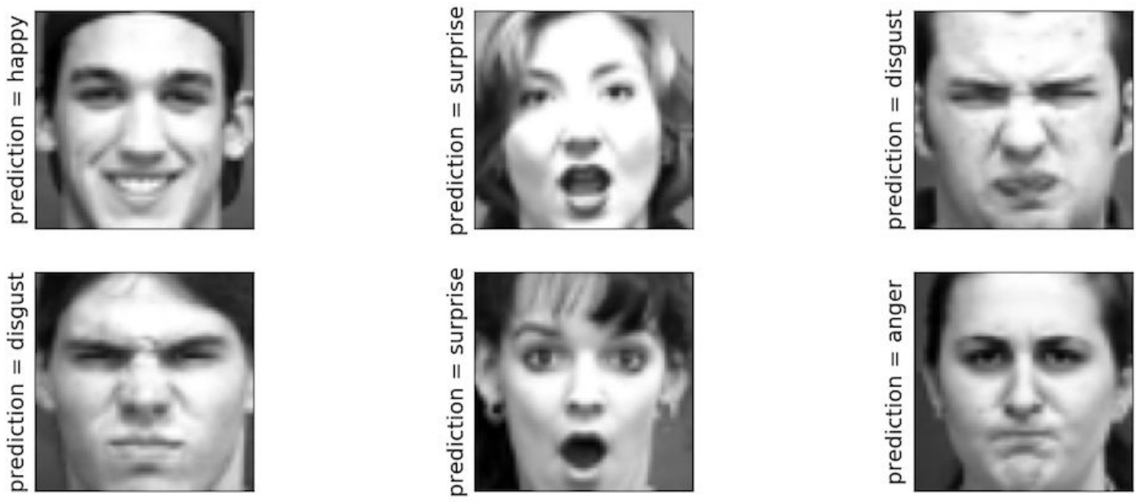
Epoch = 20





```
loss: 0.0290 - accuracy: 0.9985 - val_loss: 0.0566 - val_accuracy: 0.9907
Test Loss: 0.05661149072046909
Test accuracy: 0.9907407164573669
```

Predictions



Discussion and conclusion

In this research paper, I have presented my approach to Emotion recognition using CNN. The objective of this work was to achieve high accuracy in results.
From the experiment using an epoch of 20, the model was able to learn better and produce an accuracy of 99.07%. The aim was to make the validation loss as low as possible to avoid overfitting. But in this case, overfitting is nearly always a good thing. This often happens when the training data is quite low.
All trained models, weights and trained results are attached for further references and replication.

Future plan

In the future, I will be focusing more on emotion recognition using my images as an authentication factor to unlock my door. For this, I will be training my images of different emotions and running through an autoencoder and finding the shortest distance. Then set a threshold to allow authentication or not. Addition, by using batch normalisation, it shows an increase in learning in training process of a neural network. Also, different people have different ways of expressing their emotions, and under the influence of brightness, background and other factors that might affect the accuracy.

References

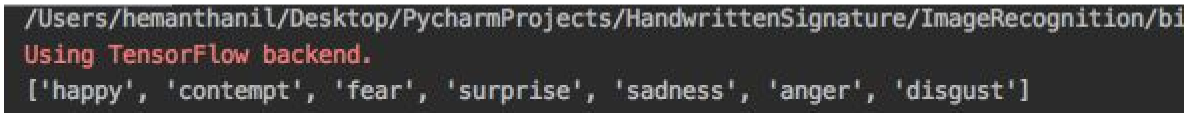
https://www.tensorflow.org/tutorials/images/classification
https://www.tensorflow.org/tutorials/keras/classification
https://youtu.be/6g4O5UOH304
https://cv-tricks.com/tensorflow-tutorial/training-convolutional-neural-network-for-image-classification/
https://stackoverflow.com/questions/44209071/python-os-listdir-reading-subfolders
https://docs.scipy.org/doc/numpy-1.13.0/user/basics.types.html
https://keras.io/utils/
https://keras.rstudio.com/articles/tutorial_save_and_restore.html
https://stackoverflow.com/questions/36952763/how-to-return-history-of-validation-loss-in-keras

Padgett, C., & Cottrell, G. (1995, November). Identifying emotion in static face images. In *Proceedings of the 2nd Joint Symposium on Neural Computation* (Vol. 5, pp. 91-101).

Additional Information

- If replicating my model. Make sure to delete the mac hidden file using the command on Terminal:
  find . -name '.DS_Store' -type f -delete
- Keras : 2.3.1 was used for this project. Therefore there are some parameters that need to be changed.
- For Windows users - Order of the labels change alphabetically in different Operating systems. Make sure to print the file list to check the correct order.

```
# List the directories in the current order
data_dir_list = os.listdir(data_path)
print(data_dir_list)
```



```
/Users/hemanthanil/Desktop/PycharmProjects/HandwrittenSignature/ImageRecognition/bi
Using TensorFlow backend.
['happy', 'contempt', 'fear', 'surprise', 'sadness', 'anger', 'disgust']
```