Suleyman Demirel University

Human Face Emotion Detection

**Instructor:** Sultanova Nazerke

**Prepared by:** Akhmetova Sayagul 160104059

Mukanova Gaukhar 160107036

Mussin Nurlybek 160107015

Kaskelen, 2018

**Content**

**Introduction**

1. Project description
2. Aim and hypothesis
3. Actuality

**Main**

1. Problem definition
2. Methods
3. Algorithm
4. Application

**Conclusion**

**References**

**Introduction**

Nowadays, one of the more investigated fields of unstructured data analytics in Machine Learning is image processing and recognition. It is used to identify unknown patterns, inherent and valuable information from images. Image classification helps to make relationships between different categories of images which are found in large image databases. These images can reveal useful information to the users.

According to our knowledge, we presented a project about human face image representing. The given work includes a possibility to detect on human faces their emotions and share with results. The project had been done according to the following Project requirements: data, ML algorithms, application and documentation.

Aim: use data to create a model that will predict and define the emotions of person.

Hypothesis: if we use the data in the right way, then the emotion of person will be detected correctly.

Actuality of this topic lies in the fact that in the age of technology, face detection is very popular and beneficial. This system is used to create protection of various buildings, safes, jewelry and money from thieves and etc. In addition, this is helpful for communication between computers and human, even a Smartphone can determine the human mood due to different emotions. Also, image classification is one of the features of artificial intelligence.

**Main**

* ***Problem definition***

The task is to create a model that will detect emotions. 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. The task is to categorize each face based on the emotion shown in the facial expression in to one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).

train.csv contains two columns, "emotion" and "pixels". The "emotion" column contains a numeric code ranging from 0 to 6, 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 our task is to predict the emotion column.

The training set consists of 28,709 examples. The public test set consists of 3,589 examples.

* ***Methods***

In this project work we have used *Keras sequential model*. The sequential API allows us to create models layer-by-layer for most problems. It is limited in that it does not allow to create models that share layers or have multiple inputs or outputs. The Sequential model API is great for developing deep learning models in most situations, but it also has some limitations. For example, it is not straightforward to define models that may have: multiple different input sources, produce multiple output destinations, or models that re-use layers.

However, there is *Convolution2D* layer that creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. If use bias is True, a bias vector is created and added to the outputs. Finally, if activation is not None, it is applied to the outputs as well.

*Core layers like Dense and Dropout*, which perform classification on the features extracted by the convolutional layers and downsampled by the pooling layers. In a cort layer, every node in the layer is connected to every node in the preceding layer.

**Algorithm**

First, import needed libraries:

import keras  
from keras.models import Sequential  
from keras.layers import Conv2D, MaxPooling2D, AveragePooling2D  
from keras.layers import Dense, Dropout, Flatten  
from keras.preprocessing.image import ImageDataGenerator  
import numpy as np

Then we read the dataset and create an instance.

num\_classes = 7 # angry, disgust, fear, happy, sad, surprise, neutral  
batch\_size = 256  
epochs = 5  
  
with open("./dataset/fer2013.csv") as f:  
 content = f.readlines()  
  
lines = np.array(content)  
  
num\_of\_instances = lines.size  
print("number of instances: ", num\_of\_instances)  
print("instance length: ", len(lines[1].split(",")[1].split(" ")))

Preprocessing data

x\_train, y\_train, x\_test, y\_test = [], [], [], []  
  
for i in range(1, num\_of\_instances):  
 try:  
 emotion, img, usage = lines[i].split(",")  
  
 val = img.split(" ")  
  
 pixels = np.array(val, 'float32')  
  
 emotion = keras.utils.to\_categorical(emotion, num\_classes)  
  
 if 'Training' in usage:  
 y\_train.append(emotion)  
 x\_train.append(pixels)  
 elif 'PublicTest' in usage:  
 y\_test.append(emotion)  
 x\_test.append(pixels)  
 except:  
 print("", end="")  
  
x\_train = np.array(x\_train, 'float32')  
y\_train = np.array(y\_train, 'float32')  
x\_test = np.array(x\_test, 'float32')  
y\_test = np.array(y\_test, 'float32')  
  
x\_train /= 255  
x\_test /= 255  
  
x\_train = x\_train.reshape(x\_train.shape[0], 48, 48, 1)  
x\_train = x\_train.astype('float32')  
x\_test = x\_test.reshape(x\_test.shape[0], 48, 48, 1)  
x\_test = x\_test.astype('float32')  
  
print(x\_train.shape[0], 'train samples')  
print(x\_test.shape[0], 'test samples')

Creating neural network

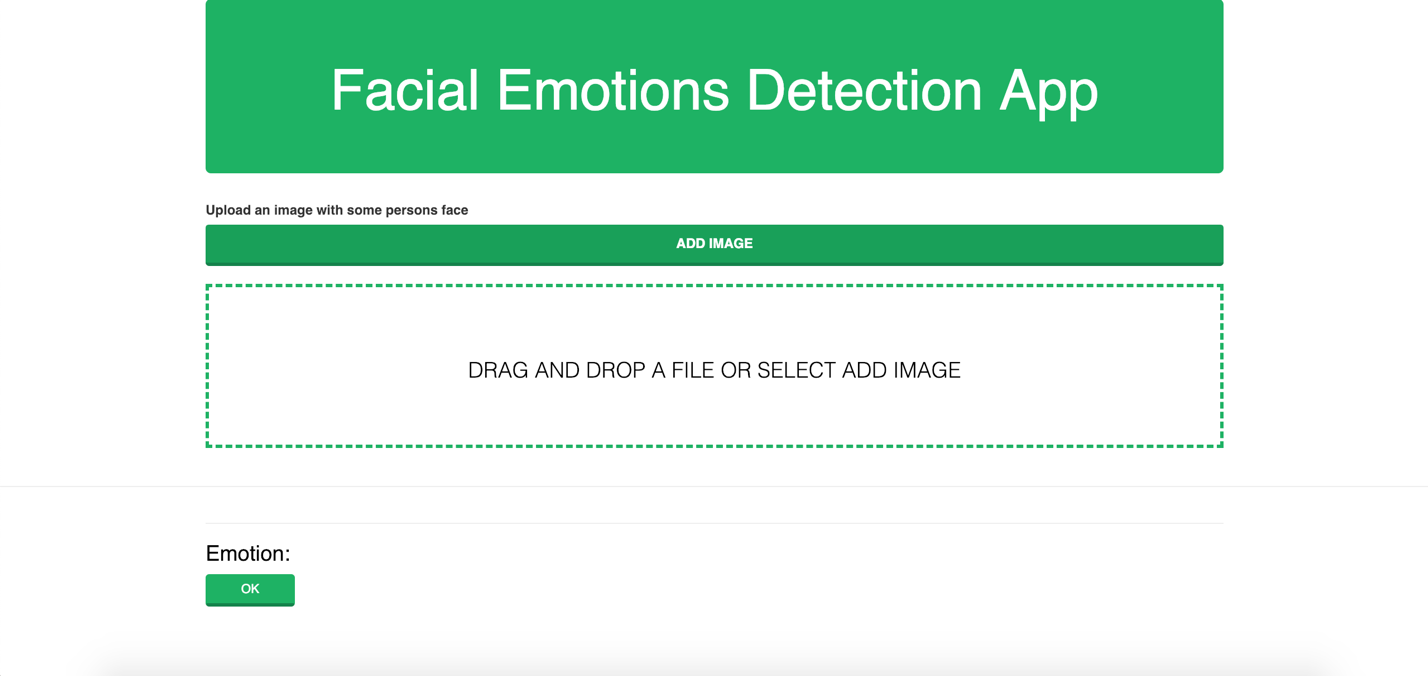
model = Sequential()  
  
# 1st convolution layer  
model.add(Conv2D(64, (5, 5), activation='relu', input\_shape=(48, 48, 1)))  
model.add(MaxPooling2D(pool\_size=(5, 5), strides=(2, 2)))  
  
# 2nd convolution layer  
model.add(Conv2D(64, (3, 3), activation='relu'))  
model.add(Conv2D(64, (3, 3), activation='relu'))  
model.add(AveragePooling2D(pool\_size=(3, 3), strides=(2, 2)))  
  
# 3rd convolution layer  
model.add(Conv2D(128, (3, 3), activation='relu'))  
model.add(Conv2D(128, (3, 3), activation='relu'))  
model.add(AveragePooling2D(pool\_size=(3, 3), strides=(2, 2)))  
  
model.add(Flatten())  
  
# fully connected neural networks  
model.add(Dense(1024, activation='relu'))  
model.add(Dropout(0.2))  
model.add(Dense(1024, activation='relu'))  
model.add(Dropout(0.2))  
  
model.add(Dense(num\_classes, activation='softmax'))  
  
gen = ImageDataGenerator()  
train\_generator = gen.flow(x\_train, y\_train, batch\_size=batch\_size)  
  
model.compile(loss='categorical\_crossentropy'  
 , optimizer=keras.optimizers.Adam()  
 , metrics=['accuracy']  
 )  
  
model.fit\_generator(train\_generator, steps\_per\_epoch=batch\_size, epochs=epochs)  
  
model\_json = model.to\_json()  
with open("./model/facial\_expression\_model\_structure.json", "w") as json\_file:  
 json\_file.write(model\_json)  
model.save\_weights("./model/facial\_expression\_model\_weights.h5")  
print("Saved model to disk")  
  
  
score = model.evaluate(x\_test, y\_test)  
print('Test loss:', score[0])  
print('Test accuracy:', 100\*score[1])

Code that display the result:

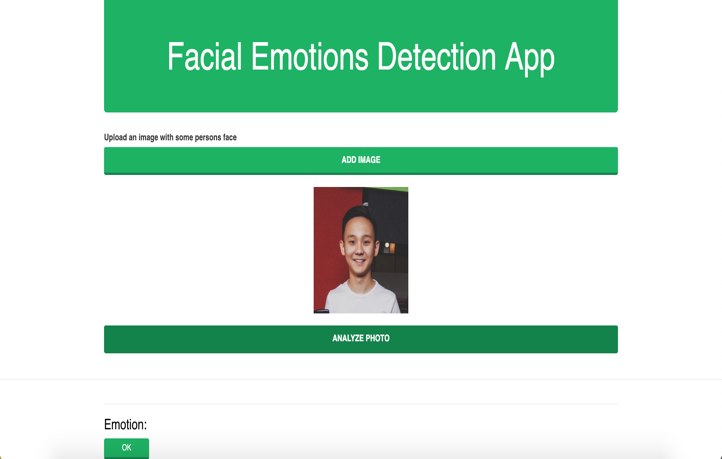
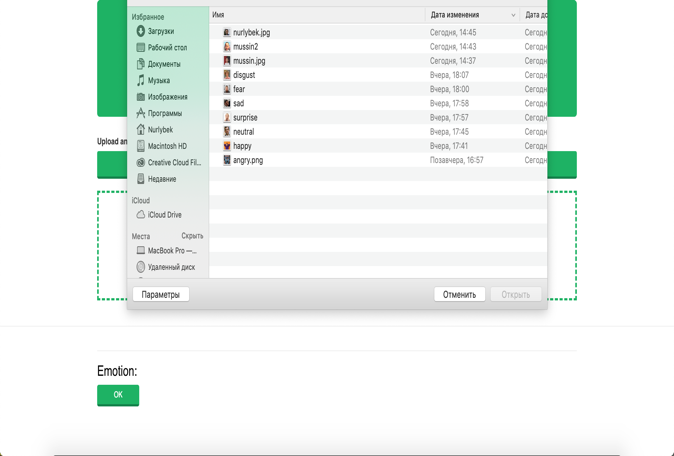
sfname = 'images/' + filename  
json\_file = open('./model/facial\_expression\_model\_structure.json', 'r')  
loaded\_model\_json = json\_file.read()  
json\_file.close()  
model = model\_from\_json(loaded\_model\_json)  
model.load\_weights("./model/facial\_expression\_model\_weights.h5")  
print("Loaded model from disk")  
  
emotion\_dict = {  
 0: 'Angry',  
 1: 'Disgust',  
 2: 'Fear',  
 3: 'Happy',  
 4: 'Sad',  
 5: 'Surprised',  
 6: 'Neutral'  
}  
  
img = image.load\_img(sfname, grayscale=True, target\_size=(48, 48))  
  
x = image.img\_to\_array(img)  
x = np.expand\_dims(x, axis=0)  
x /= 255  
  
custom = model.predict(x)  
index\_max = np.argmax(custom[0])  
  
predicted = emotion\_dict[index\_max]  
# print("Emotion :", predicted)

**Application**

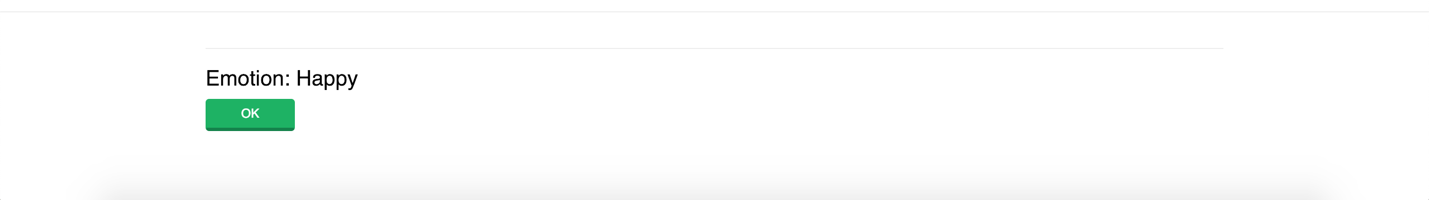
This is the design of our web application

****

Here we can upload an image then click the button to analyze the emotion of a person on a photo:



The result looks like this:



Application is built with *Flask* framework.

**Flask** is a micro [web framework](https://en.wikipedia.org/wiki/Web_framework) written in Python. It is classified as a microframework because it does not require particular tools or libraries. Flask supports extensions that can add application features as if they were implemented in Flask itself.

**Conclusion**

All work done according to Project Requirements, as following:

* Data
* ML algorithm
* Application
* Documentation

So, in the given work we used data, detected a face by pixels and defined emotions of person. In addition, we practiced our knowledge and skills that have been learnt on CSS324 Machine Learning course.

In the process of creating a project there were some difficulties related with the huge weight of data. Because of this, data takes a long time to load. Sometimes, pixels of image don’t match with image.

In the future works a project can be improved by adding some new emotions and other features like gender, approximate age of person, possible responses on an exact emotion, etc. Also, the application of Human Face Emotion detection can be installed on smartphones and other gadgets.

In contemporary century this work might be useful starting from children’s development, psychology, robotics, and ending with different studies.

**References**

* Dr.S.Vijayarani, M.Vinupriya “An Efficient Algorithm for Facial Image Classification”, 2015
* Ion Marqu´es , “Face Recognition Algorithms”, 2010
* Keras Documentation

<https://keras.io>

* Flask Docs

<http://flask.pocoo.org>

* Understanding Deep Convolutional Neural Networks with a practical use-case in Tensorflow and Keras

<https://www.kdnuggets.com/2017/11/understanding-deep-convolutional-neural-networks-tensorflow-keras.html>

* Building a Basic Keras Neural Network Sequential Model

<https://www.kdnuggets.com/2018/06/basic-keras-neural-network-sequential-model.html>