LOVELY PROFESSIONAL UNIVERSITY

FACIAL EXPRESSION RECOGNITION (Deep Learning)

Prasanth Velpula (A09)

11801519

KM073

Course Code: INT248

Github link: https://github.com/11801519lpu/prasanth-facial-expression-recogition.git

Submitted to Mrs Ankita Wandan

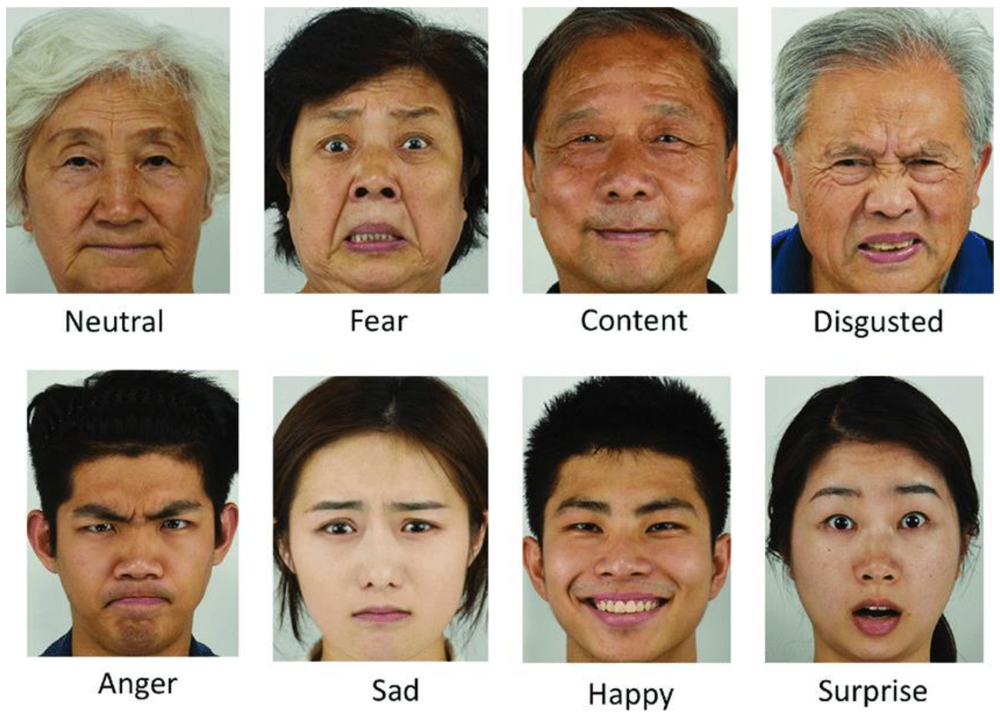
Facial Expression Recognition

Introduction:

Expression is an important aspect in the interaction and communication between people. Most of the communication between human beings involves non-verbal signs, and the social aspect of this communication is important. Humans also tend to include this social aspect when communicating with computers.

What to do:

We have given the dataset now using that data set we need to build a system for recognizing expression detection. We have used existing data and the result of their analysis were 31 to 81 percentage correct. We used sequential model to train our neural lnetwork.



Technology:

Python

Tensorflow

Algorithm:

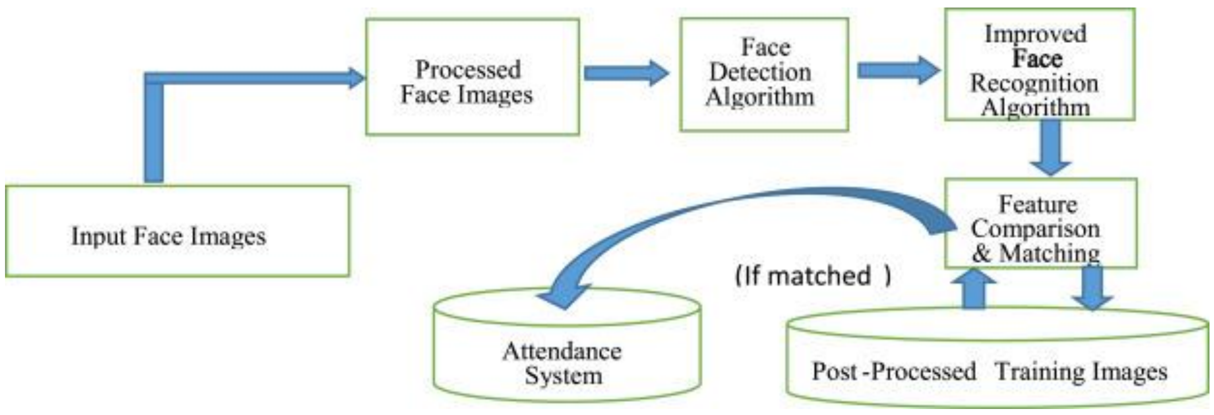

Input Face Images
Processed Face Images
Face Detection Algorithm
Improved Face Recognition Algorithm
Feature Comparison & Matching
(If matched)
Attendance System
Post -Processed Training Images

Figure shows the flowchart of the algorithm. As depicted in the flowchart, the captured input face images are processed using our proposed image processing techniques, then the face detection algorithm is applied to detect faces. Once faces are detected, the face recognition algorithm aided with our proposed method will be applied to recognize faces. Once faces are recognized, the metadata of the recognized faces will be extracted to mark attendance using the attendance system.

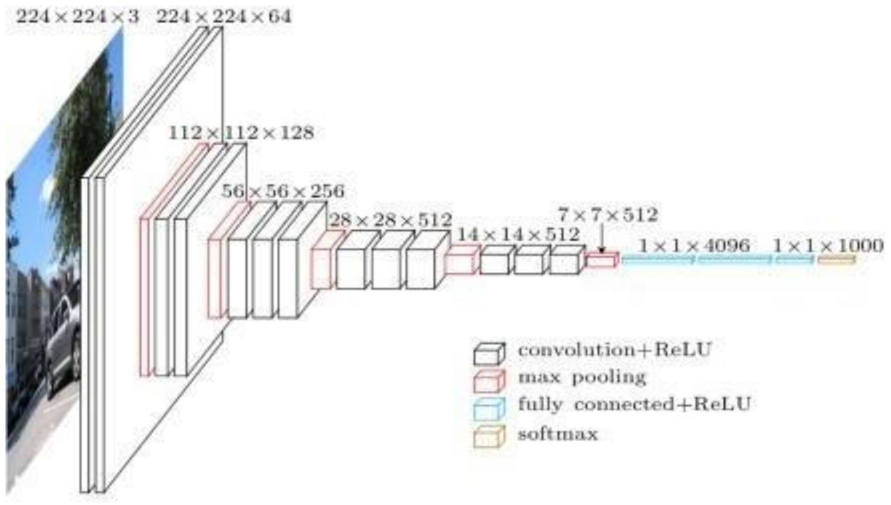
Implementation:

Part 1: Training Neural Network

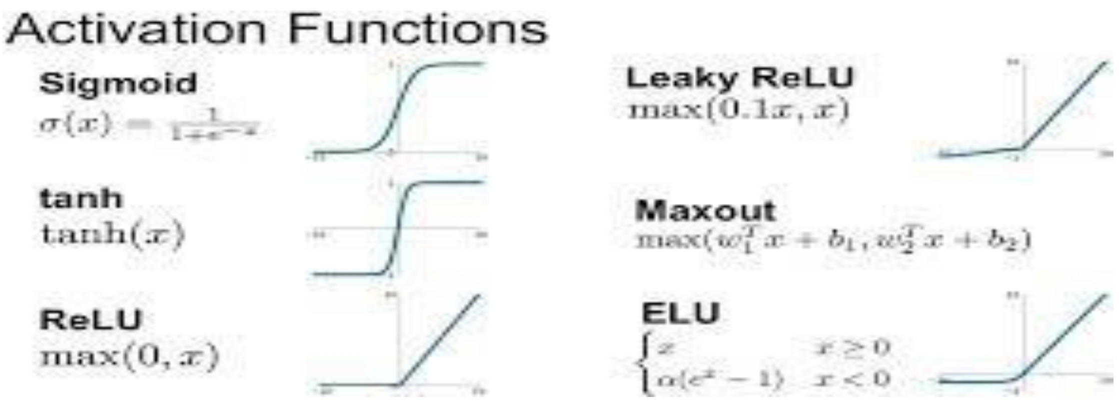
For training a NN, it required sufficient relevant data for training and testing. We use 80% data for training and 20% data for testing.

Step 1: We made 7 images from 1 image just changing its orientation and created another directory for validation.

Step 2: created a sequential model. Sequential model is NN that uses 2D data at input side and at the end of NN it changes that data into 1D. Here we made 7-layer NN model. At each layer (except input layer) it takes input from its previous layer and calculate the target value with appropriate weight and forward to next layer.



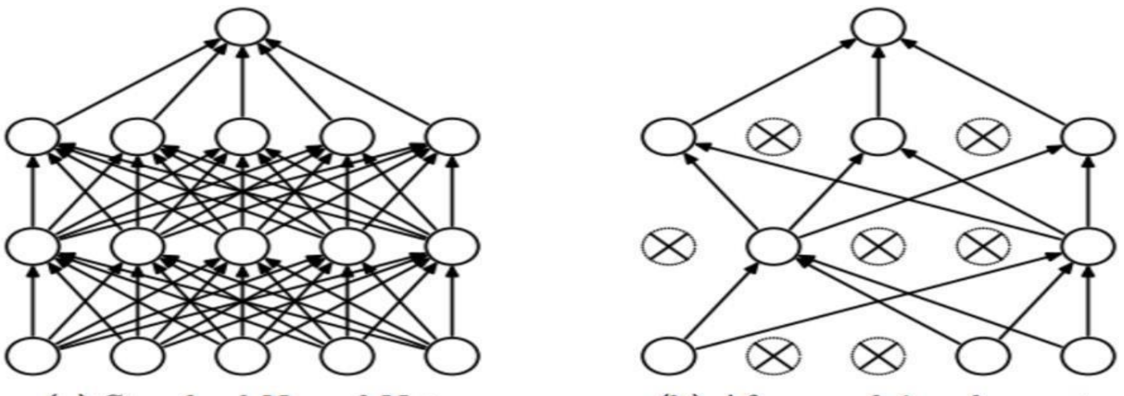
Step 3: The activation function. Activation function is function based which the weight is get updated. Here we used ELU activation function it a kind of exponential. ELU. Exponential Linear Unit or its widely known name ELU is a function that tend to converge cost to zero faster and produce more accurate results. Different to other activation functions, ELU has an extra alpha constant which should be positive number. ELU is very similar to RELU except negative inputs.



Step4:BatchNormalizationandMax Pooling:Batchnormalization is a technique fortrainingverydeepneuralnetworksthatstandardizes theinputsto a layer for each mini batch.

Max-Pooling: Maximum pooling, or max pooling, is
a pooling operation that calculates the maximum, or largest, value in eachpatch ofeach featuremap. The results are down sampled or pooled feature maps that highlight the most present feature in the patch, not theaveragepresenceofthe feature in the case of average pooling.

Dropout class
The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 arescaledupby 1/(1-rate) suchthatthesum overall inputsisunchanged.



Step 5: Repeat step 4 for 7 time but after 4th time we introduce Flatten: basically, flattenmake 2D layerof matrix data into flat data. Andin lasttwo layers Dense layer come up: A dense layer is just a regular layer of neurons in a neural network. Each neuron receives input from all the neurons in the previous layer, thus densely connected. The layer has aweight matrix W, abias vector b, and the activations of previous layer a.

Step6:Start training to theNN for defined epochand decided amount of data tobe trained. During training figured out loss and accuracy if in continue three iteration loss increases and accuracy decreases the program need to stop training. At the end model is ready to test.

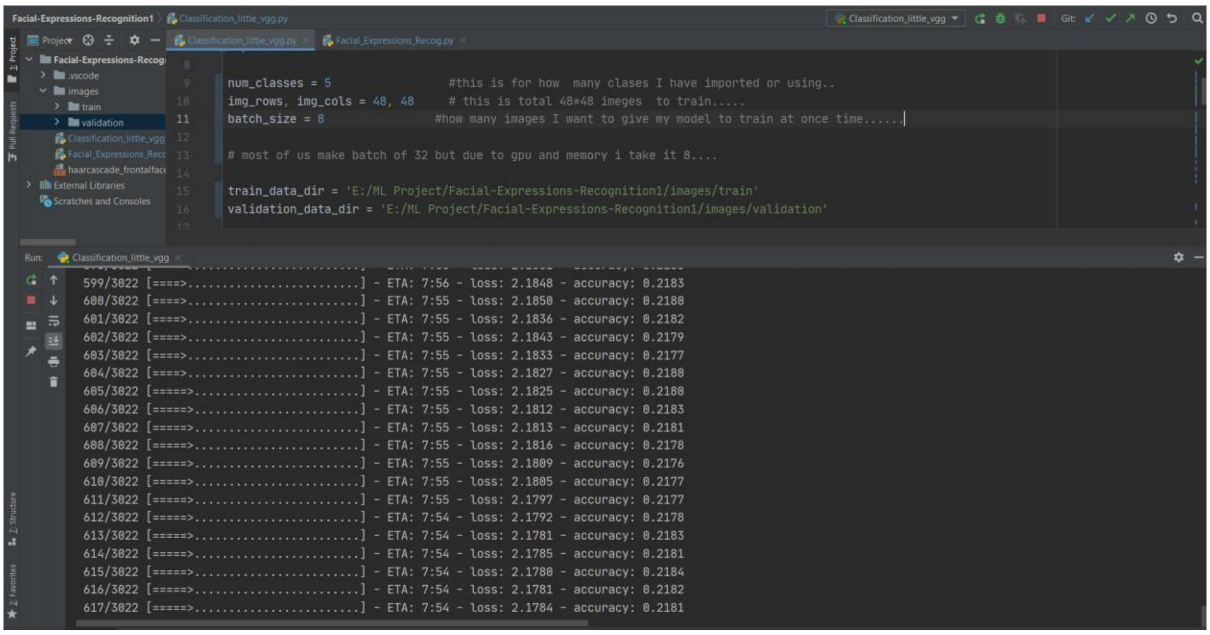
Part 2: Testing of Model

Step 1: Take input data using System webcam or from database of system. In our case we use Open CV to take image as input. Then map with haarcascade file: a file that contain information about gesture. And apply to model for testing.

Step 2: After processing with input image give the output with target ans. Haarcascade file we easily find face in image ant the provided to input layer.

Step 3: Keep Showing Output with result status.

Output:1 Training Part:


Facial-Expressions-Recognition1
Classification_little_vgg.py
Facial_Expressions_Recog.py
.vscode
images
train
validation
haarcascade_frontalface
External Libraries
Scratches and Consoles
Run: Classification_little_vgg

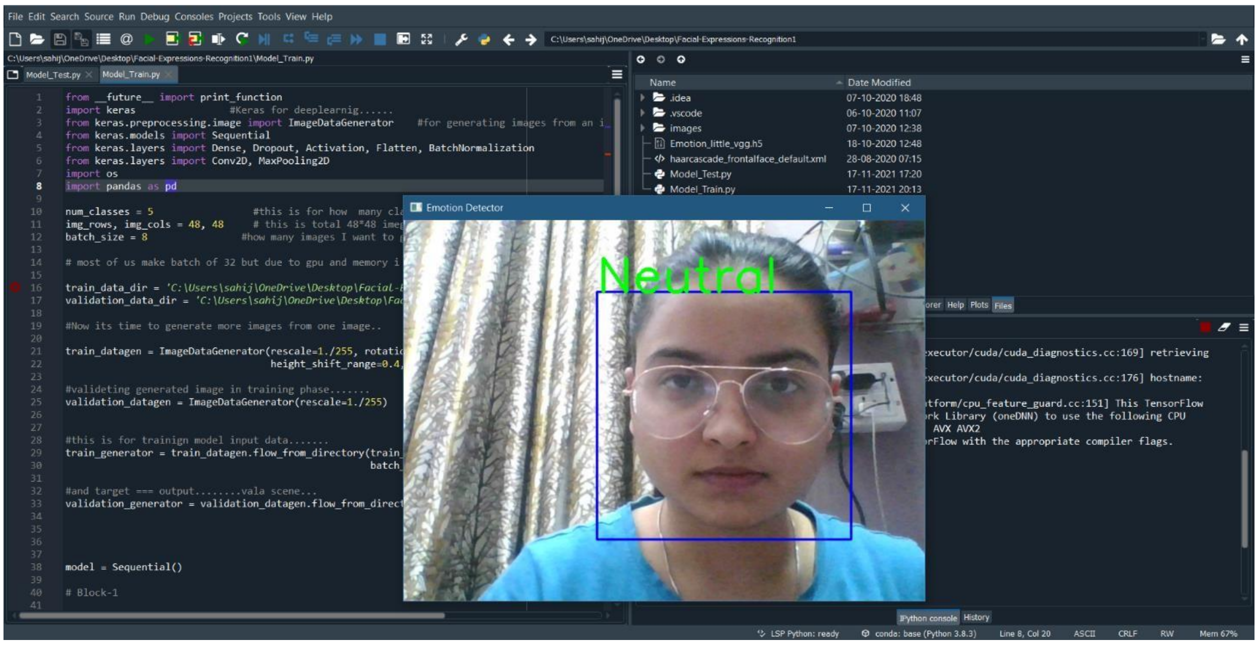
```
num_classes = 5                  #this is for how  many clases I have imported or using..
img_rows, img_cols = 48, 48      # this is total 48*48 imeges  to train.....
batch_size = 8                 #how many images I want to give my model to train at once time......

# most of us make batch of 32 but due to gpu and memory i take it 8....

train_data_dir = 'E:/ML Project/Facial-Expressions-Recognition1/images/train'
validation_data_dir = 'E:/ML Project/Facial-Expressions-Recognition1/images/validation'
```

```
599/3022 [====>.........................] - ETA: 7:56 - loss: 2.1848 - accuracy: 0.2183
600/3022 [====>.........................] - ETA: 7:55 - loss: 2.1850 - accuracy: 0.2180
601/3022 [====>.........................] - ETA: 7:55 - loss: 2.1836 - accuracy: 0.2182
602/3022 [====>.........................] - ETA: 7:55 - loss: 2.1843 - accuracy: 0.2179
603/3022 [====>.........................] - ETA: 7:55 - loss: 2.1833 - accuracy: 0.2177
604/3022 [====>.........................] - ETA: 7:55 - loss: 2.1827 - accuracy: 0.2180
605/3022 [=====>........................] - ETA: 7:55 - loss: 2.1825 - accuracy: 0.2180
606/3022 [=====>........................] - ETA: 7:55 - loss: 2.1812 - accuracy: 0.2183
607/3022 [=====>........................] - ETA: 7:55 - loss: 2.1813 - accuracy: 0.2181
608/3022 [=====>........................] - ETA: 7:55 - loss: 2.1816 - accuracy: 0.2178
609/3022 [=====>........................] - ETA: 7:55 - loss: 2.1809 - accuracy: 0.2176
610/3022 [=====>........................] - ETA: 7:55 - loss: 2.1805 - accuracy: 0.2177
611/3022 [=====>........................] - ETA: 7:55 - loss: 2.1797 - accuracy: 0.2177
612/3022 [=====>........................] - ETA: 7:54 - loss: 2.1792 - accuracy: 0.2178
613/3022 [=====>........................] - ETA: 7:54 - loss: 2.1781 - accuracy: 0.2183
614/3022 [=====>........................] - ETA: 7:54 - loss: 2.1785 - accuracy: 0.2181
615/3022 [=====>........................] - ETA: 7:54 - loss: 2.1780 - accuracy: 0.2184
616/3022 [=====>........................] - ETA: 7:54 - loss: 2.1781 - accuracy: 0.2182
617/3022 [=====>........................] - ETA: 7:54 - loss: 2.1784 - accuracy: 0.2181
```

2 Testing Part:


File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\sahij\OneDrive\Desktop\Facial-Expressions-Recognition1
C:\Users\sahij\OneDrive\Desktop\Facial-Expressions-Recognition1\Model_Train.py
Model_Test.py
Model_Train.py
Emotion Detector
Neutral
Explorer Help Plots Files
IPython console History
LSP Python: ready
conda: base (Python 3.8.3)
Line 8, Col 20
ASCII
CRLF
RW
Mem 67%

```
from __future__ import print_function
import keras                  #Keras for deeplearnig......
from keras.preprocessing.image import ImageDataGenerator    #for generating images from an i
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten, BatchNormalization
from keras.layers import Conv2D, MaxPooling2D
import os
import pandas as pd

num_classes = 5                 #this is for how  many cla
img_rows, img_cols = 48, 48     # this is total 48*48 imeg
batch_size = 8                 #how many images I want to g

# most of us make batch of 32 but due to gpu and memory i

train_data_dir = 'C:\Users\sahij\OneDrive\Desktop\Facial-E
validation_data_dir = 'C:\Users\sahij\OneDrive\Desktop\Fac

#Now its time to generate more images from one image..

train_datagen = ImageDataGenerator(rescale=1./255, rotatic
                                   height_shift_range=0.4,

#valideting generated image in training phase.......
validation_datagen = ImageDataGenerator(rescale=1./255)


#this is for trainign model input data.......
train_generator = train_datagen.flow_from_directory(train_
                                                    batch_

#and target === output........vala scene...
validation_generator = validation_datagen.flow_from_direct




model = Sequential()

# Block-1

```

Name	Date Modified
.idea	07-10-2020 18:48
.vscode	06-10-2020 11:07
images	07-10-2020 12:38
Emotion_little_vgg.h5	18-10-2020 12:48
haarcascade_frontalface_default.xml	28-08-2020 07:15
Model_Test.py	17-11-2021 17:20
Model_Train.py	17-11-2021 20:13

```
executor/cuda/cuda_diagnostics.cc:169] retrieving

executor/cuda/cuda_diagnostics.cc:176] hostname:

tform/cpu_feature_guard.cc:151] This TensorFlow
rk Library (oneDNN) to use the following CPU
 AVX AVX2
rFlow with the appropriate compiler flags.
```

Code

Main:

```
from
demo
import
demo
        from model import train_model, valid_model
        import tensorflow as tf

        flags =  tf.app.flags
        flags.DEFINE_string('MODE', 'demo',
                    'Set program to run in different mode, include train, valid and demo.')
        flags.DEFINE_string('checkpoint_dir', './ckpt',
                    'Path to model file.')
        flags.DEFINE_string('train_data', './data/fer2013/fer2013.csv',
                    'Path to training data.')
        flags.DEFINE_string('valid_data', './valid_sets/',
                    'Path to training data.')
        flags.DEFINE_boolean('show_box', False,
                    'If true, the results will show detection box')
        FLAGS = flags.FLAGS

        def main():
```

```
  assert FLAGS.MODE in ('train', 'valid', 'demo')

  if FLAGS.MODE == 'demo':
    demo(FLAGS.checkpoint_dir, FLAGS.show_box)
  elif FLAGS.MODE == 'train':
    train_model(FLAGS.train_data)
  elif FLAGS.MODE == 'valid':
    valid_model(FLAGS.checkpoint_dir, FLAGS.valid_data)

if __name__ == '__main__':
  main()
```

Model:

```
import
os
import sys
import numpy as np
import tensorflow as tf
from utils import *
EMOTIONS = ['angry', 'disgusted', 'fearful', 'happy', 'sad', 'surprised', 'neutral']
def deepnn(x):
  x_image = tf.reshape(x, [-1, 48, 48, 1])
  # conv1
  W_conv1 = weight_variables([5, 5, 1, 64])
  b_conv1 = bias_variable([64])
  h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
  # pool1
  h_pool1 = maxpool(h_conv1)
  # norm1
  norm1 = tf.nn.lrn(h_pool1, 4, bias=1.0, alpha=0.001 / 9.0, beta=0.75)
  # conv2
  W_conv2 = weight_variables([3, 3, 64, 64])
  b_conv2 = bias_variable([64])
  h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
  norm2 = tf.nn.lrn(h_conv2, 4, bias=1.0, alpha=0.001 / 9.0, beta=0.75)
  h_pool2 = maxpool(norm2)
  # Fully connected layer
  W_fc1 = weight_variables([12 * 12 * 64, 384])
  b_fc1 = bias_variable([384])
  h_conv3_flat = tf.reshape(h_pool2, [-1, 12 * 12 * 64])
  h_fc1 = tf.nn.relu(tf.matmul(h_conv3_flat, W_fc1) + b_fc1)
  # Fully connected layer
  W_fc2 = weight_variables([384, 192])
  b_fc2 = bias_variable([192])
  h_fc2 = tf.matmul(h_fc1, W_fc2) + b_fc2
  # linear
  W_fc3 = weight_variables([192, 7])
  b_fc3 = bias_variable([7])
  y_conv = tf.add(tf.matmul(h_fc2, W_fc3), b_fc3)

  return y_conv
def conv2d(x, W):
  return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')
def maxpool(x):
  return tf.nn.max_pool(x, ksize=[1, 3, 3, 1],
                        strides=[1, 2, 2, 1], padding='SAME')
def weight_variables(shape):
  initial = tf.truncated_normal(shape, stddev=0.1)
  return tf.Variable(initial)
def bias_variable(shape):
  initial = tf.constant(0.1, shape=shape)
  return tf.Variable(initial)

def train_model(train_data):
  fer2013 = input_data(train_data)
  max_train_steps = 30001

  x = tf.placeholder(tf.float32, [None, 2304])
  y_ = tf.placeholder(tf.float32, [None, 7])

  y_conv = deepnn(x)
  cross_entropy = tf.reduce_mean(
    tf.nn.softmax_cross_entropy_with_logits(labels=y_, logits=y_conv))
  train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
  correct_prediction = tf.equal(tf.argmax(y_conv, 1), tf.argmax(y_, 1))
  accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
  with tf.Session() as sess:
    saver = tf.train.Saver()
    sess.run(tf.global_variables_initializer())
    for step in range(max_train_steps):
      batch = fer2013.train.next_batch(50)
      if step % 100 == 0:
        train_accuracy = accuracy.eval(feed_dict={
          x: batch[0], y_: batch[1]})
        print('step %d, training accuracy %g' % (step, train_accuracy))
      train_step.run(feed_dict={x: batch[0], y_: batch[1]})
      if step + 1 == max_train_steps:
        saver.save(sess, './models/emotion_model', global_step=step + 1)
      if step % 1000 == 0:
        print('*Test accuracy %g' % accuracy.eval(feed_dict={
          x: fer2013.validation.images, y_: fer2013.validation.labels}))
def predict(image=[[0.1] * 2304]):
  x = tf.placeholder(tf.float32, [None, 2304])
  y_conv = deepnn(x)
  # init = tf.global_variables_initializer()
  saver = tf.train.Saver()
  probs = tf.nn.softmax(y_conv)
  y_ = tf.argmax(probs)
  with tf.Session() as sess:
    # assert os.path.exists('/tmp/models/emotion_model')
    ckpt = tf.train.get_checkpoint_state('./models')
    print(ckpt.model_checkpoint_path)
    if ckpt and ckpt.model_checkpoint_path:
      saver.restore(sess, ckpt.model_checkpoint_path)
      print('Restore ssss')
    return sess.run(probs, feed_dict={x: image})

def image_to_tensor(image):
  tensor = np.asarray(image).reshape(-1, 2304) * 1 / 255.0
  return tensor
def valid_model(modelPath, validFile):
  x = tf.placeholder(tf.float32, [None, 2304])
  y_conv = deepnn(x)
  probs = tf.nn.softmax(y_conv)
  saver = tf.train.Saver()
  ckpt = tf.train.get_checkpoint_state(modelPath)
  with tf.Session() as sess:
    print(ckpt.model_checkpoint_path)
    if ckpt and ckpt.model_checkpoint_path:
      saver.restore(sess, ckpt.model_checkpoint_path)
      print('Restore model sucsses!!')
    files = os.listdir(validFile)
    for file in files:
      if file.endswith('.jpg'):
        image_file = os.path.join(validFile, file)
        image = cv2.imread(image_file, cv2.IMREAD_GRAYSCALE)
        tensor = image_to_tensor(image)
        result = sess.run(probs, feed_dict={x: tensor})
        print(file, EMOTIONS[result.argmax()])
```

THANK

YOU!