BIOMETRIC PROJECT

TEAM 5 - SLOT G1

Topic - Facial Emotion Recognition Biometric System

Faculty Name - Sharmila Banu

TEAM MEMBERS

Ikshit Samanta - 20BCE2821

Dhruv Sethi - 20BCE2240

Ruksana Tabassum - 20BCE2650

Akshat Garg – 20BCE2291

INTRO

Emotion recognition is one of the many facial recognition technologies that have developed and grown through the years. Currently, facial emotion recognition software is used to allow certain programs to examine and process the expressions on a human's face. Using advanced image dispensation, this software functions like a human brain that makes it capable of recognizing emotions too.

The AI detects and studies the expressions depending on many factors such as location of the eyebrows and eyes, position of the mouth and distinct changes of the facial features to conclude what emotion the person is showing.

Emotion recognition is used for various purposes that some people do not even notice on a daily basis. Here are some of the areas that would show that emotion recognition is beneficial -

Security Measures, HR Assistance, Customer Service, Differently Abled Children, Audience Engagement, Video Game Testing, Healthcare

CHOOSING THE DATASET

We decided to do our project on Face Emotion Recognition Biometric Systems. We have picked the FER2013 dataset from a kaggle competition in the year 2013. Here, this was given as a challenge to the participants to perform facial emotion/expression recognition on the given dataset.

The dataset has many images along with their emotions, this dataset is extremely popular in the face detection area.

Over 35,000 photographs were collected in 2013, some of the existing photographs were taken from various media publications and some of them consist of stock photographs - all these are included within this dataset.

REVIEWS

In Review 1, we researched on several biometric projects, start-ups, research papers, and blogs. From there we got to learn about different applications of face recognition biometric systems.

In most of the projects that we referred to, libraries like Tensorflow, Keras, ScikitLearn and so on were being used. So, we learnt and implemented these libraries in our notebook.

In Review 2, we further went on to build on our code, and make it almost production ready. We learnt about applications of MLCNN (Multi-level Convolutional Neural Networks) and the use of ELU and ReLU. We have tried to recognize emotions in photographs using a keras library. Over 35,000 photographs collected in the data in 2013, some of the existing photographs were taken from various media publications and some of them consist of stock photographs. Our goal here was to recognize 7 different emotions (angry, happy, scared, natural, etc.) through photographs.

We had a few errors along with a few changes that we completed for our final Review 3.

In Review 3, we added a confusion matrix for error detection.

CODE EXPLANATION

We have used a pre-trained model to shorten the training time of the dataset; this made our model more optimised.

From the dataset, we realise that the 3 main emotions that are found are happy, sad and neutral. We display a graph for the various emotions to find this information.

We take a random photo from the dataset(FER2013), and test that image against the true emotion. So, finally as we can see in the image that gets produced as a final result, we can see that the predicted emotion matches the true emotion, almost every time.

For loading our data, we use a pretrained model from an .npy file. This is the data that we will use our model on, to predict the emotions and plot the confusion matrix.

Using the confusion matrix, we can identify the accuracy of the prediction, and as we can see, the accuracy is approximately 74.08%.

We have used an ensemble duonet model, which means that it is an aggregation of a lot of smaller Neural Networks, combined into one big Neural Network.

ERROR DETECTION

We researched error detection libraries like dlib, wherein it checks if the facial data present in the dataset matches a similarity score. If the face isn't detected in this test image, the code will throw an out of range warning, and consider that image as a void.

ROC - We have used ROC curves(Receiver Operating Characteristic curve) for signal detection. We have made use of a confusion matrix to understand the number of correct vs incorrect predictions made by the classification model with respect to the actual outcomes/ground truths

Confusion Matrix - We have also displayed a confusion matrix, which is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known.

MEMBER WISE CONTRIBUTION

Ikshit Samanta - Research + Code

Dhruv Sethi - Research + Code

Ruksana Tabassum - Research + Documentation

Akshat Garg - Research

CODE

FER

In [1]:
```
from keras.models import load_model
from keras.preprocessing.image import ImageDataGenerator

import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import itertools
import pandas as pd

import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import pyplot
```

In [2]:
```
n_classes = 7
classes=np.array(('angry', 'disgust', 'fear', 'happy', 'neutral', 'sad', 'surprise'))

img_size = 48
# batch_size = 1024
batch_size = 64

current_model = 'fer2013'
# final_model_path = current_model + '.h5'
```

In [3]:
```
df = pd.read_csv('fer2013.csv')
print(df.shape)
df.head()
```

```
(35887, 3)
```

Out[3]:

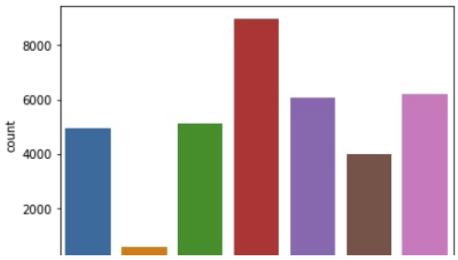
	emotion	pixels	Usage
0	0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121...	Training
1	0	151 150 147 155 148 133 111 140 170 174 182 15...	Training
2	2	231 212 156 164 174 138 161 173 182 200 106 38...	Training
3	4	24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 1...	Training
4	6	4 0 0 0 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84...	Training

In [4]:
```
df.emotion.value_counts()
```

Out[4]:
```
3    8989
6    6198
4    6077
2    5121
0    4953
5    4002
1     547
Name: emotion, dtype: int64
```

In [5]:
```
sns.countplot(df.emotion)
pyplot.show()
```

```
C:\Users\Asus\anaconda3\envs\TensorFlow\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the foll
owing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and pas
sing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [6]:

```python
fig = pyplot.figure(1, (14, 14))

k = 0
for label in sorted(df.emotion.unique()):
    for j in range(7):
        px = df[df.emotion==label].pixels.iloc[k]
        px = np.array(px.split(' ')).reshape(48, 48).astype('float32')

        k += 1
        ax = pyplot.subplot(7, 7, k)
        ax.imshow(px, cmap='gray')
        ax.set_xticks([])
        ax.set_yticks([])
        ax.set_title(classes[label])
        pyplot.tight_layout()
```

In [7]:

```python
def plot_confusion_matrix(y_test, y_pred, classes,
                          normalize=False,
                          title='Unnormalized confusion matrix',
                          cmap=plt.cm.Blues):
```

```
    cm = confusion_matrix(y_test, y_pred)

    if normalize:
        cm = np.round(cm.astype('float') / cm.sum(axis=1)[:, np.newaxis], 2)

    np.set_printoptions(precision=2)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    thresh = cm.min() + (cm.max() - cm.min()) / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True expression')
    plt.xlabel('Predicted expression')
    plt.show()
```

In [8]:

```
def int2emo(num):
    if num == 0:
        return 'Angry'
    elif num == 1:
        return 'Disgust'
    elif num == 2:
        return 'Fear'
    elif num == 3:
        return 'Happy'
    elif num == 4:
        return 'Neutral'
    elif num == 5:
        return 'Sad'
    else:
        return 'Surprise'
```

Loading the data

In the following cell, we load the test data for a pretrained model from an `.npy` file. This is the data that we will use our model on, to predict the emotions and plot the confusion matrix.

In [9]:

```
test_data_x = np.load(current_model + '_test_data_x.npy')
test_data_y_one_hot = np.load(current_model + '_test_data_y_one_hot.npy')
n_test_ins = test_data_x.shape[0]
n_test_ins
```

Out[9]: 3589

Loading the model

The model is an ensemble duonet, which means that it is an aggregation of a lot of smaller Neural Networks, combined into one big Neural Network.

In [10]:

```
BEST_WEIGHT_FILE = './' + current_model + '.hdf5'

print('> loading trained model from ' + BEST_WEIGHT_FILE + '...', end='')
model = load_model(BEST_WEIGHT_FILE)
print('done.')
```

```
> loading trained model from ./fer2013.hdf5...done.
```

In [11]:

```
model.summary()
```

```
Model: "ensemble_duonet"
________________________________________________________________________________________________
Layer (type)                    Output Shape          Param #     Connected to
================================================================================================
input (InputLayer)              [(None, 48, 48, 1)]   0
________________________________________________________________________________________________
```

conv_1_1_net_1 (Conv2D)	(None, 48, 48, 64)	640	input[0][0]
conv_1_1_net_2 (Conv2D)	(None, 48, 48, 64)	640	input[0][0]
conv_1_1_net_3 (Conv2D)	(None, 48, 48, 64)	640	input[0][0]
conv_1_2_net_1 (Conv2D)	(None, 48, 48, 64)	36928	conv_1_1_net_1[0][0]
conv_1_2_net_2 (Conv2D)	(None, 48, 48, 64)	36928	conv_1_1_net_2[0][0]
conv_1_2_net_3 (Conv2D)	(None, 48, 48, 64)	36928	conv_1_1_net_3[0][0]
pool_1_net_1 (MaxPooling2D)	(None, 24, 24, 64)	0	conv_1_2_net_1[0][0]
pool_1_net_2 (MaxPooling2D)	(None, 24, 24, 64)	0	conv_1_2_net_2[0][0]
pool_1_net_3 (MaxPooling2D)	(None, 24, 24, 64)	0	conv_1_2_net_3[0][0]
drop_1_net_1 (Dropout)	(None, 24, 24, 64)	0	pool_1_net_1[0][0]
drop_1_net_2 (Dropout)	(None, 24, 24, 64)	0	pool_1_net_2[0][0]
drop_1_net_3 (Dropout)	(None, 24, 24, 64)	0	pool_1_net_3[0][0]
conv_2_1_net_1 (Conv2D)	(None, 24, 24, 128)	73856	drop_1_net_1[0][0]
conv_2_1_net_2 (Conv2D)	(None, 24, 24, 128)	73856	drop_1_net_2[0][0]
conv_2_1_net_3 (Conv2D)	(None, 24, 24, 128)	73856	drop_1_net_3[0][0]
conv_2_2_net_1 (Conv2D)	(None, 24, 24, 128)	147584	conv_2_1_net_1[0][0]
conv_2_2_net_2 (Conv2D)	(None, 24, 24, 128)	147584	conv_2_1_net_2[0][0]
conv_2_2_net_3 (Conv2D)	(None, 24, 24, 128)	147584	conv_2_1_net_3[0][0]
conv_2_3_net_1 (Conv2D)	(None, 24, 24, 128)	147584	conv_2_2_net_1[0][0]
conv_2_3_net_2 (Conv2D)	(None, 24, 24, 128)	147584	conv_2_2_net_2[0][0]
conv_2_3_net_3 (Conv2D)	(None, 24, 24, 128)	147584	conv_2_2_net_3[0][0]
pool_2_net_1 (MaxPooling2D)	(None, 12, 12, 128)	0	conv_2_3_net_1[0][0]
pool_2_net_2 (MaxPooling2D)	(None, 12, 12, 128)	0	conv_2_3_net_2[0][0]
pool_2_net_3 (MaxPooling2D)	(None, 12, 12, 128)	0	conv_2_3_net_3[0][0]
drop_2_net_1 (Dropout)	(None, 12, 12, 128)	0	pool_2_net_1[0][0]
drop_2_net_2 (Dropout)	(None, 12, 12, 128)	0	pool_2_net_2[0][0]
drop_2_net_3 (Dropout)	(None, 12, 12, 128)	0	pool_2_net_3[0][0]
conv_3_1_net_1 (Conv2D)	(None, 12, 12, 256)	295168	drop_2_net_1[0][0]
conv_3_1_net_2 (Conv2D)	(None, 12, 12, 256)	295168	drop_2_net_2[0][0]
conv_3_1_net_3 (Conv2D)	(None, 12, 12, 256)	295168	drop_2_net_3[0][0]
conv_3_2_net_1 (Conv2D)	(None, 12, 12, 256)	590080	conv_3_1_net_1[0][0]
conv_3_2_net_2 (Conv2D)	(None, 12, 12, 256)	590080	conv_3_1_net_2[0][0]
conv_3_2_net_3 (Conv2D)	(None, 12, 12, 256)	590080	conv_3_1_net_3[0][0]
conv_3_3_net_1 (Conv2D)	(None, 12, 12, 256)	590080	conv_3_2_net_1[0][0]
conv_3_3_net_2 (Conv2D)	(None, 12, 12, 256)	590080	conv_3_2_net_2[0][0]
conv_3_3_net_3 (Conv2D)	(None, 12, 12, 256)	590080	conv_3_2_net_3[0][0]
conv_3_4_net_1 (Conv2D)	(None, 12, 12, 256)	590080	conv_3_3_net_1[0][0]
conv_3_4_net_2 (Conv2D)	(None, 12, 12, 256)	590080	conv_3_3_net_2[0][0]
conv_3_4_net_3 (Conv2D)	(None, 12, 12, 256)	590080	conv_3_3_net_3[0][0]
pool_3_net_1 (MaxPooling2D)	(None, 6, 6, 256)	0	conv_3_4_net_1[0][0]
pool_3_net_2 (MaxPooling2D)	(None, 6, 6, 256)	0	conv_3_4_net_2[0][0]
pool_3_net_3 (MaxPooling2D)	(None, 6, 6, 256)	0	conv_3_4_net_3[0][0]
drop_3_net_1 (Dropout)	(None, 6, 6, 256)	0	pool_3_net_1[0][0]
drop_3_net_2 (Dropout)	(None, 6, 6, 256)	0	pool_3_net_2[0][0]
drop_3_net_3 (Dropout)	(None, 6, 6, 256)	0	pool_3_net_3[0][0]

conv_4_1_net_1 (Conv2D)	(None, 6, 6, 256)	590080	drop_3_net_1[0][0]
conv_4_1_net_2 (Conv2D)	(None, 6, 6, 256)	590080	drop_3_net_2[0][0]
conv_4_1_net_3 (Conv2D)	(None, 6, 6, 256)	590080	drop_3_net_3[0][0]
conv_4_2_net_1 (Conv2D)	(None, 6, 6, 256)	590080	conv_4_1_net_1[0][0]
conv_4_2_net_2 (Conv2D)	(None, 6, 6, 256)	590080	conv_4_1_net_2[0][0]
conv_4_2_net_3 (Conv2D)	(None, 6, 6, 256)	590080	conv_4_1_net_3[0][0]
conv_4_3_net_1 (Conv2D)	(None, 6, 6, 256)	590080	conv_4_2_net_1[0][0]
conv_4_3_net_2 (Conv2D)	(None, 6, 6, 256)	590080	conv_4_2_net_2[0][0]
conv_4_3_net_3 (Conv2D)	(None, 6, 6, 256)	590080	conv_4_2_net_3[0][0]
conv_4_4_net_1 (Conv2D)	(None, 6, 6, 256)	590080	conv_4_3_net_1[0][0]
conv_4_4_net_2 (Conv2D)	(None, 6, 6, 256)	590080	conv_4_3_net_2[0][0]
conv_4_4_net_3 (Conv2D)	(None, 6, 6, 256)	590080	conv_4_3_net_3[0][0]
pool_4_net_1 (MaxPooling2D)	(None, 3, 3, 256)	0	conv_4_4_net_1[0][0]
pool_4_net_2 (MaxPooling2D)	(None, 3, 3, 256)	0	conv_4_4_net_2[0][0]
pool_4_net_3 (MaxPooling2D)	(None, 3, 3, 256)	0	conv_4_4_net_3[0][0]
drop_4_net_1 (Dropout)	(None, 3, 3, 256)	0	pool_4_net_1[0][0]
drop_4_net_2 (Dropout)	(None, 3, 3, 256)	0	pool_4_net_2[0][0]
drop_4_net_3 (Dropout)	(None, 3, 3, 256)	0	pool_4_net_3[0][0]
conv_5_1_net_1 (Conv2D)	(None, 3, 3, 512)	1180160	drop_4_net_1[0][0]
conv_5_1_net_2 (Conv2D)	(None, 3, 3, 512)	1180160	drop_4_net_2[0][0]
conv_5_1_net_3 (Conv2D)	(None, 3, 3, 512)	1180160	drop_4_net_3[0][0]
conv_5_2_net_1 (Conv2D)	(None, 3, 3, 512)	2359808	conv_5_1_net_1[0][0]
conv_5_2_net_2 (Conv2D)	(None, 3, 3, 512)	2359808	conv_5_1_net_2[0][0]
conv_5_2_net_3 (Conv2D)	(None, 3, 3, 512)	2359808	conv_5_1_net_3[0][0]
conv_5_3_net_1 (Conv2D)	(None, 3, 3, 512)	2359808	conv_5_2_net_1[0][0]
conv_5_3_net_2 (Conv2D)	(None, 3, 3, 512)	2359808	conv_5_2_net_2[0][0]
conv_5_3_net_3 (Conv2D)	(None, 3, 3, 512)	2359808	conv_5_2_net_3[0][0]
conv_5_4_net_1 (Conv2D)	(None, 3, 3, 512)	2359808	conv_5_3_net_1[0][0]
conv_5_4_net_2 (Conv2D)	(None, 3, 3, 512)	2359808	conv_5_3_net_2[0][0]
conv_5_4_net_3 (Conv2D)	(None, 3, 3, 512)	2359808	conv_5_3_net_3[0][0]
pool_5_net_1 (MaxPooling2D)	(None, 1, 1, 512)	0	conv_5_4_net_1[0][0]
pool_5_net_2 (MaxPooling2D)	(None, 1, 1, 512)	0	conv_5_4_net_2[0][0]
pool_5_net_3 (MaxPooling2D)	(None, 1, 1, 512)	0	conv_5_4_net_3[0][0]
flatten_2_net_1 (Flatten)	(None, 73728)	0	conv_2_3_net_1[0][0]
flatten_3_net_1 (Flatten)	(None, 36864)	0	conv_3_4_net_1[0][0]
flatten_4_net_1 (Flatten)	(None, 9216)	0	conv_4_4_net_1[0][0]
drop_5_net_1 (Dropout)	(None, 1, 1, 512)	0	pool_5_net_1[0][0]
flatten_2_net_2 (Flatten)	(None, 36864)	0	conv_3_2_net_2[0][0]
flatten_3_net_2 (Flatten)	(None, 9216)	0	conv_4_2_net_2[0][0]
flatten_4_net_2 (Flatten)	(None, 4608)	0	conv_5_2_net_2[0][0]
drop_5_net_2 (Dropout)	(None, 1, 1, 512)	0	pool_5_net_2[0][0]
flatten_2_net_3 (Flatten)	(None, 18432)	0	drop_2_net_3[0][0]
flatten_3_net_3 (Flatten)	(None, 9216)	0	drop_3_net_3[0][0]
flatten_4_net_3 (Flatten)	(None, 2304)	0	drop_4_net_3[0][0]

```
drop_5_net_3 (Dropout)          (None, 1, 1, 512)    0           pool_5_net_3[0][0]
__________________________________________________________________________________________________
dense_2_net_1 (Dense)           (None, 256)          18874624    flatten_2_net_1[0][0]
__________________________________________________________________________________________________
dense_3_net_1 (Dense)           (None, 256)          9437440     flatten_3_net_1[0][0]
__________________________________________________________________________________________________
dense_4_net_1 (Dense)           (None, 256)          2359552     flatten_4_net_1[0][0]
__________________________________________________________________________________________________
flatten_5_net_1 (Flatten)       (None, 512)          0           drop_5_net_1[0][0]
__________________________________________________________________________________________________
dense_2_net_2 (Dense)           (None, 256)          9437440     flatten_2_net_2[0][0]
__________________________________________________________________________________________________
dense_3_net_2 (Dense)           (None, 256)          2359552     flatten_3_net_2[0][0]
__________________________________________________________________________________________________
dense_4_net_2 (Dense)           (None, 256)          1179904     flatten_4_net_2[0][0]
__________________________________________________________________________________________________
flatten_5_net_2 (Flatten)       (None, 512)          0           drop_5_net_2[0][0]
__________________________________________________________________________________________________
dense_2_net_3 (Dense)           (None, 256)          4718848     flatten_2_net_3[0][0]
__________________________________________________________________________________________________
dense_3_net_3 (Dense)           (None, 256)          2359552     flatten_3_net_3[0][0]
__________________________________________________________________________________________________
dense_4_net_3 (Dense)           (None, 256)          590080      flatten_4_net_3[0][0]
__________________________________________________________________________________________________
flatten_5_net_3 (Flatten)       (None, 512)          0           drop_5_net_3[0][0]
__________________________________________________________________________________________________
early_confusion_net_1 (Concaten (None, 1280)         0           dense_2_net_1[0][0]
                                                                 dense_3_net_1[0][0]
                                                                 dense_4_net_1[0][0]
                                                                 flatten_5_net_1[0][0]
__________________________________________________________________________________________________
early_confusion_net_2 (Concaten (None, 1280)         0           dense_2_net_2[0][0]
                                                                 dense_3_net_2[0][0]
                                                                 dense_4_net_2[0][0]
                                                                 flatten_5_net_2[0][0]
__________________________________________________________________________________________________
early_confusion_net_3 (Concaten (None, 1280)         0           dense_2_net_3[0][0]
                                                                 dense_3_net_3[0][0]
                                                                 dense_4_net_3[0][0]
                                                                 flatten_5_net_3[0][0]
__________________________________________________________________________________________________
later_confusion (Concatenate)   (None, 3840)         0           early_confusion_net_1[0][0]
                                                                 early_confusion_net_2[0][0]
                                                                 early_confusion_net_3[0][0]
__________________________________________________________________________________________________
ensemble_dense_1 (Dense)        (None, 512)          1966592     later_confusion[0][0]
__________________________________________________________________________________________________
ensemble_drop_1 (Dropout)       (None, 512)          0           ensemble_dense_1[0][0]
__________________________________________________________________________________________________
ensemble_dense_2 (Dense)        (None, 512)          262656      ensemble_drop_1[0][0]
__________________________________________________________________________________________________
ensemble_drop_2 (Dropout)       (None, 512)          0           ensemble_dense_2[0][0]
__________________________________________________________________________________________________
ensemble_7_way_softmax (Dense)  (None, 7)            3591        ensemble_drop_2[0][0]
==================================================================================================
Total params: 92,825,543
Trainable params: 2,232,839
Non-trainable params: 90,592,704
__________________________________________________________________________________________________
```

In [12]:

```python
test_datagen = ImageDataGenerator(featurewise_center=True, featurewise_std_normalization=True)
test_datagen.fit(test_data_x)
test_generator = test_datagen.flow(test_data_x, test_data_y_one_hot, batch_size=batch_size, shuffle=False)

y_pred_ = model.predict_generator(generator=test_generator, steps=n_test_ins/batch_size, verbose=1)
```

```
WARNING:tensorflow:From C:\Users\Asus\AppData\Local\Temp/ipykernel_1492/2528678366.py:5: Model.predict_generator (from tensorflow.python.keras.engine.training) is deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.predict, which supports generators.
57/56 [==============================] - 57s 1s/step
```

Confusion Matrix

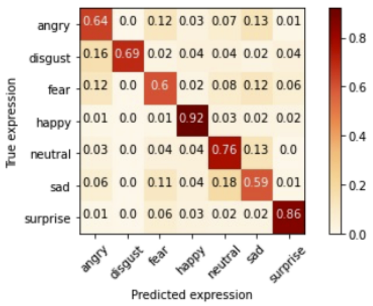
In [13]:

```python
# Predictions
y_pred = np.argmax(y_pred_, axis=1)
# Ground truth
t_te = np.argmax(test_generator.y, axis=1)

fig = plot_confusion_matrix(y_test=t_te, y_pred=y_pred,
                      classes=classes,
```

```
                        normalize=True,
                        cmap=plt.cm.OrRd,
                        title='Average accuracy: ' + str(np.sum(y_pred == t_te)/len(t_te)) + '\n')
```

Average accuracy: 0.7408748955140708



In [14]:

```
rnd_indx = np.random.randint(0, test_data_x.shape[0])
test_img = test_data_x[rnd_indx, :, :]
test_img = test_img.reshape((1, test_img.shape[0], test_img.shape[1], test_img.shape[2]))
```

In [15]:

```
pred_emo = int2emo(y_pred[rnd_indx])
true_emo = int2emo(t_te[rnd_indx])
```

In [16]:

```
plt.imshow(test_img[0, :, :, 0], cmap='gray')
plt.axis('off')
plt.title('True emotion: ' + true_emo + '\nPredicted emotion: ' + pred_emo + '\n')
plt.show()
```

True emotion: Neutral
Predicted emotion: Neutral

In [17]:

```
model.save("model2.h5")
```

In [18]:

```
# model2 = load_model("model2.h5")
```

In [19]:

```
# model2.summary()
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Loading [MathJax]/extensions/Safe.js