Emotion recognition

Serban Mihai, An 2, master 2

1. Introduction

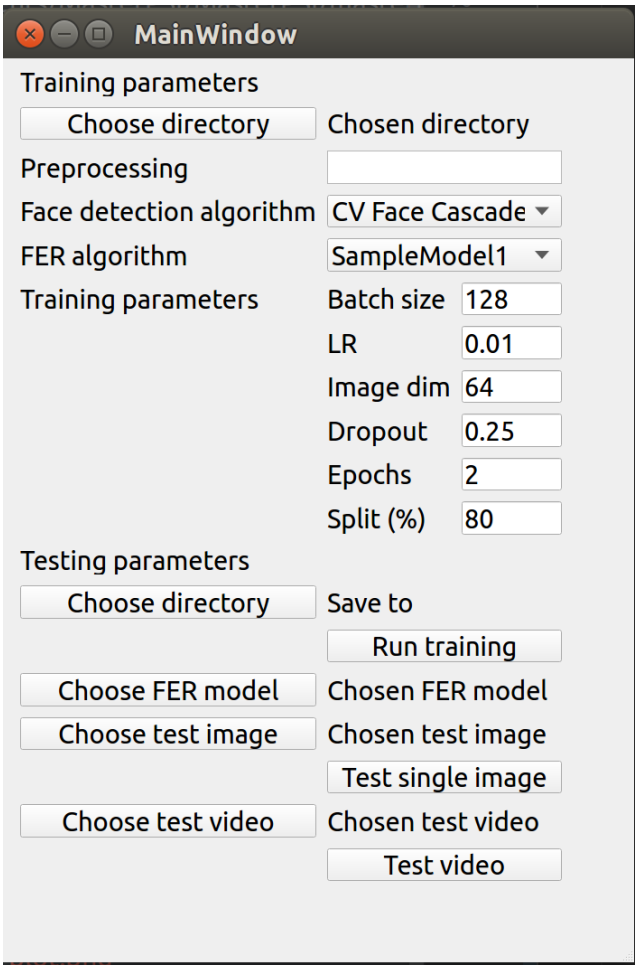
The current system has as a goal to provide a solution for the problem of recognizing emotions in images and videos. Emotion recognition by using innovative techniques such as deep learning is a challenging task, especially due to the identity legal property, which creates a lack of necessary data for creating robust and reliable architectures.

Classical approaches try to find and extract different patterns from the data, while analyzing the entire image. The literature suggests that for emotion recognition, a more performant system can be achieved if performing an initial processing of the image, consisting of the detection of existing faces in the image and only then followed by the actual task of creating a model which can further recognize emotions in unseen images.

2. Architecture of the system

The system is composed of 3 main modules:

- **The UI of the application** has the role of providing different parametrization for the application, as well as different operations which are available for the user, such as:
    - Choosing **the directory of the dataset for training**. The directory to be chosen needs to have a specific format, namely it should contain as many directories as emotional states the system proposes to recognize. All the images representing a specific emotion should lie in the corresponding folder, no sub-folders being allowed.
    - The **face detection algorithm** the system wants to use for training or for prediction.
    - The **architecture of the neural network (NN)**, which will be used for training the corresponding data.
    - The **training parameters of the (NN).**
    - Choosing the directory where the generated model should be saved.
    - Choosing the model for performing a prediction
    - Choosing an image for testing
    - Choosing a video for testing. This option will generate an avi file, in which each frame with a detected face will be labeled.
    - The UI is represented as in the following diagram

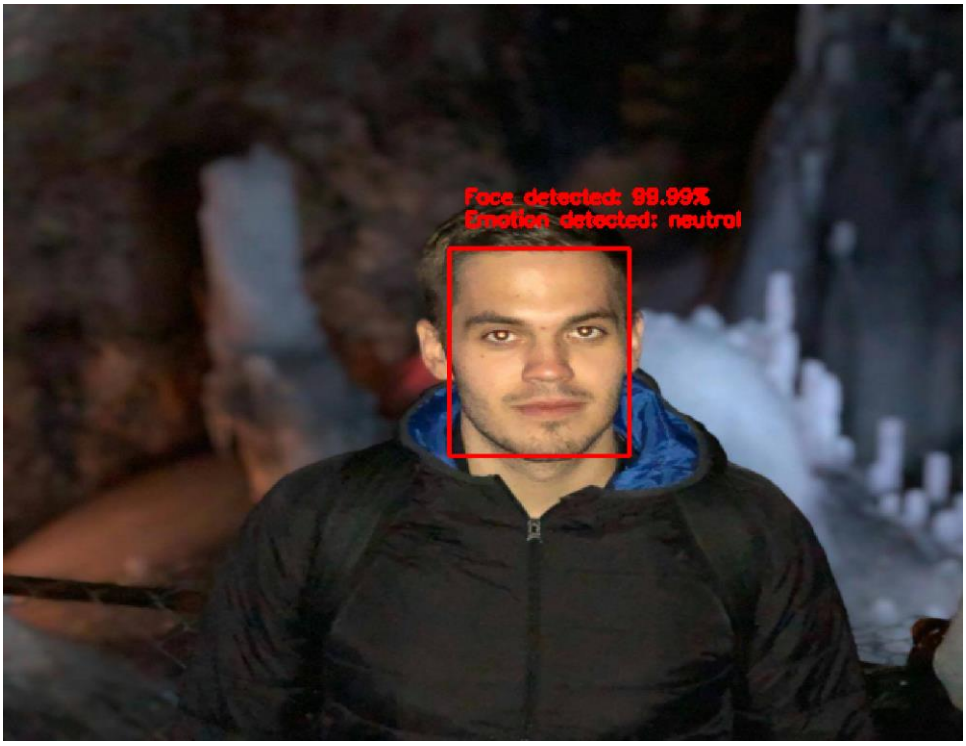


- **The face detection module**
  - A pre-processing of the raw image is done before applying any face detection algorithm, with the goal of canceling any existing noise or unbalance in the image.
  - The user has the option to choose between two face detection algorithms from the opencv library
    - The first algorithm is based on the classical Cascades classifier, which uses the Haar features in order to find matching patches and detect faces in the image. The algorithm has the disadvantage of detecting many FPs in real-world scenarios. More details can be found here https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html
    - The second algorithm is based on using a DNN, which was trained on around 3 million faces in different postures, having different orientations and face sizes. An example of how the DNN of opencv is applied to face detection is provided in the following https://www.pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning/, which was also used in the current project.

- Example of result

- **The facial expression recognition module**
  - The data handling is firstly done, the corresponding labels are obtained for each of the image in order to prepare the data for the training
  - The module has two main functionalities
    - Training a model from scratch based on the different parameters set at the beginning by the user (batch size, LR, image dimension – which will actually be the dimension of the detected face from the previous step, the dropout, the number of epochs, the split percentage for training-validation) and saving the model
    - Testing an existing model
  - The generated model is saved in the "h5" format, specific for the Keras API.
  - Example of results


Face detected: 97.97%
Emotion detected: happy
Face detected: 99.55%
Emotion detected: surprise
00:01:10:27
10-10-94


Face detected: 99.99%
Emotion detected: neutral