MOODYLYSER

Project Report

By TEAM ZAPS

EKLAVYA MENTORSHIP PROGRAMME

At

SOCIETY OF ROBOTICS AND AUTOMATION,
VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE
MUMBAI

JUNE 2020

ACKNOWLEDGMENT

We wish to express our gratitude to the **SRA team of VJTI**, for organizing the **EKLAVYA** mentorship programme, and for giving us this most treasured opportunity to participate in this programme of the highest calibre. From the selection code-offs to the weekly project reports, they were a huge motivator every step of the way. This unique learning experience has changed us for the better, and will remain an essential milestone that forever encourages us to pursue more projects of the same calibre.

Member 1 full name:
Anushree Bapusaheb Sabnis
Email id:
sabnisanushree@gmail.com

Member 2 full name:
Mohammed Saad Hashmi
Email id:
hashmis104@gmail.com

TABLE OF CONTENTS

Project Overview

USE CASE:

1. The user launches the program.
2. The user's face is in view of the webcam as it records a live video feed
3. The system detects a person's face through the live video feed.

4. The system detects 64 facial landmarks on the detected person's face and feeds it into the pre-trained model.
5. The system analyses the emotions displayed by the user during the run-time of the program.
6. The system displays the live video feed with dynamic emotion recognition.
7. The user exits the program when needed.
8. The system statistically analyses the emotions it records during runtime.
9. The system displays the statistical analyses.

TECHNOLOGY USED IN THIS PROJECT:

1. **Python:**
2. Computer vision with **OpenCV**:

   OpenCV is a library of programming functions mainly aimed at real-time computer vision. **Moodylyser** employs computer vision for **face-detection**, used as a "switch" which activates the project automatically upon detection of a face.

3. Data-handling with **NumPy:**

   NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. Moodylyser makes use of NumPy for data-handling, to store pixel values of images, to store labels of said images upon extraction from the dataframes into multidimensional arrays.

4. Statistical analyses with **Matplotlib**:

   Matplotlib is a plotting library for the Python programming language that provides an object-oriented API for embedding plots into applications. Moodylyser used Matplotlib to display graphs depicting the frequency of detected emotions during runtime.

5. Scikit-learn:

   Scikit-learn is a free software machine learning library for the Python programming language.

6. Pandas:

   pandas is a software library written for the Python programming language for data manipulation and analysis.

7. Keras:
8. Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible
9. dlib:

A facial landmark detector that can be used to detect facial landmarks in *real-time* with high quality predictions.

Moodylyser is an image processing project that analyses a dataset of images depicting various human emotions.We have programmed our model to extract features such as facial gestures related to various emotions. The emotions detected in this project are : “Negative-emotions” (“Anger”, “Disgust”, “Fear”), “Happiness”, “Sadness”, “Surprise” and “Neutral”.

For the model to run, we will be converting live web-cam(video) feed into grayscale and give it the desired dimensions.We will be saving the pixel data numpy arrays, which we will save into csv files to be read into our models.

INTRODUCTION

General :

Emotions can be defined as “instinctive or intuitive feelings as distinguished from reasoning or knowledge”. But their detection doesn’t lie outside the realm of logic and science. Using Computer Vision to detect emotions on a human face could very well be a milestone in the understanding of human emotions by Artificial Intelligence; as well as a deeper understanding of the same by humans. A Convolutional Neural Network(CNN), as used in this project, detects emotions much like human brains do. It starts off by its lower layers detecting edges and lines on a human face, moves on to grouping them into facial gestures, upon which it detects emotions linked to those gestures in its higher levels.

Machine Learning is the study of Computational Algorithms that are designed to automatically improve their performance. This project works within the domain of Artificial Intelligence by Deep Learning. Deep Learning is Machine Learning that deviates from the connectivist model which is biologically based. Not every layer inside a Deep Learning Neural network is connected to the next layer, and the “Deep” comes from the use of multiple layers within a neural network.

We endeavoured to create an unsupervised neural network, capable of extracting facial gestures by building on edge extractions from raw data layer by layer with minimal preprocessing. A CNN was the best fit for the task. Convolution is a mathematical

operation to merge two sets of information. In our case the convolution is applied on the input data using a *convolution filter* to produce a *feature map*. The convolution operation is performed by sliding the filter/kernel over the input. The size of the output(feature map) is determined by the formula:

$$Y = [(X + 2P - f)/S] + 1$$

Where:

Y x **Y = Shape of the output**

X x **X = Shape of the input**

P = ZeroPadding applied on the input

f x **f = Shape of the filter/kernel**

S = Strides applied to the layer

The types of layers used for our CNN model were : Convolutional layers, Dropouts, MaxPooling2D and Dense convolutions. [1]

Model Architecture :

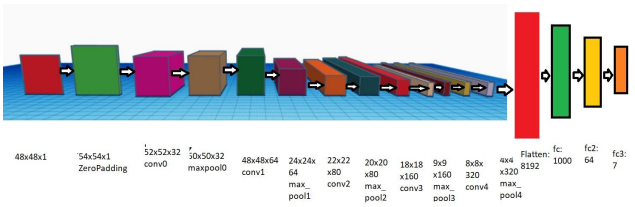


Fig 1 : Model Architecture

METHOD

We used Keras API to define our model, which consists of 1,378,363 trainable parameters. The model was trained on the **fer2013** dataset from the Kaggle challenge : CHALLENGES IN REPRESENTATION LEARNING : FACIAL EMOTION RECOGNITION CHALLENGE [2]. To better improve accuracy, we used the pre-trained facial landmark detector in the **dlib** library[3]. We controlled the learning-rate of the model by using a custom “LearningRateScheduler” callback instead of using Exponential decay in our optimizer, as this method resulted in better accuracy. An Alternative to this could be the use of the Keras Callback “ReduceLRonPlateau”

Instead of using Keras API, the proposed project could be completed using Tensorflow. We employed Keras as it is a high-level API capable of running on top of **TensorFlow**, CNTK and **Theano**. It has gained favor for its ease of use and syntactic simplicity, facilitating fast development.

An alternative method to going about this project would be to use “Transfer Learning”. The “VGG-net” has been proved to work well on this dataset. The cons of this are that, as a beginner, it could not give you a deeper understanding of CNN architecture.

The optimizer used in this project is “Adam”, which provided the best results over “SGD” and “RMSProp”

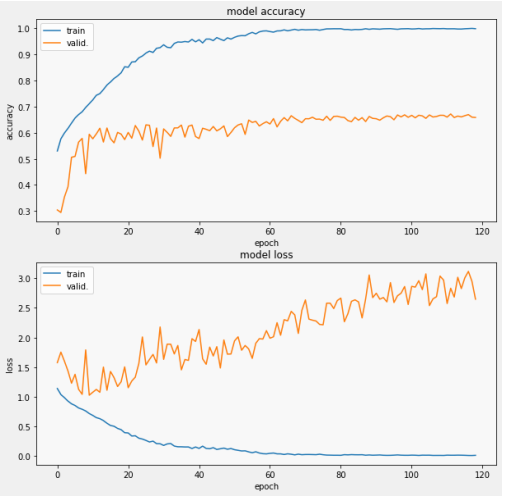
WORKFLOW OF THE PROJECT:

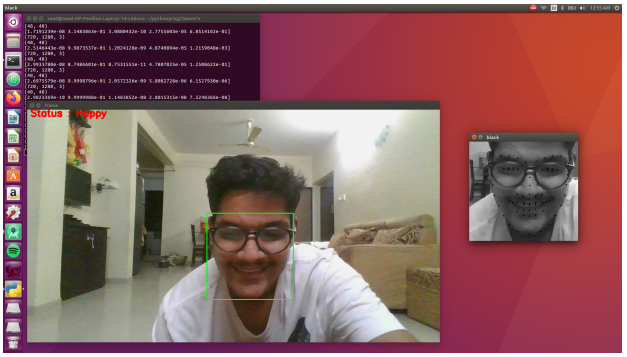
1. Weeks 1-2: Learning concepts of OpenCV and CNN
2. Week 2: Data-preprocessing and building a basic CNN model to improve upon.
3. Week 3: After hitting a dead-end at 60% validation accuracy, found ways to improve accuracy by using **dlib.**
4. Week 3-4: Trying to improve model accuracy using techniques such as Dropouts, Regularization, Kernel Constraints and experimenting with optimizers.
5. Week 4: Prepared final model using callback “LearningRateScheduler” and optimizer “Adam”.

EXPERIMENTAL RESULTS AND ANALYSIS

A variety of models with different architectures were employed[4], but we didn’t get sufficient accuracy till we made a model by modifying the dataset, making use of **dlib.** In our quest to improve the accuracy of the model, we used regularizers and dropouts, as well as kernel constraints. But they weren’t as efficient on our model as we wanted them to be. We have now identified the problem to be the underrepresentation of a certain class of emotion called “Disgust” in our database. We are working to improve upon this in our further work with this project.

Fig 2 : Model performance : (A) Model Learning Curve (metrics : accuracy | loss) (B) Emotion detection using markers





As evidenced, they display certain overfitting and signs of under-representation of classes in the dataset. We plan to remedy this by dropping the “Disgust” class and merging the images with the “Fear” class.

CONCLUSION

We have attained a 99.8% training accuracy and a 67.13% validation accuracy using our model. The model is able to work with live video feed and able to graphically plot the frequency of the abovementioned 7 emotions.

We were able to increase the accuracy of this model by employing techniques such as Learning Rate Scheduling and Pre-detection of facial gestures to boost the model's performance.

We were not able to diminish the overfitting in our trained model, but we endeavour to do that by modifying the hyperparameters and employing Image Augmentation on underrepresented classes.

FUTURE SCOPE

We aim to release this model in the form of a website/app. It has multiple applications as it takes in data from a live video feed. A suitable application that we plan to be working on is to use *MOODYLYSER* in therapy sessions, to accurately gauge the emotions a patient displays during the course of therapy.
We also plan to integrate the model with an ESP-eye[5] to create a portable desk-accessory which can be used for emotion regulation, instead of using the user's webcams.
We plan to convert this project into a mobile app as a mental-health-tracker, prompting the user to take small videos of their faces after a set duration and providing an accurate analysis of their emotions throughout the day.
The project has wide and varied applications in AI, and we plan to pursue that avenue by creating a chat-bot which has appropriate responses to your current emotional state.

REFERENCES

[1]Visualizing CNN:

https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

[2]CChallenges in Representation Learning: Facial Expression Recognition Challenge

[3]dlib C++ Library : http://dlib.net/

[4]Experimentation with model architecture:
https://www.kaggle.com/notebooks?sortBy=dateRun&group=profile&pageSize=20

[5]ESP-eye:
https://www.espressif.com/en/products/devkits/esp-eye/overview#:~:text=ESP%2DEYE%20is%20a%20development,Megapixel%20camera%20and%20a%20microphone.&text=It%20also%20supports%20image%20transmission,through%20a%20Micro%2DUSB%20port.

[6]dlib applications:
https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/