SADNESS DETECTION FOR FUTURE SMART HOMES

An Undergraduate Research Scholars Thesis

by

SIYUAN YANG

Submitted to the LAUNCH: Undergraduate Research office at
Texas A&M University
in partial fulfillment of requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by
Faculty Research Advisor: Anxiao Jiang

May 2021

Major: Computer Science

RESEARCH COMPLIANCE CERTIFICATION

Research activities involving the use of human subjects, vertebrate animals, and/or biohazards must be reviewed and approved by the appropriate Texas A&M University regulatory research committee (i.e., IRB, IACUC, IBC) before the activity can commence. This requirement applies to activities conducted at Texas A&M and to activities conducted at non-Texas A&M facilities or institutions. In both cases, students are responsible for working with the relevant Texas A&M research compliance program to ensure and document that all Texas A&M compliance obligations are met before the study begins.

I, Siyuan Yang, certify that all research compliance requirements related to this Undergraduate Research Scholars thesis have been addressed with my Research Faculty Advisor prior to the collection of any data used in this final thesis submission.

This project did not require approval from the Texas A&M University Research Compliance & Biosafety office.

TABLE OF CONTENTS

ABSTRACT

Sadness Detection for Future Smart Homes

Siyuan Yang
Department of Computer Science and Engineering
Texas A&M University

Research Faculty Advisor: Anxiao Jiang
Department of Computer Science and Engineering
Texas A&M University

This thesis focuses on sadness detection and recognition using deep learning and image processing in python. It analyzes accurate and efficient ways to collect a large set of “moments” from YouTube videos to build large-scale databases for “moments” that show the emotion of sadness. For the overall model architecture, a sequential neural network model is built with three fully connected convolutional layers and rectified linear units as our activation function. Initially, we obtain a nearly zero false positive rate and around ten percent false negative rate on this trained model. To further improve the accuracy and efficiency, the Haar Cascade classifier is used to crop only frontal face images and OpenPose is used to locate facial key points to precisely detect and analyze the facial expression. Besides, we crawl the YouTube network to acquire the video information and used natural language processing to filter the videos that are more likely to contain the emotion sadness. By incorporating the deep learning model with the above algorithms, “moments” that contain the emotion of sadness are extracted from YouTube videos and output as a JSON file, which can be viewed via the iLab AI-Human Video Database.

ACKNOWLEDGEMENTS

Contributors

I would like to thank my faculty advisor, Dr. Andrew Jiang, and our PhD student lab member, Jeff Hykin, for their guidance and support throughout the course of this research.

Thanks also go to the department faculty and staff for making my time at Texas A&M University a great experience.

Finally, thanks to my friends and colleagues for their encouragement and to my parents for their patience and love.

The dataset “Challenges in Representation Learning: Facial Expression Recognition Challenge” used for training and testing the model was provided by the machine learning and data science community: Kaggle. The iLab AI-Human Video Database used for exploring and verifying data was provided by Jeff Hykin. OpenPose used for facial key points detection was provided by CMU Perceptual Computing Lab.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

This work received no funding.

1. INTRODUCTION

Imagine in the future, we are going to have smart homes installed with all kinds of smart cameras, and these smart cameras can recognize the actions and emotions of people who live in smart houses. We hope to use these smart cameras to detect what people are doing or how they are feeling so that we can improve their lives and take care of them. For instance, does the person fall and need help? Does he/she look tired or not getting enough sleep? Does the resident look happy today, or sad? Under this scenario, the target emotion of this work is sadness. Nothing in our lifetimes can be compared with the magnitude of the COVID-19 pandemic. A variety of studies have proved that the impact of the COVID-19 crisis on mental health was severe, and there was a significant increase in suicide deaths attributed to fears of contagion, social isolation, and psychological distress. Understanding and helping humans is a key part of artificial intelligence. How can we let machines help humans during the current COVID-19 epidemic? In this thesis, we present several efficient methods to extract the moments in downloaded YouTube videos that show the emotion of sadness and build large-scale databases for sad video clips in YouTube videos. The research roadmap of this work is as follows:

1. We develop a sequential neural network model to detect the emotion of sadness in images and optimize its hyper-parameters using image data preprocessing and configuring random transformations and normalization during training.
2. We improve the accuracy of detecting the emotion of sadness by capturing frontal faces using Haar cascade classifiers and detecting and analyzing facial key points using OpenPose library and the Dictionary of Body Language.

3. We crawl the YouTube network and acquire the video information such as video titles, video length, and related videos by using the BeautifulSoup and Requests libraries to build a large graph of YouTube videos.
4. We apply natural language processing methods on YouTube titles using the spaCy library to filter videos that are more likely to contain sad moments and import the python package Text2Emotion to pull out the emotions from the titles.
5. We extract the video clips that contain the emotion of sadness from a large graph of YouTube videos and store the information on the moments in iLab AI-Human Video databases in suitable ways.

Initially, we obtain a nearly zero false positive rate and around ten percent false negative rate on the trained model. By applying the cascade classifiers and OpenPose library, the accuracy of detecting sad video clips is largely improved. The false negative rate now decreases to around 5 percent. By building such databases, we can allow machines to understand people's emotions accurately and effectively by identifying or detecting their features of facial expressions and thus providing services accordingly.

2. RELATED WORK

The paper [1] focuses on emotion detection using Image Processing in Python. The work has been implemented using Python (2.7, Open Source Computer Vision Library (OpenCV) and NumPy, specifically designed to develop a system which can analyze the image and predict the expression of the person.

The paper [2] focuses on emotion recognition using facial expressions. It analyzes the strengths and the limitations of systems based only on facial expressions or acoustic information, and it also discusses two approaches used to fuse these modalities: decision level and feature level integration.

The paper [3] focuses on the problem space for facial expression analysis, which includes level of description, transitions among expressions, eliciting conditions, etc. It presents the CMU-Pittsburgh AU-Coded Face Expression Image Database, which is the most comprehensive testbed to date for comparative studies of facial expression analysis.

The paper [4] focuses on a fuzzy relational approach to human emotion recognition from facial expressions and its control. It proposes the use of external stimulus to excite specific emotions in human subjects whose facial expressions are analyzed by segmenting and localizing the individual frames into regions of interest.

The paper [5] gives a neural network based approach for face detection. The paper [6] and [7] describes a machine learning approach for visual object detection and presents a general trainable framework for object detection in static images of cluttered scenes. The paper [8] describes a face detection framework that can process images extremely rapidly with high detection rates.

3. METHODS

3.1 Sequential Network Architecture

The proposed model is designed based on some references from Keras Library, developer guides, and online tutorials such as how to build the sequential model[1], convolutional neural network with practical implementation[2], etc. For the overall architecture, a sequential neural network model is defined with three fully connected convolutional layers and one activation function: ReLU. It is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor. [3]For the first layer, the 2D convolution layer is used to create a convolution kernel that is convolved with the layer input to produce a tensor of outputs. The input shape is set as (48, 48, 1) for 48*48 RGB pictures. Then, the max pooling 2D layer is added to this model to down-sample the input representation by taking the maximum value over the windows defined by pool size (5, 5) for each dimension along the feature axis. The window is shifted by strides set as (2, 2) in each dimension. For the second and third convolution layers, two 2D convolution layers are added to the model. Instead of using the max pooling 2D operation, the average pooling 2D operation is used this time for spatial data, and the pool size is set to (3, 3) and strides are set to (2, 2). Finally, the flatten function is called in the model to convert the pooled feature map to a single column that is passed to the fully connected layer, and the Dense layer is added to feed the fully connected layer to the neural network. More details about the model can be found in Figure 3.1 and Figure 3.2.

[1] Francois Chollet, 2020
[2] Amir Ali, 2019
[3] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, … X. Zheng, 2015

Layer (type)	Output Shape	Param #
conv2d_10 (Conv2D)	(None, 44, 44, 64)	1664
max_pooling2d_6 (MaxPooling2	(None, 20, 20, 64)	0
conv2d_11 (Conv2D)	(None, 18, 18, 64)	36928
conv2d_12 (Conv2D)	(None, 16, 16, 64)	36928
average_pooling2d_2 (Average	(None, 7, 7, 64)	0
conv2d_13 (Conv2D)	(None, 5, 5, 128)	73856
conv2d_14 (Conv2D)	(None, 3, 3, 128)	147584
average_pooling2d_3 (Average	(None, 1, 1, 128)	0
flatten_1 (Flatten)	(None, 128)	0
dense_3 (Dense)	(None, 1024)	132096
dropout_2 (Dropout)	(None, 1024)	0
dense_4 (Dense)	(None, 1024)	1049600
dropout_3 (Dropout)	(None, 1024)	0
dense_5 (Dense)	(None, 7)	7175

Total params: 1,485,831
Trainable params: 1,485,831
Non-trainable params: 0

Figure 3.1: Sequential neural network model summary

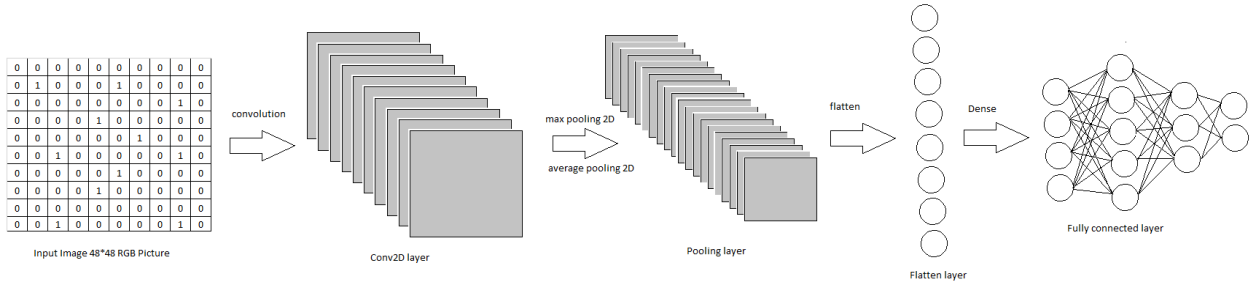

Figure 3.2: Sequential neural network model architecture

3.2 Training Dataset for Deep Learning

The training dataset can be found in Kaggle "Challenges in Representation Learning: Facial Expression Recognition Challenge[4]". The data consists of 28,709 examples of 48*48 pixels gray scale images of faces. The faces have been automatically registered so that the face is centered and occupies about the same amount of space in each image. Each face is categorized into one of seven classes (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral) based on the emotion shown in the facial expression.

3.3 Overall Model Training and Performance

To optimize the hyper-parameters, image data preprocessing is used to generate batches of image data with real-time data augmentation and configure random transformations and normalization operations during training[5]. These generators can then be used with the Keras model methods that accept data generators as inputs. Then, the model is compiled and fit to train the dataset with Adam optimization and train for randomly selected ones. To improve the accuracy of the metrics, the steps per epoch are set to 256 and repeated by 25 epochs or 6400 batches. As a result, the training accuracy is 96.37, and the training loss is 0.10; the test accuracy is 56.78, and the test loss is 3.22. There is overfitting in the training model as the test accuracy is much lower than the training accuracy. However, after conducting several experiments, it has been found that the model works well on samples with higher image quality, whole frontal face captured, or obvious emotion states. Otherwise, the accuracy is relatively low, as sometimes it could be hard for the model to detect emotions in low resolution of images or images with low facial proportions. Figures 3.3 and 3.4 are two examples that the model works well on. Overall, the model is satisfactory for practical applications with above-mentioned requirements.

[4] Pierre-Luc Carrier, Aaron Courville, 2012
[5] Francois Chollet, 2016

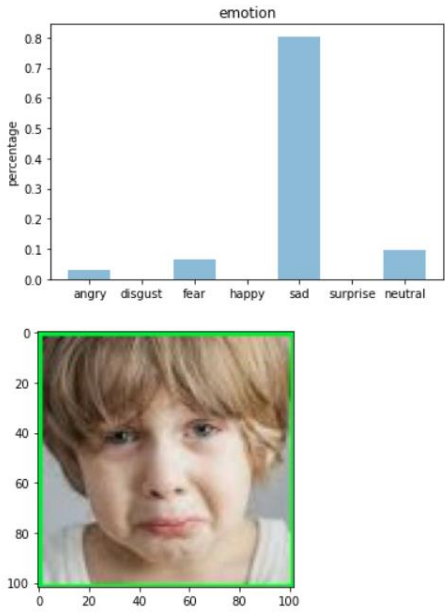

Figure 3.3: Emotion plot on a sad facial expression

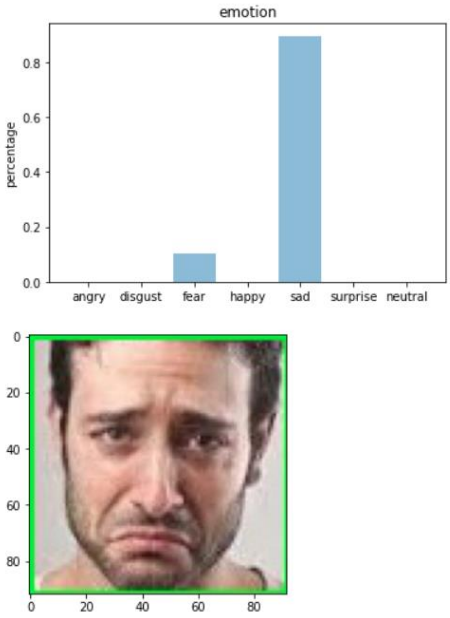

Figure 3.4: Emotion plot on a sad facial expression

3.4 Frontal Face Detection Using Haar Cascade Classifiers

To further improve the accuracy, Haar feature-based cascade classifiers are used for the facial recognition of the frontal face and the capture of the minimized frame that can hold the

face. It is an effective face detection method introduced by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features"[6] in 2001. The algorithm requires large numbers of images with and without faces to train and test the classifier. To achieve that, Haar-wavelet[7], proposed by Alfréd Haar in 1909, is frequently used in this method, which helps detect the most relevant features of facial expressions in gray-scaled images. These Haar-features are very similar to convolutional kernels in a way that each feature is a single value calculated by subtracting the average sum of the white pixel intensities from that of the black pixel intensities. Each Haar-feature is assigned to every single important feature on human frontal faces such as eyes, noses, mouths, etc. Figures 3.5 and 3.6 show the combination of white pixels and black pixels with ideal Haar-feature pixel intensities, which can be used to detect edges and lines. For instance, eyebrows are darker and thus can be regarded as a black pixel, while the forehead can be treated as a white pixel. In this case, edge features can be used to locate eyes in the frontal face. Similarly, line features can help locate the mouth in the frontal face when regarding the lip as black pixels and teeth as white pixels. By applying the Viola-Jones algorithm, the features will be categorized into different stages of the cascade of classifiers by comparing their closeness of the real image to the ideal case and applied one by one for locating the frontal face in the image. Simply put, this method allows the images to best fit the model and thus giving more accurate prediction results on the cropped frames.

[6] P. Viola, M. Jones, 2001
[7] Charles K. Chui, 1992

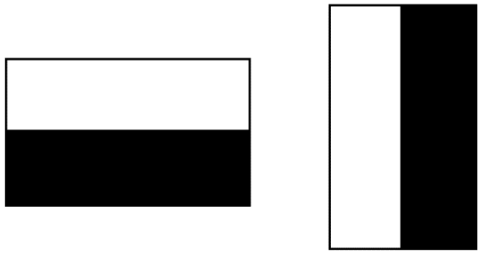

Figure 3.5: Edge features

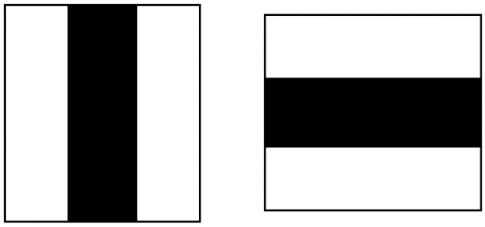

Figure 3.6: Line features

3.5 Facial Key Points Detection Using OpenPose

OpenPose, developed by CMU Perceptual Computing Lab, is used to detect multi-person facial key points on single images to precisely detect and analyze facial expressions. This method uses a nonparametric representation to learn to associate body parts with individuals in the image, and the bottom-up system achieves high accuracy and real-time performance[8]. The standard for sadness used in this method is referred to as "The Dictionary of Body Language". There are mainly three facial features indicating sadness: inner corners of the eyebrows pulled up and together, upper eyelids dropped and eyes looking down, and lip corners pulled downward. Figure 3.7 is one of the OpenPose JSON output formats, which saves the people's facial key points data onto JSON files. Each output contains facial part locations (x, y) and detection confidence (c). To detect the facial expression of sadness, we need to locate the inner corners of

[8] Joe Navarro, 2018

the eyebrows $(y20, y21, y22, y23)$, upper eyelids and eyeballs $(y36, y37, y44, y45, y68, y69)$, and lip corners $(y48, y49, y53, y54, y60, y64, y65, y67)$. We consider a frame as a sad moment when it meets all the requirements for the sadness standard. By combining this method with cascade classifiers, we improve the accuracy of detecting sad moments in YouTube videos.

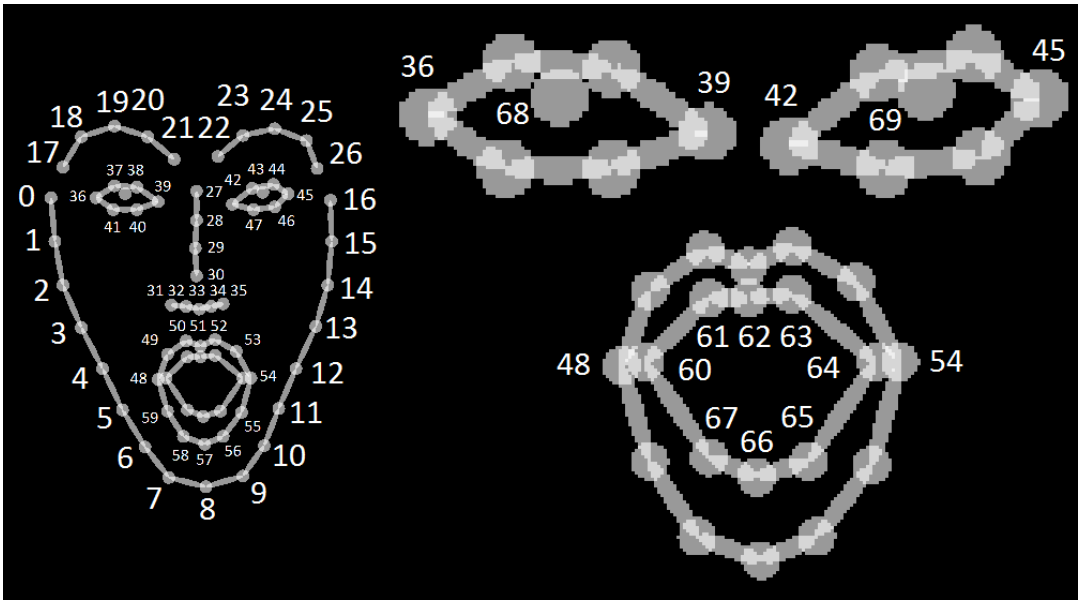

Figure 3.7: OpenPose 2D facial key points output format[9]

3.6 Natural Language Processing on YouTube Titles

3.6.1 YouTube Network Crawling Using BeautifulSoup and Requests Libraries

Some of the videos acquired through YouTube Data API do not even have sad moments or only contain few seconds. Spending time predicting these videos is not wise. To further improve the efficiency, natural language processing methods are applied to the titles of YouTube videos to search for videos that are more likely to contain the emotion of sadness. In Figure 3.8, BeautifulSoup and Requests libraries are used to crawl the YouTube network and acquire the titles of YouTube videos given their YouTube ids by sending HTTP requests.

[9] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, Y. A. Sheikh, 2019

```python
# crawl the YouTube website to get the video title
from bs4 import BeautifulSoup
import requests

def get_title(video_id):
    url = "https://www.youtube.com/watch?v=" + video_id
    r = requests.get(url)
    s = BeautifulSoup(r.text, "html.parser")
    title = s.find("title").text
    return title
```

Figure 3.8: Annotated codes for YouTube network crawling

3.6.2 YouTube Titles Preprocessing Using spaCy Library

In Figure 3.9, the spaCy library is imported for advanced natural language processing on acquired YouTube titles. The first step is to get rid of numbers, punctuations, and stop words and only keep the English words in the given title. Then, reduce the token to its lowercase lemma form for coherence. Finally, the spaCy library is used to tokenize the filtered text and break the text into meaningful units and perform the lemmatization to reduce inflected forms of a word[10].

```python
# set the limit to the token
def is_token_allowed(token):
    if (not token.is_alpha or not token.string.strip() or token.is_stop
        or token.is_punct or token.string == 'YouTube'):
        return False
    return True
```

```python
# reduce token to its lowercase lemma form
def preprocess_token(token):
    return token.lemma_.strip().lower()
```

```python
# get the processed lemma
import spacy

def get_lemma(video_id):
    doc_lemma = []
    nlp = spacy.load('en_core_web_sm')
    text = get_title(video_id)
    doc = nlp(text)
    filtered_doc = [preprocess_token(token) for token in doc if is_token_allowed(token)]
    sentence = ' '.join(filtered_doc)
    final_doc = nlp(sentence)
    for token in final_doc:
        doc_lemma.append(token.lemma_)
    lemma_sents = ' '.join(doc_lemma)
    return lemma_sents
```

Figure 3.9: Annotated codes for natural language processing on YouTube titles

[10] T. Singh, 2020

3.6.3 Sadness Detection on Processed YouTube Titles Using Text2Emotion

The python package Text2Emotion is applied to pull out the emotions from the content, process any textual data, recognize the emotion embedded in it, and provide the output in the form of a dictionary[11]. The result is well suited with 5 basic emotion categories as Happy, Angry, Sad, Surprise, and Fear. By performing Text2Emotion on the processed YouTube titles, if the filtered text contains the emotion of sadness in the returned dictionary, the YouTube ID is pushed to the list for further implementations. Figure 3.10 is an example of the output in terms of the dictionary where emotion categories along with the respective score are printed out, and the emotion embedded in it is sadness.

```
import text2emotion as te
text = get_lemma(video_id_as_a_string)
print('Filtered YouTube Title: ' + text)
print(te.get_emotion(text))
```

```
Filtered YouTube Title: emotional whatsapp status cry man video
{'Happy': 0.0, 'Angry': 0.0, 'Surprise': 0.0, 'Sad': 1.0, 'Fear': 0.0}
```

Figure 3.10: Annotated codes for pulling out emotions from YouTube titles using Text2Emotion

[11] A. Gupta, A. Band, S. Sharma, K. Bilakhiya, 2020

4. RESULTS

4.1 Steps to Detect "Moments" of Sadness on YouTube Videos

To detect the "moments" in downloaded YouTube videos, the given YouTube video is cut frame by frame and stored in the local machine. Then, the score for sadness is predicted on each frame by using the model. Since the downloaded video is cut into frames per second, the start time is just the frame number. If the prediction score for sadness on the frame is greater than or equal to 0.5 and is the maximum in the seven emotion categories, the frame will be regarded as a sad moment and the frame number will be recorded and finally printed out to the screen.

4.2 Measuring Accuracy of Detecting Sad "Moments" on YouTube Videos

The false positive rate (FPR) is a measure of accuracy for a machine learning model test[12]. The FPR is calculated by FP divided by the sum of FP and TN, where FP is the number of false positives and TN is the number of true negatives. Similarly, the false negative rate (FNR) is the probability that a true positive will be missed by the test. It is calculated by FN divided by the sum of FN and TP, where FN is the number of false negatives and TP is the number of true positives. Here are the definitions for each possible outcome.

1. True Positive: the truth is positive, and the test predicts a positive.
2. True Negative: the truth is negative, and the test predicts a negative.
3. False Negative: the truth is positive, but the test predicts a negative.
4. False Positive: the truth is negative, but the test predicts a positive.

When applying the model to YouTube videos, the numbers of false positives, false negatives, true positives, and true negatives are noted down on the prediction result of each

[12] Glossary, 2020

frame. It turns out that the false positive rate is nearly zero, and the false negative rate is around 10 percent, which means the accuracy of the model is quite satisfactory. By applying the cascade classifiers and OpenPose library, the false negative rate decreases to around 5 percent.

4.3 Measuring Efficiency of Detecting Sad "Moments" on YouTube Videos

The first step is to acquire a list of YouTube videos and create folders for each of the videos in the list so that frames of videos can be stored in separate folders shown in Figure 4.1.

```
# create folders to store frames of different videos
for i in range(len(url_list)):
    os.chdir('/home/steven/Desktop/URS/finding_happiness/image')
    path = url_list[i]
    try:
        os.mkdir(path)
    except OSError:
        print ("Creation of the directory %s failed" % path)
    else:
        print ("Successfully created the directory %s " % path)
```

Figure 4.1: Annotated codes for creating folders to store frames of different videos

The next step is to download the video objects in the list to the local machine and cut all the videos frame by frame and save them to the corresponding folders shown in Figure 4.2.

```
# loop through the whole url list
for url in url_list:
    # download the video object to local
    os.chdir('/home/steven/Desktop/URS/finding_happiness/emotion_detect')
    video_object = DatabaseVideo(url)
    video_object.frames
    # cut the videos frame by frame and save to local
    index = url_list.index(url)
    imagePath = '../image/' + url
    videoPath = '../video.nosync.cache/name_' + url + '.mp4'
    vidcap = cv2.VideoCapture(videoPath)
    os.chdir(imagePath)
    sec, frameRate, cnt = 0, 1, 1
    success = get_frames(sec)
    while success:
        cnt += 1
        sec += frameRate
        sec = round(sec, 2)
        success = get_frames(sec)
```

Figure 4.2: Annotated codes for downloading the videos and cutting them frame by frame

Finally, loop through the whole video list and extract and print out all the sad moments to the screen if the frame is regarded as a sad moment shown in Figure 4.3 and 4.4.

```
def extract_moment(url):
    time_list = []
    file_no = len(os.listdir('../image/' + url))
    os.chdir('/home/steven/Desktop/URS/finding_happiness/emotion_detect')
    # start the timer
    start = timeit.default_timer()
    # loop through all the frames
    for i in range(1, file_no):
        # crop the frames with only frontal face
        face_crop('../image/'+ url + '/image' + str(i) + '.jpg', str(i))
        # check if the frame contains the frontal face
        if os.path.exists('../crop/crop'+ str(i) + '.jpg') == True:
            sad_list = sad_plt(str(i))
            # print out the start and end time and append to the list if regarded as a sad moment
            if sad_list[4] == max(sad_list) and sad_list[4] >= 0.5:
                print('Start Time (seconds): ', i)
                print('End Time (seconds): ', i + 1)
                time_list.append([i, i+1])
    # end the timer
    stop = timeit.default_timer()
    # print out the runtime
    print('Runtime (seconds): ', stop - start)
    return time_list
```

Figure 4.3: Annotated codes for the function of extracting sad moments

```
# print out sad moments for a list of videos
import logging
import tensorflow as tf
tf.get_logger().setLevel(logging.ERROR)

for i in range(len(url_list)):
    time_list = []
    clear_crop()
    print('Video ' + str(i + 1) + ' sad moment:')
    time_list = extract_moment(url_list[i])

    print('----------------------------')

    # write to the json file
    write_to_json(time_list, url_list[i])
```

Figure 4.4: Annotated codes for printing out the sad moments and writing to JSON files

By taking the average, it takes 0.475 seconds to perform an emotion prediction on a cropped image and 0.114 seconds to crop the frame with only frontal face in the image. Within an hour, ideally, the algorithm can predict the sadness score on 6112 cropped frames. In other words, the ratio is around 1: 1.69. By applying natural language processing methods to the titles

of YouTube videos, the algorithm searches and filters the videos that are more likely to contain the emotion of sadness. As expected, the efficiency of predicting sadness score is largely improved. The algorithm avoids cropping and predicting sadness score on videos that are less likely to have the emotion of sadness, and the ratio raises up to 1: 2.33.

4.4 The iLab AI-Human Video Database

4.4.1 Ways of Storing Information on the iLab Website

The iLab website is created by one of our Ph.D. student lab members, Jeff Hykin, specifically for organizing and retrieving videos and observational data. Since directly interfacing with a database can be highly unintuitive and tedious, we used this online interface for exploring and verifying data. To store the information on the moments in the database, there are several ways to do it. The first way is to interact with the iLab website "Observation" section shown in Figure 4.6. We need to manually enter the information on the moments after clicking on the "Edit" button at the top right corner. This method is very time-consuming as there are numerous sad moments needed to enter in for each video. The other way is to save the video information such as start time, end time, and video id while extracting the sad moments and automatically generate the JSON file by using the function shown in Figure 4.5. Then, simply upload the JSON file to the iLab database, and the video information will be shown up later.

```python
# write the video information to json file
import json

def write_to_json(time_list, video_id):
    os.chdir('/home/steven/Desktop/URS/finding_happiness/iilvd_interface/json')

    # Writing to json file
    filename = video_id + '.json'
    data = []

    with open(filename, 'w') as f:
        for i in range(len(time_list)):
            dic = {
                "videoId": video_id,
                "type": "segment",
                "startTime": time_list[i][0],
                "endTime": time_list[i][1],
                "observer": "Stevens_Machine",
                "isHuman": False,
                "confirmedBySomeone": True,
                "rejectedBySomeone": False,
                "observation": {
                    "label": "sad",
                    "labelConfidence": 0.99
                }
            }
            data.append(dic)
        json.dump(data, f, indent = 4)
```

Figure 4.5: Annotated codes for generating JSON file based on the video information

4.4.2 View Extracted Moments and Videos on the iLab Website

To view the moments and videos on the iLab website, simply click on the “sad” clip. After the video list is retrieved, the first video is immediately displayed, and the site also skips the video to the time of the first "sad" observation. Looking at the video player and the blue rectangle beneath it, all the observations represented as colored rectangles are displayed according to both their start time and their duration. Thus, the prediction results can be visualized on each extracted moment. For instance, in Figure 4.6, the video “Fifty Shades of Gray Pitch Meeting” only contains one sad moment which lasts around one second. From the website, the start time is 195 seconds, and the end time is 196 seconds. This closely matches with the visualization result which is 193.527 seconds as a start time and 194.763 seconds as an end time.

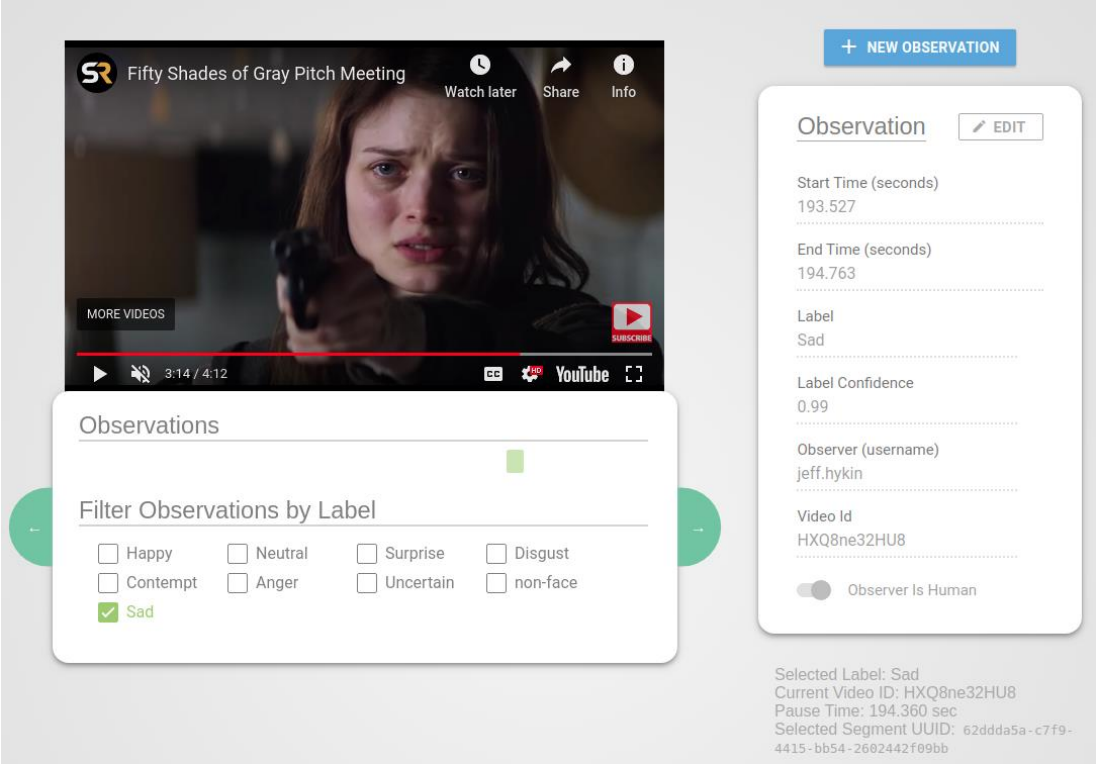

Figure 4.6: View found moments in iLab website

4.4.3 View Stats of YouTube Videos on the iLab website

To view more videos, hover the mouse over the top right corner labeled with “Videos” and click on the videos you want to watch. All the labeled moments have been added to the website in sad clips by uploading JSON files for multiple observations. Besides, the false positive rate of the extracted videos is also shown on this website, which is a measure of accuracy for the machine learning model test. For instance, in Figure 4.7, the false positive rate for ten videos is nearly zero, which means the accuracy of the model is very satisfactory.

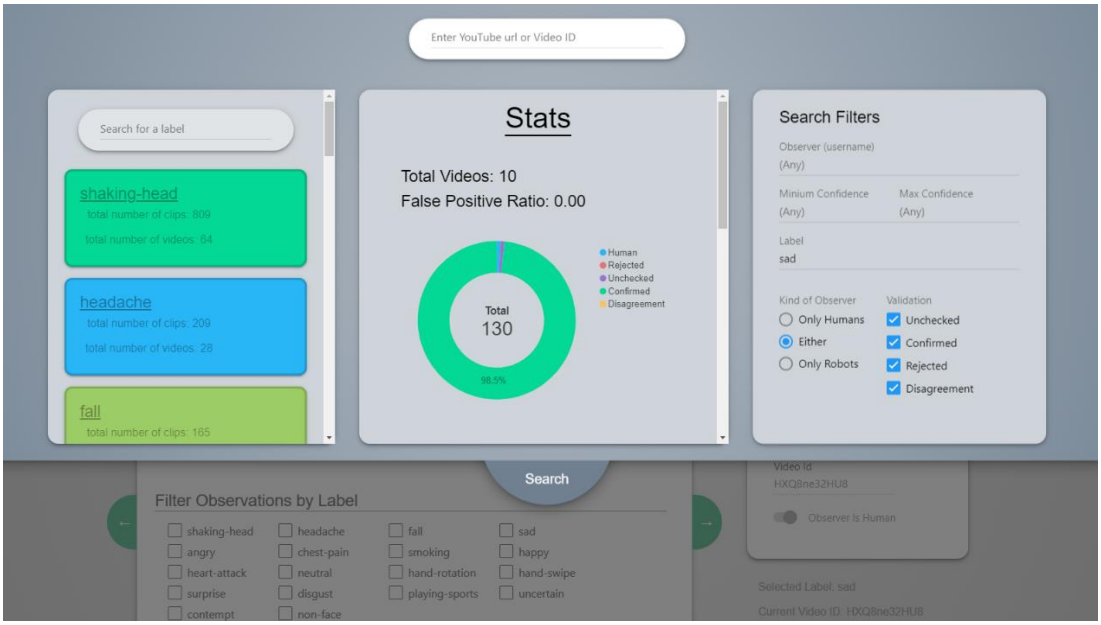

Figure 4.7: View false positive rates in iLab website

5. CONCLUSION

In this work, we studied sadness detection and recognition on YouTube videos using deep learning and image processing in python. Our results indicate that a sequential neural network model built with three fully connected convolutional layers and rectified linear units is satisfactory for practical application, but there exists overfitting in the training model as the test accuracy is much lower than the training accuracy. The reason for the overfitting is that sometimes it is hard for the model to predict emotions in low resolution of images or images with low facial proportions. To further improve the accuracy of the model, the Haar Cascade classifier is used to detect multiple line features and edge features on human faces and crop only frontal face images so that the model can precisely detect the facial expression and predict the emotion.

Initially, the efficiency of applying this model to extract sad moments is also relatively low. Within an hour, the algorithm can predict the sadness score on 6112 cropped frames. In other words, the ratio is around 1:1.69. We found that some of the videos acquired through YouTube Data API do not even have sad moments or only contain few seconds. Spending time predicting these videos is not wise. Thus, we decided to crawl the YouTube network to acquire the video information and use natural language processing to filter the videos that are more likely to contain the emotion sadness. This method largely increases the overall efficiency of extracting sad moments.

The iLab website, also known as iLab AI-Human Video Database, is designed for organizing and retrieving videos and observational data. After the video information on the moments is output as a JSON file and uploaded to this database, all the observations, prediction

results, and false positive rates will show up on the website in an elegant way. It allows us to explore and verify data intuitively so that we can find and resolve the issues effectively and further improve the model if needed.

In this work, we have not explored real-time sadness detection and recognition yet. Our future work includes applying our training models and efficient algorithms to real-time sadness detection and finding ways to provide services accordingly using smart camera systems. Besides, it is also essential to understand how to design deep learning-based smart camera systems and apply OpenCV algorithms for facial expression recognition.

REFERENCES

[1] R. Puri, A. Gupta, M. Sikri, M. Tiwari, N. Pathak, S. Goel, "Emotion Detection using Image Processing in Python", in Computer Vision and Pattern Recognition, 2020.

[2] C. Busso, Zhigang Deng, S. Yildirim, M. Bulut, C. Lee, A. Kazemzadeh, S. Lee, U. Neumann, Shrikanth S. Narayanan, "Analysis of emotion recognition using facial expressions, speech and multimodal information", in ICMI, 2004.

[3] T. Kanade, Y. Tian, J. Cohn, "Comprehensive database for facial expression analysis", Proceedings Fourth IEEE International Conference On Automatic Face And Gesture Recognition (Cat. No. PR00580), 2000.

[4] A. Arunachakraborty, U. Konar, A. Kumar, Chakraborty, Chatterjee, "Emotion Recognition From Facial Expressions and Its Control Using Fuzzy Logic", Proceedings Fourth IEEE International Conference On Automatic Face And Gesture Recognition (Cat. No. PR00580), 2000.

[5] H. Rowley, S. Baluja, T. Kanade, "Neural network-based face detection", in IEEE Patt. Anal. Mach. Intell., volume 20, pages 22–38, 1998.

[6] Paul Viola, Michael Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features", International Journal of Computer Vision, 57(2):137–154, 2004.

[7] C. Papageorgiou, M. Oren, T. Poggio, "A general framework for object detection", in International Conference on Computer Vision, 1998.

[8] K. Sung, T. Poggio, "Example-based learning for viewbased face detection", in IEEE Patt. Anal. Mach. Intell., volume 20, pages 39–51, 1998.

[9] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, Y. A. Sheikh, "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields", in IEEE Transactions on Pattern Analysis and Machine Intelligence, 2019.

[10] Pierre-Luc Carrier, Aaron Courville, 2012, "Challenges in Representation Learning: Facial Expression Recognition Challenge", Kaggle Research Prediction Competition, accessed on August 2020, <https://www.kaggle.com/c/challenges-in-representation-

learning-facial-expression-recognition-challenge/data>.

[11] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, … X. Zheng, 2015, "The Sequential model", TensorFlow: Large-scale machine learning on heterogeneous systems, accessed on Feburary 2021, Software available from tensorflow.org.

[12] Francois Chollet, 2016, "Building powerful image classification models using very little data", The Keras Blog, accessed on Feburary 2021, <https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html>.

[13] Amir Ali, 2019, "Convolutional Neural Nework(CNN) with Practical Implementation", Wavy AI Research Foundation, accessed on Feburary 2021, <https://medium.com/machine-learning-researcher/convlutional-neural-network-cnn-2fc4faa7bb63>.

[14] Francois Chollet, 2020, "The Sequential model", Keras, accessed on December 2020, <https://keras.io/guides/sequential_model/>.

[15] Glossary, 2020, "False Positive Rate", Split, accessed on December 2020, <https://www.split.io/glossary/false-positive-rate/>.

[16] A. Gupta, A. Band, S. Sharma, K. Bilakhiya, 2020, "Detecting emotions behind the text", Text2Emotion, accessed on January 2021, <https://shivamsharma26.github.io/text2emotion/>.

[17] T. Singh, 2020, "Natural Language Processing With spaCy in Python", Real Python, accessed on December 2020, <https://realpython.com/natural-language-processing-spacy-python/>.

[18] Joe Navarro, "The Dictionary of Body Language_A Field Guide to Human Behavior", HarperCollins Publishers Inc., 2018.