Image Sentiment Analysis

Report for Project (CS794)

B. Tech in Computer Science & Engineering

B. P. Poddar Institute of Management & Technology

under

Maulana Abul Kalam Azad University of Technology

Under the supervision of

Dr. Bikromaditya Mondal
and
Mr. Suvadeep Bhattacharjee

Submitted by

Name:	Ankur Karmakar	Name:	Amit Kumar Agarwal
Roll No.:	11500216054	Roll No.:	11500116127
Regn. No.:	161150110134	Regn. No.:	161150110013
Name:	Modhura Das	Name:	Mampi Das
Roll No.:	11500116083	Roll No:	11500117015
Regn. No.:	161150110057	Regn. No.:	171150120009

Academic Year: 2019 - 2020

Department of Computer Science & Engineering
B. P .Poddar Institute of Management & Technology
137, V.I.P Road. Poddar Vihar, Kolkata – 700 052.

Department of Computer Science & Engineering
B.P.Poddar Institute of Management & Technology 137,
V.I.P Road, Poddar Vihar. Kolkata – 700052

CERTIFICATE

This is to certify that the project work entitled *"Image Sentiment Analysis"* submitted by Group No 11, comprising of (Amit Kumar Agarwal, Ankur Karmakar, Modhura Das, Mampi Das), has been prepared according to the regulation of the degree B. Tech in Computer Science & Engineering of Maulana Abul Kalam Azad University of Technology, West Bengal. The candidate(s) have partially fulfilled the requirements for the submission of the project work.

(Signature of HOD)

Dept. of Computer Science & Engg.

(Signature of the Supervisor)

Dept. of Computer Science & Engg.

(Signature of External Examiner)

ACKNOWLEDGEMENT

It is a great pleasure for us to express our earnest and great appreciation to (*Dr. Bikromadittya Mondal and Mr. Suvadeep Bhattacharjee*), our project guide. We are very much grateful to them for their kind guidance, encouragement, valuable suggestions, innovative ideas, and supervision throughout this project work, without which the completion of the project work would have been a difficult one.

We would like to express our thanks to the Head of the Department, *Prof. Ananya Kanjilal* for her active support.

We also express our sincere thanks to all the teachers of the department for their precious help, encouragement, kind cooperation and suggestions throughout the development of the project work.

We would like to express our gratitude to the library staff and laboratory staff for providing us with a congenial working environment.

---------- ----------

(Amit Kumar Agarwal) (Ankur Karmakar)

---------- ----------

(Modhura Das) (Mampi Das)

(Full Signature of the Student(s))

Dept. of Computer Science & Engg.

B.P.Poddar Institute of Management & Technology

Table of Content

- Departmental Mission , Vision, PEO, PO, PSO
- Mapping with PO and PSO
- Justification of Mapping
- Abstract
- Activity Chart
- Introduction
    - What Is Sentiment Analysis
    - Types of Sentiment Analysis
        - Fine-grained Sentiment Analysis
        - Emotion Detection
        - Aspect Based Sentiment Analysis
        - Multilingual Sentiment Analysis
    - Why Perform Sentiment Analysis
    - Applications Of Sentiment Analysis
    - Semantic Analysis in Sentiment Analysis
    - Advantages of Using Sentiment Analysis
    - Disadvantages of Using Sentiment Analysis
    - Challenges For Sentiment
        - *Implicit Sentiment and Sarcasm*
        - *World Knowledge*
        - *Negation Handling*
        - *Diversity And Informality*
- Literature Review
- Gaps In Existing Work
- Theory
- Proposed System / Software
    - Advantages of Using VADER
- Feasibility Study
- Algorithm / Code
    - Code for Sentiment Analysis on Text Extracted From Image Using OCR and VADER Tool
    - Code for Sentiment Analysis on Faces
        - *Code for Sentiment Analysis on Faces Present in Image*
        - *Code for Sentiment Analysis on Faces Present in Live Video*
    - Code for CNN model
    - Code for training classifiers models
        - *Code for training emotion classifier*
        - *Code for training gender classifier*

- Results & Discussions

- Finalized Application
  - *Developing User Interface (UI)*
    - User Interface code of our finalized application
  - *Backend code of our finalized application*
  - *Converting python code to windows executable file*
  - *Running our application*
- Future Plan
- References

1.1 DEPARTMENTAL MISSION

Enrich students with sound knowledge in fundamentals and cutting edge technologies of Computer Science and Engineering to excel globally in challenging roles in industries and academics.

Emphasize quality teaching, learning and research to encourage creative thoughts through application of professional knowledge and skill.

Inspire leadership and entrepreneurship skills in evolving areas of Computer Science and Engineering with social and environmental awareness.

Instill moral and ethical values to attain the highest level of accomplishment and personal growth.

1.2 DEPARTMENTAL VISION

Developing competent professionals in Computer Science and Engineering, who can adapt to constantly evolving technologies for addressing industrial and social needs through continuous learning.

1.3 PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

Graduates of Computer Science and Engineering program will have good knowledge in the core concepts of systems, software and tools for analysing problems and designing solutions addressing the dynamic requirements of the industry and society, while employed in industries or work as entrepreneurs.

Graduates of Computer Science and Engineering program will opt for higher education and research in emerging fields of Computer Science & Engineering towards building a sustainable world.

Graduates of Computer Science and Engineering will have leadership skills, communication skills, ethical and moral values, team spirit and professionalism.

1.4 PROGRAM OUTCOMES (POs)

PO1: Engineering Knowledge: Apply knowledge of Mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem Analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/Development of Solutions: Design solutions for complex engineering problems and design system components or processes that meet specified needs with appropriate consideration for public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct Investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern Tool Usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: Engineer and Society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and Sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and Team Work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project Management and Finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long Learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

1.5 PROGRAM SPECIFIC OUTCOMES (PSO)

PSO1: Students will have proficiency in fundamental engineering and computing techniques and knowledge on contemporary topics like artificial intelligence, data science and distributed computing towards development of optimized algorithmic solutions.

PSO2: Students will have capabilities to participate in the development of software and embedded systems through synergized teams to cater to the dynamic needs of the industry and society.

2. PO & PSO MAPPING

Example given for your Reference:

PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO1	P012	PSO1	PSO2
2	3	3	2	3	3	3	2	3	1	2	2	2	2

3. JUSTIFICATIONS OF MAPPING

- The project aims at solving a major challenge of analysing human emotion from various social media to make a variety of decisions by providing and developing a solution by understanding complex problems, thus justifying *PO2, PO3, PO4, PO6, PO7*.
- The project works on modern techniques such as Machine Learning and Artificial Intelligence, thus justifying *PO5*.
- Knowledge of statistics is required to apply ML and AI theorems, thus justifying *PO1*.
- The project can achieve its objective only when each and every member works as an individual completing his/her assigned work and also by acting like a team, consulting each other, thus justifying *PO9, PO10, PO11*.
- The project can be extended and modified accordingly with requirements and time, and also the completion of this project will contribute to the team members a life-long experience and learning, thus justifying *PO12*.

4. ABSTRACT

Large numbers of users share their opinions on Social networking sites, making it a valuable platform for tracking and analyzing public sentiment. Such tracking and analysis can provide critical information for decision making in various domains. So, it has attracted attention in both academia and industry. Presently, the use of public sentiment analysis has spread to services and making applications and developments came into existence in this area and now its main target is to make computers able to identify and create emotions like human beings. It is true that a picture is worth a thousand words. The use of images to express views, opinions, feelings, emotions and sentiments has increased tremendously on social platforms like Flickr, Instagram, Twitter, Tumblr, etc. The analysis of sentiments in user generated images is of increasing importance for developing several applications. A lot of research work has been done for sentiment analysis of textual data; there has been very limited work that focuses on analyzing sentiment of image data. In this paper, we are going to propose the development of an Image Sentiment Analysis Software using a multi modal approach. We found that sentiment of an image can be more accurately determined by studying both faces of persons and text contained in the image.For getting sentiment of face in the image convolutional neural network (CNN) has been used to build a framework for designing real-time CNNs. Our models are validated by creating a real-time vision system which accomplishes the tasks of face detection, gender classification and emotion classification simultaneously in one blended step using our proposed CNN architecture. We are also going to introduce the very recent real-time enabled guided backpropagation visualization technique. Guided back-propagation uncovers the dynamics of the weight changes and evaluates the learned features.

Keywords: sentiment, sentiment analysis, human behavior analysis, opinion mining sentiment analysis of textual data .

5. ACTIVITY CHART

	Week 1 - Week 2	Week 3 - Week 5	Week 6 - Week 9	Week 10 - Week 17	Week 18 - Week 23	Week 24 - Week 25
Planning	X					
Requirements Gathering		X				
Designing			X			
Building				X		
Testing & Debugging					X	
Deployment						X

6. INTRODUCTION

Being a major platform for communication and information exchange, internet provides a rich repository of people's verdict and sentiment about a vast spectrum of topics. Hence knowledge is embedded in multiple facets such as comments, tags, browsing and shared media. The analysis of such information either in the area of opinion mining, affective computing or sentiment analysis plays an important role in behaviour sciences, which aims to understand and predict human decision making and enables applications such as brand monitoring, stock market prediction, or political voting forecasts.

Sentiment analysis (also known as opinion mining or emotion AI) refers to the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information. Sentiment analysis is widely applied to voice of the customer materials such as reviews and survey responses, online and social media, and healthcare materials for applications that range from marketing to customer service to clinical medicine.

6.1 WHAT IS SENTIMENT ANALYSIS?

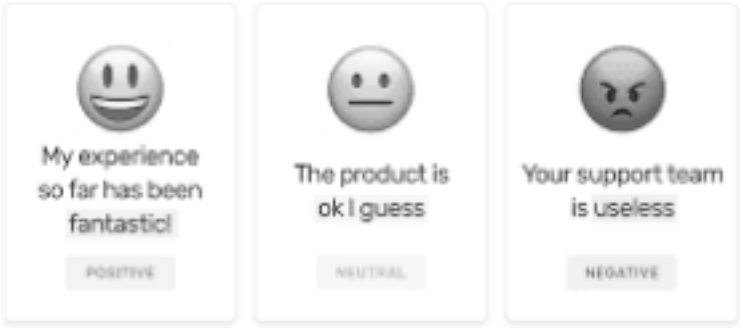


Fig. 1: Sentiment Analysis

Sentiment Analysis is a process of interpretation and classification of Labelling Emotional States (positive, negative and neutral) of data (image, voice speech, video or text) and Information Extraction task that aims to automate the process of understanding attitude, emotions and opinions. Sentiment analysis allows businesses to identify customer sentiment toward products, brands or services in online conversations and feedback. It is a very challenging task. Researchers from natural language processing and information retrieval have developed different approaches to solve this problem, achieving promising or satisfying results. Generally speaking, sentiment analysis aims to determine the attitude of a person present in an image with respect to some text or the overall tonality of the image. In recent years, the exponential increase in the Internet usage and exchange of public opinion is the driving force behind Sentiment Analysis today. The Web is a huge repository of structured and unstructured data. The analysis of this data to extract latent public opinion and sentiment is a challenging task.

Liu et al. (2009) defines a sentiment or opinion as a quintuple-

> *"<o_j , f_{jk} , so $_{ijkl}$, h_i , t_1 >, where o_j is a target object, f_{jk} is a feature of the object o_j , so $_{ijkl}$ is the sentiment value of the opinion of the opinion holder h_i on feature f_{jk} of object o_j at time t_l , so $_{ijkl}$ is +ve, -ve, or neutral, or a more granular rating, h_i is an opinion holder, t_l is the time when the opinion is expressed."*

6.2 TYPES OF SENTIMENT ANALYSIS

Sentiment analysis models focus on polarity (positive, negative, neutral) but also on feelings and emotions (angry, happy, sad, etc), and even on intentions (e.g. interested v. not interested).

Here are some of the most popular types of sentiment analysis:

a) *Fine-grained Sentiment Analysis :*

If polarity precision is important to your business, you might consider expanding your polarity categories to include:

- Very positive
- Positive
- Neutral
- Negative
- Very negative

This is usually referred to as fine-grained sentiment analysis, and could be used to interpret 5-star ratings in a review, for example:

- Very Positive = 5 stars
- Very Negative = 1 star

b) *Emotion detection :*

This type of sentiment analysis aims at detecting emotions, like happiness, frustration, anger, sadness, and so on. Many emotion detection systems use lexicons (i.e. lists of words and the emotions they convey) or complex machine learning algorithms.

One of the downsides of using lexicons is that people express emotions in different ways. Some words that typically express anger, like bad or kill (e.g. your product is so bad or your customer support is killing me) might also express happiness (e.g. this is bad ass or you are killing it).

c) *Aspect-based Sentiment Analysis :*

Usually, when analyzing sentiments of texts, let’s say product reviews, you’ll want to know which particular aspects or features people are mentioning in a positive, neutral, or negative way. That's where aspect-based sentiment analysis can help, for example in this text: "The battery life of this camera is too short", an aspect-based classifier would be able to determine that the sentence expresses a negative opinion about the feature battery life.

d) *Multilingual sentiment analysis :*

Multilingual sentiment analysis can be difficult. It involves a lot of preprocessing and resources. Most of these resources are available online (e.g. sentiment lexicons), while others need to be created (e.g. translated corpora or noise detection algorithms), but you'll need to know how to code to use them.

Alternatively, you could detect language in texts automatically with MonkeyLearn's language classifier, then train a custom sentiment analysis model to classify texts in the language of your choice. Sign up for free to try this model out.

6.3 WHY PERFORM SENTIMENT ANALYSIS

It's estimated that 80% of the world's data is unstructured, in other words it's unorganized. Huge volumes of text data (emails, support tickets, chats, social media conversations, surveys, articles, documents, etc), images are created every day but it's hard to analyze, understand, and sort through, not to mention time-consuming and expensive.

In today's environment where we're justifiably suffering from data overload (although this does not mean better or deeper insights), companies might have mountains of customer feedback collected; but for mere humans, it's still impossible to analyze it manually without any sort of error or bias.

Oftentimes, companies with the best intentions find themselves in an insights vacuum. You know you need insights to inform your decision making and you know that you're lacking them, but don't know how best to get them.

Sentiment analysis provides some answers into what the most important issues are, from the perspective of customers, at least. Because sentiment analysis can be automated, decisions can be made based on a significant amount of data rather than plain intuition that isn't always right.

<u>Impossible to analyze large amounts of data without error</u>
Imagine this scenario: you're the owner of a small delivery business and you receive about 20 responses to your email surveys every month. You could (and should), read these yourself and perform your own analysis by hand.

Now, imagine receiving 30,000 responses per month (as many large enterprises do). Whoa, that's more than a thousand responses that you would need to read and analyze each day.

Needless to say this is impossible as a part of a business owner's day job (not even the biggest shot of mocha latte will help you there!).

Then, there's the question of bias.

If you're having one of "those days" where your alarm doesn't go off, you rush out the door, spill coffee all over yourself to then be stuck in a traffic jam for 45 minutes, we can be sure that you're not in the best of moods, coming into the office late.

Fig. 2: Risk of wrong interpretation of messages

The risk of you interpreting messages and any form of communication more negatively, is rife.

You might also have your own, preconceived opinions about the topic at hand, and all of this, can influence how you interpret the text you need to analyze.

You'll also need to summarize the feedback into a few actionable insights, so that it is meaningful for your company to make use of.

Also the insights need to be translated into presentable form so that it is easy to grasp.

Sentiment analysis, however, helps businesses make sense of all this unstructured text by automatically tagging it.

6.4.1 Benefits of sentiment analysis include:

1. Sorting Data at Scale Can you imagine manually sorting through thousands of tweets, customer support conversations, or surveys? There's just too much data to process manually. Sentiment analysis helps businesses process huge amounts of data in an efficient and cost-effective way.

2. Real-Time Analysis Sentiment analysis can identify critical issues in real-time, for example is a PR crisis on social media escalating? Is an angry customer about to churn? Sentiment analysis models can help you immediately identify these kinds of situations and gauge brand sentiment, so you can take action right away.

3. Consistent criteria It's estimated that people only agree around 60-65% of the time when determining the sentiment of a particular data. Tagging text by sentiment is highly subjective, influenced by personal experiences, thoughts, and beliefs. By using a centralized sentiment analysis system, companies can apply the same criteria to all of their data, helping them improve accuracy and gain better insights.

6.4 APPLICATIONS OF SENTIMENT ANALYSIS

Word of mouth (WOM) is the process of conveying information from person to person and plays a major role in customer buying decisions. In commercial situations, WOM involves consumers sharing attitudes, opinions, or reactions about businesses, products, or services with other people. WOM communication functions based on social networking and trust. People rely on families, friends, and others in their social network. Research also indicates that people appear to trust seemingly disinterested opinions from people outside their immediate social network, such as online reviews. This is where Sentiment Analysis comes into play. Growing availability of opinion rich resources like online review sites, blogs, social networking sites have made this “decision-making process” easier for us. With the explosion of Web 2.0 platforms consumers have a soapbox of unprecedented reach and power by which they can share opinions. Major companies have realized these consumer voices affect shaping voices of other consumers.

Sentiment Analysis thus finds its use in Consumer Market for Product reviews, Marketing for knowing consumer attitudes and trends, Social Media for finding general opinion about recent hot topics in town, Movie to find whether a recently released movie is a hit.

Pang-Lee et al. (2002) broadly classifies the applications into the following categories.

A. Applications to Review-Related Websites
   Movie Reviews, Product Reviews etc.
B. Applications in Business and Government Intelligence
   Knowing Consumer attitudes and trends
C. Applications across Different Domains
   Knowing public opinions for political leaders or their notions about rules and regulations in place etc.

6.5 SEMANTIC ANALYSIS IN SENTIMENT ANALYSIS

Semantics plays an important role in the accurate analysis of the context of a sentiment expression. We analyze this role from two perspectives: the way semantics is encoded in sentiment resources, such as lexica, corpora, and ontologies, and the way it is used by automatic systems that perform sentiment analysis on social media data. Specifically, we review and discuss state-of-the-art methods and tools that rely on semantic models and resources, possibly enabling reasoning, so as to deal with some key challenges for sentiment analysis. Examples of such challenges are context dependency and finer-grained sentiment detection, which can translate into the assigning of sentiment values to aspects of objects or into the design and use of a wider spectrum of affective labels, or the use of current techniques in finer-grained semantic processing. In our dealing with semantics, lexical aspects must be combined with pragmatic and cognitive issues, and other dimensions involved in conveying sentiment, such as emotions.

6.6 ADVANTAGES OF USING SENTIMENT ANALYSIS

By using sentiment analysis, you gauge how customers feel about different areas of your business without having to read thousands of customer comments at once.

If you have thousands or even tens of thousands of survey responses per month, it is impossible for one person to read all of these responses and have an unbiased and consistent measure of customer sentiment. By using sentiment analysis and automating this process, you can easily drill down into different customer segments of your business and get a better understanding of sentiment in these segments.

6.7 DISADVANTAGES OF SENTIMENT ANALYSIS

While sentiment analysis is useful, we do not believe it is a complete replacement for reading survey responses, as there are often useful nuances in the comments themselves. Where sentiment analysis can help you further is by identifying which of these comments you should read, for example allowing you to focus on the most negative comments.

6.8 CHALLENGES FOR SENTIMENT ANALYSIS

Sentiment Analysis approaches aim to extract positive and negative sentiment bearing words from a text and image and classify these as positive, negative or else objective if it cannot find any sentiment bearing words and special facial expressions. In this respect, it can be thought of as a text categorization task. In text classification there are many classes corresponding to different topics whereas in Sentiment Analysis we have only 3 broad classes. Thus it seems Sentiment Analysis is easier than text classification which is not quite the case. The general challenges can be summarized as:

6.5.1 Implicit Sentiment and Sarcasm

Irony and sarcasm are sophisticated forms of speech in which authors write the opposite of what they mean. They have been studied in linguistics, psychology, and cognitive science. While irony is often used to emphasize occurrences that deviate from the expected, sarcasm is commonly used to convey implicit criticism. However, the detection of irony and sarcasm is a complex task, even for humans. The difficulty in recognizing irony and sarcasm causes misunderstanding in everyday communication and poses problems to many natural language processing tasks such as sentiment analysis. This is particularly challenging when one is dealing with social media messages, where the language is concise, informal, and ill-formed.
A sentence may have an implicit sentiment even without the presence of any sentiment bearing words. Consider the following examples.

How can anyone sit through this movie?

One should question the stability of mind of the writer who wrote this book.

Both the above sentences do not explicitly carry any negative sentiment bearing words although both are negative sentences. Thus identifying semantics is more important in SA than *syntax detection*.

6.5.2 World Knowledge

Often world knowledge needs to be incorporated in the system for detecting sentiments. Consider the following examples:

He is a Frankenstein.

Just finished Doctor Zhivago for the first time and all I can say is Russia sucks.

The first sentence depicts a negative sentiment, whereas the second one depicts a positive sentiment. But one has to know about Frankenstein and Doctor Zhivago to find out the sentiment.

6.5.3 Negation Handling

Negation is a challenging task in SA. Negation can be expressed in subtle ways even without the explicit use of any negative words. A method often followed in handling negation explicitly in sentences like *"I do not like the movie"*, is to reverse the polarity of all the words appearing after the negation operator (like not). But this does not work for *"I do not like the acting but I like the direction"*. So we need to consider the scope of negation as well, which extends only till but here. So the thing that can be done is to change the polarity of all words appearing after a negation word till another negation word appears. But still there can be problems. For example, in the sentence *"Not only did I like the acting, but also the direction"*, the polarity is not reversed after "not" due to the presence of "only". So this type of combination of "not" with other words like "only" has to be kept in mind while designing the algorithm.

6.5.4 Diversity And Informality

In the context of social media, there are several additional unique challenges, i.e.

- There are huge amounts of data available.
- Messages on social networks are by nature informal and short.
- People use not only textual messages, but also images and videos to express themselves.

Since images are the easiest medium through which people nowadays express their emotions on social media, sentiment analysis of such large scale visual content can extract user sentiments towards particular events or topics. A good sketch is better than a long speech(*Napoleon Bonaparte*). People with different backgrounds can easily understand the main content of an image or video. Apart from the large amount of easily available visual content, today's computational infrastructure is also much cheaper and more powerful to make the analysis of computationally intensive visual content analysis feasible. In this era of big data, it has been shown that the integration of visual content can provide us more reliable or complementary online social signals.

6.3.5 Predict or identify the sentiments of unlabelled images

The major challenge is to predict or identify the sentiments of unlabelled images. To overcome this challenge deep learning techniques are used for sentiment analysis, as deep learning models have the capability for effectively learning the image behavior or polarity. Image recognition, image prediction, image sentiment analysis, and image classification are some of the fields where Neural Network (NN) has performed well implying significant performance of deep learning in image sentiment analysis.

6.3.6 Problem of labelling images with emotion

The problem of labeling images with the emotion they depict is very subjective and can differ from person to person. Also, due to cultural or geographical differences some images might invoke a different emotion in different people - like in India people light candles to celebrate a festival called "Diwali", however in western countries candles are lit, most of the time,to mark an occasion of mourning.

6.3.7 *Multipolarity*

Sometimes, a given sentence or document—or whatever unit of text we would like to analyze—will exhibit multipolarity. In these cases, having only the total result of the analysis can be misleading, very much like how an average can sometimes hide valuable information about all the numbers that went into it.

Picture when authors talk about different people, products, or companies (or aspects of them) in an article or review. It's common that within a piece of text, some subjects will be criticized and some praised.

Here, the total sentiment polarity will be missing key information. This is why it's necessary to extract all the entities or aspects in the sentence with assigned sentiment labels and only calculate the total polarity if needed.

Let's consider an example which consists of multiple polarities:

The audio quality of my new laptop is so cool but the display colors are not too good.

Some sentiment analysis models will assign a negative or a neutral polarity to this sentence. To deal with such situations, a sentiment analysis model must assign a polarity to each aspect in the sentence; here, “audio” is an aspect assigned a positive polarity and “display” is a separate aspect with a negative polarity

7. LITERATURE REVIEW

It has been a widely used area over the years and still it leaves a lot to be researched. Fried, Surdeanu, Kobourov, Hingle, Bell[1] investigated the predictive power behind the language of food on social media. They collected a corpus of over three million food-related posts from Twitter and demonstrate that many latent population characteristics can be directly predicted from this data: overweight rate, diabetes rate, political leaning, and home geographical location of authors. For all tasks, their language-based models significantly outperform the majority class baselines. Logunov, Panchenko[2] generated Twitter sentiment indices by analysing a stream of Twitter messages and categorising messages in terms of emoticons, pictorial representations of facial expressions in messages. Based on emoticons they generated daily indices. Then they explored the time-series properties of these indices by focusing on seasonal and cyclical patterns, persistence and conditional heteroscedasticity. Zhang, Parikh, Singh, Sundaresan[3] chose a particular global ecommerce platform (eBay) and a particular global social media platform (Twitter). They quantified the characteristics of the two individual trends as well as the correlations between it. They provided evidence that about 5% of general eBay query streams show strong positive correlations with the corresponding Twitter mention streams, while the percentage jumps to around 25% for trending eBay query streams.Gonçalves, Araújo, Benevenuto, Cha[4] There are multiple methods for measuring sentiments, including lexical-based approaches and supervised machine learning methods. Despite the wide use and popularity of some methods, it is unclear which method is better for identifying the polarity (i.e., positive or negative) of a message as the current literature does not provide a method of comparison among existing methods. Such a comparison is crucial for understanding the potential limitations, advantages, and disadvantages of popular methods in analyzing the content of OSNs messages[6].

8.1 RELATED WORK

Commonly used CNNs for feature extraction include a set of fully connected layers at the end. Fully connected layers tend to contain most of the parameters in a CNN. Specifically, VGG16 [7] contains approximately 90% of all its parameters in their last fully connected layers. Recent architectures such as Inception V3 [8], reduced the amount of parameters in their last layers by including a Global Average Pooling operation. Global Average Pooling reduces each feature map into a scalar value by taking the average over all elements in the feature map. The average operation forces the network to extract global features from the input image. Modern CNN architectures such as Xception [9] leverage from the combination of two of the most successful experimental assumptions in CNNs: the use of residual modules [10] and depth-wise separable convolutions [11]. Depth-wise separable convolutions reduce further the amount of parameters by separating the processes of feature extraction and combination within a convolutional layer.

Furthermore, the state-of-the-art model for the FER2-2013 dataset is based on CNN trained with square hinged loss[12].

This model achieved an accuracy of 71% [1] using approximately 5 million parameters. In this architecture 98% of all parameters are located in the last fully connected layers.

The second-best methods presented in [1] achieved an accuracy of 66% using an ensemble of CNNs.

8.2 GAPS IN EXISTING WORK

1. Multi modal approach towards image sentiment analysis using text and face.
2. Analyzing emotions of multiple faces present in a photo frame, i.e. detection of multiple faces and analyzing the sentiment of each face.
3. Summing up the sentiments of multiple faces present in a photo frame to get the overall sentiment of all people present within the photo frame, i.e. generalization of overall sentiment of the entire group consisting of multiple faces.
4. Analyzing the sentiment of the text present in an image
5. Sentiment analysis is done on images that are based on emotions, i.e. we try to find the facial expression of a person and based upon this we take a decision but we do not consider the external factors i.e. surroundings of a person in our decision making.

We argue that the careful implementation of modern CNN architectures, the use of the current regularization methods and the visualization of previously hidden features are necessary in order to reduce the gap between slow performances and real-time architectures.

9. THEORY

We have used a multi-modal approach for sentiment analysis of an image. So there are two parts in analyzing sentiments -

- First one is getting sentiment from the faces of the people in the image. In this part, faces are detected and then convolutional neural networks (CNN) are used to find the sentiment and gender of faces present in the image. For this CNNs were trained using FER-2013 emotion dataset and gender classification was done using IMDB gender dataset. [5].
- Second one is getting sentiment from text present in images. For this Optical Character Recognition (OCR) technique was used with the help of pytesseract tool in Python for extracting the text from image. Then to analyse the sentiment of the extracted text VADER(Valence Aware Dictionary and Sentiment Reasoner) tool is applied on it that uses Natural Language Processing (NLP) to get the sentiment from the extracted text.

Fig. 3: Samples of the FER-2013 emotion dataset [1]

Fig 4: Samples of IMDB gender dataset [2]

After presenting the details of the training procedure setup we proceed to evaluate on standard benchmark sets. We report accuracy of 95.8% in the IMDB gender dataset, 66.2% in the FER-2013 emotion dataset and 96.35% in the face sentiment detection.

10. PROPOSED SYSTEM / SOFTWARE

Our proposed architecture for face sentiment analysis is a mini-Xception model [5] with standard fully-convolutional neural network composed of 9 convolution layers, ReLUs [3], Batch Normalization [4] and Global Average Pooling. This model contains approximately 600,000 parameters. It was trained on the IMDB gender dataset, which contains 460,723 RGB images where each image belongs to the class “woman” or “man”, and it achieved an accuracy of 96% in this dataset. We also validated this model in the FER-2013 emotion dataset. This dataset contains 35,887 grayscale images where each image belongs to one of the following classes {“angry”, “disgust”, “fear”, “happy”, “sad”, “surprise”, “neutral”}. The model achieved an accuracy of 66% in this dataset.

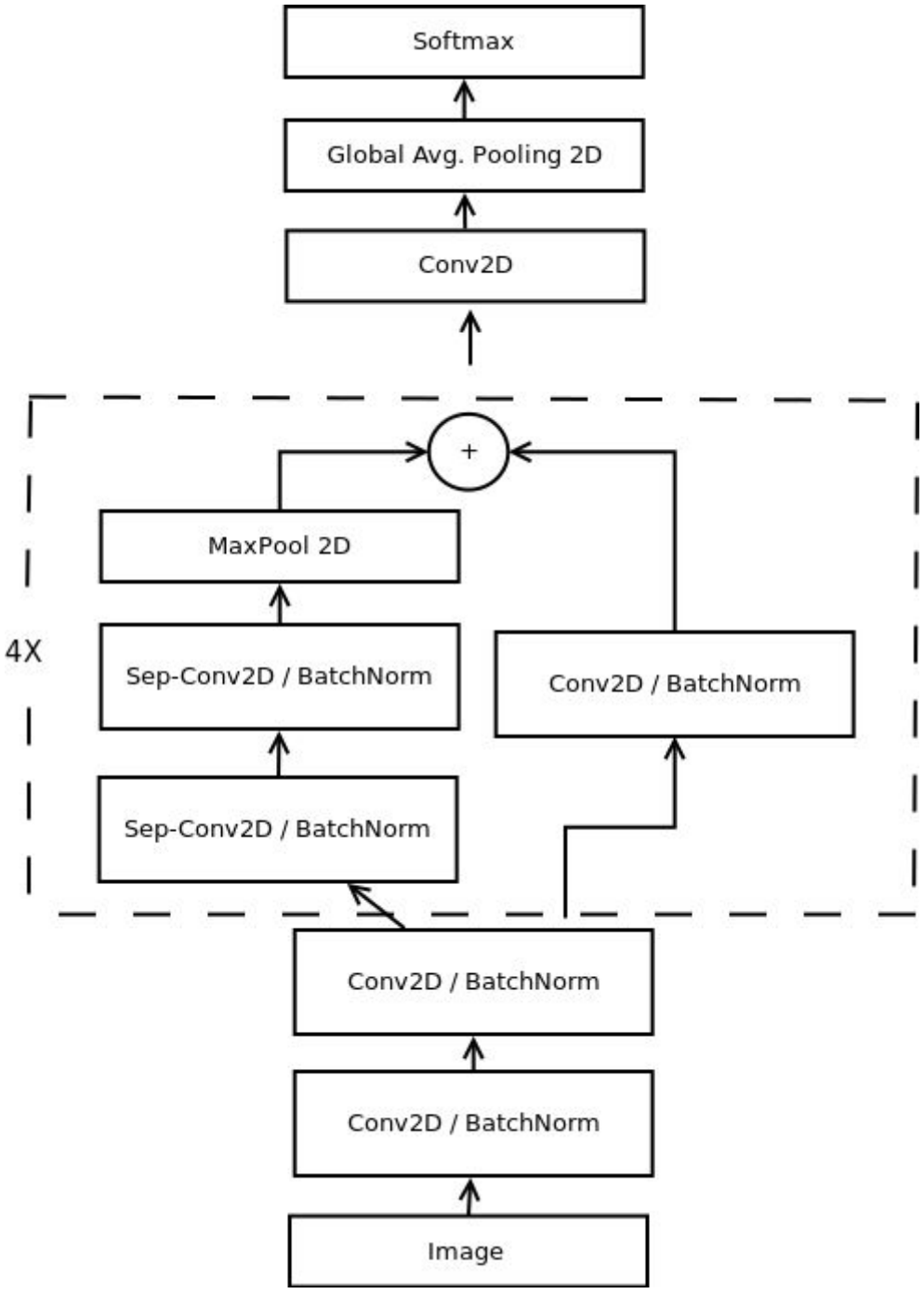


Fig. 5: Model for real time classification of faces [5]

VADER (Valence Aware Dictionary and Sentiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. VADER uses a combination of A sentiment lexicon is a list of lexical features (e.g., words) which are generally labeled according to their semantic orientation as either positive or negative.

VADER has been found to be quite successful when dealing with social media texts, NY Times editorials, movie reviews, and product reviews. This is because VADER

not only tells about the Positivity and Negativity score but also tells us about how positive or negative a sentiment is.
It is fully open-sourced under the MIT License. The developers of VADER have used Amazon's Mechanical Turk to get most of their ratings, You can find complete details on their Github Page.

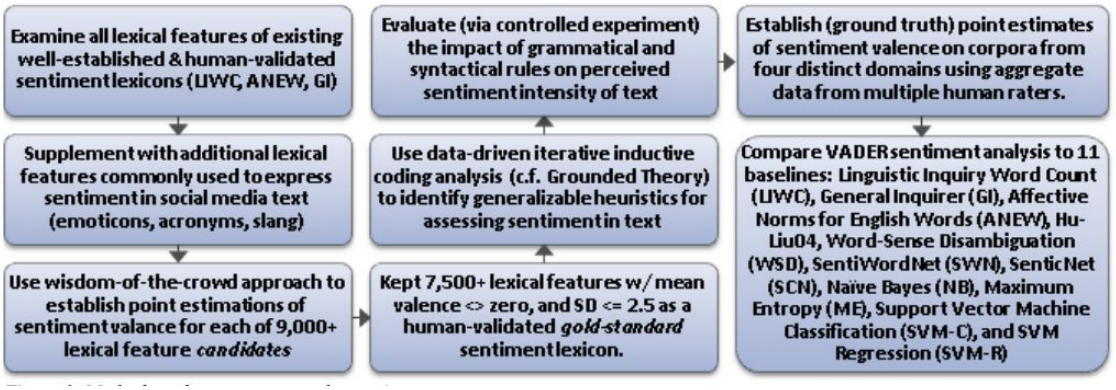


Fig. 5:

10.1 ADVANTAGES OF USING VADER

VADER has a lot of advantages over traditional methods of Sentiment Analysis, including:

- ➢ It works exceedingly well on social media type text, yet readily generalizes to multiple domains.
- ➢ It doesn't require any training data but is constructed from a generalizable, valence-based, human-curated gold standard sentiment lexicon.
- ➢ It is fast enough to be used online with streaming data.
- ➢ It does not severely suffer from a speed-performance tradeoff.

Let us check how VADER performs on a given review

sentiment_analyzer_scores("The phone is super cool.")

The phone is super cool---------- {'neg': 0.0, 'neu': 0.326, 'pos': 0.674, 'compound': 0.7351}

Putting in a Tabular form:

Sentiment Metric	Score
Positive	0.674
Neutral	0.326
Negative	0.0
Compound	0.735

The Positive, Negative and Neutral scores represent the proportion of text that falls in these categories. This means our sentence was rated as 67% Positive, 33% Neutral and 0% Negative. Hence all these should add up to 1. The Compound score is a metric that calculates the sum of all the lexicon ratings which have been normalized between -1(most extreme negative) and +1 (most extreme positive). In the case above, lexicon ratings for and supercool are 2.9 and 1.3 respectively. The compound score turns out to be 0.75 , denoting a very high positive sentiment.

1. **positive sentiment:** `compound` score >= 0.05
2. **neutral sentiment:** (`compound` score > -0.05) and (`compound` score < 0.05)
3. **negative sentiment:** `compound` score <= -0.05

10.2 Python-tesseract

Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and "read" the text embedded in images. Python-tesseract is a wrapper for google's Tesseract-OCR (http://code.google.com/p/tesseract-ocr/). It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Python Imaging Library, including jpeg, png, gif, bmp, tiff, and others, whereas tesseract-ocr by default only supports tiff and bmp. Additionally, if used as a script, Python-tesseract will print the recognized text instead of writing it to a file. Support for confidence estimates and bounding
box data is planned for future releases.

11. FEASIBILITY STUDY

11.1 Technical feasibility

In sentiment analysis of text present in an image first pytesseract tool is needed and for analyzing the sentiment of the same VADER(Valence Aware Dictionary for Sentiment Reasoning) tool is needed. In sentiment analysis of faces present in an image, we have used python libraries like keras, openCV, numpy and utils.

11.2 Time feasibility

The entire project, i.e, solving all gaps, will require 2-3 months for completion.

11.3 Economic feasibility

The project does not require any additional hardware to perform (except its' performing platform), so there is no any added economic feasibility.

12. ALGORITHM / CODE

Required codes are described below.

12.1 Code for sentiment analysis on text extracted from image using OCR and VADER tool

```
# Import modules
from PIL import Image
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
import pytesseract
# Include tesseract executable in your path
pytesseract.pytesseract.tesseract_cmd = r"C:\Program
Files\Tesseract-OCR\tesseract.exe"
# Create an image object of PIL library
image = Image.open('H:\proj\ecc.jpg')
# pass image into pytesseract module
# pytesseract is trained in many languages
image_to_text = pytesseract.image_to_string(image, lang='eng')
# Print the text
print(image_to_text)

analyser = SentimentIntensityAnalyzer()
def sentiment_analyzer_scores(sentence):
    score = analyser.polarity_scores(sentence)
    print("{:-<40} {}".format(sentence, str(score)))

sentiment_analyzer_scores(image_to_text) #image_to_text contains extracted text
```

12.2 Code for sentiment analysis on faces

Sentiment can be analysed by studying faces present both in image and live video. Required codes are mentioned below:

12.2.1 Code for sentiment analysis on faces present in image

```
import sys
import cv2
from keras.models import load_model
import numpy as np

from utils.datasets import get_labels
from utils.inference import detect_faces
from utils.inference import draw_text
from utils.inference import draw_bounding_box
from utils.inference import apply_offsets
```

```
from utils.inference import load_detection_model
from utils.inference import load_image
from utils.preprocessor import preprocess_input

# parameters for loading data and images
image_path = sys.argv[1]
detection_model_path =
'../trained_models/detection_models/haarcascade_frontalface_default.xml'
emotion_model_path =
'../trained_models/emotion_models/fer2013_mini_XCEPTION.102-0.66.hdf5'
gender_model_path = '../trained_models/gender_models/simple_CNN.81-0.96.hdf5'
emotion_labels = get_labels('fer2013')
gender_labels = get_labels('imdb')
font = cv2.FONT_HERSHEY_SIMPLEX

# hyper-parameters for bounding boxes shape
gender_offsets = (30, 60)
gender_offsets = (10, 10)
emotion_offsets = (20, 40)
emotion_offsets = (0, 0)

# loading models
face_detection = load_detection_model(detection_model_path)
emotion_classifier = load_model(emotion_model_path, compile=False)
gender_classifier = load_model(gender_model_path, compile=False)

# getting input model shapes for inference
emotion_target_size = emotion_classifier.input_shape[1:3]
gender_target_size = gender_classifier.input_shape[1:3]

# loading images
rgb_image = load_image(image_path, grayscale=False)
gray_image = load_image(image_path, grayscale=True)
gray_image = np.squeeze(gray_image)
gray_image = gray_image.astype('uint8')

#extracting each face and storing in faces
faces = detect_faces(face_detection, gray_image)
for face_coordinates in faces:
        x1, x2, y1, y2 = apply_offsets(face_coordinates, gender_offsets)
        rgb_face = rgb_image[y1:y2, x1:x2]

        x1, x2, y1, y2 = apply_offsets(face_coordinates, emotion_offsets)
        gray_face = gray_image[y1:y2, x1:x2]

        try:
        rgb_face = cv2.resize(rgb_face, (gender_target_size))
        gray_face = cv2.resize(gray_face, (emotion_target_size))
```

```
        except:
        continue
        #gender classification
        rgb_face = preprocess_input(rgb_face, False)
        rgb_face = np.expand_dims(rgb_face, 0)
        gender_prediction = gender_classifier.predict(rgb_face)
        gender_label_arg = np.argmax(gender_prediction)
        gender_text = gender_labels[gender_label_arg]
        #sentiment analyzer
        gray_face = preprocess_input(gray_face, True)
        gray_face = np.expand_dims(gray_face, 0)
        gray_face = np.expand_dims(gray_face, -1)
        emotion_label_arg = np.argmax(emotion_classifier.predict(gray_face))
        emotion_text = emotion_labels[emotion_label_arg]

        if gender_text == gender_labels[0]:
        color = (0, 0, 255)
        else:
        color = (255, 0, 0)

        draw_bounding_box(face_coordinates, rgb_image, color)
        draw_text(face_coordinates, rgb_image, gender_text, color, 0, -20, 1, 2)
        draw_text(face_coordinates, rgb_image, emotion_text, color, 0, -50, 1, 2)

bgr_image = cv2.cvtColor(rgb_image, cv2.COLOR_RGB2BGR)
cv2.imwrite('../images/predicted_test_image.png', bgr_image)
```

12.2.2 Code for sentiment analysis on faces present in live video:

```
from statistics import mode

import cv2
from keras.models import load_model
import numpy as np

from utils.datasets import get_labels
from utils.inference import detect_faces
from utils.inference import draw_text
from utils.inference import draw_bounding_box
from utils.inference import apply_offsets
from utils.inference import load_detection_model
from utils.preprocessor import preprocess_input

# parameters for loading data and images
detection_model_path =
'../trained_models/detection_models/haarcascade_frontalface_default.xml'
emotion_model_path =
'../trained_models/emotion_models/fer2013_mini_XCEPTION.102-0.66.hdf5'
emotion_labels = get_labels('fer2013')

# hyper-parameters for bounding boxes shape
frame_window = 10
emotion_offsets = (20, 40)

# loading models
face_detection = load_detection_model(detection_model_path)
emotion_classifier = load_model(emotion_model_path, compile=False)

# getting input model shapes for inference
emotion_target_size = emotion_classifier.input_shape[1:3]

# starting lists for calculating modes
emotion_window = []

# starting video streaming
cv2.namedWindow('window_frame')
video_capture = cv2.VideoCapture(0)
while True:
    bgr_image = video_capture.read()[1]
    gray_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2GRAY)
    rgb_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2RGB)
    #extracting each face and storing in faces
    faces = detect_faces(face_detection, gray_image)

    for face_coordinates in faces:
```

```
        x1, x2, y1, y2 = apply_offsets(face_coordinates, emotion_offsets)
        gray_face = gray_image[y1:y2, x1:x2]
        try:
            gray_face = cv2.resize(gray_face, (emotion_target_size))
        except:
            continue

        gray_face = preprocess_input(gray_face, True)
        gray_face = np.expand_dims(gray_face, 0)
        gray_face = np.expand_dims(gray_face, -1)
        emotion_prediction = emotion_classifier.predict(gray_face)
        emotion_probability = np.max(emotion_prediction)
        emotion_label_arg = np.argmax(emotion_prediction)
        emotion_text = emotion_labels[emotion_label_arg]
        emotion_window.append(emotion_text)

        if len(emotion_window) > frame_window:
            emotion_window.pop(0)
        try:
            emotion_mode = mode(emotion_window)
        except:
            continue

        if emotion_text == 'angry':
            color = emotion_probability * np.asarray((255, 0, 0))
        elif emotion_text == 'sad':
            color = emotion_probability * np.asarray((0, 0, 255))
        elif emotion_text == 'happy':
            color = emotion_probability * np.asarray((255, 255, 0))
        elif emotion_text == 'surprise':
            color = emotion_probability * np.asarray((0, 255, 255))
        else:
            color = emotion_probability * np.asarray((0, 255, 0))

        color = color.astype(int)
        color = color.tolist()

        draw_bounding_box(face_coordinates, rgb_image, color)
        draw_text(face_coordinates, rgb_image, emotion_mode,
                  color, 0, -45, 1, 1)

    bgr_image = cv2.cvtColor(rgb_image, cv2.COLOR_RGB2BGR)
    cv2.imshow('window_frame', bgr_image)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
video_capture.release()
cv2.destroyAllWindows()
```

12.3 Code for CNN model:

```
from keras.layers import Activation, Convolution2D, Dropout, Conv2D
from keras.layers import AveragePooling2D, BatchNormalization
from keras.layers import GlobalAveragePooling2D
from keras.models import Sequential
from keras.layers import Flatten
from keras.models import Model
from keras.layers import Input
from keras.layers import MaxPooling2D
from keras.layers import SeparableConv2D
from keras import layers
from keras.regularizers import l2


def simple_CNN(input_shape, num_classes):

    model = Sequential()
    model.add(Convolution2D(filters=16, kernel_size=(7, 7), padding='same',
                            name='image_array', input_shape=input_shape))
    model.add(BatchNormalization())
    model.add(Convolution2D(filters=16, kernel_size=(7, 7), padding='same'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(AveragePooling2D(pool_size=(2, 2), padding='same'))
    model.add(Dropout(.5))

    model.add(Convolution2D(filters=32, kernel_size=(5, 5), padding='same'))
    model.add(BatchNormalization())
    model.add(Convolution2D(filters=32, kernel_size=(5, 5), padding='same'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(AveragePooling2D(pool_size=(2, 2), padding='same'))
    model.add(Dropout(.5))

    model.add(Convolution2D(filters=64, kernel_size=(3, 3), padding='same'))
    model.add(BatchNormalization())
    model.add(Convolution2D(filters=64, kernel_size=(3, 3), padding='same'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(AveragePooling2D(pool_size=(2, 2), padding='same'))
    model.add(Dropout(.5))

    model.add(Convolution2D(filters=128, kernel_size=(3, 3), padding='same'))
    model.add(BatchNormalization())
    model.add(Convolution2D(filters=128, kernel_size=(3, 3), padding='same'))
    model.add(BatchNormalization())
```

```
    model.add(Activation('relu'))
    model.add(AveragePooling2D(pool_size=(2, 2), padding='same'))
    model.add(Dropout(.5))

    model.add(Convolution2D(filters=256, kernel_size=(3, 3), padding='same'))
    model.add(BatchNormalization())
    model.add(Convolution2D(
        filters=num_classes, kernel_size=(3, 3), padding='same'))
    model.add(GlobalAveragePooling2D())
    model.add(Activation('softmax', name='predictions'))
    return model


def simpler_CNN(input_shape, num_classes):

    model = Sequential()
    model.add(Convolution2D(filters=16, kernel_size=(5, 5), padding='same',
                            name='image_array', input_shape=input_shape))
    model.add(BatchNormalization())
    model.add(Convolution2D(filters=16, kernel_size=(5, 5),
                            strides=(2, 2), padding='same'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(Dropout(.25))

    model.add(Convolution2D(filters=32, kernel_size=(5, 5), padding='same'))
    model.add(BatchNormalization())
    model.add(Convolution2D(filters=32, kernel_size=(5, 5),
                            strides=(2, 2), padding='same'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(Dropout(.25))

    model.add(Convolution2D(filters=64, kernel_size=(3, 3), padding='same'))
    model.add(BatchNormalization())
    model.add(Convolution2D(filters=64, kernel_size=(3, 3),
                            strides=(2, 2), padding='same'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(Dropout(.25))

    model.add(Convolution2D(filters=64, kernel_size=(1, 1), padding='same'))
    model.add(BatchNormalization())
    model.add(Convolution2D(filters=128, kernel_size=(3, 3),
                            strides=(2, 2), padding='same'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(Dropout(.25))
```

```
    model.add(Convolution2D(filters=256, kernel_size=(1, 1), padding='same'))
    model.add(BatchNormalization())
    model.add(Convolution2D(filters=128, kernel_size=(3, 3),
                            strides=(2, 2), padding='same'))

    model.add(Convolution2D(filters=256, kernel_size=(1, 1), padding='same'))
    model.add(BatchNormalization())
    model.add(Convolution2D(filters=num_classes, kernel_size=(3, 3),
                            strides=(2, 2), padding='same'))

    model.add(Flatten())
    # model.add(GlobalAveragePooling2D())
    model.add(Activation('softmax', name='predictions'))
    return model


def tiny_XCEPTION(input_shape, num_classes, l2_regularization=0.01):
    regularization = l2(l2_regularization)

    # base
    img_input = Input(input_shape)
    x = Conv2D(5, (3, 3), strides=(1, 1), kernel_regularizer=regularization,
               use_bias=False)(img_input)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Conv2D(5, (3, 3), strides=(1, 1), kernel_regularizer=regularization,
               use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    # module 1
    residual = Conv2D(8, (1, 1), strides=(2, 2),
                      padding='same', use_bias=False)(x)
    residual = BatchNormalization()(residual)

    x = SeparableConv2D(8, (3, 3), padding='same',
                        kernel_regularizer=regularization,
                        use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = SeparableConv2D(8, (3, 3), padding='same',
                        kernel_regularizer=regularization,
                        use_bias=False)(x)
    x = BatchNormalization()(x)

    x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
    x = layers.add([x, residual])
```

```
# module 2
residual = Conv2D(16, (1, 1), strides=(2, 2),
                  padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)

x = SeparableConv2D(16, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(16, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)

x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])

# module 3
residual = Conv2D(32, (1, 1), strides=(2, 2),
                  padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)

x = SeparableConv2D(32, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(32, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)

x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])

# module 4
residual = Conv2D(64, (1, 1), strides=(2, 2),
                  padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)

x = SeparableConv2D(64, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(64, (3, 3), padding='same',
```

```
                kernel_regularizer=regularization,
                use_bias=False)(x)
    x = BatchNormalization()(x)

    x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
    x = layers.add([x, residual])

    x = Conv2D(num_classes, (3, 3),
            # kernel_regularizer=regularization,
            padding='same')(x)
    x = GlobalAveragePooling2D()(x)
    output = Activation('softmax', name='predictions')(x)

    model = Model(img_input, output)
    return model


def mini_XCEPTION(input_shape, num_classes, l2_regularization=0.01):
    regularization = l2(l2_regularization)

    # base
    img_input = Input(input_shape)
    x = Conv2D(8, (3, 3), strides=(1, 1), kernel_regularizer=regularization,
            use_bias=False)(img_input)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Conv2D(8, (3, 3), strides=(1, 1), kernel_regularizer=regularization,
            use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    # module 1
    residual = Conv2D(16, (1, 1), strides=(2, 2),
                      padding='same', use_bias=False)(x)
    residual = BatchNormalization()(residual)

    x = SeparableConv2D(16, (3, 3), padding='same',
                kernel_regularizer=regularization,
                use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = SeparableConv2D(16, (3, 3), padding='same',
                kernel_regularizer=regularization,
                use_bias=False)(x)
    x = BatchNormalization()(x)

    x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
    x = layers.add([x, residual])
```

```
# module 2
residual = Conv2D(32, (1, 1), strides=(2, 2),
                  padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)

x = SeparableConv2D(32, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(32, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)

x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])

# module 3
residual = Conv2D(64, (1, 1), strides=(2, 2),
                  padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)

x = SeparableConv2D(64, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(64, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)

x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])

# module 4
residual = Conv2D(128, (1, 1), strides=(2, 2),
                  padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)

x = SeparableConv2D(128, (3, 3), padding='same',
                    kernel_regularizer=regularization,
                    use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(128, (3, 3), padding='same',
```

```
                kernel_regularizer=regularization,
                use_bias=False)(x)
    x = BatchNormalization()(x)

    x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
    x = layers.add([x, residual])

    x = Conv2D(num_classes, (3, 3),
           # kernel_regularizer=regularization,
           padding='same')(x)
    x = GlobalAveragePooling2D()(x)
    output = Activation('softmax', name='predictions')(x)

    model = Model(img_input, output)
    return model


def big_XCEPTION(input_shape, num_classes):
    img_input = Input(input_shape)
    x = Conv2D(32, (3, 3), strides=(2, 2), use_bias=False)(img_input)
    x = BatchNormalization(name='block1_conv1_bn')(x)
    x = Activation('relu', name='block1_conv1_act')(x)
    x = Conv2D(64, (3, 3), use_bias=False)(x)
    x = BatchNormalization(name='block1_conv2_bn')(x)
    x = Activation('relu', name='block1_conv2_act')(x)

    residual = Conv2D(128, (1, 1), strides=(2, 2),
                      padding='same', use_bias=False)(x)
    residual = BatchNormalization()(residual)

    x = SeparableConv2D(128, (3, 3), padding='same', use_bias=False)(x)
    x = BatchNormalization(name='block2_sepconv1_bn')(x)
    x = Activation('relu', name='block2_sepconv2_act')(x)
    x = SeparableConv2D(128, (3, 3), padding='same', use_bias=False)(x)
    x = BatchNormalization(name='block2_sepconv2_bn')(x)

    x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
    x = layers.add([x, residual])

    residual = Conv2D(256, (1, 1), strides=(2, 2),
                      padding='same', use_bias=False)(x)
    residual = BatchNormalization()(residual)

    x = Activation('relu', name='block3_sepconv1_act')(x)
    x = SeparableConv2D(256, (3, 3), padding='same', use_bias=False)(x)
    x = BatchNormalization(name='block3_sepconv1_bn')(x)
    x = Activation('relu', name='block3_sepconv2_act')(x)
    x = SeparableConv2D(256, (3, 3), padding='same', use_bias=False)(x)
```

```
    x = BatchNormalization(name='block3_sepconv2_bn')(x)

    x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
    x = layers.add([x, residual])
    x = Conv2D(num_classes, (3, 3),
               # kernel_regularizer=regularization,
               padding='same')(x)
    x = GlobalAveragePooling2D()(x)
    output = Activation('softmax', name='predictions')(x)

    model = Model(img_input, output)
    return model


if __name__ == "__main__":
    input_shape = (64, 64, 1)
    num_classes = 7
    # model = tiny_XCEPTION(input_shape, num_classes)
    # model.summary()
    # model = mini_XCEPTION(input_shape, num_classes)
    # model.summary()
    # model = big_XCEPTION(input_shape, num_classes)
    # model.summary()
    model = simple_CNN((48, 48, 1), num_classes)
    model.summary()
```

12.4 Code for training classifier models:

12.4.1 Code for training emotion classifier:

```
from keras.callbacks import CSVLogger, ModelCheckpoint, EarlyStopping
from keras.callbacks import ReduceLROnPlateau
from keras.preprocessing.image import ImageDataGenerator

from models.cnn import mini_XCEPTION
from utils.datasets import DataManager
from utils.datasets import split_data
from utils.preprocessor import preprocess_input

# parameters
batch_size = 32
num_epochs = 10000
input_shape = (64, 64, 1)
validation_split = .2
verbose = 1
num_classes = 7
patience = 50
base_path = '../trained_models/emotion_models/'

# data generator
data_generator = ImageDataGenerator(
                        featurewise_center=False,
                        featurewise_std_normalization=False,
                        rotation_range=10,
                        width_shift_range=0.1,
                        height_shift_range=0.1,
                        zoom_range=.1,
                        horizontal_flip=True)

# model parameters/compilation
model = mini_XCEPTION(input_shape, num_classes)
model.compile(optimizer='adam', loss='categorical_crossentropy',
              metrics=['accuracy'])
model.summary()


datasets = ['fer2013']
for dataset_name in datasets:
    print('Training dataset:', dataset_name)

    # callbacks
    log_file_path = base_path + dataset_name + '_emotion_training.log'
    csv_logger = CSVLogger(log_file_path, append=False)
```

```
early_stop = EarlyStopping('val_loss', patience=patience)
reduce_lr = ReduceLROnPlateau('val_loss', factor=0.1,
                              patience=int(patience/4), verbose=1)
trained_models_path = base_path + dataset_name + '_mini_XCEPTION'
model_names = trained_models_path + '.{epoch:02d}-{val_acc:.2f}.hdf5'
model_checkpoint = ModelCheckpoint(model_names, 'val_loss', verbose=1,
                                   save_best_only=True)
callbacks = [model_checkpoint, csv_logger, early_stop, reduce_lr]

# loading dataset
data_loader = DataManager(dataset_name, image_size=input_shape[:2])
faces, emotions = data_loader.get_data()
faces = preprocess_input(faces)
num_samples, num_classes = emotions.shape
train_data, val_data = split_data(faces, emotions, validation_split)
train_faces, train_emotions = train_data
model.fit_generator(data_generator.flow(train_faces, train_emotions,
                                        batch_size),
                    steps_per_epoch=len(train_faces) / batch_size,
                    epochs=num_epochs, verbose=1, callbacks=callbacks,
                    validation_data=val_data)
```

12.4.2 Code for training gender classifier:

```
from keras.callbacks import CSVLogger, ModelCheckpoint, EarlyStopping
from keras.callbacks import ReduceLROnPlateau
from utils.datasets import DataManager
from models.cnn import mini_XCEPTION
from utils.data_augmentation import ImageGenerator
from utils.datasets import split_imdb_data

# parameters
batch_size = 32
num_epochs = 1000
validation_split = .2
do_random_crop = False
patience = 100
num_classes = 2
dataset_name = 'imdb'
input_shape = (64, 64, 1)
if input_shape[2] == 1:
    grayscale = True
images_path = '../datasets/imdb_crop/'
log_file_path = '../trained_models/gender_models/gender_training.log'
trained_models_path =
'../trained_models/gender_models/gender_mini_XCEPTION'

# data loader
data_loader = DataManager(dataset_name)
ground_truth_data = data_loader.get_data()
train_keys, val_keys = split_imdb_data(ground_truth_data, validation_split)
print('Number of training samples:', len(train_keys))
print('Number of validation samples:', len(val_keys))
image_generator = ImageGenerator(ground_truth_data, batch_size,
                                 input_shape[:2],
                                 train_keys, val_keys, None,
                                 path_prefix=images_path,
                                 vertical_flip_probability=0,
                                 grayscale=grayscale,
                                 do_random_crop=do_random_crop)

# model parameters/compilation
model = mini_XCEPTION(input_shape, num_classes)
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.summary()

# model callbacks
```

```
early_stop = EarlyStopping('val_loss', patience=patience)
reduce_lr = ReduceLROnPlateau('val_loss', factor=0.1,
                              patience=int(patience/2), verbose=1)
csv_logger = CSVLogger(log_file_path, append=False)
model_names = trained_models_path + '.{epoch:02d}-{val_acc:.2f}.hdf5'
model_checkpoint = ModelCheckpoint(model_names,
                                   monitor='val_loss',
                                   verbose=1,
                                   save_best_only=True,
                                   save_weights_only=False)
callbacks = [model_checkpoint, csv_logger, early_stop, reduce_lr]

# training model
model.fit_generator(image_generator.flow(mode='train'),
                    steps_per_epoch=int(len(train_keys) / batch_size),
                    epochs=num_epochs, verbose=1,
                    callbacks=callbacks,
                    validation_data=image_generator.flow('val'),
                    validation_steps=int(len(val_keys) / batch_size))
```

13. RESULTS & DISCUSSIONS

Results of sentiment analysis on faces can be observed in figure 5 and figure 6. Along with sentiment analysis we have also done gender classification.

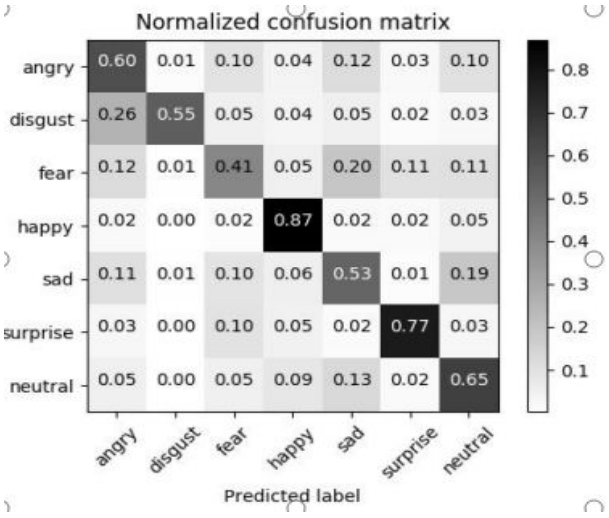


Fig. 6: Confusion matrix results of emotion classification is

We can observe several common misclassifications such as predicting “sad” instead of “fear” and predicting “angry” instead of “disgust”.

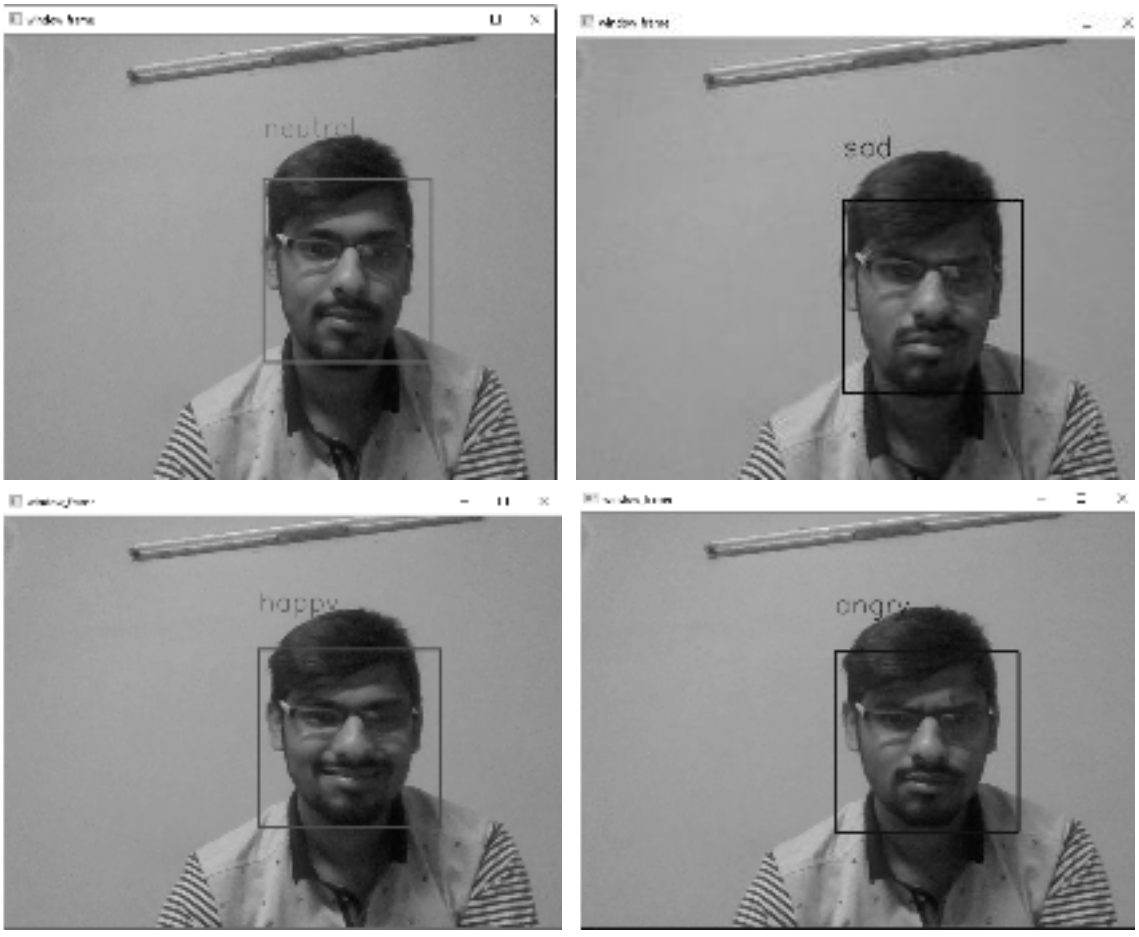


Fig. 7: Sentiment Analysis of face present in live video

In figure 6, faces present in an image frame of live video stream are detected and then emotion is analysed. Here, 4 kinds of emotions are detected - neutral, sad, happy and angry.

Fig. 8: Sentiment analysis of faces present in an image

In figure 7, faces present in an image are detected and then emotion is analyzed along with the gender.

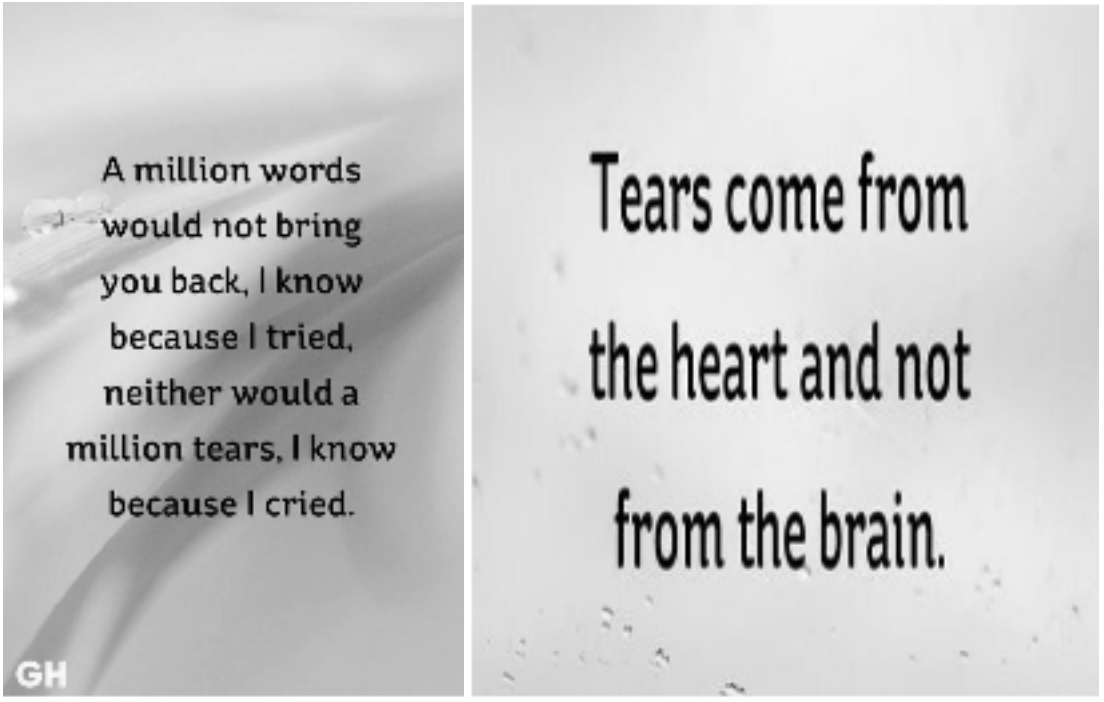
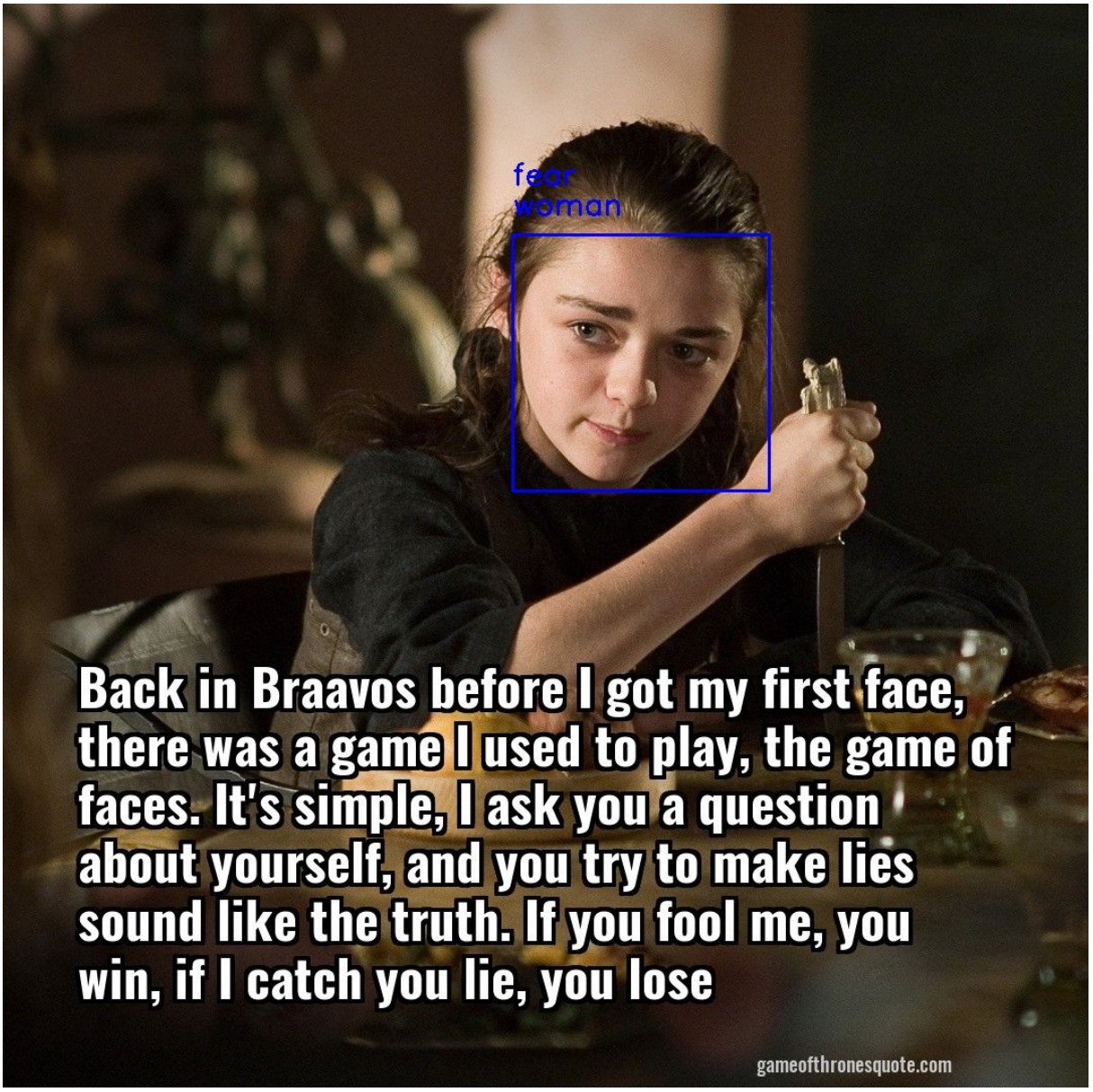


Fig. 9: Samples of images containing texts for sentiment analysis

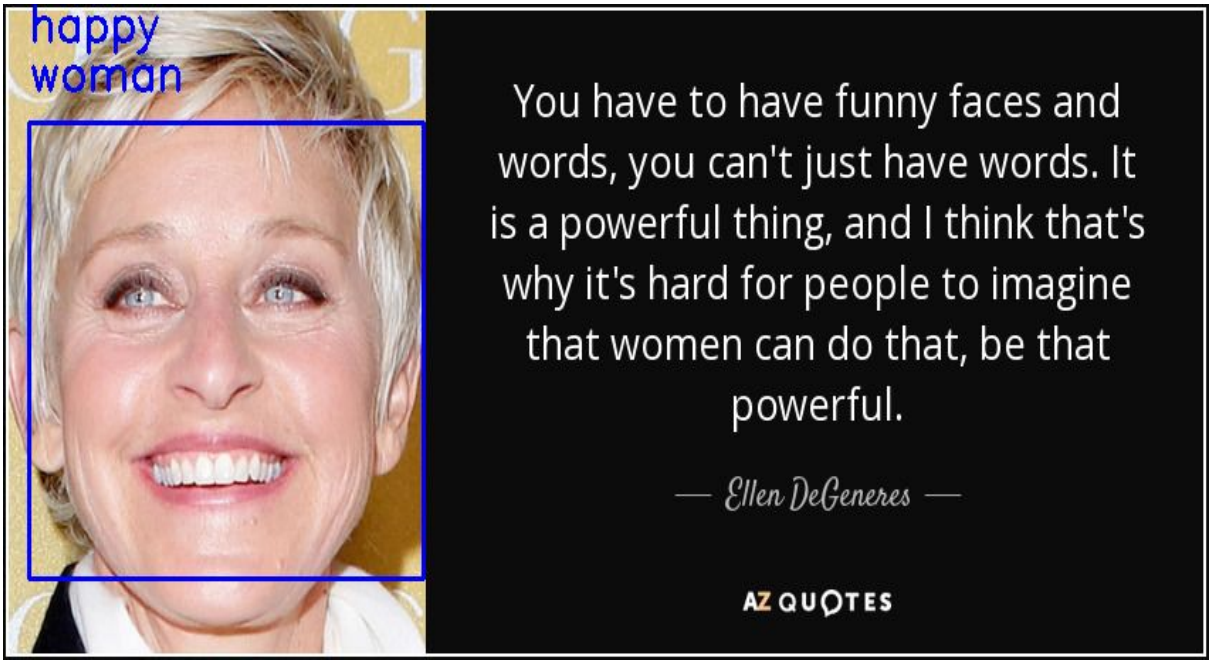
Fig. 10: Result of sentiment analysis of text present in image

The compound value decides finally whether the sentiment is positive, negative or neutral. If the compound value is greater than or equal to 0.05 then it is positive sentiment, if the compound value is less than or equal to -0.05 then it is negative sentiment, otherwise it is neutral sentiment sentiment.


fear
woman
Back in Braavos before I got my first face, there was a game I used to play, the game of faces. It's simple, I ask you a question about yourself, and you try to make lies sound like the truth. If you fool me, you win, if I catch you lie, you lose
gameofthronesquote.com

```
Back in Braavos aia Par LES 6 oe
there.was a game)ljused to play, the game of ~

LETT It's simple}l/ask TNA question !
about yourself; and you|try'to make lies
SUT CUT RCTS LACT C Ian cL

VTL oe

PEO US RU
Back in Braavos aia Par LES 6 oe
there.was a game)ljused to play, the game of ~

LETT It's simple}l/ask TNA question !
about yourself; and you|try'to make lies
SUT CUT RCTS LACT C Ian cL

VTL oe

PEO US RU {'neg': 0.128, 'neu': 0.82, 'pos': 0.052, 'compound': -0.5461}
---------negative sentiment-----------

C:\Users\Ankur Karmakar\Desktop\Multi-Modal-Image-Sentiment-Analysis-master\src>
```

Fig. 11: Sentiment analysis of image containing both text and face



```
You have to have funny faces and
words, you can't just have words. It
is a powerful thing, and | think that's
why it's hard for people to imagine

that women can do that, be that
eon aiele

— Glen DeGeneres —

VAL Oba t)
You have to have funny faces and
words, you can't just have words. It
is a powerful thing, and | think that's
why it's hard for people to imagine

that women can do that, be that
eon aiele

— Glen DeGeneres —

VAL Oba t) {'neg': 0.029, 'neu': 0.855, 'pos': 0.116, 'compound': 0.6486}
---------positive sentiment-----------
```

Fig. 12: Sentiment analysis of image containing both text and face

13.1 Finalized Application

13.1.1 Developing User Interface (UI)

We have designed a user interface so that users can easily use our application to find and perform a multi-modal sentiment analysis on images. We have used tkinter python library to develop the UI. Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

13.1.1.1 User Interface code of our finalized application

```
import sys
import os
from tkinter import filedialog
from tkinter import *

window=Tk()

window.title("Multi Modal Sentiment Analysis")
window.geometry('550x200')
def search_for_file_path ():
  currdir = os.getcwd()
  tempdir = filedialog.askopenfilename(parent=window, initialdir=currdir,
title='Please select a directory')
  if len(tempdir) > 0:
     print ("You chose: %s" % tempdir)
  return tempdir
def browse():
  textBox.delete(1.0, "end")
  textBox.insert(1.0, search_for_file_path())
def run():
  inputValue=textBox.get("1.0","end-1c")
  os.system('python image_emotion_gender_demo.py '+inputValue)
textBox=Text(window, height=2, width=60)
textBox.place(x=30,y=20)
#textBox.pack()
btn2 = Button(window, text="Browse", bg="black", fg="white",command=browse)
btn = Button(window, text="Analyze", bg="black", fg="white",command=run)
btn2.place(x=250,y=70)
btn.place(x=250,y=130)
window.mainloop()
```

13.1.2 Backend code of our finalized application

```
import sys

import cv2
from keras.models import load_model
import numpy as np

from utils.datasets import get_labels
from utils.inference import detect_faces
from utils.inference import draw_text
from utils.inference import draw_bounding_box
from utils.inference import apply_offsets
from utils.inference import load_detection_model
from utils.inference import load_image
from utils.preprocessor import preprocess_input

# parameters for loading data and images
image_path = sys.argv[1]
detection_model_path =
'../trained_models/detection_models/haarcascade_frontalface_default.xml'
emotion_model_path =
'../trained_models/emotion_models/fer2013_mini_XCEPTION.102-0.66.hdf5'
gender_model_path =
'../trained_models/gender_models/simple_CNN.81-0.96.hdf5'
emotion_labels = get_labels('fer2013')
gender_labels = get_labels('imdb')
font = cv2.FONT_HERSHEY_SIMPLEX

# hyper-parameters for bounding boxes shape
gender_offsets = (30, 60)
gender_offsets = (10, 10)
emotion_offsets = (20, 40)
emotion_offsets = (0, 0)

# loading models
face_detection = load_detection_model(detection_model_path)
emotion_classifier = load_model(emotion_model_path, compile=False)
gender_classifier = load_model(gender_model_path, compile=False)

# getting input model shapes for inference
emotion_target_size = emotion_classifier.input_shape[1:3]
gender_target_size = gender_classifier.input_shape[1:3]

# loading images
rgb_image = load_image(image_path, grayscale=False)
gray_image = load_image(image_path, grayscale=True)
```

```
gray_image = np.squeeze(gray_image)
gray_image = gray_image.astype('uint8')

faces = detect_faces(face_detection, gray_image)
for face_coordinates in faces:
    x1, x2, y1, y2 = apply_offsets(face_coordinates, gender_offsets)
    rgb_face = rgb_image[y1:y2, x1:x2]

    x1, x2, y1, y2 = apply_offsets(face_coordinates, emotion_offsets)
    gray_face = gray_image[y1:y2, x1:x2]

    try:
        rgb_face = cv2.resize(rgb_face, (gender_target_size))
        gray_face = cv2.resize(gray_face, (emotion_target_size))
    except:
        continue

    rgb_face = preprocess_input(rgb_face, False)
    rgb_face = np.expand_dims(rgb_face, 0)
    gender_prediction = gender_classifier.predict(rgb_face)
    gender_label_arg = np.argmax(gender_prediction)
    gender_text = gender_labels[gender_label_arg]

    gray_face = preprocess_input(gray_face, True)
    gray_face = np.expand_dims(gray_face, 0)
    gray_face = np.expand_dims(gray_face, -1)
    emotion_label_arg = np.argmax(emotion_classifier.predict(gray_face))
    emotion_text = emotion_labels[emotion_label_arg]

    if gender_text == gender_labels[0]:
        color = (0, 0, 255)
    else:
        color = (255, 0, 0)

    draw_bounding_box(face_coordinates, rgb_image, color)
    draw_text(face_coordinates, rgb_image, gender_text, color, 0, -20, 1, 2)
    draw_text(face_coordinates, rgb_image, emotion_text, color, 0, -50, 1, 2)

bgr_image = cv2.cvtColor(rgb_image, cv2.COLOR_RGB2BGR)
cv2.imwrite('../images/predicted_test_image.png', bgr_image)


# Import modules
import tkinter as tk
from tkinter import *
from PIL import Image
from PIL import ImageTk, Image
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
```

```
import pytesseract
# Include tesseract executable in your path
pytesseract.pytesseract.tesseract_cmd = r"C:\Program Files
(x86)\Tesseract-OCR\tesseract.exe"
# Create an image object of PIL library
image = Image.open('../images/predicted_test_image.png')
# pass image into pytesseract module
# pytesseract is trained in many languages
image_to_text = pytesseract.image_to_string(image, lang='eng')
# Print the text
print(image_to_text)

analyser = SentimentIntensityAnalyzer()
def sentiment_analyzer_scores(sentence):
  score = analyser.polarity_scores(sentence)
  print("{:-<40} {}".format(sentence, str(score)))
  s=""
  if score['compound'] >= 0.05:
     print("---------positive sentiment-----------")
     s="positive sentiment"
  elif score['compound'] <= -0.05:
     print("---------negative sentiment-----------")
     s="negative sentiment"
  else:
     print("---------neutral sentiment------------")
     s="neutral sentiment"
  root1 = tk.Tk()
  root2 = tk.Tk()
  root1.title("Face Sentiment")
  root1.geometry('1024x1024')
  root2.title("Text Sentiment")
  root2.geometry('1024x1024')
  img =
ImageTk.PhotoImage(Image.open("../images/predicted_test_image.png"))
  panel = Label(root1, image=img)
  panel.pack(side="bottom", fill="both", expand="yes")

  x = tk.Label(root2,text="Extracted text= "+image_to_text)
  x.config(font=("Arial", 20))
  x.pack()
  w = tk.Label(root2, text="Sentiment of text= "+s)
  w.config(font=("Arial", 40))
  w.pack()
  root2.mainloop()
  root1.mainloop()

sentiment_analyzer_scores(image_to_text) #image_to_text contains extracted text
```

13.1.3 Converting python code to windows executable file

We have converted python script to windows executable so that users can easily use our application. For this we have used **auto-py-to-exe** which is an amazing application for making .exe files out of your project whether it is one .py file or any number of them.

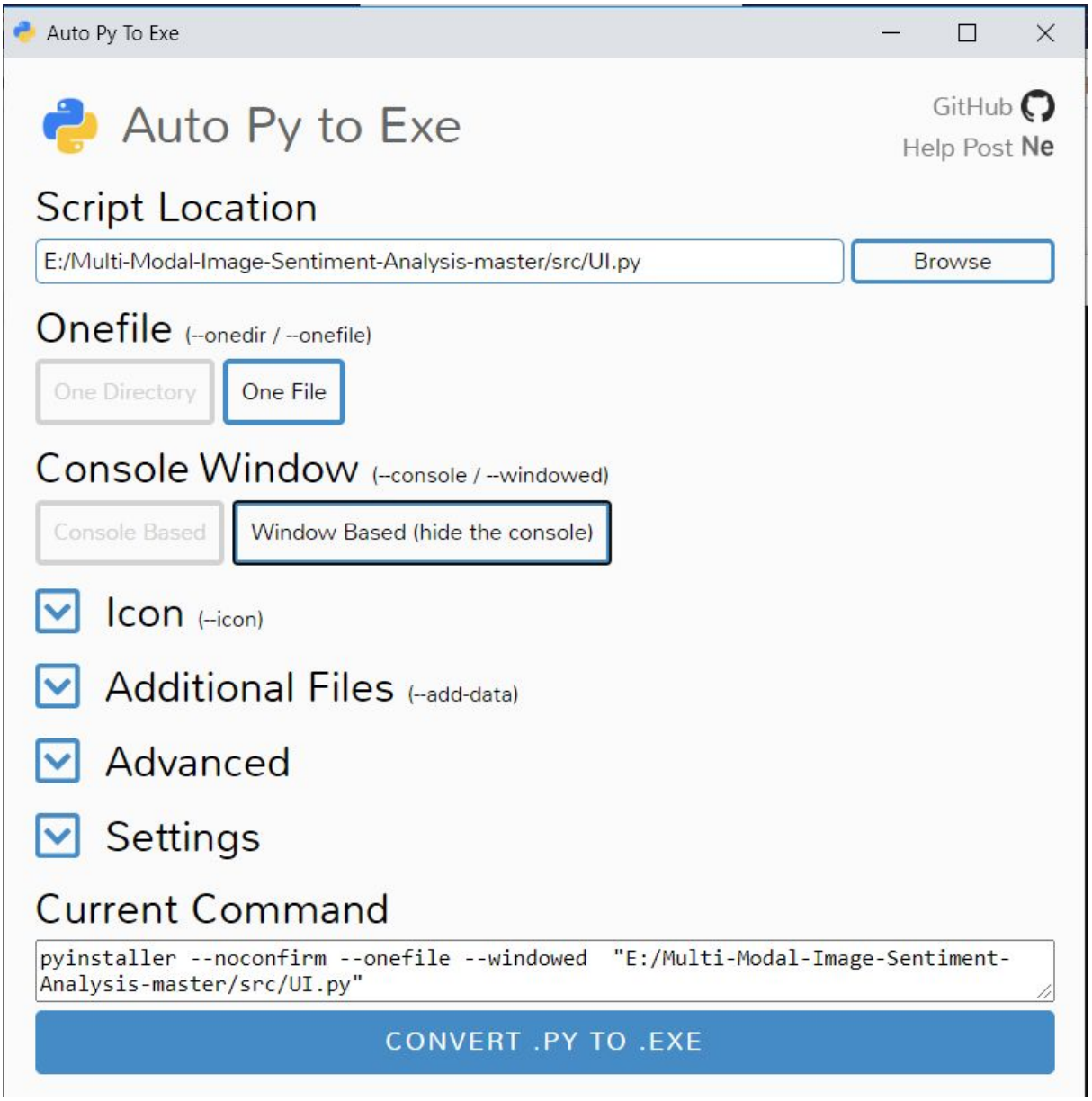


Fig. 13: auto-py-to-exe application

Fig. 14: converted windows executable

13.1.4 Running our application

For running our application, users need to click on UI.exe and the application will start. Then the user has to browse the image which he wants to analyze and then click on the Analyze button. The results will pop up as separate windows - one for text sentiment analysis and for face sentiment analysis.

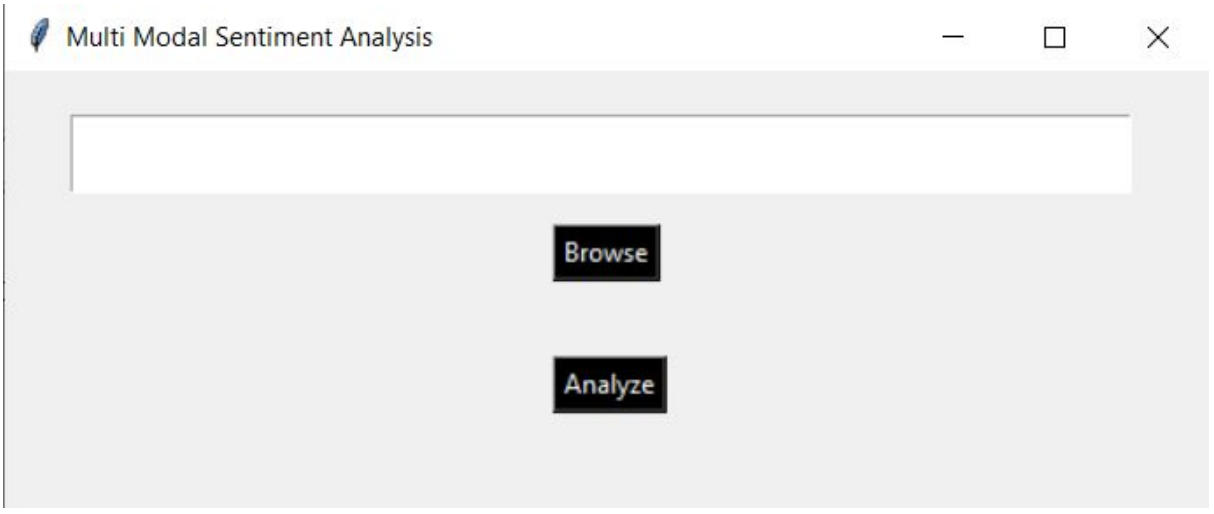


Fig. 15: UI of our application

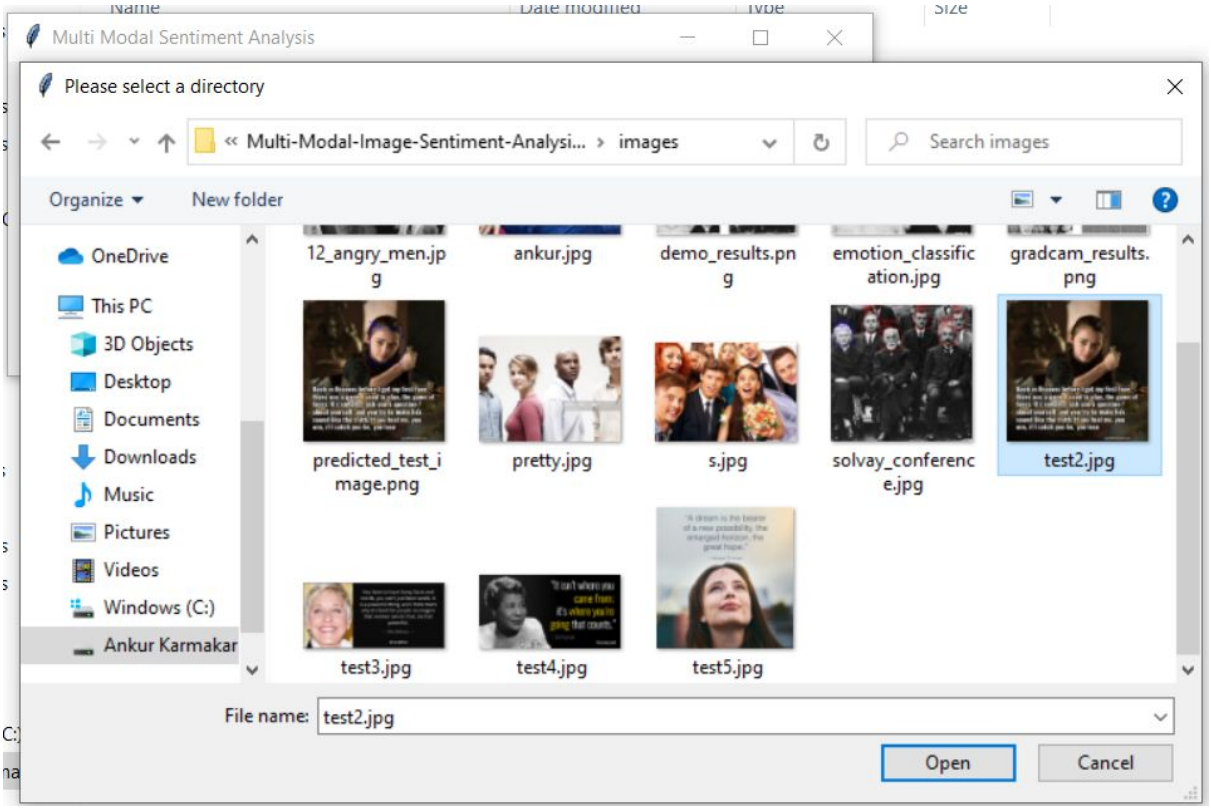


Fig. 16: Browse pictures for sentiment analysis

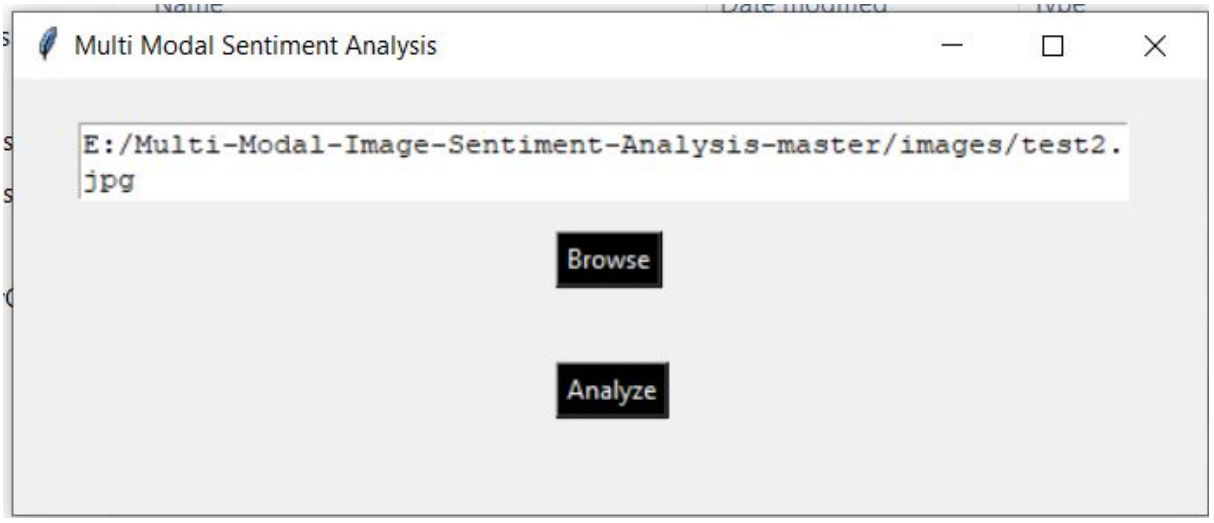


Fig. 17: The path of selected image appears in text

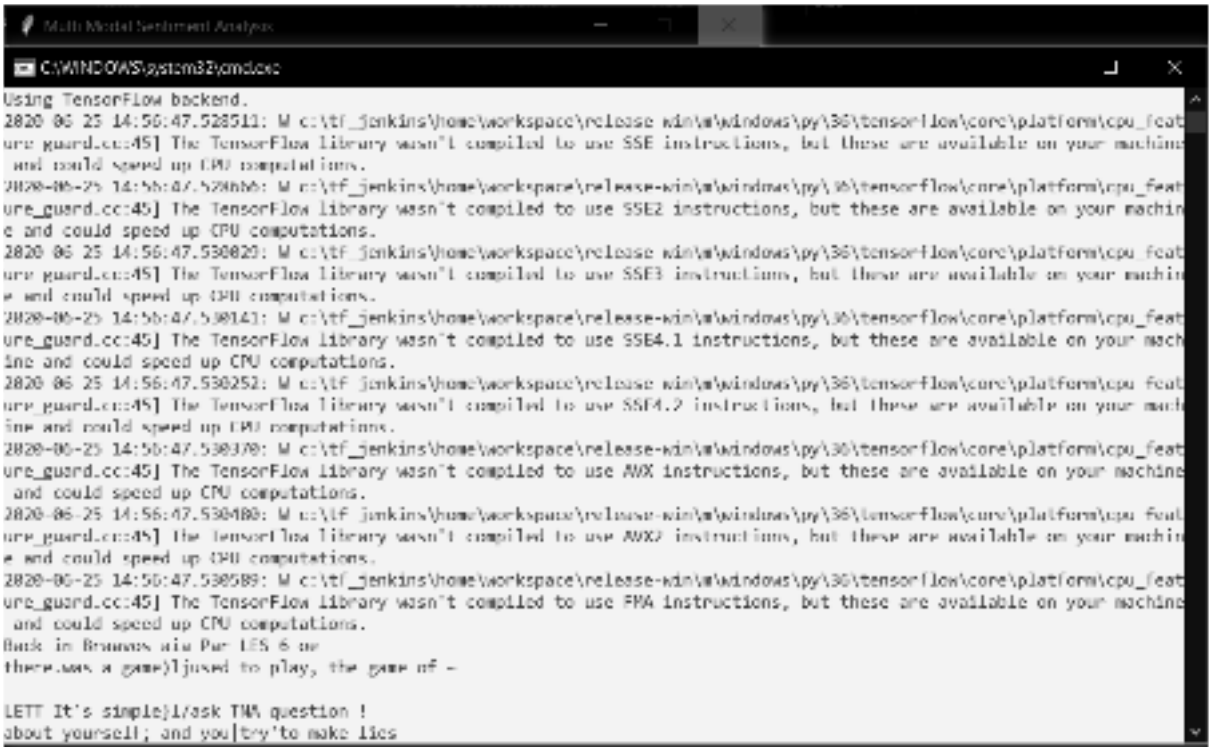


Fig. 18: Command prompt pops up for backend processing

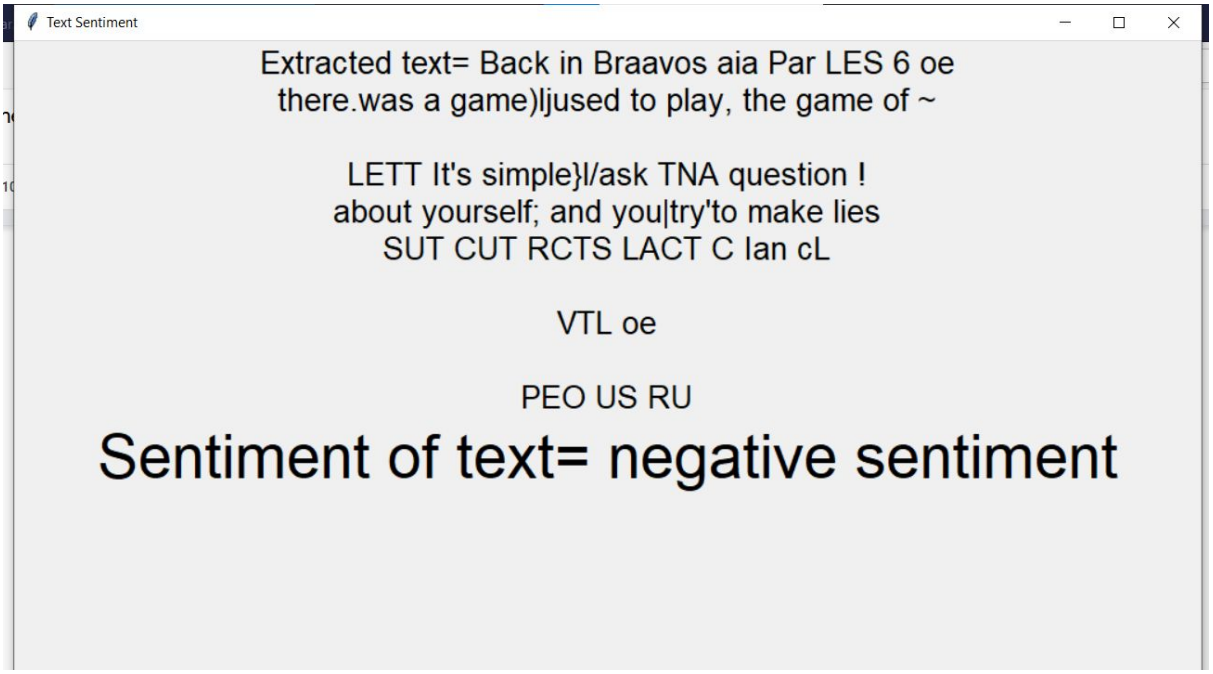


Fig. 19: Result of sentiment analysis on text present in image

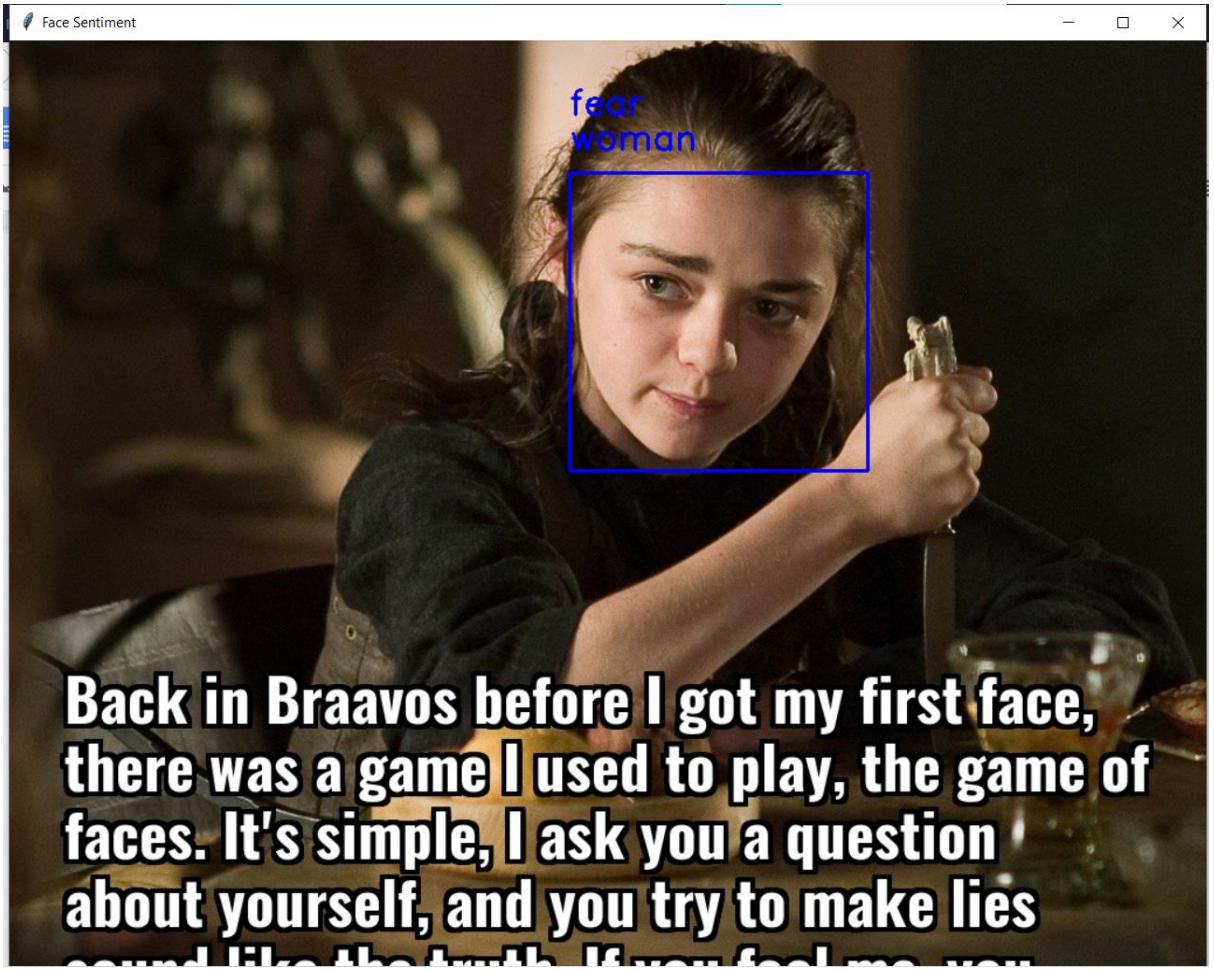


Fig. 20: Result of sentiment analysis on face present in image

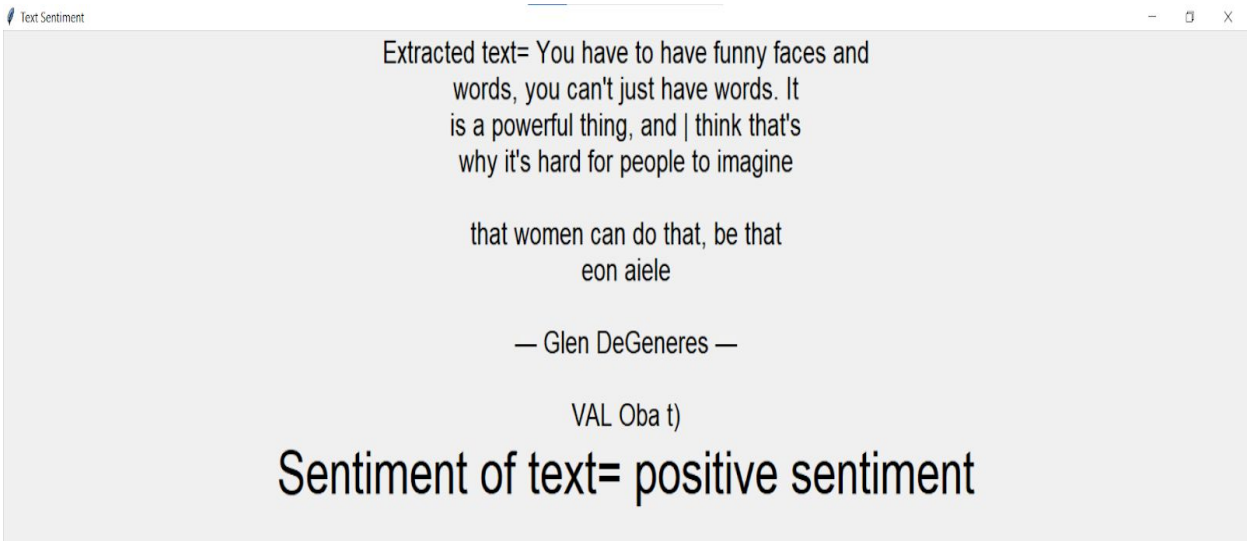


Fig. 21: Result of text sentiment analysis performed on different image

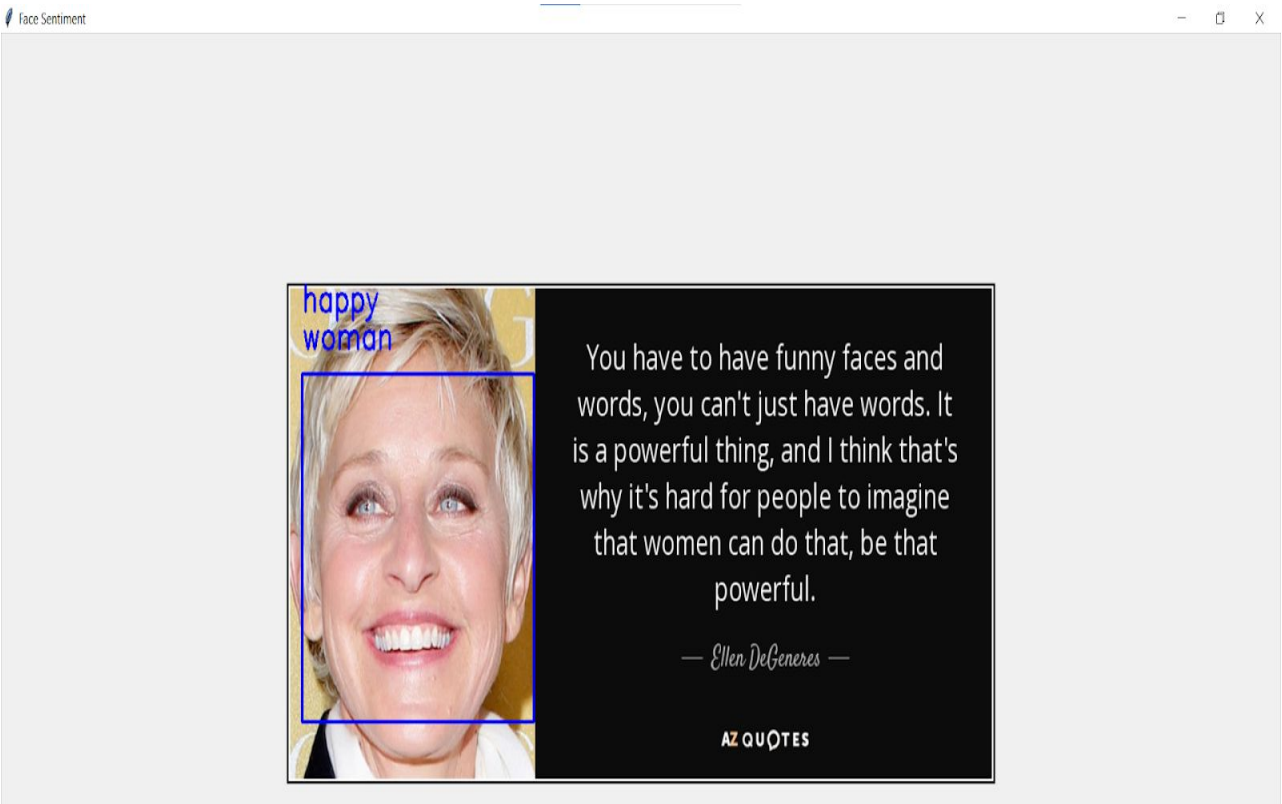


Fig. 22: Result of face sentiment analysis performed on different image

14. FUTURE PLAN

Sentiment analysis is not all that smooth after all. There are several issues related to Sentiment analysis that could lead to the loss of popularity of the technique I.e. opinion spam, e result measure, lack of complete information etc. Despite all the challenges and potential problems that threatens Sentiment analysis, one cannot ignore the value that it adds to the industry. Because Sentiment analysis bases its results on factors that are so inherently humane, it is bound to become one of the major drivers of many business decisions in the future. Improved accuracy and consistency in text mining techniques can help overcome some current problems faced in Sentiment analysis. Looking ahead, what we can see is a true social democracy that will be created using Sentiment analysis, where we can harness the wisdom of the crowd rather than a select few “experts”. A democracy where every opinion counts and every sentiment affects decision making.

Machine learning models are biased in accordance to their training data. In our specific application we have empirically found that our trained CNNs for gender classification are biased towards western facial features and facial accessories. We hypothesize that this misclassification occurs since our training dataset consists of mostly western: actors, writers and cinematographers as observed in Figure 2. Furthermore, as discussed previously, the use of glasses might affect the emotion classification by interfering with the features learned. However, the use of glasses can also interfere with the gender classification. This might be a result from the training data having most of the images of people wearing glasses assigned with the label “man”. We believe that uncovering such behaviours is of extreme importance when creating robust classifiers, and that the use of visualization techniques such as guided back-propagation will become invaluable when uncovering model biases.

<u>Opinion Spam Detection in Social Networks</u>

As social media websites have emerged as popular platforms for sharing and spreading real-time information on the Internet, impostors see huge opportunities in taking advantage of such systems to spread distorted information. Online social networks and review websites have become targets of opinion spamming. More and more traditional review websites allow users to “friend” or “follow” each other so as to enhance overall user experience. This has brought significant advances of opinion spam detection using users’ social networks or heterogeneous networks of various entities in a broader sense. These methods will be implemented via the application of spam detection on review-hosting websites and popular social media platforms.

Sentiment Analysis of Audio

Internet based multimedia has become the main source of presenting one's opinion.This is primarily because regular Internet users have a wider sphere of influence through larger social circles.It is no surprise that among Internet users peer recommendation forms one of the most important sentiment or opinion. Hence work can be done on audio sentiment analysis by converting audio to text and then analyzing the sentiment of the text.

Sentiment Analysis of Video

The vocal inflections and facial expressions in the visual data, along with the language appearing in a textual transcript, provide important cues to better identify affective states of opinion holders, creating a more robust emotion recognition model. We have to classify sentiment in videos (positive, neutral, or negative), their findings can be incorporated into the more specific task of emotion classification across a wider spectrum of basic emotional categories (anger, happiness, sadness, neutral, excitement, frustration, fear, surprise, and other). Example business applications include the development of virtual assistants, analysis of YouTube and other digital or social videos (ex. product reviews, advertising campaigns), analysis of news videos, individual emotion monitoring for mental health professionals, and surely many others.

15. REFERENCES

[1] Ian Goodfellow et al. Challenges in Representation Learning: A report on three machine learning contests, 2013.
[2] Rasmus Rothe, Radu Timofte, and Luc Van Gool. Deep expectation of real and apparent age from a single image without facial landmarks.
[3] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pages 315–323, 2011.
[4] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International Conference on Machine Learning, pages 448–456, 2015.
[5] Octavio Arriaga, Paul G. Ploger, Matias Valdenegro: Real-time Convolutional Neural Networks for Emotion and Gender Classification
[6] Sentiment Analysis on Social Networking: A Literature Review by Prof. Durgesh M. Sharma (Department Of Computer Technology, M.I.E.T, Gondia, India) and Prof. Moiz M. Baig (Department of Computer Science and Engineering, J.D.C.O.E.M, Nagpur, India)
[7] Karen Simonyan and Andrew Zisserman. Very deep convolu-tional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.
[8] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2818–2826, 2016.
[9] Franc¸ois Chollet. Xception: Deep learning with depth wise separable convolutions. CoRR, abs/1610.02357, 2016.
[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.
[11] Andrew G. Howard et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. CoRR, abs/1704.04861, 2017.
[12] Yichuan Tang. Deep learning using linear support vector machines. arXiv preprint arXiv:1306.0239, 2013.