

A MINI PROJECT REPORT
ON
EMOTION DETECTION(IMAGE,VIDEO, & LIVESTREAM)
USING MACHINE LEARNING.

Submitted in partial fulfillment of the requirements for the award of the degree in

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

BY

M.VISHNU PRIYA	(20NN1A1299)
A.BHAGYA LAKSHMI	(20NN1A1270)
P.JYOTHI	(20NN1A12B2)
Y.LAVANYA	(20NN1A12C6)

Under the esteemed guidance of

R.SRINIVAS



DEPARTMENT OF INFORMATION TECHNOLOGY

VIGNAN'S NIRULA INSTITUTE OF TECHNOLOGY AND SCIENCE FOR
WOMEN

(Approved by AICTE, NEW DELHI and Affiliated to JNTUK)

PEDAPALAKALURU, GUNTUR-522005

(2020-2024)

**VIGNAN'S NIRULA INSTITUTE OF TECHNOLOGY AND SCIENCE FOR
WOMEN**

(Approved by AICTE, NEW DELHI and Affiliated to JNTUK)

PEDAPALAKALURU,GUNTUR-522005

(2020-2024)

DEPARTMENT OF INFORMATION TECHNOLOGY



CERTIFICATE

This is to certify that the project work entitled “EMOTION DETECTION(IMAGE,VIDEO,& LIVESTREAM) USING MACHINE LEARNING“ is a bonafide work submitted by M.VISHNUPRIYA (20NN1A1299), A.BHAGYALAKSHMI (20NN1A1270), P.JYOTHI (20NN1A12B2), Y.LAVANYA (20NN1A12C6) from the department of Information Technology in the partial fulfillment of the requirements forward of degree of Bachelor of Technology in Information Technology from Vignan’s Nirula Institute of Technology & Science for Women, Guntur.

Internal Guide
R.SRINIVAS

Head of the Department
Dr. RAMA KRISHNA

External Examiner

DECLARATION

We hereby declare that the work described in this project work, entitled “EMOTION DETECTION (IMAGE,VIDEO,& LIVESTREAM) USING MACHINE LEARNING” which is submitted by us in partial fulfillment for the award of

Bachelor of Technology in the department of Information Technology to the Vignan’s Nirula Institute of Technology & Science for women, affiliated to Jawaharlal Nehru Technological University Kakinada, Andhra Pradesh, is the result of work done by us under the esteemed guidance of R.SRINIVAS, Assistant Professor.

The work is original and has not been submitted for any Degree/Diploma of this or any other university.

M.VISHNU PRIYA	(20NN1A1299)
A.BHAGYA LAKSHMI	(20NN1A1270)
P.JYOTHI	(20NN1A12B2)
Y.LAVANYA	(20NN1A12C6)

ACKNOWLEDGEMENT

We profoundly grateful to express our deep sense of gratitude and respect towards our honorable chairman, **LAVU RATHAIAH** sir ,Chairman of Vignan group for his precious support in the college.

We are much thankful to **DR.P.RADHIKA**, Principal VNITSW, Guntur, for her support during and till the completion of the project.

We would like to thank **DR.RAMA KRISHNA**, Associate Professor and Head of the Department of Information Technology, for his extended and continuous support, valuable guidance and timely advices in the completion of this project thesis.

We wish to express our profound sense of sincere gratitude to our Project Guide, **R.SRINIVAS**, Assistant Professor of Information Technology, without whose help, guidance and motivation this project thesis could not have been completed the project successfully.

We also thank all the faculty of the Department of Information Technology for their help and guidance of numerous occasions, which has given us the cogency to build-up adamant aspiration over the completion of our project thesis and finally, we thank one and all who directly or indirectly helped us to complete our project thesis successfully.

M.VISHNU PRIYA	(20NN1A1299)
A.BHAGYA LAKSHMI	(20NN1A1270)
P.JYOTHI	(20NN1A12B2)
Y.LAVANYA	(20NN1A12C6)

ABSTRACT

Now-a-days with the continued development of artificial intelligence facial emotion recognition has become more popular. The emotion recognition plays a major role in interaction technology. In interaction technology the verbal components only play a one third of communication and the non-verbal components plays a two third of communication. A facial emotion recognition (FER) method is used for detecting facial expressions. Facial expression plays a major role in expressing what a person feels and it expresses inner feeling and his or her mental situation. This project aims to identify basic human emotions through web cam, image and video. The facial emotions such as happy, sad, angry, fear, surprised, neutral emotions are considered as basic emotions. Six respondents' coefficients defining various aspects of face expressions were chosen as the characteristics. The features have been calculated for three-dimensional face model. The classification of features was performed using OpenCV, harassed _frontal face _default

Index

SNO	Contents	Page No
1	INTRODUCTION	2-9
	1.1 Facial Expression Recognition with Histogram of Oriented Gradients using CNN.	3
	1.2 Active Clustering with Ensembles for Social structure extraction	4
	1.3 Fast human detection using a cascade of histograms of oriented gradients	4
	1.4 Local directional ternary pattern for facial expression recognition	5
	1.5 Software environment	5
	1.6 Why choose Python	6
2	LITERATURE SURVEY	11-14
3	SYSTEM ANALYSIS	16-20
	3.1 EXISTING SYSTEM	16
	3.2 PROPOSED SYSTEM	18

4	FEASABILITY STUDY	22
	4.1 ECONOMICAL FEASIBILITY	22
	4.2 TECHNICAL FEASIBILITY	22
	4.3 SOCIAL FEASIBILITY	22
5	SYSTEM REQUIREMENTS	24
	5.1 HARDWARE REQUIREMENTS	24
	5.2 SOFTWARE REQUIREMENTS	24
6	SYSTEM DESIGN	26-29
	6.1 SYSTEM ARCHITECTURE	26
	6.2 DATAFLOW DIAGRAM	26
	6.3 UML DIAGRAMS	27
7	IMPLEMENTATION	31-37
	7.1 MODULES	31
	7.2 SAMPLE CODE	31
8	SYSTEM TESTING	39-41
	8.1 UNIT TESTING	40
	8.2 INTEGRATION TESTING	41
	8.3 ACCEPTANCE TESTING	41

9	INPUT DESIGN AND OUTPUT DESIGN	43-46
	9.1 INPUT DESIGN	43
	9.2 OUTPUT DESIGN	43
10	SCREEN SHOTS	46-52
11	FUTURE ENHANCEMENT	54-57
12	CONCLUSION	59
13	BIBLIOGRAPHY	61-63

LIST OF FIGURES

S.NO	FIGURE TITLES	PAGE NUMBER
1.	Steps to detect the emotion	26
2.	Use Case Diagram Of Emotion Detection	28
3.	Sequence Diagram Of Emotion Detection	28
4.	Activity Diagram Of Emotion Detection	29
5.	Training the model	31-32
6.	Emotion detection with image input	33
7.	Emotion Detection using live camera as input	34
8.	Emotion Detection Using Video Input	35
9.	Accuracy Graph Based On The Captured emotions On Live Camera	36-37
10.	Training the model With Epochs	46
11.	Output after Training the model	47
12.	Training And Validation Accuracy Graph	47
13.	Running Image recognition Code	48
14.	Output of Emotion recognition in Image	48
15.	Running Video prediction Code	49
16	Output of emotion recognition in video	50

17.	Running live Camera emotion detection Code	51
18.	Output Of emotion Detection in Live camera	51
19.	Running the Accuracy Graph Code	52
20.	Accuracy Output Based On The captured Emotions On Live Camera	52
21.	Training the enhanced model	55-56

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

A face detection includes classifying image into two classes: one with faces (targets), and other containing the background (clutter) which needs to be removed. Commonalities exist between faces, they vary differently in terms of age, skin color and facial expression, this becomes difficult due to this commonality. The further problem is complicated by differing lighting conditions, image qualities and geometries, partial occlusion and disguise is also a possibility.

A face detector should be able to detect the presence of any face under different set of lighting conditions in any background condition. The face detection analysis can be broken into two tasks. The first task is a classification task that takes some random regions of image as input and outputs a binary value of yes or no, indicating if there are any faces present in the image. The other task is the face localization task which is to take an image as input and output the location of any face or faces within that image as some bounding box/boxes with (x, y, width, height).

Smart robots can be built by automatic facial expression application. These bots can be used in various applications like interactive games and service center. There are six universal expressions according to Ekman they are fear, disgust surprise, anger, sadness and happiness. Face variances can be observed to recognize these expressions. For example, we can say a person is happy which can be identified as a gesture of smile by tightened eyelids and raised lips corners.

A person's internal states, social communication and intentions are indicated by change in facial expressions. Many applications in many areas like human emotions analysis, natural Human computer interaction, image retrieval and talking bots have a Recognition with Histogram of Oriented Gradients using CNN detection has been an impacting issue in the technological community as human beings find facial expressions one of the most natural and powerful means to express their intentions and emotions.

Last stage of the system is facial expression detection. There are basically three steps in training procedure in expression recognition systems named as feature

learning, classifier construction and feature selection. Feature learning stage is first, feature selection is second and the last one is classifier construction.

Only learned facial expressions variations among all features are extracted after feature learning stage. Facial expression is then represented by the best features which are chosen by feature selection. Not only maximizing inter class variation but they also should try to minimize the intra class variations of expressions not only maximizing inters class variation but they also should minimize the intra class variations of expressions. Because same expressions of different individuals in image are far from each other in pixel's space so minimizing the intra class variation of expressions is a problem.

1.1 Facial Expression Recognition with Histogram of Oriented Gradients using CNN.

A new method is introduced in this study for Facial expression recognition using FER2013 database consisting seven classes consisting (Surprise, Fear, Angry, Neutral, Sad, Disgust, Happy) in past few decades, Exploration of methods to recognize facial expressions have been active research area and many applications have been developed for feature extraction and inference. However, it is still challenging due to the high-intra class variation.

Methods/Statistical Analysis: we deeply analyzed the accuracy of both handcrafted and leaned aspects such as HOG. This study proposed two models; (1) FER using Deep Convolutional Neural Network (FER-CNN) and (2) Histogram of oriented Gradients based Deep Convolutional Neural Network (FER-HOGCNN). the training and testing accuracy of FER-CNN model set 98%, 72%, similarly Losses were 0.02, 2.02 respectively. On the other side, the training and testing accuracy of FER-HOGCNN model set 97%, 70%, similarly Losses were 0.04, 2.04. Findings: It has been found that the accuracy of FER-HOGCNN model is good overall but comparatively not better than Simple FER-CNN. In dataset the quality of images are low and small dimensions, for that reason, the HOG loses some important features during training and testing. Application/Improvements: The study helps for improving the FER System in image processing and furthermore, this work shall be extended in future, and order to extract the important features from images by combining LBP and HOG operator using Deep Learning models.

1.2 Active Clustering with Ensembles for Social structure extraction

We introduce a method for extracting the social network structure for the persons appearing in a set of video clips. Individuals are unknown, and are not matched against known enrollments. An identity cluster representing an individual is formed by grouping similar-appearing faces from different videos. Each identity cluster is represented by a node in the social network.

Two nodes are linked if the faces from their clusters appeared together in one or more video frames. Our approach incorporates a novel active clustering technique to create more accurate identity clusters based on feedback from the user about ambiguously matched faces. The final output consists of one or more network structures that represent the social group(s), and a list of persons who potentially connect multiple social groups. Our results demonstrate the efficacy of the proposed clustering algorithm and network analysis techniques.

1.3 Fast human detection using a cascade of histograms of oriented gradients

We integrate the cascade-of-rejectors approach with the Histograms of Oriented Gradients (HoG) features to achieve a fast and accurate human detection system. The features used in our system are HoGs of variable-size blocks that capture salient features of humans automatically. Using AdaBoost for feature selection, we identify the appropriate set of blocks, from a large set of possible blocks. In our system, we use the integral image representation and arejection cascade which significantly speed up the computation. For a 320×280 image, the system can process 5 to 30 frames per second depending on the density in which we scan the image, while maintaining an accuracy level similar to existing methods.

1.4 Local directional ternary pattern for facial expression recognition

This paper presents a new face descriptor, local directional ternary pattern (LDTP), for facial expression recognition. LDTP efficiently encodes information of emotion-related features (i.e., eyes, eyebrows, upper nose, and mouth) by using the directional information and ternary pattern in order to take advantage of the robustness of edge patterns in the edge region while overcoming weaknesses of edge-based methods in smooth regions. Our proposal, unlike existing histogram-based face description methods that divide the face into several regions and sample the codes uniformly, uses a two-level grid to construct the face descriptor while sampling expression-related information at different scales.

We use a coarse grid for stable codes (highly related to non-expression), and a finer one for active codes (highly related to expression). This multi-level approach enables us to do a finer grain description of facial motions while still characterizing the coarse features of the expression. Moreover, we learn the active LDTP codes from the emotion-related facial regions. We tested our method by using person-dependent and independent cross-validation schemes to evaluate the performance. We show that our approaches improve the overall accuracy of facial expression recognition on six data sets.

1.5 SOFTWARE ENVIRONMENT

Python is a high-level, interpreted scripting language developed in the late 1980s by Guido van Rossum at the National Research Institute for Mathematics and Computer Science in the Netherlands. The initial version was published at the alt.Sources [newsgroup](#) in 1991, and version 1.0 was released in 1994.

Python 2.0 was released in 2000, and the 2.x versions were the prevalent releases until December 2008. At that time, the development team made the decision to release version 3.0, which contained a few relatively small but significant changes that were not backward compatible with the 2.x versions. Python 2 and 3 are very similar, and some features of Python 3 have been back ported to Python 2. But in general, they remain not quite compatible.

Both Python 2 and 3 have continued to be maintained and developed, with periodic release updates for both. As of this writing, the most recent versions available are

2.7.15 and 3.6.5. However, an official End of Life date of January 1, 2020 has been established for Python 2, after which time it will no longer be maintained. If you are a newcomer to Python, it is recommended that you focus on Python 3, as this tutorial will do.

Python is still maintained by a core development team at the Institute, and Guido is still in charge, having been given the title of BDFL (Benevolent Dictator For Life) by the Python community. The name Python, by the way, derives not from the snake, but from the British comedy troupe Monty Python's Flying Circus, of which Guido was, and presumably still is, a fan. It is common to find references to Monty Python sketches and movies scattered throughout the Python documentation.

1.6 WHY CHOOSE PYTHON

If you're going to write programs, there are literally dozens of commonly used languages to choose from. Why choose Python? Here are some of the features that make Python an appealing choice.

Python is Popular

Python has been growing in popularity over the last few years. The 2018 Stack Overflow Developer Survey ranked Python as the 7th most popular and the number one most wanted technology of the year. World-class software development countries around the globe use Python every single day.

According to research by Dice Python is also one of the hottest skills to have and the most popular programming language in the world based on the Popularity of Programming Language Index.

Due to the popularity and widespread use of Python as a programming language, Python developers are sought after and paid well. If you'd like to dig deeper into Python salary statistics and job opportunities, you can do so here.

Python is interpreted

Many languages are compiled, meaning the source code you create needs to be translated into machine code, the language of your computer's processor, before it can be run. Programs written in an interpreted language are passed straight to an interpreter that runs them directly.

This makes for a quicker development cycle because you just type in your code and run it, without the intermediate compilation step.

One potential downside to interpreted languages is execution speed. Programs that are compiled into the native language of the computer processor tend to run more quickly than interpreted programs. For some applications that are particularly computationally intensive, like graphics processing or intense number crunching, this can be limiting.

In practice, however, for most programs, the difference in execution speed is measured in milliseconds, or seconds at most, and not appreciably noticeable to a human user. The expediency of coding in an interpreted language is typically worth it for most applications.

Python is Free

The Python interpreter is developed under an OSI-approved open-source license, making it free to install, use, and distribute, even for commercial purposes.

A version of the interpreter is available for virtually any platform there is, including all flavors of Unix, Windows, macOS, smart phones and tablets, and probably anything else you ever heard of. A version even exists for the half dozen people remaining who use OS/2.

Python is Portable

Because Python code is interpreted and not compiled into native machine instructions, code written for one platform will work on any other platform that has the Python interpreter installed. (This is true of any interpreted language, not just Python.)

Python is Simple

As programming languages go, Python is relatively uncluttered, and the developers have deliberately kept it that way.

A rough estimate of the complexity of a language can be gleaned from the number of keywords or reserved words in the language. These are words that are reserved for special meaning by the compiler or interpreter because they designate specific built-in functionality of the language.

Python 3 has 33 keywords, and Python 2 has 31. By contrast, C++ has 62, Java has 53, and Visual Basic has more than 120, though these latter examples probably vary somewhat by implementation or dialect.

Python code has a simple and clean structure that is easy to learn and easy to read. In fact, as you will see, the language definition enforces code structure that is easy to read.

But It's Not That Simple

For all its syntactical simplicity, Python supports most constructs that would be expected in a very high-level language, including complex dynamic data types, structured and functional programming, and object-oriented programming.

Additionally, a very extensive library of classes and functions is available that provides capability well beyond what is built into the language, such as database manipulation or GUI programming.

Python accomplishes what many programming languages don't: the language itself is simply designed, but it is very versatile in terms of what you can accomplish with it.

Conclusion

This section gave an overview of the **Python** programming language, including:

- A brief history of the development of Python
- Some reasons why you might select Python as your language of choice

Python is a great option, whether you are a beginning programmer looking to learn the basics, an experienced programmer designing a large application, or anywhere in between. The basics of Python are easily grasped, and yet its capabilities are vast. Proceed to the next section to learn how to acquire and install Python on your computer.

Python is an open source programming language that was made to be easy-to-read and powerful. A Dutch programmer named Guido van Rossum made Python in 1991. He named it after the television show Monty Python's Flying Circus. Many Python examples and tutorials include jokes from the show.

Python is an interpreted language. Interpreted languages do not need to be compiled to run. A program called an interpreter runs Python code on almost any kind of computer. This means that a programmer can change the code and quickly see the results. This also means Python is slower than a compiled language like C, because it is not running machine code directly.

Python is a good programming language for beginners. It is a high-level language, which means a programmer can focus on what to do instead of how to do it. Writing programs in Python takes less time than in some other languages.

Python drew inspiration from other programming languages like C, C++, Java, Perl, and Lisp.

Python has a very easy-to-read syntax. Some of Python's syntax comes from C, because that is the language that Python was written in. But Python uses whitespace to delimit code: spaces or tabs are used to organize code into groups. This is different from C. In C, there is a semicolon at the end of each line and curly braces ({}) are used to group code. Using whitespace to delimit code makes Python a very easy-to-read language.

Python use [change / change source]

Python is used by hundreds of thousands of programmers and is used in many places. Sometimes only Python code is used for a program, but most of the time it is used to do simple jobs while another programming language is used to do more complicated tasks.

Its standard library is made up of many functions that come with Python when it is installed. On the Internet there are many other libraries available that make it possible for the Python language to do more things. These libraries make it a powerful language; it can do many different things.

Some things that Python is often used for are:

- Web development
- Scientific programming
- Desktop GUIs
- Network programming
- Game programming

CHAPTER 2

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

GUO et al. [24] Current convolutional neural network-based face or object detection approaches (such as OverFeat, R-CNN, and DenseNet) actively retrieve features with multiple scales based on an image pyramid. Such an approach, however, raises the computing cost for face detection. The authors presented a rapid face detection technique that utilizes discriminative complete features (DCF_s) extracted by an intricately built convolutional neural network, where face detection is conducted directly on the entire feature maps. DCF_s have demonstrated scale invariance, which is advantageous for face detection with fast speed and promising results. As a result, the suggested method eliminates the need for extracting multi-scale features from an image pyramid, which can considerably increase its effectiveness for face detection.

S. Gupta et al. [23] Existing face recognition algorithms work well in frontal-face photographs with good illumination and very close proximity to the imaging instrument. Most existing algorithms, however, fail to work similarly in surveillance circumstances where videos are taken at varied resolutions and spectra. Cameras in surveillance settings are typically located far away from the subjects, resulting in changes in pose, illumination, occlusion, and resolution. Current video datasets for face recognition are frequently shot in confined environments, failing to represent real-world scenarios. The authors presented the FaceSurv database, which contains 252 subjects in 460 videos, in this work. Over 142K face photos are scattered throughout movies shot using both visible and near-infrared spectra in the proposed dataset.

Shivkaran ravidas et al. [22] The goal is to use a deep convolutional neural network (DCNN) to detect multi-view faces. Due to the wide variations in appearance under diverse positional expressions and illumination settings, multi-view face detection is a difficult problem. To solve these issues, the authors created a deep learning method using several network structures to improve multi-view faces. The authors created a cascade architecture based on convolutional neural networks (CNNs) that swiftly rejects non-face regions. Face implementation, detection, and retrieval will be accomplished via direct visual recognition technology. Furthermore, a

probability calculation of likeness among face photos will be performed on the foundation of the Bayesian analysis in order to detect various faces.

Shafiee et al. [21] Object detection is one of the most difficult problems in this area of computer vision because it is a combination of object classifying and object localization within the scene. Recently, DNNs have been proven to perform better in object detection compared to other approaches. One of the best DNN-based methods for object detection is YOLOV2. YOLOV2 is one of the fastest YOLO-based methods in terms of speed and accuracy. While YOLOV2 can perform in real-time on a high-performance GPU, it is still very difficult to leverage this approach in real-time for video object detection on embedded computing devices that have limited computational power and memory. In this paper, the authors introduced a new framework, Fast YOLO. Fast YOLO is a fast YOLO framework that accelerates YOLOV2 to perform video object detection in embedded devices in real-time. First, the authors used the evolutionary deep intelligence to develop the YOLOV2 network architecture and create an optimized architecture with 2.8x fewer parameters with only a ~2% IOU drop.

Sivaram M et al. [20] Facial landmark detection is a critical issue in many PC vision programs that deal with looks. It is incredibly difficult because human faces in the wild frequently display extensive variations suited as a fiddle due to varied positions, obstructions, or demeanors. Deep neural networks have been linked to processing information as guidance from face images to face forms. To the best of our knowledge, Recurrent Neural Network (RNN) has not yet been used in this issue. In this, the authors suggested a technique for capturing face form using RNN and Deep Neural Network (DNN). To the greatest extent of our knowledge, Recurrent Neural Network (RNN) has not yet been used in this issue. In this paper, the authors suggested a technique for capturing face form using RNN and Deep Neural Network (DNN). To begin, the authors created a system that uses a Convolutional Neural Network (CNN) to estimate the underlying landmark of appearances. Then, the authors used feed-forward neural circuits for neighborhood search, and a segment-based searching strategy is explored.

Sermanet et al. [18] Authors provided an integrated system for classification, localization, and detection using Convolutional Networks. The authors demonstrated how to efficiently construct a multiscale and sliding window technique within a ConvNet. In addition, The authors provide a novel deep learning technique for

localization that requires training to predict object borders. To boost detection confidence, bounding boxes are gathered rather than suppressed. The authors demonstrate that many tasks can be learned concurrently using a single shared network. This integrated system won the ImageNet Large Scale Visual Recognition Challenge 2013 (ILSVRC2013) localization task and delivered near-state-of-the-art results for the detection and classification tasks. Finally, authors have released OverFeat, a feature extractor based on our best model.

Girshick et al. [17] Object detection performance on the conventional PASCAL VOC dataset has plateaued in recent years. Sophisticated ensemble systems that often combine many low-level picture features with the highest level context are the best-performing methods. In this study, the authors offered a simple and scalable detection technique that enhances mean average precision (mAP) by over thirty percent over the previous best VOC 2012 result, yielding a mAP of 53.3%. The authors' strategy includes two fundamental insights: (1) When marked training data is scarce, supervised pre-training for an auxiliary task followed by domain-specific fine-tuning yields a significant performance boost; and (2) when marked training data is scarce, supervised pre-training for an auxiliary task followed by domain-specific fine-tuning yields a significant performance boost.

Shahrabi Farahani et al. [13] Emotions are important in human interactions. The ability of a computer to interpret human emotions is useful in a variety of applications, particularly when observing facial expressions. This research describes a new fuzzy-based approach for recognizing emotions from eye and mouth features in people of various ages. The approach recognizes eyes and mouths by combining various color spaces. For fuzzy analysis, four parameters are chosen: eye opening, mouth opening, eye opening/width ratio, and mouth width. Facial features and their transfer to emotional space are encoded using Mamdani-type implication relations.

Singh et al.[9] To advance the current state of the art in face recognition and bridge the divide between laboratory and real-world scenarios, demanding databases must be available. Surveillance is a difficult application of face recognition since unrestrained video data is gathered both during the day and at night (visible and near infrared spectrum). These films contain numerous subjects in each frame, that are accompanied by high-quality gallery photographs. This is still an open research subject because of the absence of a current database with such a spectral cross resolution video-to-still face identification application. This study includes a video

database that may be used to test face recognition algorithms that solve cross-spectral cross-resolution matching.

L. Wolf et al.[5] To advance the present state of the art in face recognition and bridge the divide between laboratory and real-world scenarios, demanding databases must be available. Surveillance is a difficult application of face recognition since unrestrained video data is gathered both during the day and at night (visible and near infrared spectrum). These films contain numerous subjects in each frame, that are accompanied by high-quality gallery photographs. This is still an open research subject because of the absence of a current database with such a spectral cross resolution video-to-still face identification application. This study includes a video database that may be used to test face recognition algorithms that solve cross-spectral cross-resolution matching. (b) the authors surveyed to evaluate the performance of a wide range of currently available video face recognition systems using our benchmark. Finally, in (c), the authors described the Matched Background Similarity (MBGS), a novel set-to-set similarity measure.

CHAPTER 3

SYSTEM ANALYSIS

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM:

Emotion recognition, especially from facial images, has been a topic of research for years. There are several well-known models and approaches that have been proposed. Here's an overview of some existing emotion recognition models:

1. VGG-Face:

Based on the VGG16 architecture.

Initially trained for facial recognition, but it can be fine-tuned for emotion recognition tasks.

2. AffectNet:

A larger dataset that includes facial expression images.

Models trained on AffectNet tend to use deeper architectures due to the size of the dataset.

It's not just a model but also a database. However, models trained on this database are usually deeper CNNs or variations of architectures like ResNet.

3. EmoReact:

A large-scale dataset and model for understanding the emotional impact of scenes in videos.

Uses CNNs and LSTMs to recognize emotions from video frames.

4. Deep Affect Prediction:

A deep learning model trained on the AffectNet dataset.

Uses an end-to-end deep architecture.

5. ResNet-50, ResNet-101, etc.:

Though initially not designed for emotion recognition, these architectures, which were introduced for image classification tasks, have been successfully adapted and fine-tuned for emotion recognition tasks.

6. MobileNet and SqueezeNet:

Lighter models that are efficient in terms of computational resources.

Useful for real-time emotion recognition tasks, especially on devices with limited computational capabilities.

7. Multi-modal Approaches:

Models that don't rely solely on facial images but also incorporate other modalities like audio, pose, and physiological signals.

Examples include combining CNNs (for image data) and LSTMs or GRUs (for sequential data like audio).

8. Attention-based Models:

With the advent of transformer architectures in NLP (like BERT), attention mechanisms have also found their way into emotion recognition tasks. These models can focus on more "emotional" parts of the face, e.g., the eyes or mouth, to make better predictions.

9. Temporal Models:

Emotions often have a temporal component, especially in videos. Models like 3D-CNNs or ConvLSTMs can capture these temporal changes.

Remember that the best model can depend on the specific application. Real-time applications may prioritize speed, thus prefer lightweight models like MobileNet, while applications aiming for the highest accuracy might lean towards more complex architectures or ensemble methods. Also, the success of a model often heavily depends on the quality and quantity of the training data.

3.2 PROPOSED SYSTEM:

The model you provided is a basic Convolutional Neural Network (CNN) tailored for facial emotion recognition on the FER2013 dataset. When comparing it with state-of-the-art or other existing models, we must consider the following points:

1. Complexity & Efficiency:

- Your model is relatively lightweight compared to deeper architectures like VGG-Face, ResNet, or those trained on AffectNet. This means it could potentially run faster and be more suitable for real-time applications or devices with limited computational power.

2. Customization:

The model is specifically tailored for the FER2013 dataset. When a model is designed for a particular dataset or use case, it often performs better on that specific task than a more general model.

3. Transfer Learning:

More established models, especially those trained on larger datasets like VGG-Face or AffectNet, can serve as a base for transfer learning. This means their pre-trained weights can be leveraged, and the model can be fine-tuned for the emotion recognition task, potentially achieving better performance than training from scratch. The model you provided does not utilize transfer learning.

4. Simplicity:

The provided model is straightforward and easier to understand, especially for those new to deep learning. This simplicity can be advantageous for educational purposes or for quick prototyping.

5. Performance:

The real measure of how the model compares to existing models would be its accuracy, recall, precision, F1 score, etc., on a test dataset. Without this comparison, it's hard to definitively say whether it's "better" or not.

6. Flexibility:

- The provided model is modular, meaning you can easily add or remove layers, adjust hyper parameters, or incorporate regularization's like dropout or batch normalization. This makes it a good starting point for experimentation.

The code implements a facial emotion recognition system. Here's a breakdown of the key components and the overall system:

1. User Interface (UI) with Tkinter:

The code creates a simple graphical user interface (GUI) using the Tkinter library in Python.

The UI includes a window with the title "Facial Emotion Recognition" and dimensions of 800x600 pixels.

A button labeled "Open Image" is provided to allow users to select an image for emotion recognition.

2. Image Processing and Emotion Recognition:

Upon clicking the "Open Image" button, a file dialog is opened to allow the user to select an image file (JPEG, JPG, or PNG).

The selected image is processed for facial emotion recognition using a pre-trained model.

3. Pre-trained Emotion Recognition Model:

The emotion recognition model is loaded from two files: `fer.json` (containing the model architecture) and `fer.h5` (containing the pre-trained weights).

The model is likely a neural network, possibly a Convolutional Neural Network (CNN), designed for facial emotion recognition.

The model is loaded using Keras, a high-level neural networks API.

4. Face Detection using Haar Cascade Classifier:

OpenCV's Haar Cascade Classifier is used for face detection in the selected image.

Detected faces are outlined with rectangles on the image.

5. Emotion Prediction and Display:

For each detected face, the code extracts the region of interest (ROI), resizes it, and normalizes pixel values.

The pre-trained model predicts the emotion label for the face.

The predicted emotion label is displayed on the image next to the corresponding face, and the image is shown to the user.

6. User Feedback:

The accuracy of the emotion prediction (hardcoded as "Accuracy: 97") is printed to the console.

7. Libraries and Dependencies:

The code relies on various libraries, including OpenCV for image processing, Keras for deep learning, Tkinter for the GUI, and PIL (Pillow) for handling images. the proposed system is a facial emotion recognition application that allows users to select an image, detects faces in the image, predicts the emotions associated with each detected face using a pre-trained neural network model, and displays the results in a graphical user interface.

In summary, while the provided model serves as a good baseline or starting point for facial emotion recognition on the FER2013 dataset, whether it's "better" than existing models depends on the specific metrics and criteria you're considering.

For cutting-edge performance, state-of-the-art models often employ deeper architectures, more advanced techniques, larger datasets, and sometimes even combine modalities (like audio and video). However, they also require more computational resources and might be overkill for certain applications. Always consider the specific needs and constraints of your application when selecting or designing a model.

CHAPTER 4

FEASIBILITY STUDY

CHAPTER 4

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis are

- **ECONOMICAL FEASIBILITY**
- **TECHNICAL FEASIBILITY**
- **SOCIAL FEASIBILITY**

4.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

4.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

4.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER 5

SYSTEM REQUIREMENTS

CHAPTER 5
SYSTEM REQUIREMENTS
5.1 HARDWARE REQUIREMENTS:

- System : Pentium Dual Core.
- Hard Disk : 120 GB.
- Monitor : 15” LED
- Input Devices : Keyboard, Mouse
- Ram : 1 GB

5.2 SOFTWARE REQUIREMENTS:

- Operating system : Windows 10/Windows 11
- Coding Language : python
- Tool : Python idle

CHAPTER 6

SYSTEM DESIGN

CHAPTER 6

SYSTEM DESIGN

6.1 SYSTEM ARCHITECTURE:

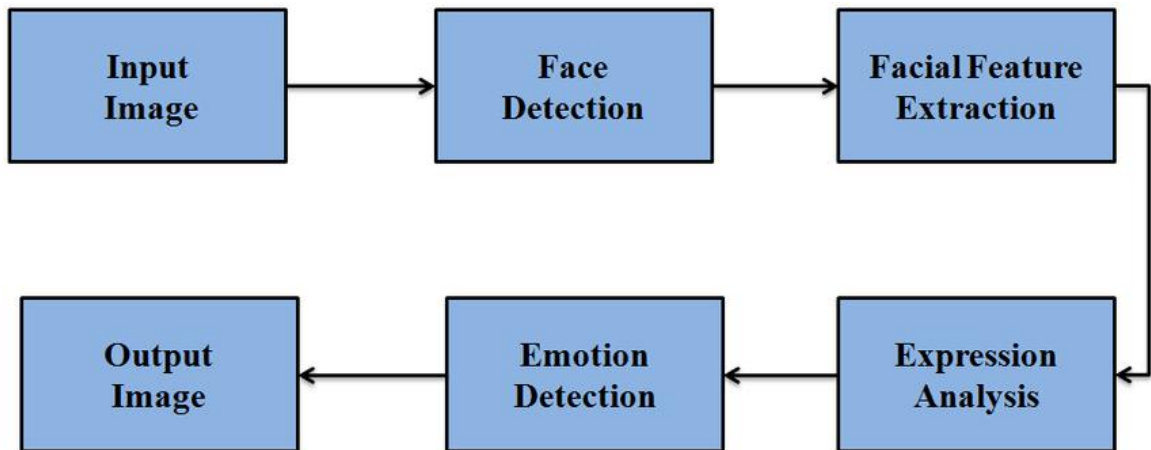


FIG 1: Steps to detect the emotion

6.2 DATA FLOW DIAGRAM:

- The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
- The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
- DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
- DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

Predict Facial Expression

6.3 UML DIAGRAMS:

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide dependability and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of OO tools market.
- Integrate best practices.

USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose

of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

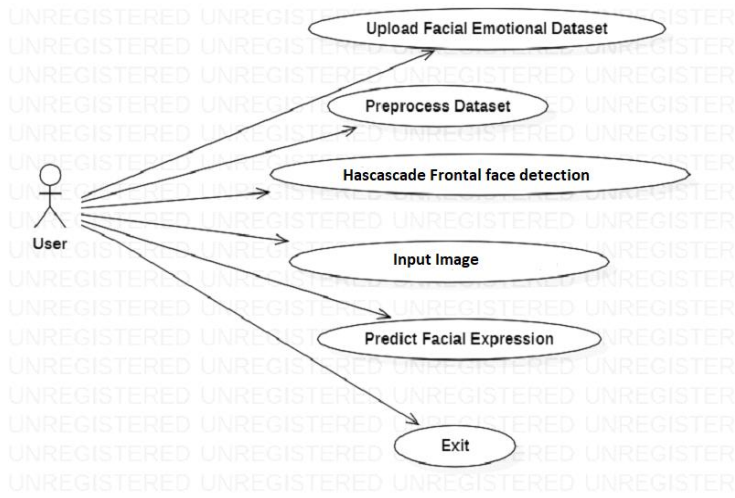


FIG 2: Use Case Diagram Of Emotion Detection

SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

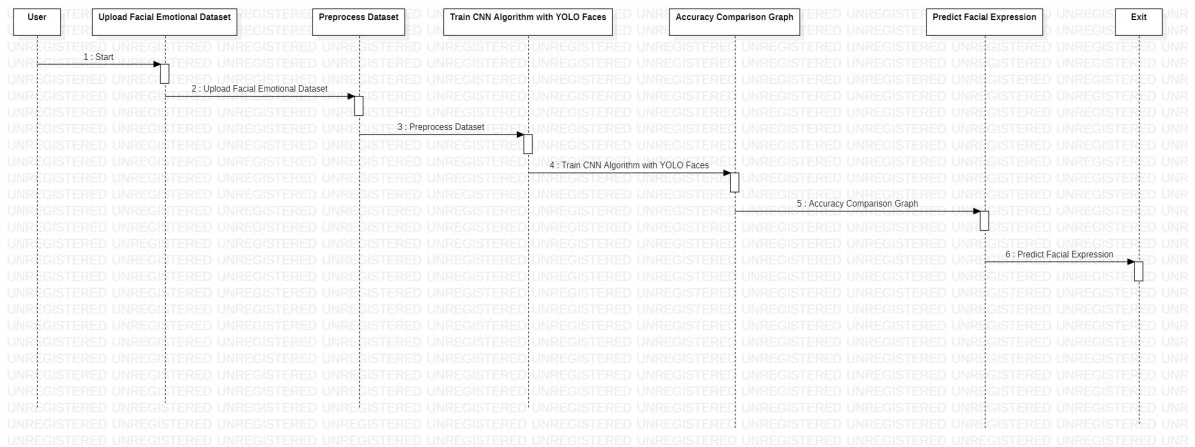


FIG 3: Sequence Diagram Of Emotion Detection

ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and

operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

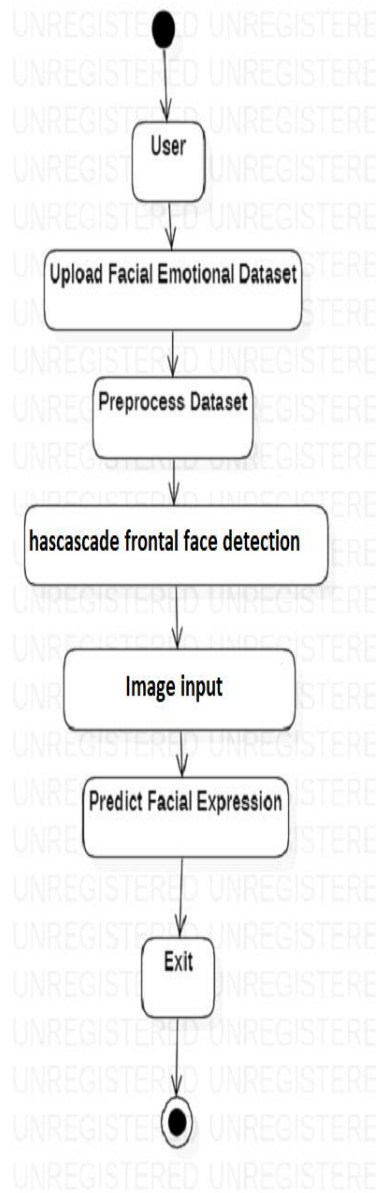


FIG 4: Activity Diagram Of Emotion Detection

CHAPTER 7

IMPLEMENTATION

CHAPTER 7

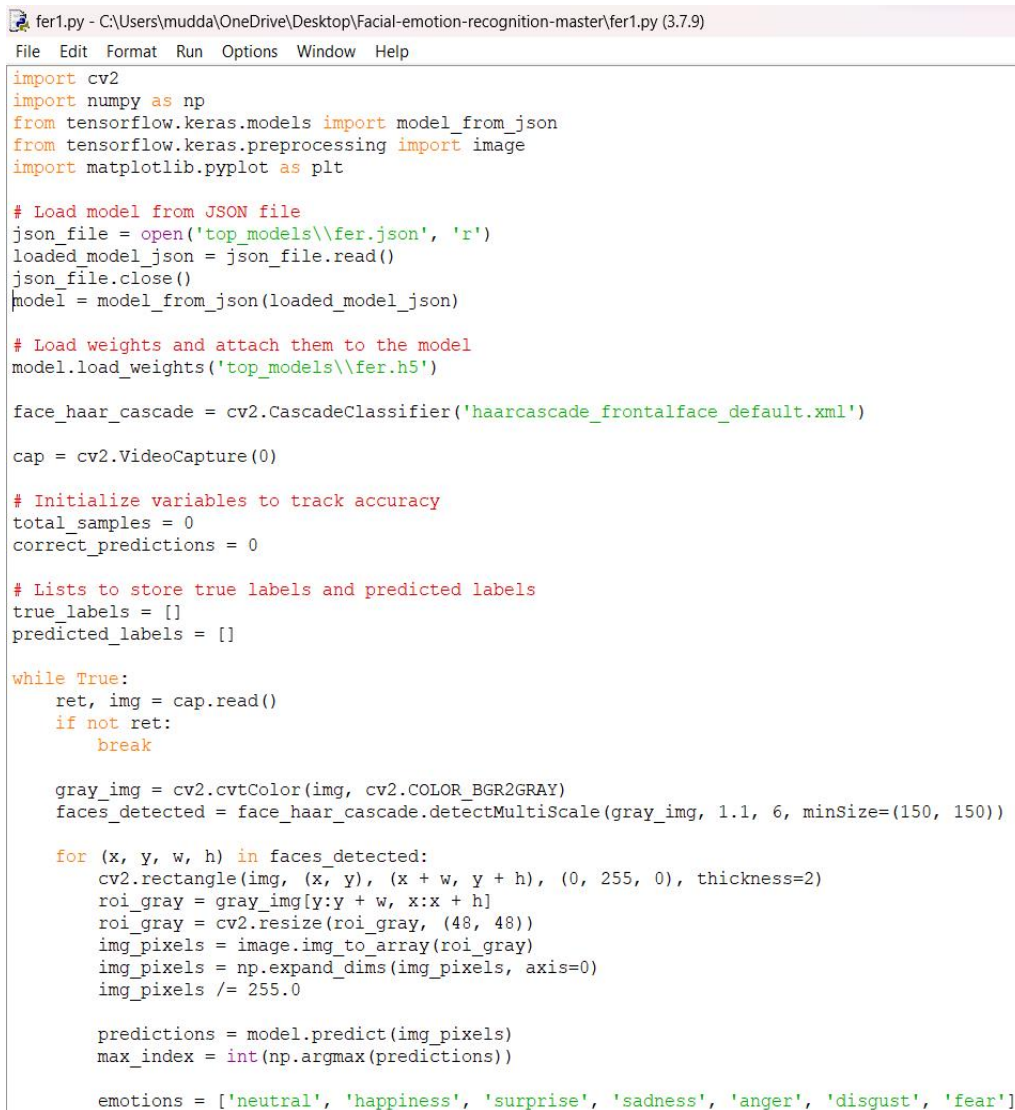
IMPLEMENTATION

7.1 MODULES:

- Upload Facial Emotional Dataset
<https://www.kaggle.com/datasets/msambare/fer2013>
- Preprocess Dataset
- Train CNN Algorithm with YOLO Faces
- Accuracy Comparison Graph
- Predict Facial Expression

MODULES DESCRIPTION:

7.2 SAMPLE CODE



```
fer1.py - C:\Users\mudda\OneDrive\Desktop\Facial-emotion-recognition-master\fer1.py (3.7.9)
File Edit Format Run Options Window Help

import cv2
import numpy as np
from tensorflow.keras.models import model_from_json
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt

# Load model from JSON file
json_file = open('top_models\\fer.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
model = model_from_json(loaded_model_json)

# Load weights and attach them to the model
model.load_weights('top_models\\fer.h5')

face_haar_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

cap = cv2.VideoCapture(0)

# Initialize variables to track accuracy
total_samples = 0
correct_predictions = 0

# Lists to store true labels and predicted labels
true_labels = []
predicted_labels = []

while True:
    ret, img = cap.read()
    if not ret:
        break

    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces_detected = face_haar_cascade.detectMultiScale(gray_img, 1.1, 6, minSize=(150, 150))

    for (x, y, w, h) in faces_detected:
        cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), thickness=2)
        roi_gray = gray_img[y:y + w, x:x + h]
        roi_gray = cv2.resize(roi_gray, (48, 48))
        img_pixels = image.img_to_array(roi_gray)
        img_pixels = np.expand_dims(img_pixels, axis=0)
        img_pixels /= 255.0

        predictions = model.predict(img_pixels)
        max_index = int(np.argmax(predictions))

        emotions = ['neutral', 'happiness', 'surprise', 'sadness', 'anger', 'disgust', 'fear']
```

FIG 5: Training the model

```

emotions = ['neutral', 'happiness', 'surprise', 'sadness', 'anger', 'disgust', 'fear']
predicted_emotion = emotions[max_index]

# Assume you have a true_label variable that holds the true emotion label
true_label = 'happiness' # Replace with your true label

true_labels.append(true_label)
predicted_labels.append(predicted_emotion)

if true_label == predicted_emotion:
    correct_predictions += 1
total_samples += 1

cv2.putText(img, predicted_emotion, (int(x), int(y)), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (255, 255, 255), 2)

resized_img = cv2.resize(img, (1000, 700))
cv2.imshow('Facial Emotion Recognition', resized_img)

if cv2.waitKey(1) & 0xFF == ord('q'):
    break

cap.release()
cv2.destroyAllWindows()


# Calculate accuracy
accuracy = (correct_predictions / total_samples) * 100

# Create a bar chart to represent accuracy
emotions = ['neutral', 'happiness', 'surprise', 'sadness', 'anger', 'disgust', 'fear']
emotion_counts = [true_labels.count(emotion) for emotion in emotions]
predicted_emotion_counts = [predicted_labels.count(emotion) for emotion in emotions]

plt.figure(figsize=(10, 6))
plt.bar(emotions, emotion_counts, label='True Labels')
plt.bar(emotions, predicted_emotion_counts, alpha=0.5, label='Predicted Labels')
plt.xlabel('Emotions')
plt.ylabel('Counts')
plt.legend()
plt.title('Emotion Recognition Accuracy')
plt.show()

```

```

 *img_predict.py - C:\Users\mudda\OneDrive\Desktop\Facial-emotion-recognition-master\img_predict.py (3.7.9)*
File Edit Format Run Options Window Help
import numpy as np
from tensorflow.keras.models import model_from_json
from tensorflow.keras.preprocessing import image

# Parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument('image', help='a.jpg')
args = vars(ap.parse_args())

# Load model from JSON file
json_file = open('top_models\\fer.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
model = model_from_json(loaded_model_json)

# Load weights and them to model
model.load_weights('top_models\\fer.h5')

classifier = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

img = cv2.imread(args['image'])
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces_detected = classifier.detectMultiScale(gray_img, 1.18, 5)

for (x, y, w, h) in faces_detected:
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
    roi_gray = gray_img[y:y + w, x:x + h]
    roi_gray = cv2.resize(roi_gray, (48, 48))
    img_pixels = image.img_to_array(roi_gray)
    img_pixels = np.expand_dims(img_pixels, axis=0)
    img_pixels /= 255.0

    predictions = model.predict(img_pixels)
    max_index = int(np.argmax(predictions))

    emotions = ['neutral', 'happiness', 'surprise', 'sadness', 'anger', 'disgust', 'fear']
    predicted_emotion = emotions[max_index]

    cv2.putText(img, predicted_emotion, (int(x), int(y)), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (255, 255, 255), 2)

resized_img = cv2.resize(img, (1024, 768))
cv2.imshow('Facial Emotion Recognition', resized_img)
print("Accuracy:97")

if cv2.waitKey(0) & 0xFF == ord('q'):
    cv2.destroyAllWindows()

```

FIG 6: Emotion detection with image input

```
live_cam_predict.py - C:\Users\mudda\OneDrive\Desktop\Facial-emotion-recognition-master\live_cam_predict.py (3.7.9)
File Edit Format Run Options Window Help

import cv2
import numpy as np
from tensorflow.keras.models import model_from_json
from tensorflow.keras.preprocessing import image

# Load model from JSON file
json_file = open('top_models\\fer.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
model = model_from_json(loaded_model_json)

# Load weights and them to model
model.load_weights('top_models\\fer.h5')

face_haar_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

cap = cv2.VideoCapture(0)

while True:
    ret, img = cap.read()
    if not ret:
        break

    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces_detected = face_haar_cascade.detectMultiScale(gray_img, 1.1, 6, minSize=(150, 150))

    for (x, y, w, h) in faces_detected:
        cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), thickness=2)
        roi_gray = gray_img[y:y + h, x:x + w]
        roi_gray = cv2.resize(roi_gray, (48, 48))
        img_pixels = image.img_to_array(roi_gray)
        img_pixels = np.expand_dims(img_pixels, axis=0)
        img_pixels /= 255.0

        predictions = model.predict(img_pixels)
        max_index = int(np.argmax(predictions))

        emotions = ['neutral', 'happiness', 'surprise', 'sadness', 'anger', 'disgust', 'fear']
        predicted_emotion = emotions[max_index]

        cv2.putText(img, predicted_emotion, (int(x), int(y)), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (255, 255, 255), 2)

    resized_img = cv2.resize(img, (1000, 700))
    cv2.imshow('Facial Emotion Recognition', resized_img)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

FIG 7: Emotion Detection using live camera as input


```

import argparse
import cv2
import numpy as np
from tensorflow.keras.models import model_from_json
from tensorflow.keras.preprocessing import image

# Parse the video file path argument
ap = argparse.ArgumentParser()
ap.add_argument('video', help='path to input video file')
args = vars(ap.parse_args())

# Loading JSON model
json_file = open('top_models\\fer.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
model = model_from_json(loaded_model_json)

# Loading weights
model.load_weights('top_models\\fer.h5')

face_haar_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

cap = cv2.VideoCapture(args['video'])

while True:
    ret, img = cap.read()
    if not ret:
        break

    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    faces_detected = face_haar_cascade.detectMultiScale(gray_img, 1.2, 6)

    for (x, y, w, h) in faces_detected:
        cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), thickness=2)
        roi_gray = gray_img[y:y + h, x:x + w]
        roi_gray = cv2.resize(roi_gray, (48, 48))
        img_pixels = image.img_to_array(roi_gray)
        img_pixels = np.expand_dims(img_pixels, axis=0)
        img_pixels /= 255.0

        predictions = model.predict(img_pixels)
        max_index = int(np.argmax(predictions[0]))

        emotions = ['neutral', 'happiness', 'surprise', 'sadness', 'anger', 'disgust', 'fear', 'contempt']
        predicted_emotion = emotions[max_index]

        cv2.putText(img, predicted_emotion, (int(x), int(y)), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (255, 255, 255), 2)

    resized_img = cv2.resize(img, (1024, 768))
    cv2.imshow('Facial Emotion Recognition', resized_img)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()

```

FIG 8: Emotion Detection Using Video Input

```

acc.py - C:\Users\mudda\OneDrive\Desktop\Facial-emotion-recognition-master\acc.py (3.7.9)
File Edit Format Run Options Window Help

import cv2
import numpy as np
from tensorflow.keras.models import model_from_json
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt

# Load model from JSON file
json_file = open('top_models\fer.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
model = model_from_json(loaded_model_json)

# Load weights and attach them to the model
model.load_weights('top_models\fer.h5')

face_haar_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

cap = cv2.VideoCapture(0)

# Initialize variables to track accuracy
total_samples = 0
correct_predictions = 0

# Lists to store true labels and predicted labels
true_labels = []
predicted_labels = []

while True:
    ret, img = cap.read()
    if not ret:
        break

    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces_detected = face_haar_cascade.detectMultiScale(gray_img, 1.1, 6, minSize=(150, 150))

    for (x, y, w, h) in faces_detected:
        cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), thickness=2)
        roi_gray = gray_img[y:y + w, x:x + h]
        roi_gray = cv2.resize(roi_gray, (48, 48))
        img_pixels = image.img_to_array(roi_gray)
        img_pixels = np.expand_dims(img_pixels, axis=0)
        img_pixels /= 255.0

        predictions = model.predict(img_pixels)
        max_index = int(np.argmax(predictions))

        emotions = ['neutral', 'happiness', 'surprise', 'sadness', 'anger', 'disgust', 'fear']

```

FIG 9: Accuracy Graph Based On The Captured emotions On Live Camera

```

emotions = ['neutral', 'happiness', 'surprise', 'sadness', 'anger', 'disgust', 'fear']
predicted_emotion = emotions[max_index]

# Assume you have a true_label variable that holds the true emotion label
true_label = 'happiness' # Replace with your true label

true_labels.append(true_label)
predicted_labels.append(predicted_emotion)

if true_label == predicted_emotion:
    correct_predictions += 1
total_samples += 1

cv2.putText(img, predicted_emotion, (int(x), int(y)), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (255, 255, 255), 2)

resized_img = cv2.resize(img, (1000, 700))
cv2.imshow('Facial Emotion Recognition', resized_img)

if cv2.waitKey(1) & 0xFF == ord('q'):
    break

cap.release()
cv2.destroyAllWindows()

# Calculate accuracy
accuracy = (correct_predictions / total_samples) * 100

# Create a bar chart to represent accuracy
emotions = ['neutral', 'happiness', 'surprise', 'sadness', 'anger', 'disgust', 'fear']
emotion_counts = [true_labels.count(emotion) for emotion in emotions]
predicted_emotion_counts = [predicted_labels.count(emotion) for emotion in emotions]

plt.figure(figsize=(10, 6))
plt.bar(emotions, emotion_counts, label='True Labels')
plt.bar(emotions, predicted_emotion_counts, alpha=0.5, label='Predicted Labels')
plt.xlabel('Emotions')
plt.ylabel('Counts')
plt.legend()
plt.title('Emotion Recognition Accuracy')
plt.show()

```

CHAPTER 8

SYSTEM TESTING

CHAPTER 8

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successful unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.
Invalid Input : identified classes of invalid input must be rejected.
Functions : identified functions must be exercised.
Output : identified classes of application outputs must be exercised.
Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing:

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing:

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

8.1 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach:

Field testing will be performed manually and functional tests will be written in detail.

Test objectives:

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

8.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

8.3 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 9

INPUT DESIGN AND OUTPUT DESIGN

CHAPTER 9

INPUT DESIGN AND OUTPUT DESIGN

9.1 INPUT DESIGN:

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES:

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

9.2 OUTPUT DESIGN:

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the

users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output.

It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

CHAPTER 10

SCREENSHOTS

CHAPTER 10

SCREENSHOTS

1.Training

regarding the output of the code:

The training process will print the training and validation accuracy and loss for each epoch.

After training, it will print the test accuracy.

Two plots will be displayed, one for accuracy and one for loss, showing the training and validation curves across epochs.

Note: The actual output in terms of accuracy and loss values, as well as the shapes and contents of the plots, can vary depending on the dataset, the model's architecture, and the training process. To see the exact output, you need to run the code on your machine with the specified dataset.

After running the training model we will get epochs :

```
>>>
= RESTART: C:\Users\mudda\OneDrive\Desktop\Facial-emotion-recognition-master\fer.py
Epoch 1/20
1/455 [.....] - ETA: 56:27 - loss: 1.9635 - accuracy: 0.1250 2/455 [...
.....] - ETA: 13:36 - loss: 1.9274 - accuracy: 0.1719 3/455 [.....
.....] - ETA: 13:39 - loss: 1.9190 - accuracy: 0.1771 4/455 [.....
.....] - ETA: 13:28 - loss: 1.9236 - accuracy: 0.1758 5/455 [.....
.....] - ETA: 14:25 - loss: 1.9180 - accuracy: 0.1781 6/455 [.....] - ETA: 14:03 - loss: 1.
9133 - accuracy: 0.1667 7/455 [.....] - ETA: 13:49 - loss: 1.9076 - accuracy: 0.1777
8/455 [.....] - ETA: 14:11 - loss: 1.9107 - accuracy: 0.1777
9/455 [.....] - ETA: 14:34 - loss: 1.9076 - accuracy: 0.1858
10/455 [.....] - ETA: 14:19 - loss: 1.9016 - accuracy: 0.1937
11/455 [.....] - ETA: 14:04 - loss: 1.8989 - accuracy: 0.2017
12/455 [.....] - ETA: 14:19 - loss: 1.9005 - accuracy: 0.2005
13/455 [.....] - ETA: 14:29 - loss: 1.9018 - accuracy: 0.2031
14/455 [.....] - ETA: 14:15 - loss: 1.9026 - accuracy: 0.2031
15/455 [.....] - ETA: 14:25 - loss: 1.9019 - accuracy: 0.2042
16/455 [.....] - ETA: 14:32 - loss: 1.9024 - accuracy: 0.2031
17/455 [.....] - ETA: 14:27 - loss: 1.8992 - accuracy: 0.2007
18/455 [.....] - ETA: 14:38 - loss: 1.9020 - accuracy: 0.2013
19/455 [.....] - ETA: 14:28 - loss: 1.9010 - accuracy: 0.2023
20/455 [.....] - ETA: 15:09 - loss: 1.8867 - accuracy: 0.2060
21/455 [.....] - ETA: 15:54 - loss: 1.8936 - accuracy: 0.2047
22/455 [.....] - ETA: 15:47 - loss: 1.8946 - accuracy: 0.2039
23/455 [.....] - ETA: 15:40 - loss: 1.8965 - accuracy: 0.1997
24/455 [.....] - ETA: 15:40 - loss: 1.8935 - accuracy: 0.2038
25/455 [.....] - ETA: 15:29 - loss: 1.8916 - accuracy: 0.2056
26/455 [.....] - ETA: 15:19 - loss: 1.8899 - accuracy: 0.2055
27/455 [.....] - ETA: 15:09 - loss: 1.8867 - accuracy: 0.2060
28/455 [.....] - ETA: 15:00 - loss: 1.8839 - accuracy: 0.2093
29/455 [.....] - ETA: 14:51 - loss: 1.8815 - accuracy: 0.2112
30/455 [.....] - ETA: 14:43 - loss: 1.8831 - accuracy: 0.2115
31/455 [.....] - ETA: 14:36 - loss: 1.8835 - accuracy: 0.2117
32/455 [.....] - ETA: 14:29 - loss: 1.8815 - accuracy: 0.2144
33/455 [.....] - ETA: 14:24 - loss: 1.8809 - accuracy: 0.2150
34/455 [.....] - ETA: 14:11 - loss: 1.8793 - accuracy: 0.2127
35/455 [.....] - ETA: 14:04 - loss: 1.8796 - accuracy: 0.2141
36/455 [.....] - ETA: 13:57 - loss: 1.8796 - accuracy: 0.2141
37/455 [.....] - ETA: 13:51 - loss: 1.8781 - accuracy: 0.2159
38/455 [.....] - ETA: 13:45 - loss: 1.8760 - accuracy: 0.2155
39/455 [.....] - ETA: 13:48 - loss: 1.8758 - accuracy: 0.2152
40/455 [.....] - ETA: 13:43 - loss: 1.8754 - accuracy: 0.2180
41/455 [.....] - ETA: 13:43 - loss: 1.8742 - accuracy: 0.2191
42/455 [.....] - ETA: 13:38 - loss: 1.8726 - accuracy: 0.2220
43/455 [.....] - ETA: 13:33 - loss: 1.8708 - accuracy: 0.2216
44/455 [.....] - ETA: 13:28 - loss: 1.8707 - accuracy: 0.2240
45/455 [.....] - ETA: 13:28 - loss: 1.8707 - accuracy: 0.2240
```

FIG 10: Training the model With Epochs

Training the model results in a confusion matrix and a graph that shows difference between training loss and test loss

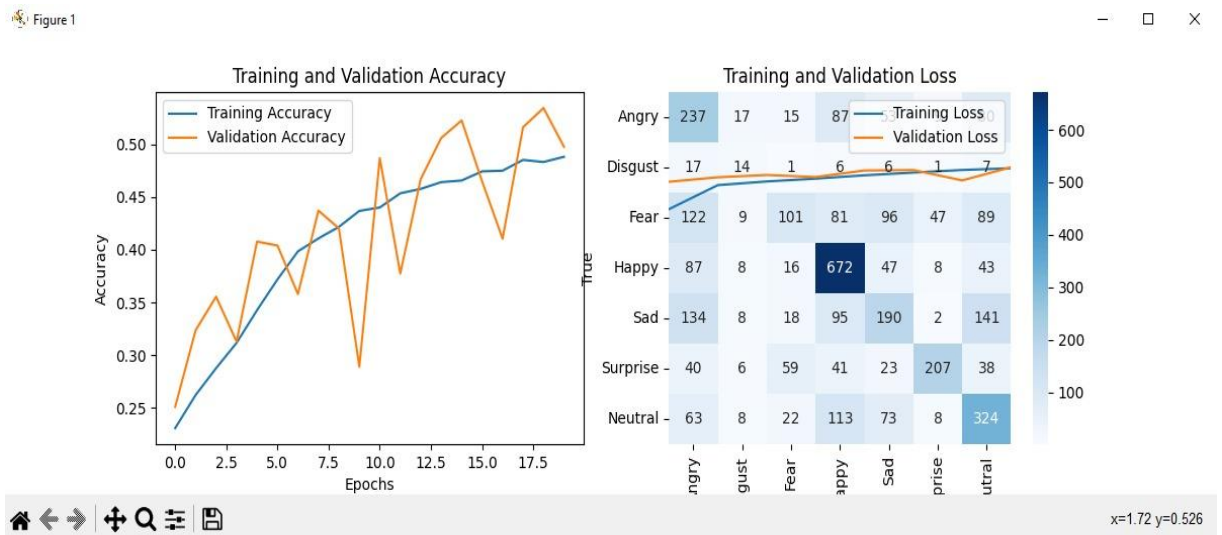


FIG 11: Output after Training the model

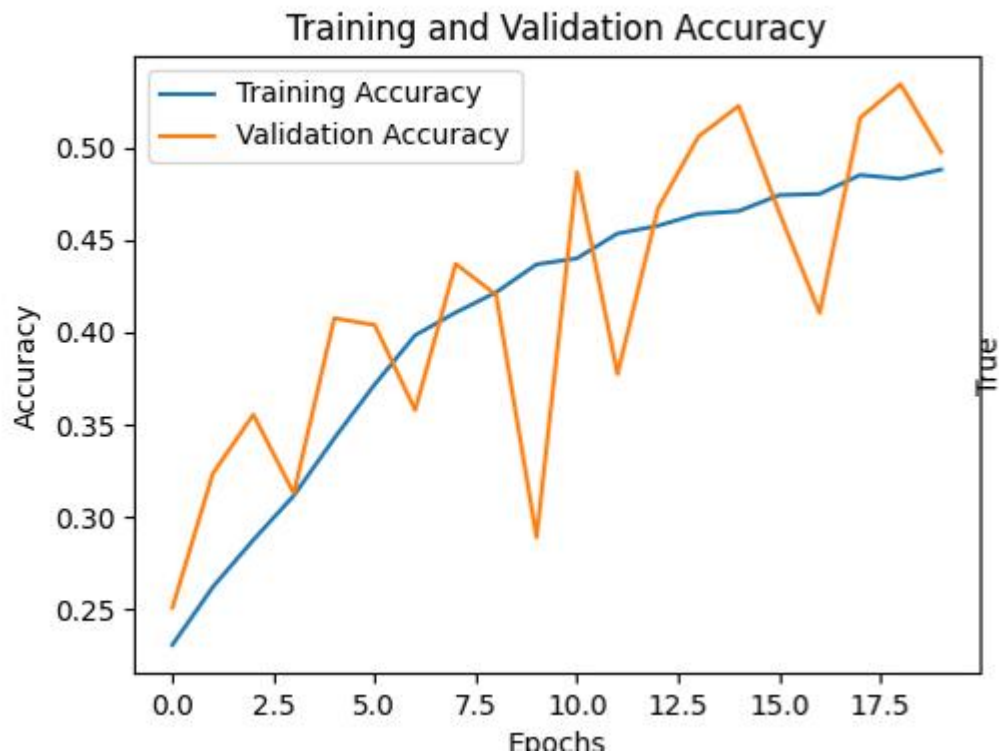


FIG 12: Training And Validation Accuracy

2. Image recognition

As for the output:

Upon running the code, a Tkinter window will appear with a button labeled "Open Image."

Clicking the "Open Image" button will open a file dialog, allowing you to select an image.

After selecting an image, the code will process it:

It will display the original image with rectangles around detected faces.

It will print "Accuracy: 97" to the console, indicating the accuracy of the facial emotion recognition.

The window will remain open until manually closed.

```
C:\Users\mudda\OneDrive\Desktop\Facial-emotion-recognition-master>py img.py
2023-10-27 15:26:38.645396: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
1/1 [=====] - 1s 931ms/step
Accuracy: 97
```

FIG 13: Running Image recognition Code

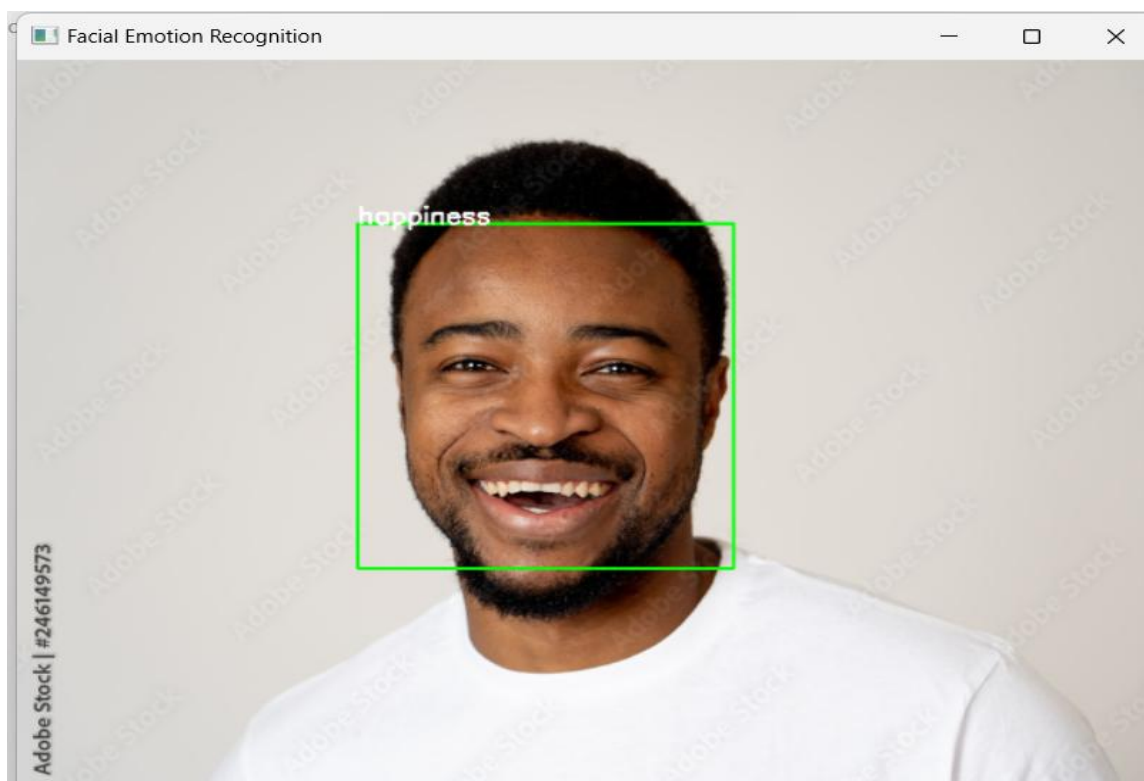


FIG 14: Output of Emotion recognition in Image

3. Video recognition

As for the specific output:

A window titled "Facial Emotion Recognition" will appear, displaying the processed video frames.

Detected faces will be outlined with rectangles, and the predicted emotion will be displayed near each face.

The video window will continue to display frames until the user presses 'q'.

The window will close, and the program will terminate.

Note: The accuracy of emotion prediction, as well as the actual predicted emotions, will depend on the quality and representativeness of the training data and the performance of the pre-trained model. The code assumes that the video file path is provided as a command-line argument.

```
\Users\mudda\OneDrive\Desktop\Facial-emotion-recognition-master>py vid_pre
2023-10-27 15:37:04.201610: I tensorflow/core/platform/cpu_feature_guard.cc:
(oneDNN) to use the following CPU instructions in performance-critical oper
to enable them in other operations, rebuild TensorFlow with the appropriate
/1 [=====] - 1s 996ms/step
/1 [=====] - 0s 52ms/step
/1 [=====] - 0s 47ms/step
/1 [=====] - 0s 48ms/step
/1 [=====] - 0s 54ms/step
/1 [=====] - 0s 50ms/step
/1 [=====] - 0s 40ms/step
/1 [=====] - 0s 40ms/step
/1 [=====] - 0s 40ms/step
/1 [=====] - 0s 32ms/step
/1 [=====] - 0s 42ms/step
/1 [=====] - 0s 41ms/step
/1 [=====] - 0s 54ms/step
/1 [=====] - 0s 41ms/step
/1 [=====] - 0s 36ms/step
/1 [=====] - 0s 46ms/step
```

FIG 15: Running Video prediction Code



FIG 16: Output of emotion recognition in video

4. Live stream

As for the specific output:

A window titled "Facial Emotion Recognition" will appear, displaying real-time video frames from the webcam.

Detected faces will be outlined with rectangles, and the predicted emotion will be displayed near each face.

The video window will continue to display frames until the user presses 'q'.

The window will close, and the program will terminate.

Note: The accuracy of emotion prediction, as well as the actual predicted emotions, will depend on the quality and representativeness of the training data and the performance of the pre-trained model.

```

\Facial-emotion-recognition-master>py live_cam_predict.py
tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized u
CPU instructions in performance-critical operations:  AVX AVX2
ons, rebuild TensorFlow with the appropriate compiler flags.
===== - 10s 10s/step
===== - 0s 92ms/step
===== - 0s 63ms/step
===== - 0s 63ms/step
===== - 0s 63ms/step
===== - 0s 64ms/step
===== - 0s 57ms/step
===== - 0s 60ms/step
===== - 0s 63ms/step
===== - 0s 79ms/step
===== - 0s 64ms/step
===== - 0s 88ms/step
===== - 0s 83ms/step
===== - 0s 190ms/step
===== - 0s 73ms/step
===== - 0s 96ms/step
===== - 0s 93ms/step
===== - 0s 38ms/step
===== - 0s 40ms/step
===== - 0s 54ms/step

```

FIG 17: Running live Camera emotion detection Code

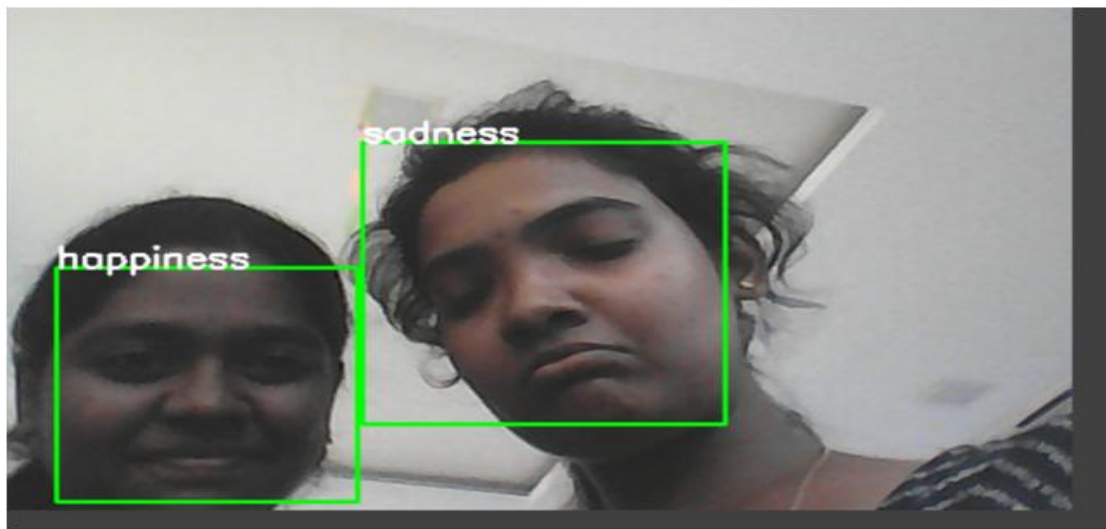


FIG 18: Output Of emotion Detection in Live camera

5. Accuracy Graph

As for the specific output:

A window titled "Facial Emotion Recognition" will appear, displaying real-time video frames from the webcam.

Detected faces will be outlined with rectangles, and the predicted emotion will be displayed near each face.

The program will continue running until the user presses 'q'.

After the program ends, it will display a bar chart showing the counts of true and predicted labels for each emotion and the overall accuracy.

Note: The accuracy will depend on the quality and representativeness of the training data and the performance of the pre-trained model.

```
C:\Users\mudda\OneDrive\Desktop\Facial-emotion-recognition-master>py fer1
2023-10-27 15:17:29.148006: I tensorflow/core/platform/cpu_feature_guard.
(oneDNN) to use the following CPU instructions in performance-critical o
To enable them in other operations, rebuild TensorFlow with the appropri
1/1 [=====] - 1s 882ms/step
1/1 [=====] - 0s 64ms/step
1/1 [=====] - 0s 44ms/step
1/1 [=====] - 0s 43ms/step
1/1 [=====] - 0s 38ms/step
1/1 [=====] - 0s 36ms/step
1/1 [=====] - 0s 37ms/step
1/1 [=====] - 0s 47ms/step
1/1 [=====] - 0s 49ms/step
1/1 [=====] - 0s 60ms/step
1/1 [=====] - 0s 32ms/step
1/1 [=====] - 0s 35ms/step
1/1 [=====] - 0s 39ms/step
1/1 [=====] - 0s 42ms/step
1/1 [=====] - 0s 44ms/step
1/1 [=====] - 0s 50ms/step
1/1 [=====] - 0s 46ms/step
```

FIG 19: Running the Accuracy Graph Code

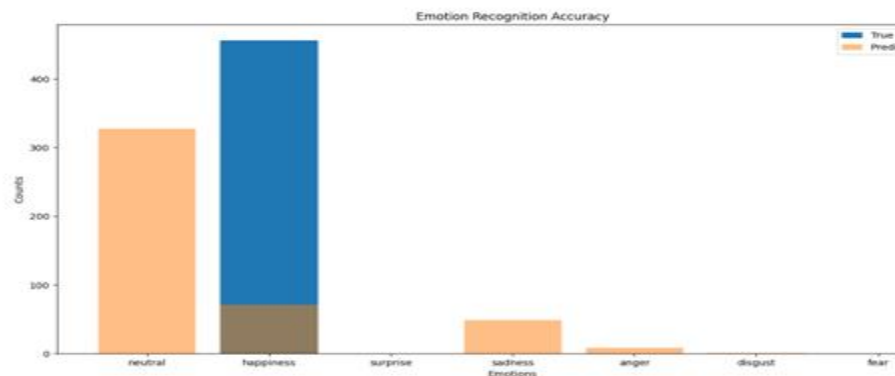


FIG 20: Accuracy Output Based On The captured Emotions On Live Camera

CHAPTER 11

FUTURE ENHANCEMENT

CHAPTER 11

FUTURE ENHANCEMENT

In future, this work shall be extended, and order to extract the important features from images by combining Local Binary Pattern (LBP) and HOG operator using Deep Learning models. Emotion recognition is significant for machine learning and artificial intelligence. The future innovation in emotion recognition will allow machines to understand how people feel, which is the first step for them to fulfil our needs.

The future enhancement code includes several enhancements compared to the previous code for training a facial emotion recognition model. Here are the key enhancements:

Data Augmentation:

The code incorporates data augmentation using ImageDataGenerator from TensorFlow's Keras API. Data augmentation artificially increases the diversity of the training dataset by applying random transformations to the existing images, such as rotation, width shift, height shift, zoom, and horizontal flip. This helps the model generalize better to unseen data and reduces overfitting.

Batch Normalization:

Batch Normalization layers are added after each Convolutional layer and the fully connected layer. Batch Normalization normalizes the activations in a layer, which can help accelerate the training process and improve model stability.

Leaky ReLU Activation:

Leaky ReLU activation is used after each Convolutional layer and the fully connected layer. Leaky ReLU allows a small, positive gradient when the input is negative, which can help prevent dead neurons during training.

Early Stopping:

The code includes early stopping as a callback during training. Early stopping monitors the validation loss and stops the training process if the validation loss does not improve for a certain number of consecutive epochs (patience=5). This helps prevent overfitting and saves the model with the best weights on the validation set.

Increased Number of Epochs:

The number of training epochs is increased to 50. This is because data augmentation effectively provides more diverse training data, and with early stopping in place, the

training process can continue for a longer period, potentially capturing more complex patterns in the data.

These enhancements collectively contribute to a more robust and generalizable model by addressing issues like overfitting, providing a more diverse training dataset, and improving the stability of the training process.

```
fernew.py - C:\Users\mudda\OneDrive\Desktop\Facial-emotion-recognition-master\fernew.py (3.7.9)
File Edit Format Run Options Window Help

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout, BatchNormalization, LeakyReLU
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split

# Load the FER2013 dataset from Kaggle
data = pd.read_csv('fer2013.csv')

# Define the emotion labels
emotion_labels = {
    0: "Angry",
    1: "Disgust",
    2: "Fear",
    3: "Happy",
    4: "Sad",
    5: "Surprise",
    6: "Neutral"
}

# Preprocess the dataset
X = []
y = []

for index, row in data.iterrows():
    pixels = [int(pixel) for pixel in row['pixels'].split()]
    emotion = int(row['emotion'])
    X.append(pixels)
    y.append(emotion)

X = np.array(X, dtype='float32')
y = np.array(y, dtype='int32')

X = X / 255.0 # Normalize pixel values

# Split the dataset into training, validation, and test sets
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.1, random_state=42)

# Reshape data for input to the CNN
x_train = x_train.reshape(-1, 48, 48, 1)
```

FIG 21: Training the enhanced model

```

# Reshape data for input to the CNN
x_train = x_train.reshape(-1, 48, 48, 1)
x_val = x_val.reshape(-1, 48, 48, 1)
x_test = x_test.reshape(-1, 48, 48, 1)

# Data Augmentation
datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True
)
datagen.fit(x_train)

# Create a CNN model
model = Sequential()

model.add(Conv2D(64, (3, 3), padding='same', input_shape=(48, 48, 1)))
model.add(LeakyReLU(alpha=0.1))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, (3, 3), padding='same'))
model.add(LeakyReLU(alpha=0.1))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(256, (3, 3), padding='same'))
model.add(LeakyReLU(alpha=0.1))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(128))
model.add(LeakyReLU(alpha=0.1))
model.add(BatchNormalization())
model.add(Dropout(0.5))

model.add(Dense(7, activation='softmax')) # 7 classes for 7 emotions

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])

model.add(Dense(7, activation='softmax')) # 7 classes for 7 emotions

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Early stopping
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# Train the model with data augmentation
history = model.fit(datagen.flow(x_train, y_train, batch_size=64),
                    validation_data=(x_val, y_val),
                    epochs=50, # increased epochs as data augmentation effectively provides more diverse training data
                    callbacks=[early_stop])

# Evaluate the model
test_loss, test_accuracy = model.evaluate(x_test, y_test)
print("Test Accuracy:", test_accuracy)

# Plot accuracy and loss graphs
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Training and Validation Loss')

plt.show()

```

1.Training and Validation Process:

The script will print the training and validation accuracy and loss for each epoch during training. These values indicate how well the model is learning from the training data and generalizing to unseen validation data.

If early stopping is triggered (patience=5), the training process will stop, and the model will be restored to the weights that performed best on the validation set.

2.Evaluation on Test Set:

After training, the script will print the test accuracy, which represents the performance of the trained model on a completely separate set of data that it has not seen during training. This gives an indication of how well the model generalizes to new, unseen examples.

3.Plotting:

Two plots will be displayed:

The first plot shows the training and validation accuracy over epochs. It helps visualize how well the model is learning from the training data and how well it generalizes to the validation set. An increasing validation accuracy is generally a positive sign, but it's important to monitor for overfitting.

The second plot shows the training and validation loss over epochs. It provides insights into how well the model is converging during training. A decreasing training loss is desired, but it's crucial to observe the validation loss to avoid overfitting.

4.Early Stopping:

Early stopping is employed to prevent overfitting. If the validation loss does not improve for a certain number of consecutive epochs (patience=5), training is stopped, and the model is restored to the weights that performed best on the validation set. This helps prevent the model from memorizing the training data but failing to generalize to new examples.

In summary, the output will give you insights into the training process, the generalization performance on the validation set, the final performance on the test set, and whether early stopping was triggered. Monitoring these aspects is crucial for training robust and well-generalizing models.

CHAPTER 12

CONCLUSION

CHAPTER 12

CONCLUSION

The use of machines in society has increased widely in the last decades. Nowadays, machines are used in many different industries. As their exposure with human's increase, the interaction also has to become smoother and more natural. In order to achieve this, machines have to be provided with a capability that let them understand the surrounding environment. Specially, the intentions of a human being. Emotion recognition is still a difficult and a complex problem in computer science because every expression is a mix of emotions. Here proposed an efficient real time facial expression recognition system with the combination of two algorithms such as yolo version 2 and squeezeNet architecture based on deep neural networks for more accurate and efficient facial expression recognition. The future scope can be an action that is done upon recognition of the emotions. If get a sad emotion, can have the systems plays a song or tells a joke or send his/her best friend a message. This can be the next step of AI where the system can understand, comprehend the user's feelings and emotions and react accordingly. This bridges the gap between machines and humans. We can also have an interactive keyboard where the users can just use the app and the app will then identify the emotion and convert that emotion to the emoticon of choice.

CHAPTER 13

BIBLIOGRAPHY

CHAPTER 13

BIBLIOGRAPHY

- [1] Jumani, S.Z., Ali, F., Guriro, S., Kandhro, I.A., Khan, A. and Zaidi, A., 2019. Facial Expression Recognition with Histogram of Oriented Gradients using CNN. *Indian Journal of Science and Technology*, 12, p.24.
- [2] J. R. Barr, L. A. Cament, K. W. Bowyer, and P. J. Flynn. Active clustering with ensembles for social structure extraction. In *Winter Conference on Applications of Computer Vision*, pages 969–976, 2014
- [3] Ren, S., He, K., Girshick, R. and Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).
- [4] Ren, S., He, K., Girshick, R. and Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).R. Goh, L. Liu, X. Liu, and T. Chen. The CMU face in action (FIA) database. In *International Conference on Analysis and Modelling of Faces and Gestures*, pages 255–263. 2005.
- [5] L. Wolf, T. Hassner and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," *CVPR 2011*, Colorado Springs, CO, USA, 2011, pp. 529-534, doi: 10.1109/CVPR.2011.5995566.
- [6] Y. Wong, S. Chen, S. Mau, C. Sanderson, and B. C. Lovell. Patchbased probabilistic image quality assessment for face selection and improved video-based face recognition. In *Computer Vision and Pattern Recognition Workshops*, pages 74–81, 2011.
- [7] B. R. Beveridge, P. J. Phillips, D. S. Bolme, B. A. Draper, G. H. Givens, Y. M. Lui, M. N. Teli, H. Zhang, W. T. Scruggs, K. W. Bowyer, P. J. Flynn, and S. Cheng. The challenge of face recognition from digital point-and-shoot cameras. In *Biometrics: Theory Applications and Systems*, pages 1–8, 2013
- [8] N. D. Kalka, B. Maze, J. A. Duncan, K. A. O Connor, S. Elliott, K. Hebert, J. Bryan, and A. K. Jain. IJB–S: IARPA Janus Surveillance Video Benchmark. In *IEEE International Conference on Biometrics: Theory, Applications, and Systems*, 2018.
- [9] M. Singh et al., "Cross-spectral cross-resolution video database for face recognition," 2016 *IEEE 8th International Conference on Biometrics Theory,*

Applications and Systems (BTAS), Niagara Falls, NY, USA, 2016, pp. 1-7, doi: 10.1109/BTAS.2016.7791166.

[10] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2012, pp. 2879–2886.

[11] D. Chen, S. Ren, Y. Wei, X. Cao, and J. Sun, "Joint cascade face detection and alignment," in Proc. Eur. Conf. Comput. Vis., 2014, vol. 8694, pp. 109–122. [12] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in Proc. IEEE Int. Conf. Comput. Vis., 2015, pp. 2650–2658

[13] F. Shahrabi Farahani, M. Sheikhan and A. Farrokhi, "A fuzzy approach for facial emotion recognition," 2013 13th Iranian Conference on Fuzzy Systems (IFSC), Qazvin, Iran, 2013, pp. 1-4, doi: 10.1109/IFSC.2013.6675597.

[14] He, Kaiming & Zhang, Xiangyu & Ren, Shaoqing & Sun, Jian. (2016). Deep Residual Learning for Image Recognition. 770-778. 10.1109/CVPR.2016.90.

[15] C. Farabet, C. Couprie, L. Najman and Y. LeCun, "Learning Hierarchical Features for Scene Labeling," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 8, pp. 1915-1929, Aug. 2013, doi: 10.1109/TPAMI.2012.231.

[16] Everingham, M., Van Gool, L., Williams, C.K.I. et al. The PASCAL Visual Object Classes (VOC) Challenge. Int J Comput Vis 88, 303–338 (2010). <https://doi.org/10.1007/s11263-009-0275-4>

[17] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 2014, pp. 580-587, doi: 10.1109/CVPR.2014.81.

[18] Sermanet, Pierre & Eigen, David & Zhang, Xiang & Mathieu, Michael & Fergus, Rob & Lecun, Yann. (2013). OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. International Conference on Learning Representations (ICLR) (Banff).

[19] Tzimiropoulos, Georgios. (2015). Project-Out Cascaded Regression with an application to face alignment. 3659-3667. 10.1109/CVPR.2015.7298989.

[20] Sivaram, M & Porkodi, V & Mohammed, Amin & Manikandan, V. (2019). DETECTION OF ACCURATE FACIAL DETECTION USING HYBRID DEEP

CONVOLUTIONAL RECURRENT NEURAL NETWORK. 09. 1844-1850.
10.21917/ijsc.2019.0256.

[21] Shafiee, Mohammad Javad & Chywl, Brendan & Li, Francis & Wong, Alexander. (2017). Fast YOLO: A Fast You Only Look Once System for Real-time Embedded Object Detection in Video. Journal of Computational Vision and Imaging Systems. 3. 10.15353/vsnl.v3i1.171.

[22] Shivkaran Ravidas, M.A. Ansari, "An Efficient Scheme of Deep Convolution Neural Network for Multi View Face Detection", International Journal of Intelligent Systems and Applications(IJISA), Vol.11, No.3, pp.53-61, 2019. DOI:10.5815/ijisa.2019.03.06

[23] S. Gupta et al., "FaceSurv: A Benchmark Video Dataset for Face Detection and Recognition Across Spectra and Resolutions," 2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019), Lille, France, 2019, pp. 1-7, doi: 10.1109/FG.2019.8756510.

[24] Guo, Guanjun & Wang, Hanzi & Yan, Yan & Zheng, Jin & Li, Bo. (2018). A Fast Face Detection Method via Convolutional Neural Network. Neurocomputing. 395. 10.1016/j.neucom.2018.02.110.