Project Report on

Human sentiment Analysis Using Machine Learning And Image processing

Submitted in partial fulfillment for the award of the degree
of
BACHELOR OF ENGINEERING
in
ELECTRONICS & TELECOMMUNICATION ENGINEERING
BY

Kartik Mishra

Abhishek Gupta

Pravinkumar Chauhan

Under the Guidance of

Mrs. Aradhana Manekar

Deputy HOD, E&TC & Associate Professor

Thakur College of Engineering and Technology

A-Block, Thakur Educational Campus, Shyamnarayan Thakur Marg, Thakur Village, Kandivali East,

Mumbai, Maharashtra 400101

Year 2020-21

B.E.

E&TC

Project

Human sentiment Analysis Using Machine Learning And Image processing

2021-22

TCET
DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING (E&TC)
(Accredited by NBA for 3 years, 3rd Cycle Accreditation w.e.f. 1st July 2019)
Choice Based Credit Grading System with Holistic Student Development (CBCGS - H 2019)
Under TCET-Autonomy Scheme - 2019

Project Report on

Human sentiment Analysis Using Machine Learning And Image processing

Submitted in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS & TELECOMMUNICATION ENGINEERING

BY

Kartik Mishra

Pravinkumar Chauhan

Abhishek Gupta

Under the Guidance of

Mrs. Aradhana Manekar

Assistant Professor

Thakur College of Engineering and Technology

Address

Year 2021-22

tcet E&TC DYNAMIC ENGINEERS

TCET

DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING (E&TC)

(Accredited by NBA for 3 years, 3rd Cycle Accreditation w.e.f. 1st July 2019)

Choice Based Credit Grading System with Holistic Student Development (CBCGS - H 2019)

Under TCET-Autonomy Scheme - 2019

CERTIFICATE

This is to certify that Kartik Mishra, Pravinkumar Chauhan and Abhishek Gupta are the bonafide students of Thakur College of Engineering and Technology, Mumbai. They have successfully carried out the project titled "Human Sentiment Analysis Using Machine Learning And Image processing" in partial fulfillment of the requirement of B. E. Degree in Electronics and telecommunication of Mumbai University during the academic year 2021-22. The work has not been presented elsewhere for the award of any other degree or diploma prior to this.

Kartik Mishra **Pravinkumar Chauhan** **Abhishek Gupta**

Mrs. Aradhana Manaker
(Internal guide) (Internal Examiner) (External Examiner)

Dr. Payel Saha
(Project Co-ordinator) (HOD-EXTC)

Thakur College of Engineering & Technology
Kandivali (East), Mumbai - 400101

Date:

Place:

2.2.3 (a) Industry certificate for outhouse project

As per the format followed in industry but certificate must clearly state name of students and work done by them

ACKNOWLEDGEMENT

It would be unfair if we do not acknowledge the help and support given by Professors, students, friends etc.

We sincerely thank our guide Aradhana Manekar for her guidance and constant support and also for the stick to our backs. We also thank the project coordinators for arranging the necessary facilities to carry out the project work.

We thank the HOD, Dr.Payal Saha, Principal, Dr. B. K. Mishra and the college management for their support.

Kartik Mishra (A-67)

Pravinkumar Chauhan (A-68)

Abhishek Gupta (A-70)

ABSTRACT

Facial expressions reveal a great deal about a person's feelings. One of the most difficult aspects of interpersonal relationships is effectively reading facial emotions. Automatic emotion identification based on facial expression recognition has become a hot topic in domains including computer science, medicine, and psychology. For improved outcomes, HCI research communities employ an automated face expression recognition system. For the recognition of expressions in both static photos and real-time films, many feature extraction algorithms have been developed. Love, happiness, rage, fear, and sadness are all examples of human emotions. These photographs are quite diverse from one another, but they all show the same human emotion. In this study, we look into the prospect of utilizing machine learning to predict an image's emotion. These kind of predictions can be employed in applications such as automatic tag predictions for social media photographs. Websites, as well as a buyer purchasing a product from a store or through a social media platform, can be useful for determining product quality.

CONTENTS

List of Figure

List of tables

Chapter 1: INTRODUCTION

1.1 Importance of the Project

As we know emotions has been used by humans to display through their facial expressions. Though nothing is said verbal but there is more to understand about the messages and signs we send and receive through the use of nonverbal communication. Although humans can recognize facial expressions of their known without any difficulty but reliable expression recognition by machine is still a challenge. There have been several development in the past few years in terms of feature extraction, face detection methods used for expression classification. The input into our system is an image and then we use to predict the emotion. In order to get an enhanced image and to extract sum useful information out of it, the method of Image Processing can be used. It's very efficient way through which an image can be converted into its digital form to performing various operations on it. Techniques like CNN, RAFT is proved to be highly accurate to determine emotions from the image.

Facial emotion recognition is the process of detecting human emotions from expressions shown by a human through its facial muscles. This technology is becoming more and more accurate with time and will eventually be able to read emotions as well as our brains do.As with any Machine Learning problem, your results are only as good as your database, garbage in means garbage out.

1.2 Literature Survey

In [1], Kai-Biao He, Jing Wen, and Bin Fang introduced the Adaboost Algorithm for face detection, which is based on MB-LBP characteristics and skin colour segmentation. Face detection is based on the Adaboost algorithm in this paper, and instead of Haar-features, extracted MB-LBP characteristics are used to train the Adaboost classifier. Skin colour is also integrated with Adaboost to lower the probability of false alarms. To begin, a skin Gaussian model in Cg-Cr colour space is created, and some constraints are applied to obtain the candidates' faces.

In [2], Enrique Correa, Arnoud Jonker, Michxael Ozo, and Rob Stolk presented their Convolutional Neural Network-based emotion recognition article. A few hundred high-resolution photographs to tens of thousands of lesser images are used in this procedural. The size of the training dataset from FERC needs be raised from 9000 to 20000 photos in order to improve the accuracy of the emotions recognized. The findings are compared to those obtained using other approaches such as SVM and LVQ. It has a 90 percent joyful, 80 percent neutral, and 77 percent surprised accuracy.

In [3], Kartika Candra Kirana, Slamet Wibawanto, and Heru Wahyu Herwanto proposed using the Viola Jones Algorithm to detect emotions. Though the Viola Jones method is widely used for face detection, it is employed here for both face detection and emotion recognition. These procedures classify emotions using Russel's Circumple, which is more efficient in classifying emotions. This method has three stages: first, an image from a video is taken, then undesirable rectangular sections are removed, and last, the emotion in the image is detected. The prediction had a 74 percent accuracy rate.

In [4], Hernández-Pérez suggested a method that combined oriented FAST and rotated BRIEF (ORB) characteristics with facial expression-derived Local Binary Patterns (LBP) features. To begin, each image is subjected to a face identification algorithm to extract more useful characteristics. Second, the ORB and LBP features are extracted from the face region to boost computational speed; particularly, region division is used in a novel way in the classic ORB to prevent feature concentration. The characteristics are unaffected by changes in size, grayscale, or rotation. Finally, a Support Vector Machine is used to classify the collected characteristics (SVM). The suggested technique is put to the test on several challenging datasets, including the CK+, JAFFE, and MMI databases.

In [5], Zhang Qinhu proposes a paper that first introduces the self attention mechanism based on the residual network concept and calculates the relative importance of a location by calculating the weighted average of all location pixels, then introduces channel attention to be told completely different options on the channel domain, and generates channel attention to target the interactive options in a variety of channels. The accuracy of this study on the CK+ and FExR2013 datasets, respectively, is 97.89% and 74.15

In [6], Khaled merit & Abdelmalik have explained an additional way of processing and detecting facial expression, Discrete Wavelet Transform(DWT) for reducing processing time. In this emotions recognition system CNN scored highest 97.6%.

In [7], Dipika Raval have provided a technique for facial expression detection using Principal Components Analysis (PCA) and Facial Action Coding System (FACS)for facial muscles detection.

In [8], G.Kalaivani ,S.Sathyapriya ,Dr.D.Anitha have worked on different types of facial movements related only to mouth region is mentioned and briefly explained.

In [9], Nicu Sebe, Mxichael S. Lew, Ira Cohen, Ashutosh Garg, and Thomas S. Huang focused on recognizing emotions via a Cauchy nave based classifier. As an approach, use the Cauchy distribution as a model. for detecting emotions through facial expressions in sequences of video The Gaussian vs. Cauchy Taking a guess was used to conduct the experiments. select a random example from the training set and do an analysis initial categorization The Cauchy distribution was shown to be superior. The testing set was then classified using this information. Experiments that are dependent on the person and experiments that are independent of the person Experiments were carried out that proved unequivocally that showing the Cauchy-based classifier was much more effective better outcomes

1.3 Motivation

Motivation for Sentiment Analysis is two fold. Both consumers and producers highly value “customer's opinion” about products and services. Thus, Sentiment Analysis has a considerable effort from industry as well as academia.

1.4 Scope of the project

Social media is exploding and the amount of information available is unprecedented. In order to correctly understand the collective sentiment, we need to change the analysis paradigm. Instead of calculating and attribute a numerical value to sentiments, we will be better off if we assigned a spectrum of values to sentiments. The overall flavor of the documents will change from; Positive, Neutral or Negative, to a more comprehensive and colorful sentiment output.

1.5 Organization of the report

Research and development work related to the project was done on weekly bases. We were working on the training of the model and implementing different ML techniques to enhance the accuracy of the model. Along with these bi-weekly meetings were held with the project guide and feedback was shared with us for the project related work. Presentation related to business canvas, pitch presentation was also given. We also performed SWOC analysis for our project.

Chapter 2: Proposed Work

2.1 Problem Definition

In real life, people express their emotion on their face to show their psychological activities and attitudes in the interaction with other people. The primary focus of this project is to determine which emotion an input image that contains one facial emotion belongs to. Because human face is complex to interpret, emotion recognition can be specifically divided into classification of basic emotion and classification of compound emotion . For the goals of our project, the essential problem is to focus on the classification of 7 basic emotions.

Goal of the project is to gain good accuracy as it will help to gain usable backend support for future up gradation. Presence of a good hardware device like computer/laptop is required to run the software. Camera, webcam is essential to capture an image of the person whose sentiments has to be judged.

2.2 Data Flow Diagram

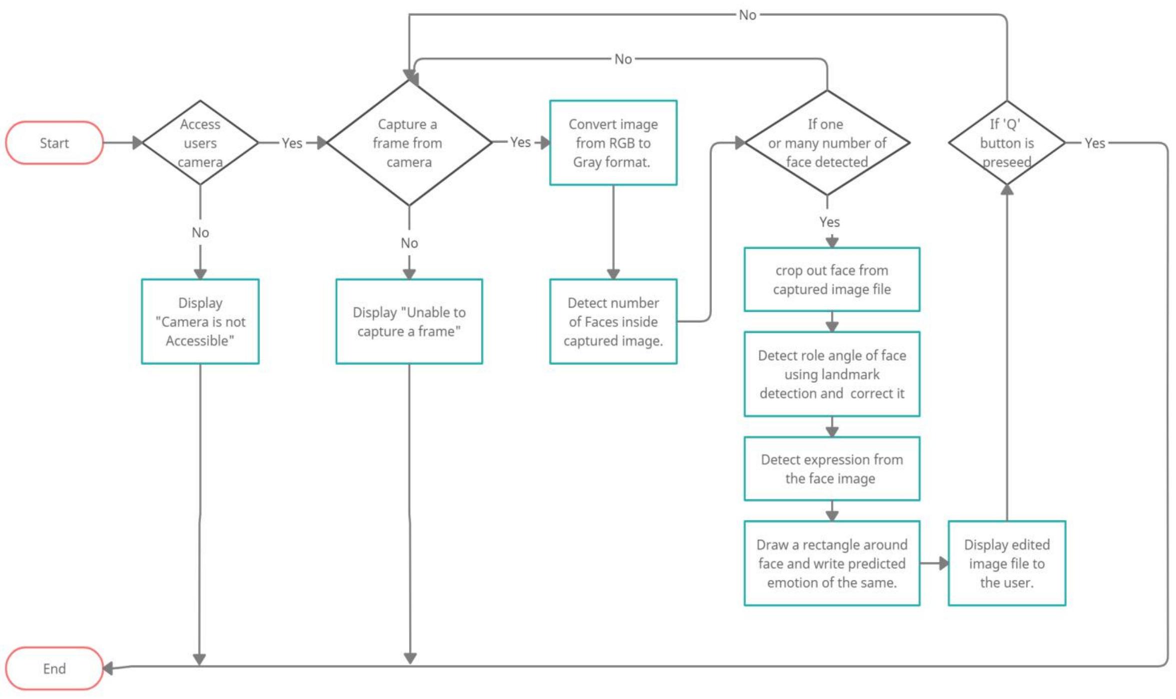

Start
Access users camera
Yes
No
Display "Camera is not Accessible"
Capture a frame from camera
Yes
No
Display "Unable to capture a frame"
Convert image from RGB to Gray format.
Detect number of Faces inside captured image.
If one or many number of face detected
No
Yes
crop out face from captured image file
Detect role angle of face using landmark detection and correct it
Detect expression from the face image
Draw a rectangle around face and write predicted emotion of the same.
Display edited image file to the user.
If 'Q' button is preseed
No
Yes
End

Figure 2.1. Data flow diagram of the Project

In the above Fig. 2.1, As shown in data flow diagram the first step is to take the input image from user, in that we are accessing the camera and taking the image from the user. The second step is the image processing part, we are converting the into gray scale provided by the user and detecting the landmark of the facial data, and by using landmark points get role angle of the face the we are correcting role of the face detected to increase the accuracy of the model and we are resizing the image by 48x48 to match the train model as the dataset consist of 48x48 size of image. The third is the CNN part that we are predicting the expression provided by the facial data and in the last part we are collecting the number of faces, location, and expression in the captured image, editing the information on the image captured and displaying the image using CSV.

CHAPTER 3: ANALYSIS AND PLANNING

3.1 Feasibility Study

STRENTHS

- In this project we have used A Large dataset And also Multiple Detection layer has been used
- to detect various human Sentiments. So this project Can Recognized multiple emotions.

- This project has been developed in a Stage wise manner. We have used multitple Layers os CNN to detect the emotions from hundred of images. Due to this It Gives a better accuracy.

- Currently, to Capture the output user has to give access to the device’s camera. In the Future scope This project can be integrated with Web application, so that a persons emotions can be detected through the web application

WEAKNESS

- To detect the sentiments from various Human faces so that its Accuracy can be increased we have used Hundreds of dataset images from various of database sources, due to this Process the project becomes heavy and its Performance can be decreased.

- To Capture the output it is Necessary to have all the files present in the device. This project is completely dependent on dataset , without the availability the dataset the project is useless.

- We are using Multiple dataset of images from Multiple sources, therefore the dataset is heavy. So to run this dataset a good Processor is needed.

CHALLENGES

- Instead of taking the dataset and code from the internet , we have Manually created the accurate dataset so that its accuracy can be increased, due to this it was little bit difficult to Make the project

- The Dataset which is generated through the execution is also available for the user
- The project is based upon Positive Emotions as well as Negative Emotions so We have segregated it into positive and negative emotions

OPPORTUNITIES

- This Project can be integrated with Hadoop Framework
- Using Hadoop Framework , the large data can be divided into many smaller parts so that it can processed in a very short time, so it can be upgraded with Hadoop for better processing time.
- Hadoop Enviroment is User friendly , so it would be easy to access the dataset and results in Hadoop ecosystem.

3.2 Project Planning

The first approach considered was Machine Learning, image recognition using machine learning was taken up and research was done on how it could be implemented for the project. Other alternatives were brought up to notice while researching, Deep Learning algorithm was one which could make our work easier and thus we finally selected it for our image classification problem.

A reliable and a generic dataset was required which will provide us with a wide range of various emotions over which the model could be trained. To make the training set more precise, some images were needed to be manually added so that the model would be well aware of the kind of day-to-day images the general audience might use while. Pre-processing was to be done over the final dataset as the images present could differ in dimensions, to maintain uniformity, the images were reshaped and resized.

The project would use Convolutional Neural Network for image classification. Convolutional Neural Network is a Deep Learning algorithm which can take in an input image, assign weights and biases to various aspects/objects in the image and then is able to differentiate among them. The Convolutional Neural Network model was created using Python's Keras framework and training and validation data were fitted on the model. To decide upon the correct selection of parameters hit and trail approach was to be done to select appropriate values for model. Various combinations of number of filters, activation functions and number of epochs were to do, and changes would be observed in validation accuracy and loss. Training Accuracy and Training Loss are metrics associated with model's performance with training dataset i.e., seen data, whereas Validation accuracy and validation loss refers to performance of model w.r.t. unseen data. The output of CNN model is an integer corresponding to label defined in training.

Real time operating Facial expression detection system was the parameter set for the projected development considering it's a must to have a skill to achieve the goals needed to be accomplished by the project. other parameters needed to be as good as possible were high accuracy and multiple facial expression detection for higher clarity of the information outcome.

Must to have parameters:

- Real time facial expression detection system
- Device and platform independent algorithm.
- Easy installation procedure.
- Environment error free detection of facial expression.

Parameters to be as high as possible:

- Accuracy of the model.
- Frequency of output of detected facial expression.
- Able to detect expressions on diverse target audience.

3.3 Scheduling

1. Model training phase has to be the first priority
    - Eight month solely for model training which was planned ass follows.
    - Three months where occupied with dataset gathering.
    - Two months for dataset filtering.
    - Three months for model training.
    - Five month were planned for application development.

- Three months width background error free facial capturing system.
- Two months for facial role correction (for higher output accuracy).
- Two months for GUI designing and real time facial detection representation.

2. In Fig. 3.1 we have explained our Scheduling Process, Nine months were devoted for research and development phase This phase was divided into consecutive repetition of multiple ML techniques implementation and model training and each technique was carefully sorted according to its performance on accuracy, frequency and reliability score's achieved on the real time facial expression detection application and sorted 800 facial expression dataset of 8 different facial expressions.

3. Three months where used to wrap up final selected trained model, selected best training technique (Convolution Neural Network) and merging it with final distribution executable for computer devices.

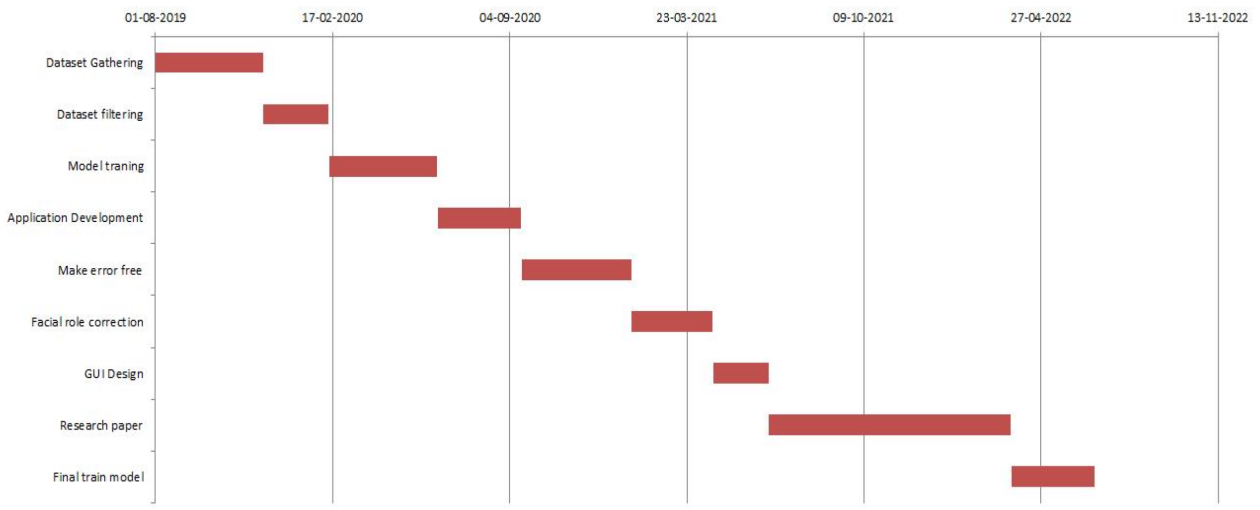


Figure 3.1: Scheduling Diagram of Project

CHAPTER 4: DESIGN

4.1 Technology

Jupyter Notebook, TensorFlow, Google Colab and Visual Studio Code Infrastructure will be used in development of application.

1. Jupyter Notebook

Jupiter Notebook is an open document format based on JSON. They have a complete record of user sessions and include code, descriptive text, equations and rich outputs. JupyterLab is a web-based interactive development environment for Jupiter notebooks, code and data. JupyterLab is compatible with: Configure and manage the user interface to support a wide range of workflows in data science, scientific computing and machine learning. JupyterLab Expandable and Modular: Write plugins that add new components and integrate with existing ones.

2. Tensorflow

TensorFlow offers several levels of abstraction so you can choose the one that best suits your needs. Build and train models using the advanced Keras API, making it easy to get started with TensorFlow and Machine Learning.

If you need more flexibility, an enthusiastic execution allows for instant repetition and intuitive debugging. For large ML training tasks, use the Distribution Strategy API for training delivered over different hardware configurations without changing the model definition.

3. Google Colab

Collaborative, or "Colab" for short, allows you to write and run Python in your browser

- Zero configuration required
- Free access to the GPU
- Easy sharing

We use Google Colab to train the model because training on the local machine Colab takes many times longer, 10 times less time.

4.2 Model Development

The Image processing model was implemented using Convolutional Neural Network model through Keras and TensorFlow in python. The basic idea of CNN is to extract low level and high- level features from training dataset, assign random weights and biases and eventually fine-tune and learn these values. These weights and biases along with suitable activation functions are stored in the model which is generated by batchwise training over train dataset. The model can classify for an unseen input data, generally referred as test data because of these values.

In Fig. 4.1 we have shown a General Convolution Operation. we created our own CNN architecture using Keras. The model type was Sequential, which allows to build a model layer by layer. Four convolutional layers along with pooling, flattening and dense layer. Gradient, color, dimensions, edges are few low-level features that are extracted by convolutional layer. The addition of more layers enables the model to capture high level features and hence a proper understanding of each image is established. Convolutional layer is accompanied by Pooling layer. There are two types of Pooling operations – increment in dimension and decrement in dimensions. A holistic and effective understanding of image dataset is achieved because pooling layer has the capability to learn high level rotational features.

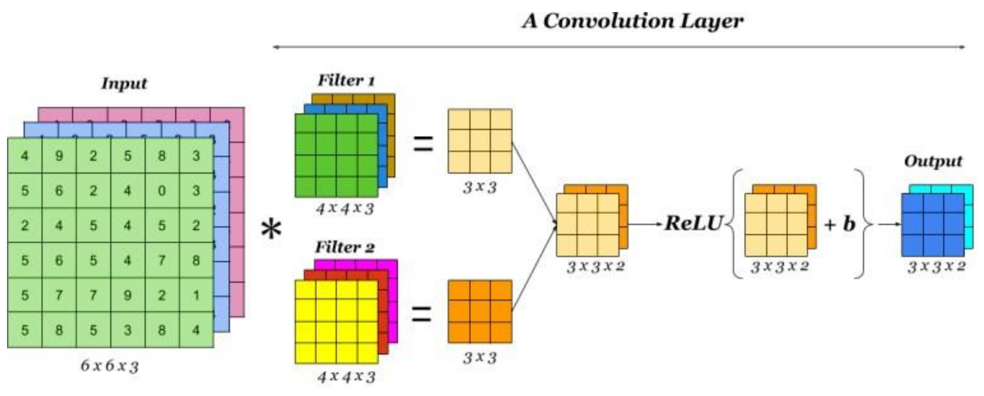


Figure 4.1. Diagram of General Convolution Operation

In Fig. 4.2 Max Pooling and Average Pooling operations is displayed. The max pooling operations results in maximum value from the product of filter and area of image covered by the filter. In contrast to this, average pooling method returns the mean of all the values obtained by convolution operation in each portion of image. The above process ensures that model has successfully learnt all features of training dataset. Now, the feature map is flattened and feed to a regular Artificial Neural Network (ANN) to perform classification operation. The ANN employs back propagation technique for each epoch. Eventually, the algorithm is capable enough to distinguish between features of input images and hence perform classification task for unseen data. The next process involved compilation of model using loss function, metrics, and optimizer. We selected 'adam' optimizer. The purpose of optimizer is to constantly adjust the learning rate throughout the training. The learning rate of model impacts the values of weights and biases, which eventually impacts the accuracy of model. The Fig. 3, is the output of total epoch cycle of the trained model and Fig. 4 and Fig. 5 are training and validation curves of loss and accuracy of same training session.

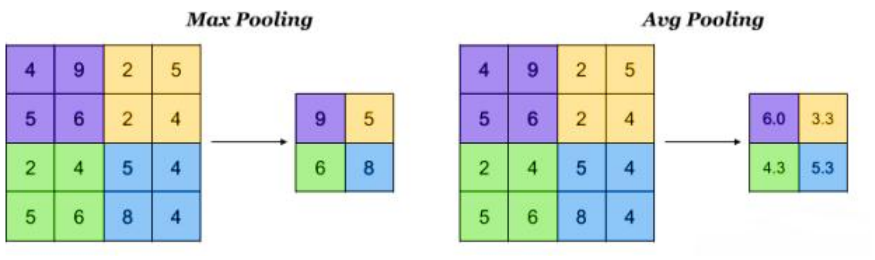
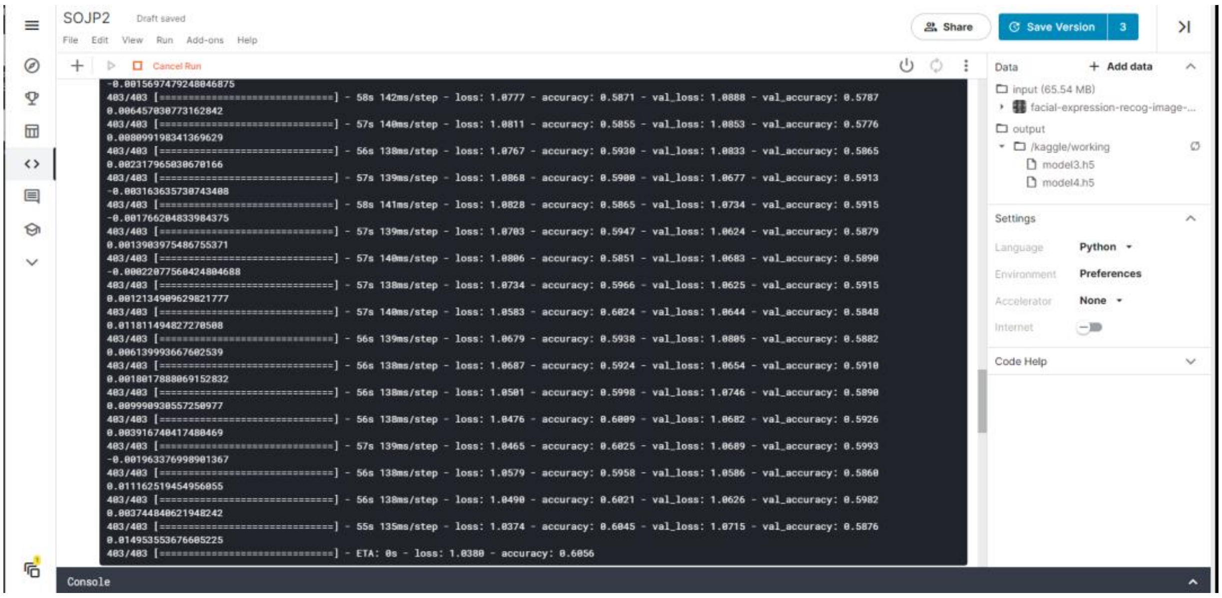


Figure 4.2. Pooling Operation



```
-0.0015697479248046875
403/403 [==============================] - 58s 142ms/step - loss: 1.0777 - accuracy: 0.5871 - val_loss: 1.0888 - val_accuracy: 0.5787
0.006457030773162842
403/403 [==============================] - 57s 140ms/step - loss: 1.0811 - accuracy: 0.5855 - val_loss: 1.0853 - val_accuracy: 0.5776
0.008099198341369629
403/403 [==============================] - 56s 138ms/step - loss: 1.0767 - accuracy: 0.5930 - val_loss: 1.0833 - val_accuracy: 0.5865
0.002317965030670166
403/403 [==============================] - 57s 139ms/step - loss: 1.0868 - accuracy: 0.5900 - val_loss: 1.0677 - val_accuracy: 0.5913
-0.003163635730743408
403/403 [==============================] - 58s 141ms/step - loss: 1.0828 - accuracy: 0.5865 - val_loss: 1.0734 - val_accuracy: 0.5915
-0.001766204833984375
403/403 [==============================] - 57s 139ms/step - loss: 1.0703 - accuracy: 0.5947 - val_loss: 1.0624 - val_accuracy: 0.5879
0.0013903975486755371
403/403 [==============================] - 57s 140ms/step - loss: 1.0806 - accuracy: 0.5851 - val_loss: 1.0683 - val_accuracy: 0.5890
-0.0002207756042480468
403/403 [==============================] - 57s 138ms/step - loss: 1.0734 - accuracy: 0.5966 - val_loss: 1.0625 - val_accuracy: 0.5915
0.0012134909629821777
403/403 [==============================] - 57s 140ms/step - loss: 1.0583 - accuracy: 0.6024 - val_loss: 1.0644 - val_accuracy: 0.5848
0.011811494827270508
403/403 [==============================] - 56s 139ms/step - loss: 1.0679 - accuracy: 0.5938 - val_loss: 1.0805 - val_accuracy: 0.5882
0.006139993667602539
403/403 [==============================] - 56s 138ms/step - loss: 1.0687 - accuracy: 0.5924 - val_loss: 1.0654 - val_accuracy: 0.5910
0.0018017888069152832
403/403 [==============================] - 56s 138ms/step - loss: 1.0501 - accuracy: 0.5998 - val_loss: 1.0746 - val_accuracy: 0.5890
0.009990930557250977
403/403 [==============================] - 56s 138ms/step - loss: 1.0476 - accuracy: 0.6009 - val_loss: 1.0682 - val_accuracy: 0.5926
0.003916740417480469
403/403 [==============================] - 57s 139ms/step - loss: 1.0465 - accuracy: 0.6025 - val_loss: 1.0689 - val_accuracy: 0.5993
-0.001963376998901367
403/403 [==============================] - 56s 138ms/step - loss: 1.0579 - accuracy: 0.5958 - val_loss: 1.0586 - val_accuracy: 0.5860
0.011162519454956055
403/403 [==============================] - 56s 138ms/step - loss: 1.0490 - accuracy: 0.6021 - val_loss: 1.0626 - val_accuracy: 0.5982
0.003744840621948242
403/403 [==============================] - 55s 135ms/step - loss: 1.0374 - accuracy: 0.6045 - val_loss: 1.0715 - val_accuracy: 0.5876
0.014953553676605225
403/403 [==============================] - ETA: 0s - loss: 1.0380 - accuracy: 0.6056
```

Figure 4.3. Epoch cycle of the training model

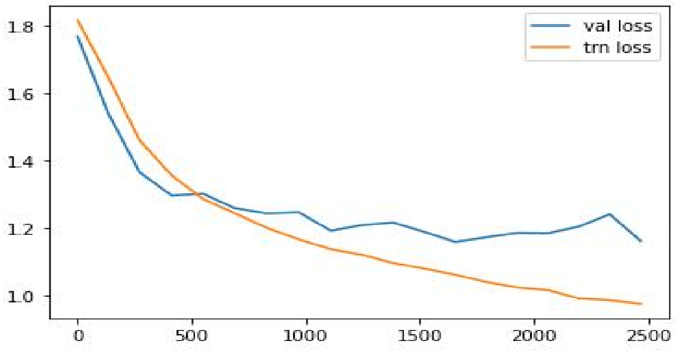


Figure 4.4. Training and Validation curve of loss

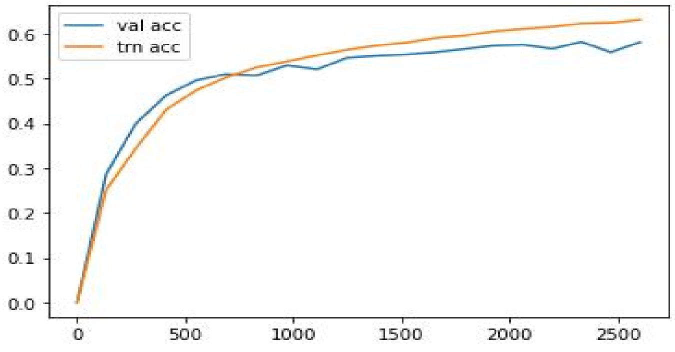


Figure 4.5. Training and Validation curve of accuracy

Keras Conv2D: Keras Conv2D is a two-dimensional convolution layer that generates a tensor of outputs by winding a convolution kernel with the layers input.

Kernel:In image processing, a kernel is a convolution matrix or masks that can be used to blur, sharpen, emboss, identify edges, and more by convolutioning a kernel with an image.

The Keras Conv2D class constructor has the following arguments:

```
keras.layers.Conv2D(
filters,
kernel_size,
strides=(1, 1),
padding="valid',
data_format=None,
dilation_rate= (1, 1),
activation=None,
use_bias=True,
kernel_initializer='glorot_uniform',
bias_initializer='zeros',
kernel_regularizer=None,
bias_regularizer=None,
activity_regularizer=None,
kernel_constraint=None,
bias_constraint=None
)
```

In below Fig. 4.6 different types of Layers has been shown like Input, Maxpooling & Output Layers.

Input layer : The input layer takes features from the user. It feeds the network information from the outside world; no computation is done here; nodes just pass the information (features) on to the hidden layer.

Hidden layer: This layer's nodes are not visible to the outside world; they are part of the abstraction offered by any neural network. The hidden layer computes all of the features entered through the input layer and sends the results to the output layer.

Output Layer: This layer communicates the network's learned information to the outside world.

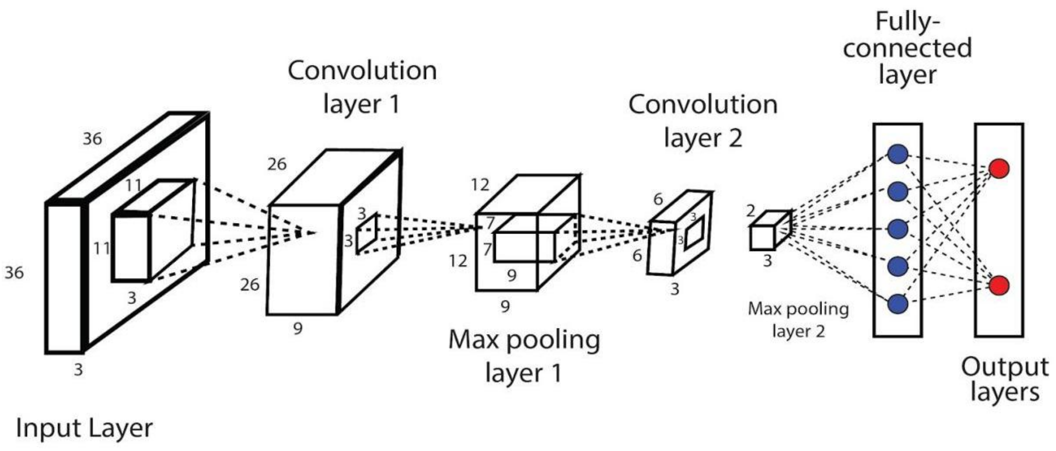


Figure 4.6. Diagram of Input, Hidden and Output layers

kernel size:The dimensions of the kernel are determined by this parameter. 11, 33, 55, and 77 are common dimensions that may be supplied as (1, 1), (3, 3), (5, 5), or (7, 7) tuples. It specifies the height and width of the 2D convolution window as an integer or a tuple/list of two numbers. This value has to be an odd number.

Stride: In below Fig. 4.7 Max Pool operation with stride has been shown. Stride is a feature of convolutional neural networks, which are neural networks optimised for image and video compression. Stride is a filter parameter in a neural network that controls the amount of movement in an image or video. When the stride of a neural network is set to 1, for example, the filter moves one pixel (or unit) at a time. Because the size of the filter has an impact on the encoded output volume, stride is usually set to a whole number rather than a fraction or decimal.

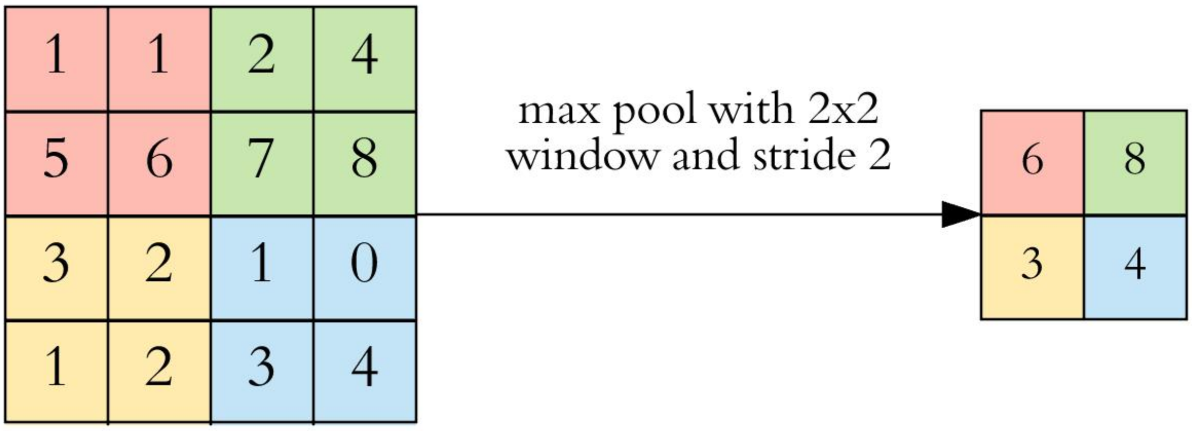


Figure 4.7. Diagram of MaxPooling on 4x4 matrix

Padding: In below Fig. 4.8 Padding operation has been performed on 4X4 Matrix. Padding is a term relevant to convolutional neural networks as it refers to the amount of pixels added to an image when it is being processed by the kernel of a CNN. For example, if the padding in a CNN is set to zero, then every pixel value that is added will be of value zero. If, however, the zero padding is set to one, there will be a one pixel border added to the image with a pixel value of zero.

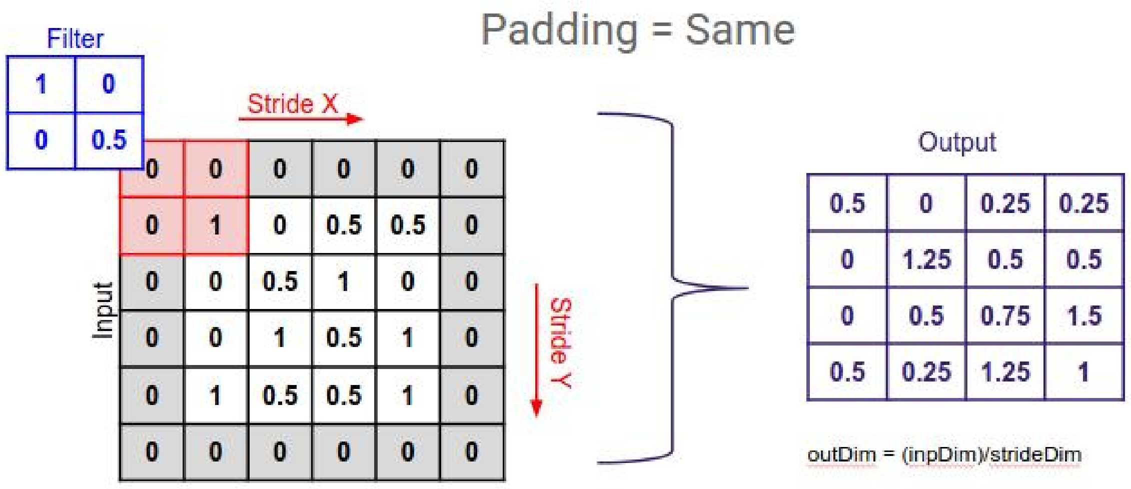


Figure 4.8.Diagram of Padding on 4x4 matrix

Data format: A string with one of the following values: channels last (default) or channels first. The dimensions in the inputs are ordered. Inputs with shape (batch size, height, width,

channels) go to channels last, whereas inputs with shape (batch size, channels, height, width) go to channels first. The image data format setting in your Keras config file at /.keras/keras.json is used by default. If you don't set it, it defaults to channels last.

Activation Function: In below Fig. 4.9 different types of activation functions has been explained. The activation function is the final component of the convolutional layer, and it increases the output non-linearity. In a convolution layer, the ReLu or Tanh function is commonly employed as an activation function.

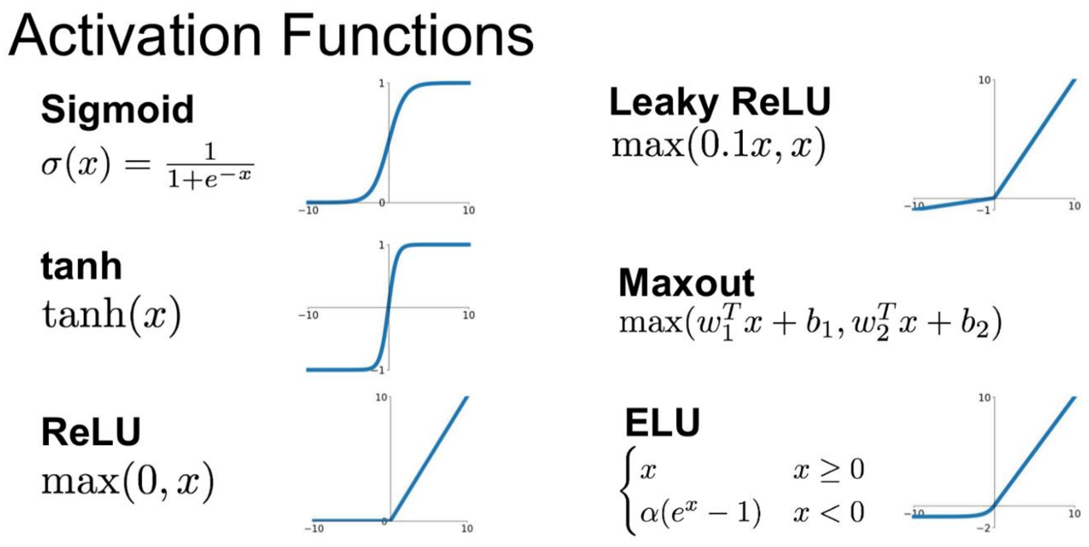


Figure 4.9. Diagram of Activation functions on neural network layers

Pooling Layer: In below Fig. 4.10 Pooling Techniques has been applied on 4x4 Matrix. To minimize the size of the input image, a pooling layer is employed. A convolutional layer is frequently followed by a pooling layer in a convolutional neural network. A pooling layer is generally implemented to speed up computation and improve the robustness of certain of the discovered characteristics. Kernel and stride are also used in the pooling procedure. The

2X2 filter is used in the example image below to pool the 4X4 input image with a stride of 2. Different methods of pooling exist. The most often used pooling methods in a convolutional neural network are max pooling and average pooling.

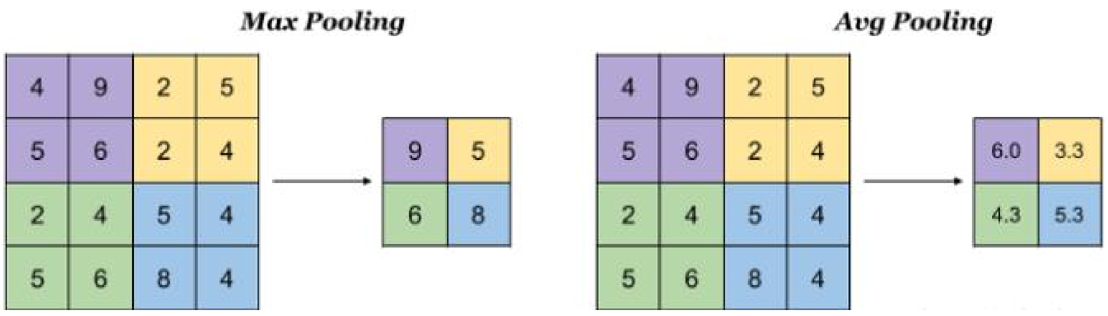


Figure 4.10. Images of various Pooling Techniques on 4x4 matrix

Max Pooling: Pooling that chooses the maximum element from the region of the feature map covered by the filter is known as max pooling. As a result, the output of the max-pooling layer would be a feature map with the most prominent features from the preceding feature map

Average Pooling : The average of the items present in the region of the feature map covered by the filter is computed using average pooling. As a result, while max pooling returns the most prominent feature in a feature map patch, average pooling returns the average of all features present in that patch.

Flattening: In below Fig. 4.11 Flattening has been applied on 3x3 Matrix. Flattening is the process of turning data into a one-dimensional array for use in the following layer. To construct a single lengthy feature vector, we flatten the output of the convolutional layers. It's also linked to the final categorization model, known as a fully-connected layer.

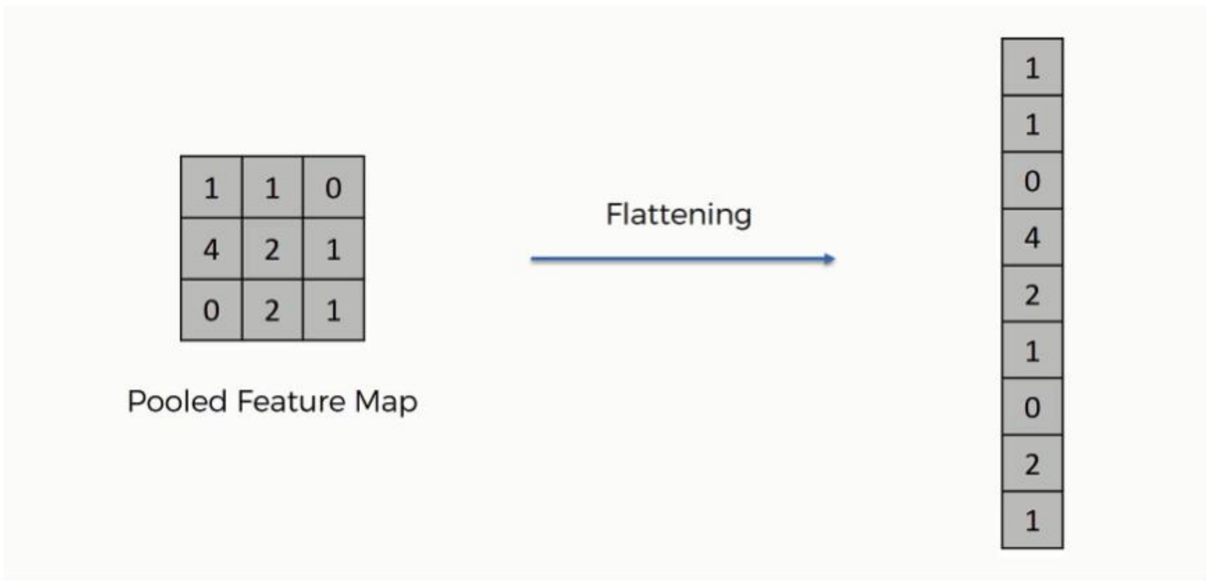


Figure 4.11. Image of Flattening layer on 3x3 matrix

Dense Layer: The typical deeply linked neural network layer is the dense layer. It is the most popular and often utilised layer. The following operation is performed on the input by the dense layer, and the output is returned. Dense provides the following operation: output = activation(dot(input, kernel) + bias), where activation is the element-wise activation function supplied as the activation parameter, kernel is the layer's weights matrix, and bias is the layer's bias vector (only relevant if use bias is True). Dense has all of these characteristics.

Batch Normalisation : In Fig. 4.12 Batch Normalization has been performed on 6x6 Matrix, Batch Norm, Layer Norm,Instance Norm & Group Norm has been perfomed. Batch normalization is a network layer that allows each layer to learn more independently. It's used to make the output of the preceding layers more natural. In normalisation, the activational scale the input layer. Learning becomes more efficient when batch normalisation is utilized, and it may also be used as a regularisation to prevent model over-fitting. To standardize the inputs or outputs, the layer is added to the sequential model. It may be utilized at numerous locations within the model's layers. It is frequently put immediately following the definition of the sequential model and before the convolution and pooling layers.

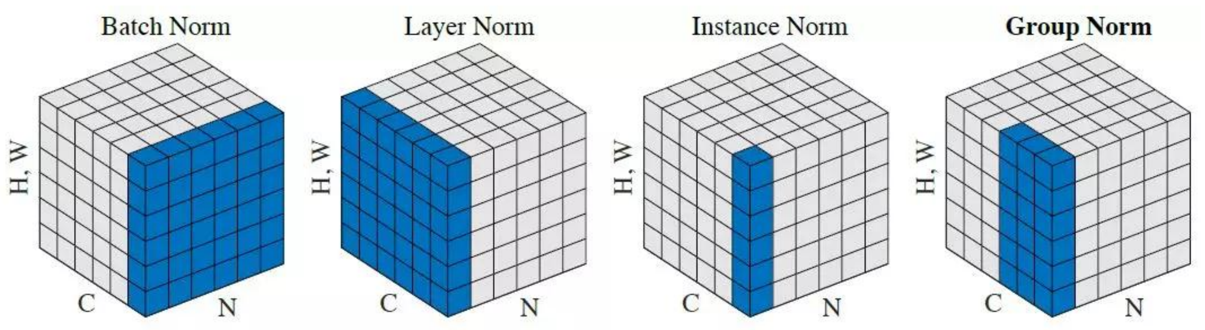


Figure 4.12. Images of Batch normalization on 6x6 matrix

Regularisation: In Fig. 4.13 An Example of Regularization has been shown. In machine learning, regularisation is widely used to reduce overfitting. Regularization approaches like DropBlock and Shake-Shake have shown to increase the generalisation performance of convolutional neural networks (CNNs). These systems, on the other hand, lack the potential to self-adapt throughout training. That is, the regularisation strength is set according to a predetermined timetable, and human modifications are necessary to accommodate different network designs.

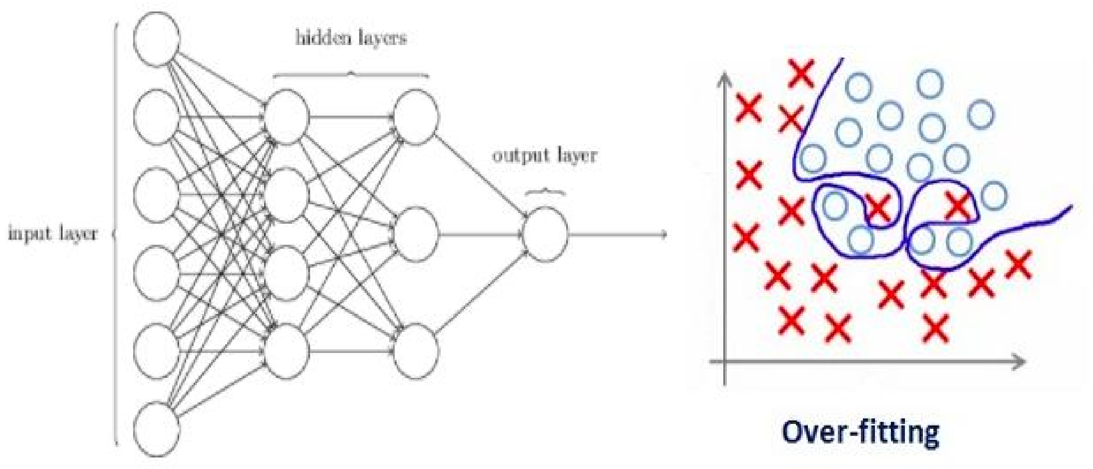


Figure 4.13. Example of Over-fitting on a neural network

Batch size: batch size is a gradient descent hyperparameter that determines how many training samples must be processed before the model's internal parameters are changed. The number of epochs is a gradient descent hyperparameter that determines how many full runs of the training dataset are made.For both the datasets, the best accuracy was achieved by the 1024 batch size, and the worst result was with the 16 batch size. The author stated that based on their results, the higher the batch size the higher the network accuracy, meaning that the batch size has a huge impact on the CNN performance.

Width shift range: The width shift range is a floating point value between 0.0 and 1.0 that indicates the upper bound of the percentage of the overall width by which the picture will be randomly moved left or right.

Epochs:An epoch is a term used in machine learning and indicates the number of passes of the entire training dataset the machine learning algorithm has completed. Datasets are usually grouped into batches (especially when the amount of data is very large). Some people use the term iteration loosely and refer to putting one batch through the model as an iteration.

If the batch size is the whole training dataset then the number of epochs is the number of iterations. For practical reasons, this is usually not the case. Many models are created with more than one epoch. The general relation where dataset size is d, number of epochs is e, number of iterations is i, and batch size is b would be d*e = i*b.

Determining how many epochs a model should run to train is based on many parameters related to both the data itself and the goal of the model, and while there have been efforts to turn this process into an algorithm, often a deep understanding of the data itself is indispensable When a whole dataset is only transported forward and backward through the neural network ONCE, it is called an Epoch. We break the epoch into numerous smaller batches since one epoch is too large to provide to the computer all at once.

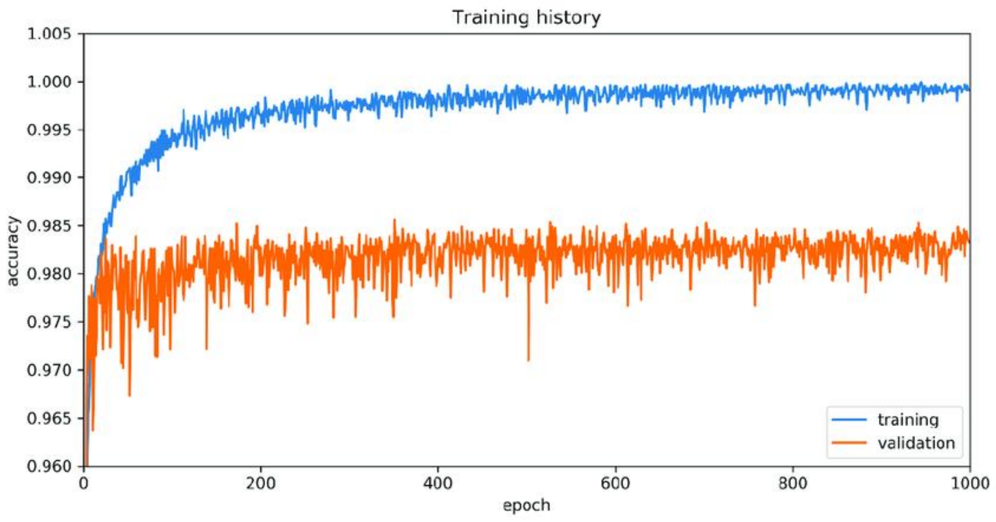


Figure 4.14. Example of Training and validation accuracy

Steps per epoch: In Fig. 4.14 We have shown a sample Image of Training & Validation accuracy and contains two graph lines, Orange one belongs to validation & Blue one belongs to training. steps per epoch is the number of samples to train per epoch. It specifies how many batches of samples should be utilized in a single epoch. It's used to signal the end of one era and the start of the next. You can disregard it if you have a fixed-size training set.

validation dataset : A validation dataset is a sample of data from your model's training that is used to evaluate model competence while tweaking the model's hyperparameters.

Sequence: A neural network takes in a variable number of sequence data and outputs a variable number of predictions in sequence modelling. Typically, the data is loaded into a recurrent neural network (RNN). Sequence models may be divided into four types: one-to-one (one input, one output), two-to-one (two inputs, two outputs), and four-to-four (four inputs, four outputs).

Dilation rate: The Conv2D class's dilation rate argument is a 2-tuple of integers that regulates the dilation rate for dilated convolution.

The fundamental convolution applied to the input volume with defined gaps is called Dilated Convolution. When working with higher quality photos and fine-grained features are crucial to you, or when building a network with fewer parameters, you can utilise this option.

kernel constraint and bias constraint: The Constraint function is applied to the kernel matrix via kernel constraint. The Constraint function is applied to the bias vector via bias constraint. A constraint is a requirement that the solution must meet in order to solve an optimization issue. Constraints are divided into three categories: equality constraints, inequality constraints, and integer constraints.You can use these options to limit the Conv2D layers.Unless you have a special purpose to confine the Con2D layers, these settings are normally left alone.

Groups: A positive integer indicating how many groups the input is divided into along the channel axis. Filters / groups filters are used to convolve each group independently. The output is the result of concatenating the results of all the groups along the channel axis. Both the input channels and the filters must be divided into groups.

kernel initializer: Before actually training the model, this option determines the initialization procedure, which is used to initialise all of the variables in the Conv2D class. It's the kernel weights matrix's initializer.

Bias initializer: The bias initializer, on the other hand, determines how the bias vector is actually initialised before the training begins.It is the bias vector's initializer.

Filters:Multiple filters are used in convolutional networks to slice through an image and map them one by one, learning distinct regions of an input picture. Consider a little filter that moves left to right over the image from top to bottom, seeking for, instance, a dark edge.

Bias initializer: The bias initializer, on the other hand, determines how the bias vector is actually initialized before the training begins. It is the bias vector's initializer.

Image Data Generator: The Image Data Generator is a Keras class that may be imported just like any other library object. The generator will go through your picture data and perform random changes to each image before passing it on to the model, ensuring that the model never sees the same image again during training.

TensorFlow: TensorFlow is a set of procedures for developing and training models in Python or JavaScript, as well as deploying them in the cloud, on-prem, in the browser, or on-device, regardless of the language. .TensorFlow is an end-to-end open source platform for machine learning. TensorFlow is a rich system for managing all aspects of a machine learning system; however, this class focuses on using a particular TensorFlow API to develop and train machine learning models.

The tensorflow. data API allows you to create complicated input pipelines out of reusable components TensorFlow APIs are arranged hierarchically, with the high-level APIs built on the low-level APIs. Machine learning researchers use the low-level APIs to create and explore new machine learning algorithms. In this class, you will use a high-level API named tf.keras to define and train machine learning models and to make predictions. Tensorflow .keras is the TensorFlow variant of the open-source Keras API.

NumPy: NumPy is the most important Python module for scientific computing. It is a Python library that includes a multidimensional array object, derived objects (such as masked arrays and matrices), and a variety of routines for performing fast array operations, such as mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation, and much more.

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

At the core of the NumPy package, is the ndarray object. This encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an ndarray will create a new array and delete the original. The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.

NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.

A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today's scientific/mathematical Python-based software, just knowing how to use Python's built-in sequence types is insufficient - one also needs to know how to use NumPy arrays. In Fig. 4.15 An Example of Neural Network predicting a numerical image has been shown as an example of layers in CNN.

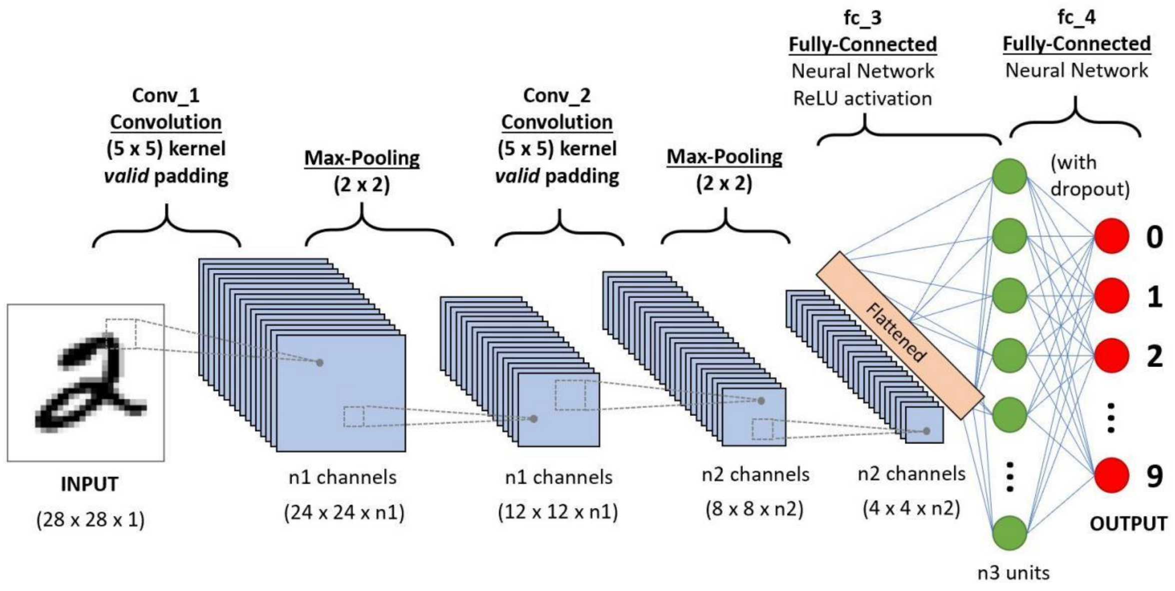


Figure 4.15. Neural Network predicting a numerical image

Matplotlib Pyplot: matplotlib. pyplot is a set of routines that allow matplotlib to behave similarly to MATLAB. Each pyplot function modifies a figure in some way, such as creating a figure, a plotting area in a figure, charting certain lines in a plotting area, decorating the plot with labels, and so on. Various states are retained throughout function calls in matplotlib.pyplot, allowing it to keep track of things like the current figure and plotting area, as well as directing plotting functions to the current axes (please note that "axes" here and in most places in the documentation refers to the axes part of a figure and not the strict mathematical term for more than one axis).

Sklearn: In Python, Scikit-learn (Sklearn) is the most usable and robust machine learning package. It uses a Python consistency interface to give a set of fast tools for machine learning and statistical modelling, such as classification, regression, clustering, and dimensionality reduction. NumPy, SciPy, and Matplotlib are the foundations of this Python-based toolkit.

Confusion Matrix: Looking at the confusion matrix is a much better technique to assess a classifier's performance. The basic concept is to keep track of how many times examples of class A are categorized as class B. For example, you may check at the 5th row and 3rd column of the confusion matrix to see how many times the classifier confused photos of 5s with images of 3s.

Dropout: Dropout is a strategy for preventing overfitting in a model. At each update of the training phase, the outgoing edges of hidden units (neurons that make up hidden layers) are set to 0 at random.

Real time emotion detection software.

In this part of the section working of the all the parts of the facial emotion detection software is explained in parts by parts. This software consists of different detection and filtering phases performed on a captured image explained below.

Capturing user facial image using OpenCV VideoCapture function

Python provides various libraries for image and video processing. One of them is OpenCV. OpenCV is a vast library that helps in providing various functions for image and video operations. With OpenCV, we can capture a video from the camera. It lets you create a video capture object which is helpful to capture videos through webcam and then you may perform desired operations on that video.

Open video file or a capturing device or a IP video stream for video capturing with API Preference. This is only a visual illustration of the haar feature traversal concept. The haar feature would traverse the image pixel by pixel in its actual work. The haar characteristics will also be applied in all feasible sizes.

Then video stream is used to chop user real time image into different images to perform all detection and prediction algorithm listed below. Also additional functions provided by openCV like rectangle and putText is used to print output on the image which has been captured. Representation the output the same chopped image data is edited and shown to the user for a high frame rate to create a sense of video being played. As shown in Fig. 4.16 a simple window is created with the help of openCV and the target users captured face is been represented in that window frame by frame to create a constant stream of real time facial features capturing system.

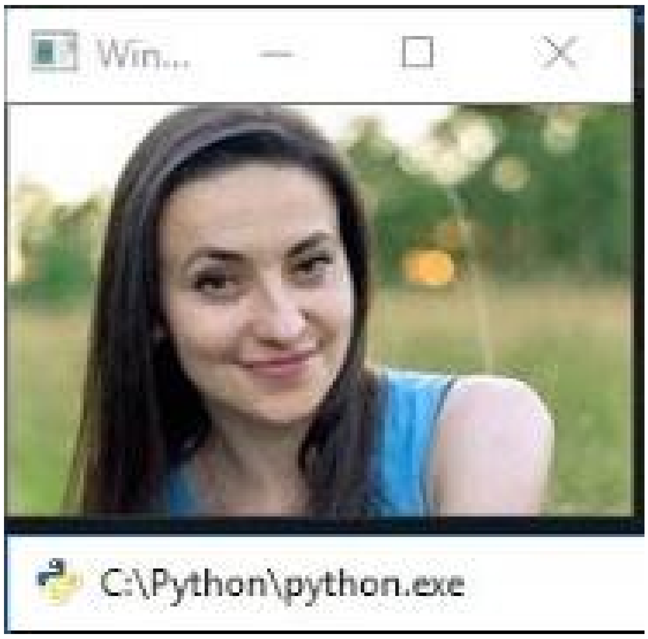


Figure 4.16. OpenCV image capturing and real time display

Steps to capture a video:

- Use cv2.VideoCapture() to get a video capture object for the camera.
- Set up an infinite while loop and use the read() method to read the frames using the above created object.
- Use cv2.imshow() method to show the frames in the video.
- Breaks the loop when the user clicks a specific key.

Facial detection using Haar Cascade frontal face detection algorithm.

Below Fig. 4.17 shows an example of Harcascade scanning with different kernels. Haar Cascade is a Object Detection Algorithm that detects faces in images and real-time videos. Viola and Jones proposed edge or line detection features in their research paper "Rapid Object Detection using a Boosted Cascade of Simple Features," published in 2001. To train on, the algorithm is given a large number of positive photos with faces and a large number of negative images with no faces. The model developed as a result of this training can be found on the OpenCV GitHub repository.

Face detection, eye detection, upper and lower body detection, licence plate detection, and other models are among them. A sample of Haar features used by Viola and Jones in their Original Research Paper.

The haar feature moves from the top left of the image to the bottom right in order to find a certain feature. This is only a visual illustration of the haar feature traversal concept. The haar feature would traverse the image pixel by pixel in its actual work. The haar characteristics will also be applied in all feasible sizes.

These are divided into three groups based on the attribute that each person is seeking for. The first pair of two rectangular features is in charge of determining whether the edges are horizontal or vertical (as shown above). The second set of three rectangle features is in charge of determining whether a lighter zone is flanked on either side by darker sections or vice versa. The final set of four rectangular characteristics is in charge of determining how pixel intensities fluctuate across diagonals.

To execute the same process, use the Integral Image. An Integral Image is created by multiplying each pixel in the Original Image by the sum of all pixels to its left and above. The accompanying GIF shows how to calculate a pixel in the Integral Image. The sum of all the pixels in the Original Image will be the last pixel in the Integral Image's bottom right corner.

Haar Cascade Detection is a sophisticated face detection technology that has been around for a long time. It has existed for a long time, long before Deep Learning became popular. Haar Features were employed to recognise faces as well as eyes, lips, licence number plates, and other features. The models are hosted on GitHub, and we can use OpenCV methods to retrieve them.

Unlike this one, which was mainly about visualising the features and implementation, I'm preparing another post on the same topic that will be mostly in code to see how things operate. The original work and concepts are complex and difficult to imagine, yet they have already proven to be extremely effective and accurate, and they examine approaches in a straightforward manner.

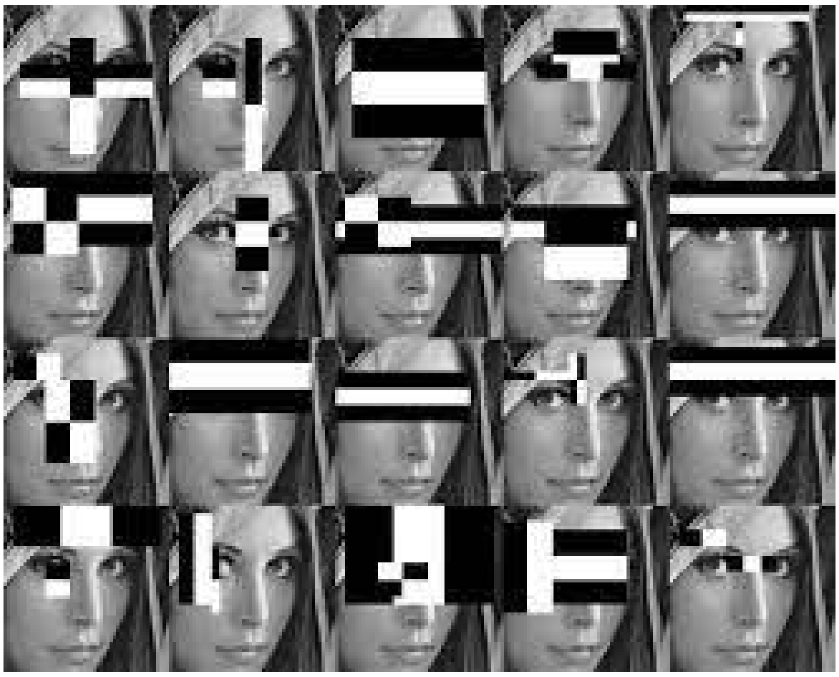


Figure 4.17. Harcascade scanning with different kernels

Facial key points (Landmarks) detection using lbfmodel.yaml 68 facial keypoints detection algorithm.

The process of detecting spots of interest in an image of a human face is known as face landmark detection. It has lately sparked interest in the computer vision field, as it offers a wide range of applications, including recognising emotion through facial motions, as well as identifying facial yaw, role, and pitch angles. Many of these uses can be found in today's cellphones and PC web-camera software. The landmark detector must discover dozens of places on the face to achieve these applications, including the corners of the mouth, corners of the eyes, the silhouette of the jaws, and many others. Many algorithms were created for this purpose, and a few were implemented in OpenCV.

A pre-trained model is required to execute the facemark detector. Although using the OpenCV APIs to train the detector model is undoubtedly doable, certain pre-trained models are available for download. One such model may be found at https://raw.githubusercontent.com/kurnianggoro/GSOC2017/maste r/data/lbfmodel.yaml, which was provided by the OpenCV algorithm implementation contributor (during the 2017 Google Summer of Code).

We can use a predetermined dataset of facial images and videos to benchmark facemark algorithms, but the facemark detector can function with any image.

Facial landmark detection techniques locate significant landmark points on facial photos automatically.As shown in Fig. 4.18 all the landmarks has been detected and numbered accurding to the facial point it's represented. To gain a higher-level comprehension of the face form, those important points are usually significant places locating a facial component, such as an eye corner or a mouth corner. Points surrounding the jawline, lips, eyes, and brows, for example, are required to identify a wide range of facial expressions. Finding face landmarks is challenging due to a multitude of factors, including subject variability, lighting circumstances, and occlusions. Over the last three decades, computer vision researchers have proposed dozens of landmark identification techniques.

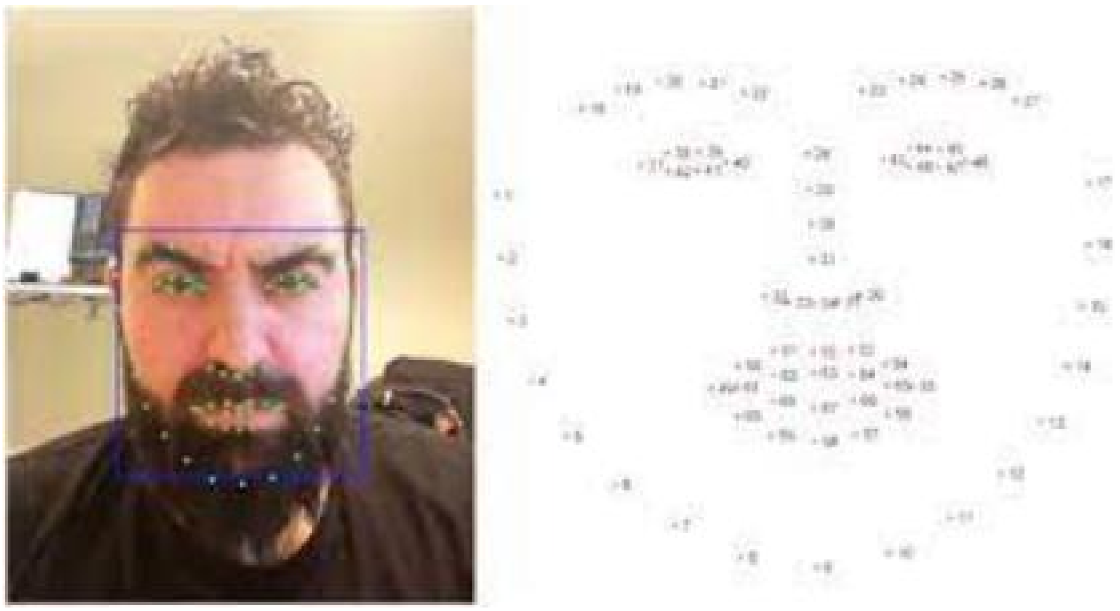


Figure 4.18. Diagram of Facial Landmark Points

Using these landmarks technique it is possible to get the role angle of the target face and by using method named rotate provided by imutils package in python library. Using this method the correction of the image file role angle has been performed.

Predicting emotions by using .H5 model file.

The HDF5 binary data format has a Pythonic interface called h5py. It allows you to store and handle large amounts of numerical data using NumPy. Slice into multi-terabyte datasets saved on disc, for example, as if they were genuine NumPy arrays. Thousands of datasets can be kept in a single file, which can be categorised and labelled as needed.

To save the architecture, weights, and training configuration of a model in a single file/folder, use model.save function of the tensorflow module. This allows you to export a model and use it without having access to the Python code. You can resume training from where you left off now that the optimizer state has been restored.model.predict function is used to implement this saved .h5 model for detecting the classification or value of the provided information and it's available on tensorflow module of python.

While predicting the outcome with model.predict function the input data stream should satisfy certain conditions. The dimensions of the nested data container or numpy array shapee should be the exact same dimension which has be used while training the model. The datatype provided in the input stream should not mismatch with the trained data type or the predictions would be irrelevant from the actual output.

Implementation of TensorBoard

To enhance something in machine learning, you typically need to be able to measure it. TensorBoard is a platform that provides the measurements and visualisations required for machine learning. It allows you to keep track of experiment parameters such as loss and accuracy, visualise the model graph, project embeddings to a lower-dimensional space, and much more.

In machine learning, to improve something you often need to be able to measure it. TensorBoard is a tool for providing the measurements and visualizations needed during the machine learning workflow. It enables tracking experiment metrics like loss and accuracy, visualizing the model graph, projecting embeddings to a lower dimensional space, and much more.

```
def plot_confusion_matrix(cm, class_names):
figure = plt.figure(figsize=(8, 8))
```

```
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)


plt.title("Confusion matrix")
plt.colorbar()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names, rotation=45)
plt.yticks(tick_marks, class_names)
labels = np.around(cm.astype('float') / cm.sum(axis=1)[:, np.newaxis], decimals=2)
threshold = cm.max() / 2.
for i, j in product(range(cm.shape[0]), range(cm.shape[1])):
color = "white" if cm[i, j] > threshold else "black"
plt.text(j, i, labels[i, j], horizontalalignment="center", color=color)


plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')
return figure


def plot_to_image(figure):
buf = io.BytesIO()
plt.savefig(buf, format='png')
plt.close(figure)
buf.seek(0)
image = tf.image.decode_png(buf.getvalue(), channels=4)
image = tf.expand_dims(image, 0)
return image


def log_confusion_matrix(epoch, logs):
test_pred_raw = model.predict_generator(validation_dataset, len(validation_dataset.classes))
test_pred = np.argmax(test_pred_raw, axis=1)
```

```
cm = conM(validation_dataset.classes, test_pred)
figure = plot_confusion_matrix(cm, class_names=target_names)
cm_image = plot_to_image(figure)
with file_writer_cm.as_default():
tf.summary.image("Confusion Matrix", cm_image, step=epoch)
!rm -rf logs
logdir = "logs/image/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=logdir)
file_writer_cm = tf.summary.create_file_writer(logdir + '/cm')


cm_callback = tf.keras.callbacks.LambdaCallback(on_epoch_end=log_confusion_matrix)
```

Tensorboard is responsible for output metric visualization and confusion matrix calculation in real time. It also covers a large number of other parameters like bias weights convolution layer weights and saving model training history in a easy to distribute file system. Fig. 19 shows some of the data visualization available in in tensorboard.

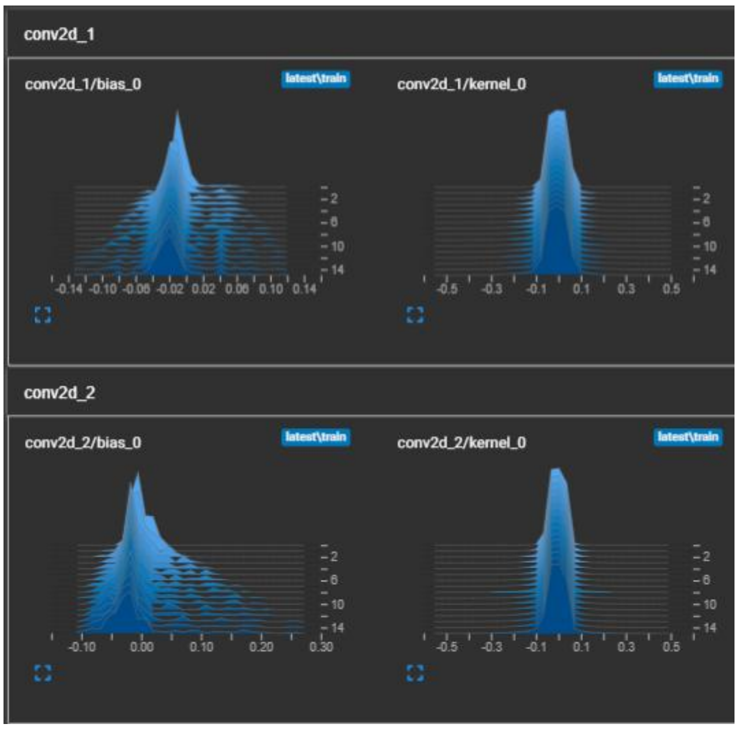


Figure 4.19. Histogram of first and second 2D convolution layer in the trained model.

CHAPTER 5: PROGRESS

Project has been successfully established and working as planned.

STEPS FOR PROTOTYPE DEVELOPMENT:

STAGE - I

Table 5.1. Stage 1 description of scheduling

Sr. No.	**Description**	**Remark**
1.	Selection of parameters for training	**DONE**
2.	Rough estimation of output Accuracy.	
3.	Generation of method of capturing users image and presenting output data to user	

STAGE – II

Table 5.2. Stage 2 description of scheduling

Sr. No.	**Description**	**Remark**
1.	Building of image capture and processor	**DONE**
2.	Building a rough trained model of CNN	

STAGE –III

Table 5.3. Stage 3 description of scheduling

Sr. No.	**Description**	**Remark**
1.	Increasing accuracy of model for all emotions	**DONE**
2.	Selection of Image Processing technique to enhance CNN output	
3.	Updating model with the power of (CNN/ML) and IP	
4.	Implementation of Facial Landmark detection	

	for multiple face detection and correction of facial roll angle.	

STAGE –IV

Table 5.4. Stage 4 description of scheduling

Sr. No.	**Description**	**Remark**
1.	Resolving issues raised in compatibility of model in training and implementing phase	**DONE**

CHAPTER 6: RESULT AND DISCUSSION

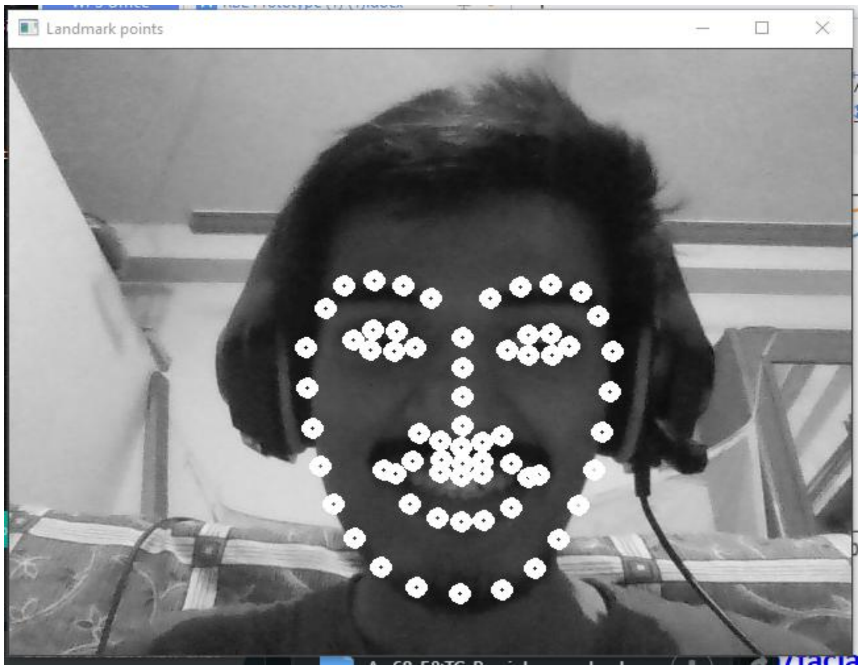


Figure 6.1. Real time facial landmark detection

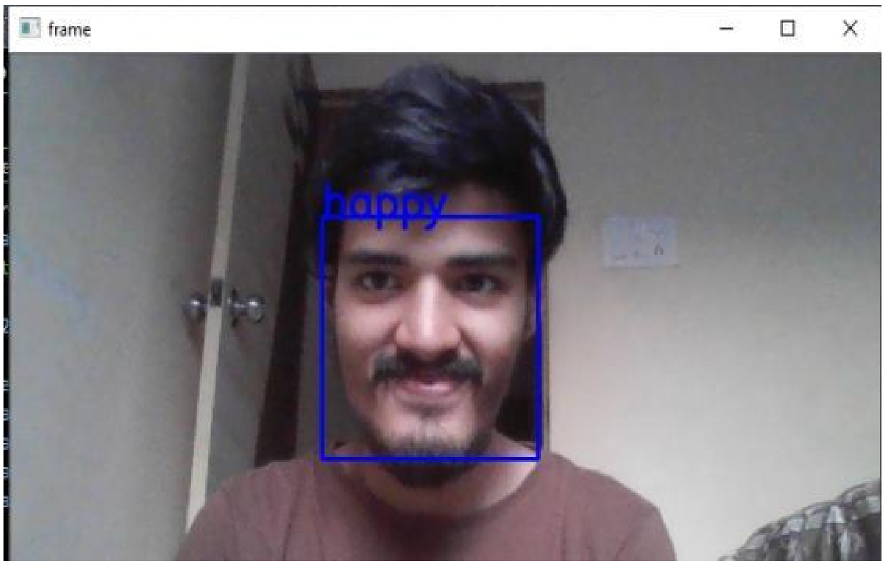
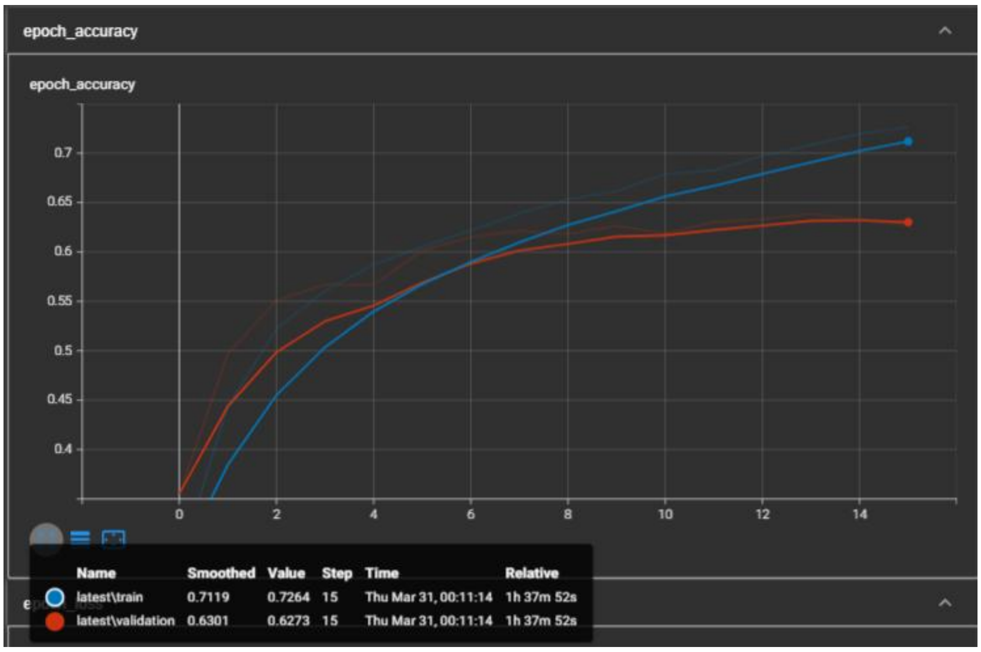


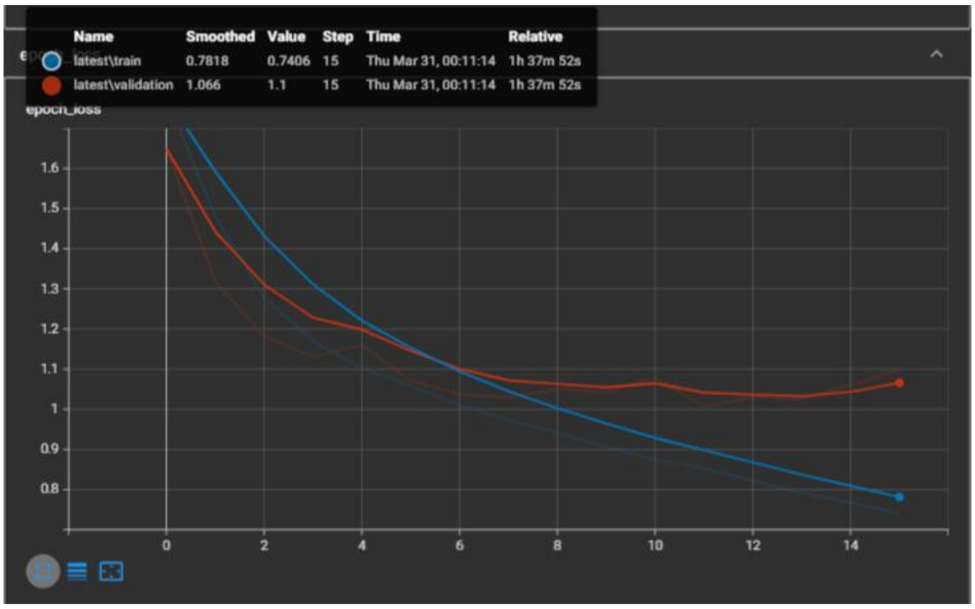
Figure 6.2. Real time facial emotion detection

Accuracy of this project is about 60% for validation dataset and 65% for training dataset. Picture as input can be taken on real time basis, we can upload the picture in the project as well. For the real time ,user will be asked to allow the camera permission Fig. 6.1 is the visualization of real time landmark detection and Fig. 6.2 is real time facial emotion detection of the trained model in the users laptop or desktop.



	Name	Smoothed	Value	Step	Time	Relative
🔵	latest\train	0.7119	0.7264	15	Thu Mar 31, 00:11:14	1h 37m 52s
🔴	latest\validation	0.6301	0.6273	15	Thu Mar 31, 00:11:14	1h 37m 52s

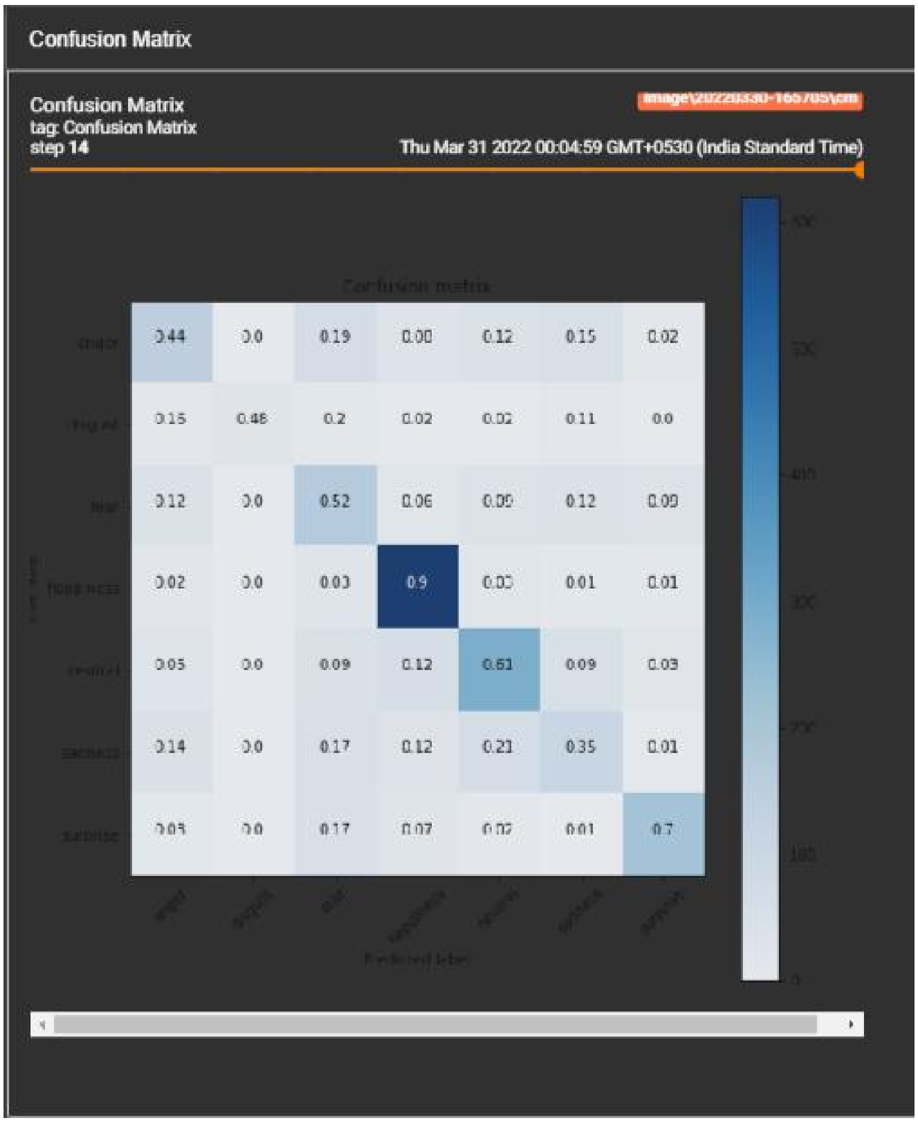
Figure 6.2. Model Training and Validation dataset accuracy curve of model trained



	Name	Smoothed	Value	Step	Time	Relative
🔵	latest\train	0.7818	0.7406	15	Thu Mar 31, 00:11:14	1h 37m 52s
🔴	latest\validation	1.066	1.1	15	Thu Mar 31, 00:11:14	1h 37m 52s

Figure 6.3. Model Training and Validation dataset loss curve of model trained

In the above Fig. 6.2 & Fig. 6.3 we can see the training curve of the loss and accuracy metrics of the trained model. As its visible in the accuracy of the final epoch for the training dataset is 0.72 (72%) and for validation dataset of 0.63 (63%). The loss function also shows the same training phase where traning dataset shows 0.78 units for loss and 1.06 units of loss for validaton dataset.


Confusion Matrix

Confusion Matrix
tag: Confusion Matrix
step 14

image\20220330-165705\cm

Thu Mar 31 2022 00:04:59 GMT+0530 (India Standard Time)

Confusion matrix

anger	0.44	0.0	0.19	0.08	0.12	0.15	0.02
disgust	0.15	0.48	0.2	0.02	0.02	0.11	0.0
fear	0.12	0.0	0.52	0.06	0.05	0.12	0.09
happiness	0.02	0.0	0.03	0.9	0.03	0.01	0.01
neutral	0.05	0.0	0.09	0.12	0.61	0.09	0.03
sadness	0.14	0.0	0.17	0.12	0.21	0.35	0.01
surprise	0.03	0.0	0.17	0.07	0.02	0.01	0.7

Predicted label

Figure 6.4. Confusion matrix of trained model

Above Fig. 6.4 shows the representation of the confusion matrix of the trained model. By evaluating the outcome of the model the emotion which has the highest accuracy is happiness and with the least accuracy is disgust. The reason for this evaluation is found out to be less number of training dataset available for the disgust data field after the filtration. Representing availability of high number of noise in disgust facial images.

CHAPTER 7: CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

Detection of Human Facial Expression with 7 different types of expressions is been able to be predicted in this project. The algorithm is able to detected human face in a noise background and correct its role angle with the help of landmark detection provided by dilb module of python. The accuracy of the model is around 55% accuracy with good processing time. The output consists of a rectangular box representing detected face and emotion predicted is written on the top of the box. The application version of the project is going to be made by pyinstaller package which supports Windows, Mac and Linux desktop and laptops.

7.2 Future Scope

Emotion detection is an essential aspect of computer vision, and it may be used to execute a wide range of jobs and processes if one understands the complexities and limitless possibilities afforded by the field of emotion detection

A. App and product development:-

Emotion detection has the potential to help optimise a variety of software engineering processes, including testing the simplicity with which a product may be utilised.

B. Improved learning practices:-

Evidence reveals that while some emotional states promote better learning habits, others try to repress them. The distinction between the two groups with differing emotional states is not easily discernible.

C. Improvised web development:-

Because of the massive size at which the internet is growing, service providers are interested in gathering massive amounts of data from customers. As a result, all information and adverts are tailored to the user's profile. As a result, including extensive details about various human emotions can result in considerably more precise behavioural models of various sorts of users.

D. Immersive gaming :-

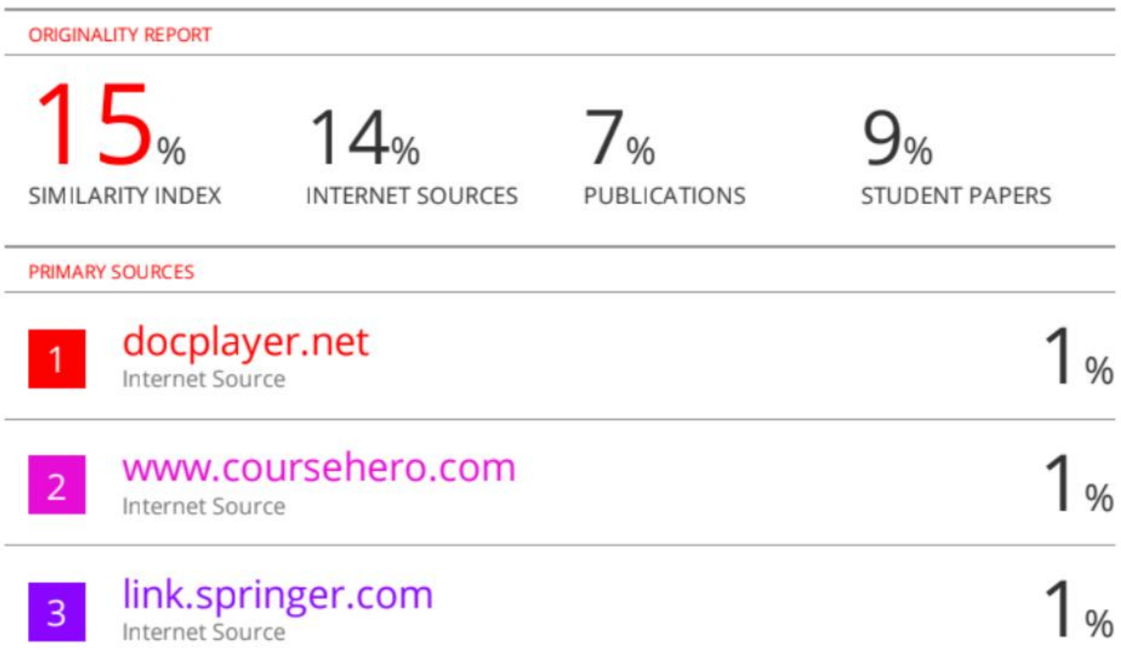
Video games account for a sizable portion of the entertainment sector. As a result, video game producers centre their research on various forms of human emotions that are regularly encountered in order to make these games much more intense and intrusive. To entice more players, video games are designed in such a way that they organically blend human emotions into game play.

Appendix A

Abbreviation and symbols

1. ML: Machine Learning
2. TF: Tensorflow
3. IP: Image Processing
4. CNN: Convolution Neural Network
5. DL: Deep Learning

Plagiarism Report



ORIGINALITY REPORT

15% SIMILARITY INDEX

14% INTERNET SOURCES

7% PUBLICATIONS

9% STUDENT PAPERS

PRIMARY SOURCES

1	docplayer.net Internet Source	1%
2	www.coursehero.com Internet Source	1%
3	link.springer.com Internet Source	1%

Appendix B

Definitions

1. Model: A model in data science refers to a file (saved or trained) on a given dataset using pre- defined algorithms to understand and learn certain type of pattern.

2. Accuracy: It's a measurement of correct predictions that are made by model. It is defined as ratio of sum of True Positive (TP) and True Negative (TN) to sum of True Positive (TP), False Positive(FP), True Negative (TN), False Negative (FN).

$$\text{Accuracy}=\frac{TP+TN}{TP+FP+TN+FN}$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

3. Loss: It a number denoting measure of bad predictions. An ideal model should have zero loss. Itrefers to penalty for incorrect predictions.

4. Training Accuracy: The value of accuracy calculated for model over seen dataset i.e., train datasetis called training accuracy.

5. Validation Accuracy: The value of accuracy calculated for model over unseen dataset i.e.,validation dataset is called validation accuracy.

6. Kernel/Filter: Kernel refers to a matrix which is used to convolute (dot product) with patch of animage (local receptive field) of similar dimension.

7. Epochs: The hyper parameter that decides the number of times an algorithm will iterate and learnon a particular dataset is called as epochs.

References

[1] Michael Revina, W.R. Sam Emmanuel , “A Survey on Human Face Expression Recognition Techniques, ” 2018 ,Reg No. 12417, N.M. Christian College, Marthandam Affiliated to Manonmaniam Sunadaranar University, Abishekapatti, Tirunelveli – 627012, Tamil Nadu, IndiaDepartment of Computer Science, N.M. Christian College, Marthandam Affiliated to Manonmaniam Sunadaranar University, Abishekapatti, Tirunelveli – 627012, Tamil Nadu, India

[2] Anjali, Prachi Chaudhary, "A Survey on Different Types of Techniques uses for Face Expression Recognition,"International Journal of Machine Learning and Computing,2019, Vol. 9, No. 1

[3] Dhwani Mehta, Mohammad Faridul Haque Siddiqui and Ahmad Y. Javaid, " Facial Emotion Recognition: A Survey and Real-World User Experiences in Mixed Reality,"2018, Sensors 18 (2), 416

[4] Wisal Hashim Abdulsalam, Rafah Shihab Alhamdani, and Mohammed Najm Abdullah, "Facial Emotion Recognition from Videos Using Deep Convolutional Neural Networks, " 2019, International Journal of Machine Learning and Computing 9 (1), 14-19

[5] James Pao, “Emotion Detection Through Facial Feature Recognition,” in International Journal of Multimedia and Ubiquitous Engineering, November 2017

[6] Aitor Azcarate, Felix Hageloh, Koen van de Sande, Roberto Valenti, “Automatic facial emotion recognition,” January 2005

[7] Dan Duncan, Gautam Shine, Chris English, "Facial Emotion Recognition in Real-Time," November 2016

[8] Shivam Gupta, “Facial emotion recognition in real-time and static images,” in 2nd International Conference on Inventive Systems and Control (ICISC) IEEE, 28 June 2018

[9] Aneta Kartali, Miloš Roglić, Marko Barjaktarović, Milica Đurić- Jovičić, Milica M. Janković, “Real-time Algorithms for Facial Emotion Recognition: A Comparison of Different Approaches,” in 2018 14th Symposium on Neural Networks and Applications (NEURAL), Nov 2018