Chapter 1

Introduction

1.1 Introduction

Facial key points detection is a computer technology being used in a variety of applications that identifies human faces in digital images. Facial key points detection also refers to the psychological process by which humans locate and attend to faces in a visual scene. Face detection can be regarded as a specific case of object-class detection. In object-class detection, the task is to find the locations and sizes of all objects in an image that belong to a given class.[1] Detect facial keypoints is a critical element in face recognition. However, there is difficulty to catch keypoints on the face due to complex influences from original images, and there is no guidance to suitable algorithms. In this paper, we study different algorithms that can be applied to locate keyponits. Specifically: our framework (1) Prepare the data for further investigation (2) Using PCA and LBP to process the data (3) Apply different algorithms to analysis data, including linear regression models, tree-based model, neural network and convolutional neural network, etc. Finally, we will give our conclusion and further research topic. A comprehensive set of experiments on dataset demonstrates the effectiveness of our framework.[2]

1.2 Motivation

Detecting facial key points is a very challenging problem. Facial features vary greatly from one individual to another, and even for a single individual, there is a large amount of variation due to 3D pose, size, position, viewing angle, and illumination conditions. Computer vision research has come a long way in addressing these difficulties, but there remain many opportunities for improvement.[3] Recognizing faces is a very challenging problem in the field of image processing. The techniques presently being used are Biometric recognition (but Iris scanners are far too expensive), Eigenfaces(inaccurate with varying image factors like intensity, camera angles), line edge maps. While facial features in images depend on a lot many conditions, faces can be recognized easily if we use the relevant keypoints and landmarks for identification. The unchanging ratios and distances between these mark the importance of this approach. [4]

1.3 Objective

The objective of this task is to predict key points positions on face images. Detection of facial keypoints is building block for many applications in computer vision. Research has been done on this part but still there is hope for improvement. What really motivated me was that this problem works as the first steps for several applications, such as:

- Tracking faces in images and video
- Analyzing facial expressions
- Detecting dysmorphic facial signs for medical diagnosis
- Biometrics / face recognition
- Smart banking
- Forensic Investigation
- Security

1.4 Scope of the Work

Importance of Face Recognition System as a Security Solution Face is considered as the most important part of human body. Research shows that even face can speak and it has different words for different emotions. It plays a very crucial role for interacting with people in the society. It conveys people's identity, so it can be used as a key for security solutions in many organizations. Nowadays, face recognition system is getting increasing trend across the world for providing extremely safe and reliable security technology. It is gaining significant importance and attention by thousands of corporate and government organizations only because of its high level of security and reliability. Moreover, this system is providing vast benefits when compared to other biometric security solutions like palm print and finger print. As computation processing powers increases and large storage are available to store data, hence demand of it increases as it is used in several real-world applications.[5] Use of deep learning made more interested in the thesis. Convolutional Neural Network is chosen because it is more accurate and allow us to modify at various stages.

1.5 Outline of the Thesis

In "Introduction", the 1st chapter of the book contains the information about project motivation, the objective of the project and so on.

In "Literature Review", the 2nd chapter of the book contains the information about Facial key points detection mechanism, Existing algorithms and its pros and cons, and a gentle overview of Machine learning, Deep learning and Convolutional Neural Network.

In "Existing System Overview", the 3rd chapter of the book contains the information about Existing system's technical details and workflow, its pros and cons and features.

In "Proposed Machine Learning Based Facial Key Points Detection System", the 4th chapter of the book contains the information about Approach of proposed system, Diagram, Workflow.

In "Implementation and Testing", the 5th chapter of the book contains the information about Platform, data, model and result of the proposed system with features, merits and requirements.

In "Conclusion and Future Works", the 6th chapter of the book shows conclusion of this project.

Chapter 2

Literature Review

2.1 Introduction

Face detection is a computer technology that is being applied for many different applications that require the identification of human faces in digital images or video. It can be regarded as a specific case of object-class detection, where the task is to find the locations and sizes of all objects in an image that belong to a given class. The technology is able to detect frontal or near-frontal faces in a photo, regardless of orientation, lighting conditions or skin color.[1]

2.2 How Facial Key Points can be Detected?

Face detection applications use algorithms that decides whether an image is a positive image also called face image or negative image also called non-face image. This is called a classifier. To classify a new image correctly, it is trained on hundreds of thousands of face and non-face images. This feature answers the question “Where are the faces in this picture?”. For each face detected, you get a complete analysis of key points also called landmarks around the eyes, eye brows, jaw, nose and mouth.

2.3 Existing Key Points Detection Algorithms

OpenCV is a popular computer vision library started by Intel in 1999. The cross-platform library sets its focus on real-time image processing and includes patent-free implementations of the latest computer vision algorithms. OpenCV 2.3.1 now comes with a programming interface to C, C++, Python and Android. OpenCV 2.4 now comes with the very newFaceRecognizerclass for face recognition, so you can start experimenting with face recognition right away.[6] The currently available algorithms are:

- Eigenfaces
- Fisherfaces
- Local Binary Patterns Histograms

Among above mentioned available algorithms, Local Binary Patterns Histograms is the most widely used and most efficient algorithm.

2.3.1 Advantages

Some advantages of OpenCV face detection methods are:

- Computationally simple and fast
- Shorter training time
- Low false positive rate
- Better performance in offline learning system
- Vision based system
- Lower configuration PC required

2.3.2 Disadvantages

Some disadvantages of OpenCV face detection methods are:

- Not a knowledge-based system
- Cannot be used in online learning system
- Difficult to evolve

2.4 Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.[7] Machine learning is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that which makes it more similar to humans: The ability to learn. Machine learning is actively being used today, perhaps in many more places than one would expect.

2.4.1 Supervised Machine Learning

Supervised learning: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs. The training process continues until the model achieves the desired level of accuracy on the training data. Some real-life examples are:

Image Classification: It is about train images with labels. Then in the future you give a new image expecting that the computer will recognize the new object.

Market Prediction/Regression: It is about train the computer with historical market data and ask the computer to predict the new price in the future.[8]

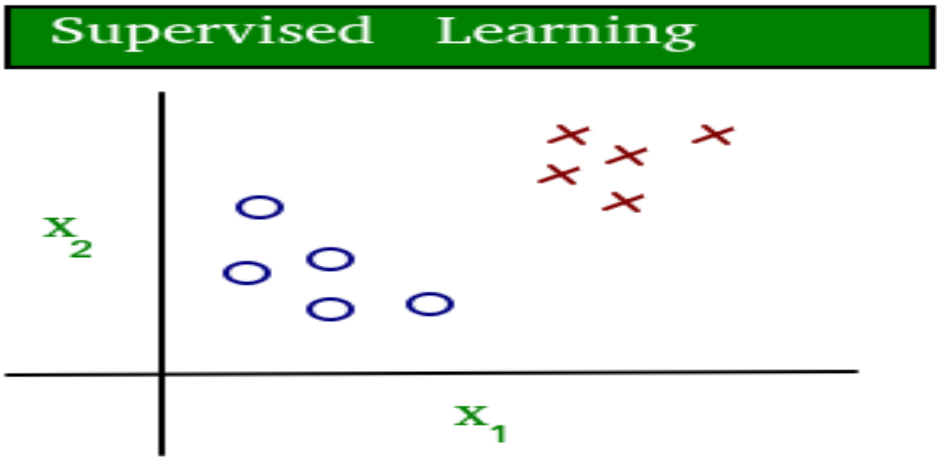

Fig 2.1: Supervised learning [8]

2.4.2 Unsupervised Machine Learning

Unsupervised learning: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. It is used for clustering population in different groups. Unsupervised learning can be a goal in itself (discovering hidden patterns in data).

Clustering: It is about to ask the computer to separate similar data into clusters, this is essential in research and science.

High Dimension Visualization: Use the computer to help us visualize high dimension data.

Generative Models: After a model captures the probability distribution of your input data, it will be able to generate more data. It can be very useful to make your classifier more robust.[8]

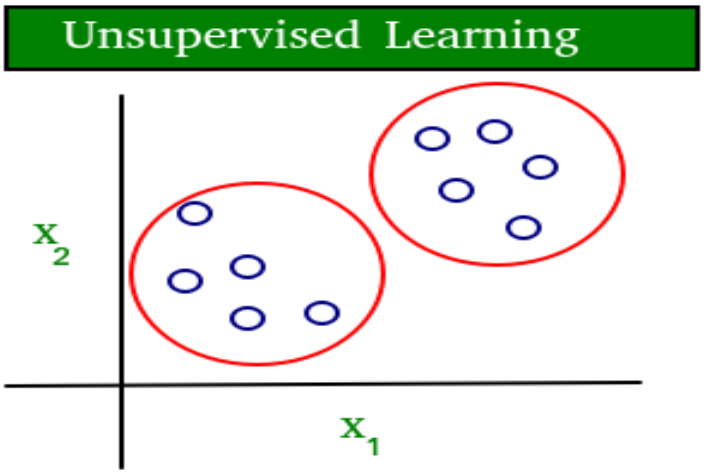

Fig 2.2: Unsupervised learning [8]

2.5 Deep Learning

Deep learning is a subset of machine learning in Artificial Intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as Deep Neural Learning or Deep Neural Network.[5]

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It's achieving results that were not possible before.

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction.[9]

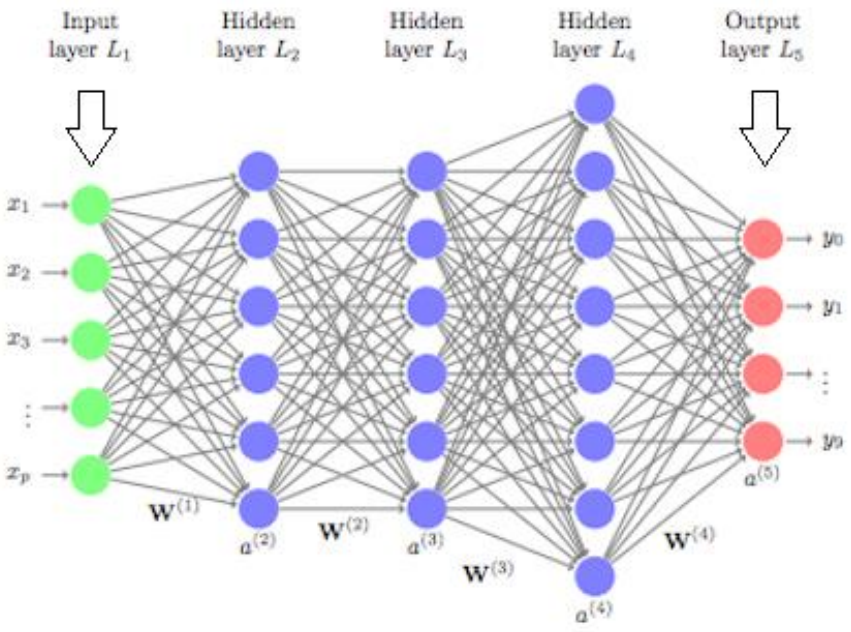

Fig 2.3: Layers of Deep Neural Network [10]

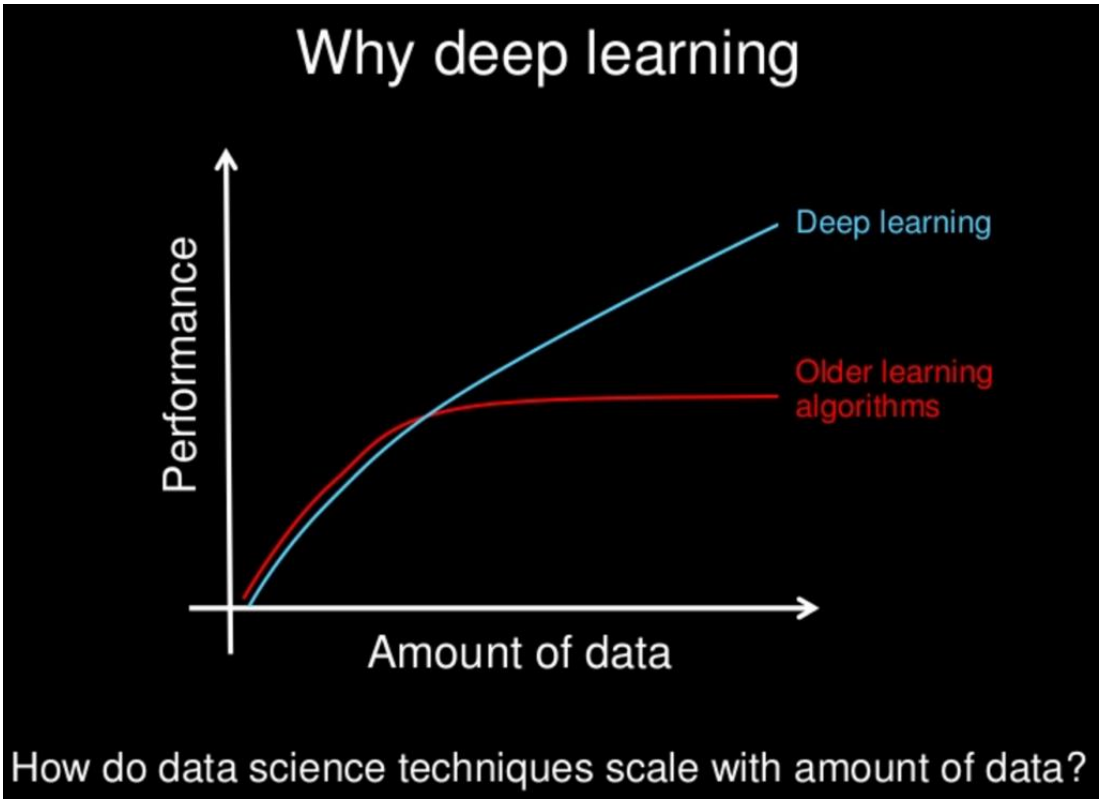

Fig 2.4: Performance of Deep Neural Network [10]

2.6 Convolutional Neural Network

A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data.
CNNs are powerful image processing, artificial intelligence (AI) that use deep learning to perform both generative and descriptive tasks, often using computer vision that includes image and video recognition, along with recommender systems and natural language processing.[5]

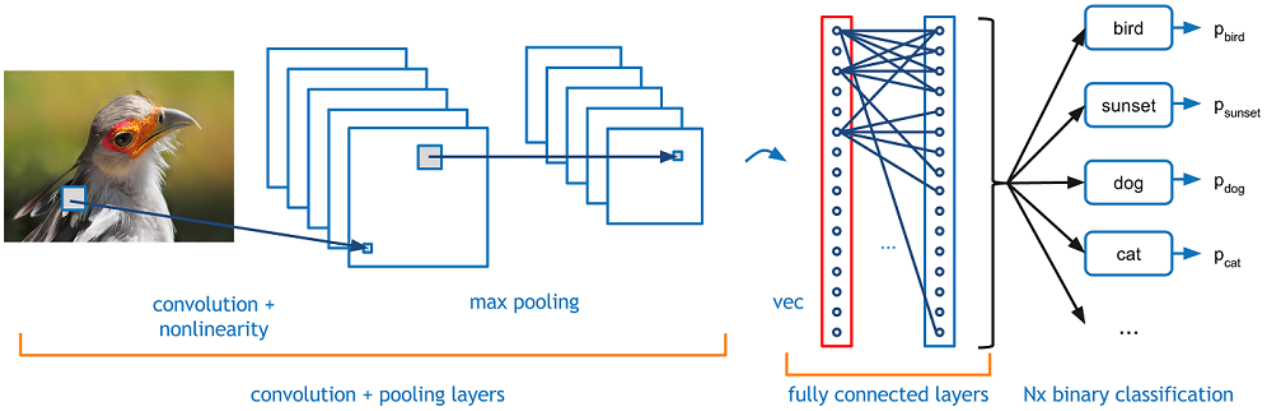

Fig 2.5: Working Principle of CNN [10]

Artificial Intelligence has been witnessing a monumental growth in bridging the gap between the capabilities of humans and machines. Researchers and enthusiasts alike, work on numerous aspects of the field to make amazing things happen. One of many such areas is the domain of Computer Vision. The agenda for this field is to enable machines to view the world as humans do, perceive it in a similar manner and even use the knowledge for a multitude of tasks such as Image & Video recognition, Image Analysis & Classification, Media Recreation, Recommendation Systems, Natural Language Processing, etc. The advancements in Computer Vision with Deep Learning has been constructed and perfected with time, primarily over one particular algorithm a Convolutional Neural Network. A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics. The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlaps to cover the entire visual area. In deep learning, a convolutional neural network is a class of deep neural networks, most commonly applied to analyzing visual imagery. CNNs are regularized versions of multilayer perceptrons.[10]

2.7 Summary

In this chapter it is discussed where the system is being applied for. It is also shown that the system can be regarded as a specific case of object-class detection. Hundreds of thousands of face and non-face images data are fed to the system to classify new image correctly. OpenCV classifiers are existing solution to the system. Its pros and cons are also described in the chapter. Finally, a gentle overview to the Machine Learning, Deep Learning and Convolutional Neural Network is provided.

Chapter 3

Existing System Overview

3.1 Introduction

OpenCV is a popular computer vision library started by Intel in 1999. The cross-platform library sets its focus on real-time image processing and includes patent-free implementations of the latest computer vision algorithms. OpenCV 2.3.1 now comes with a programming interface to C, C++, Python and Android. OpenCV 2.4 now comes with the very newFaceRecognizerclass for face recognition, so you can start experimenting with face recognition right away.[6] The currently available algorithms are:

- Eigenfaces
- Fisherfaces
- Local Binary Patterns Histograms

Among above mentioned available algorithms, Local Binary Patterns Histograms is the most widely used and most efficient algorithm.

3.2 OpenCV

Local Binary Patterns methodology is most widely used over the world for face detection. It is famous for its high accuracy. Hence it is used in this thesis for its high accuracy. Local Binary Patterns methodology has its roots in 2D texture analysis. The basic idea of Local Binary Patterns is to summarize the local structure in an image by comparing each pixel with its neighbourhood. Take a pixel as centre and threshold its neighbours against. If the intensity of the centre pixel is greater-equal its neighbour, then denote it with 1 and 0 if not. You'll end up with a binary number for each pixel, just like 11001111. So with 8 surrounding pixels, you'll end up with 2^8 possible combinations, called *Local Binary Patterns* or sometimes referred to as *LBP codes*. The first LBP operator described in literature actually used a fixed 3 x 3 neighbourhood just like this:

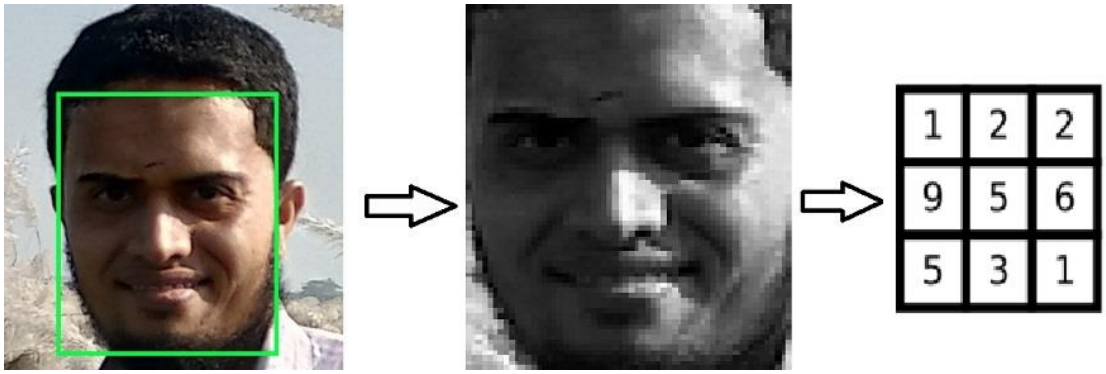

Fig 3.1: Image Conversion

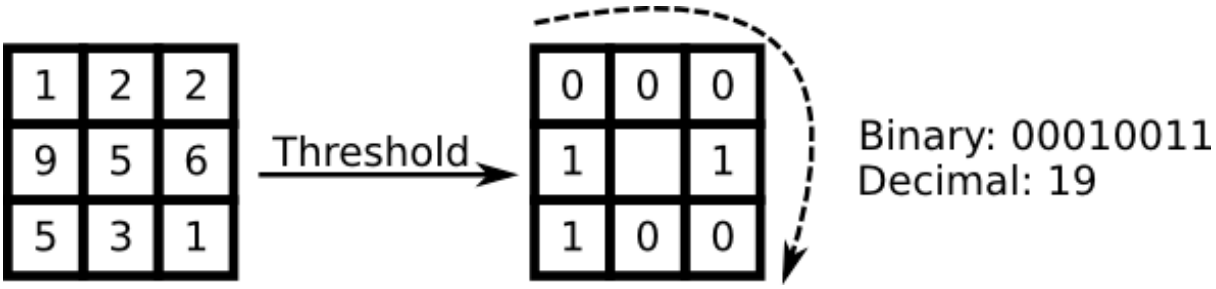

Fig 3.2: LBP Method [6]

A more formal description of the LBP operator can be given as:

$$LBP(x_c, y_c) = \sum_{p=0}^{P-1} 2^p s(i_p - i_c)$$

, with (x_c, y_c) ascentral pixel with intensity i_c; and i_n being the intensity of the neighbor pixel. s is the sign function defined as:

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{else} \end{cases}$$

By definition, the LBP operator is robust against monotonic grey scale transformations. Dividing the LBP image into m local regions and extract a histogram from each. These histograms are called *Local Binary Patterns Histograms*.[6]

3.3 Workflow

Face Recognition process is about three steps:

- Prepare Training Data: Read training images for each person/subject along with their labels, detect faces from each image and assign each detected face an integer label of the person it belongs.
- Train Face Recognizer: Train OpenCV's LBPH recognizer by feeding it the data we prepared in step 1.
- Apply Test Data: Image data that will be tested against the model are applied to the model.
- Prediction: Introduce some test images to face recognizer and see if it predicts them correctly.
- Result: Get the result after prediction and it is further processed in the next phase of the system.
- Analysis: After get the result of the system, it is analyzed with respect to some predefined threshold value.

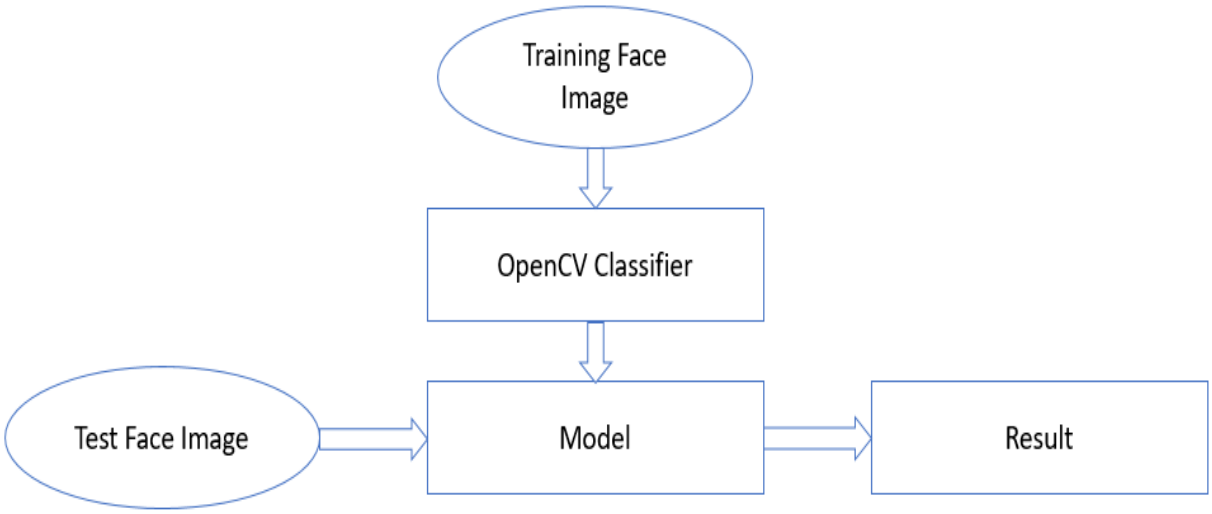

Fig 3.3: Workflow of Existing System

3.4 Advantages and Disadvantages

3.4.1 Advantages

Some advantages of OpenCV face detection methods are:

- Computationally simple and fast
- Shorter training time
- Low false positive rate
- Better performance in offline learning system

3.4.2 Disadvantages

Some disadvantages of OpenCV face detection methods are:

- Not a knowledge-based system
- Cannot be used in online learning system
- Difficult to evolve

3.5 Summary

In this chapter it is discussed what OpenCV is. It is also shown that what methods of face detection are offered by OpenCV and how to use that methods. OpenCV's technical details, to be specific how Local Binary Pattern method works are discussed here. It also depicts workflow of OpenCV face detection and its pros and cons are also described in the chapter.

Chapter 4

Proposed Machine Learning Based Facial Key Points Detection System

4.1 Introduction

In this proposed system, the combined knowledge of computer vision techniques and deep learning architectures will be applied to build a facial key points detection system. Facial key points include points around the eyes, nose, and mouth on a face and are used in many applications. This system has several applications including facial tracking, facial pose recognition, facial filters, and emotion recognition. The final system should be able to look at any image, detect faces, and predict the locations of facial key points on each face.[11]

4.2 Approach of the Proposed System

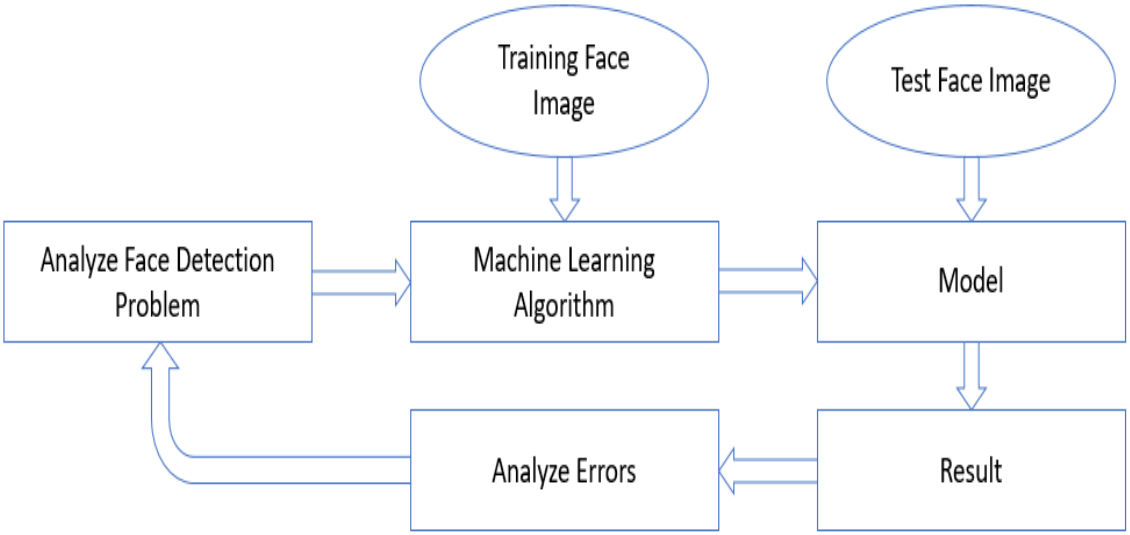

Fig 4.1: Proposed ML Based System

In the proposed system, analysis of the face detection problem is performed at first. It includes data cleaning, data visualization and features extraction. If this phase is completed successfully then the next phase of the system is performed which is applying Machine Learning algorithm. An appropriate Machine Learning algorithm is applied on training data such as Logistic regression, Deep Learning, Convolutional Neural Network etc. After applying the Machine Learning algorithm, a model can be obtained and test data are applied on that model. Result are obtained after scientific computation of the particular algorithms. And then error analysis of the model is performed. If error rate is greater than a threshold value then changes to model are required else the model is deployed. Otherwise, the problem is analyzed again in similar fashion. This process is repetitive.[12]

4.3 Workflow of the Proposed System

4.3.1 User Perspective

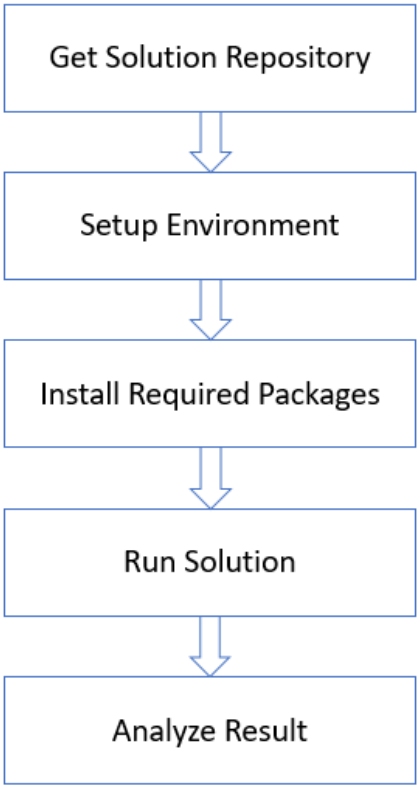

Fig 4.2: User Perspective Workflow

User needs to get the solution repository either from git or from other sources. Then he should install Python 3.5, Tensorflow latest version, Keras Latest version and install other packages through pip package manager. These packages are required to perform scientific computation, visualization etc. Then it is required to run the solution along with data. Finally, it needs to collect result and analyze it.[13]

4.4 Convolutional Neural Network

4.4.1 What Convolutional Neural Network is?

Convolutional Neural Networks (CNNs), like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output. The whole network has a loss function and all the tips and tricks that we developed for neural networks still apply on CNNs.[14]

4.4.2 CNNs Operate Over Volumes

Unlike neural networks, where the input is a vector, here the input is a multi-channeled image (3 channeled in this case). There are other differences that we will talk about in a while. Before we go any deeper, we have to understand what convolution means.[14]

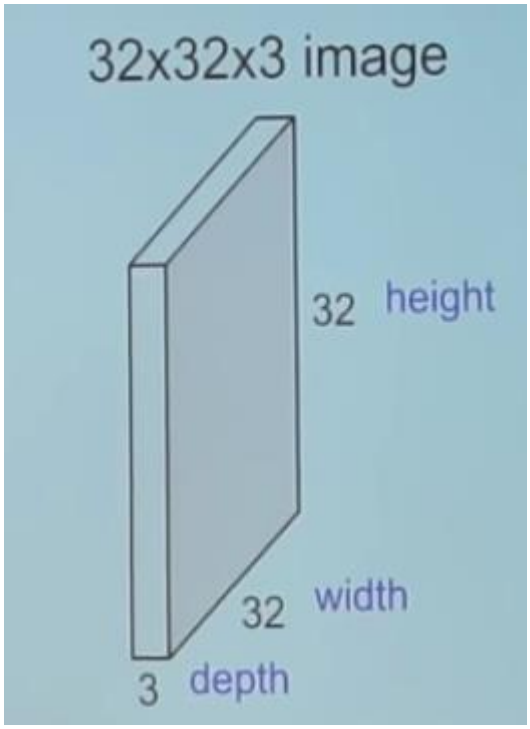

Fig 4.3: Example of an RGB Image [14]

4.4.3 Convolution

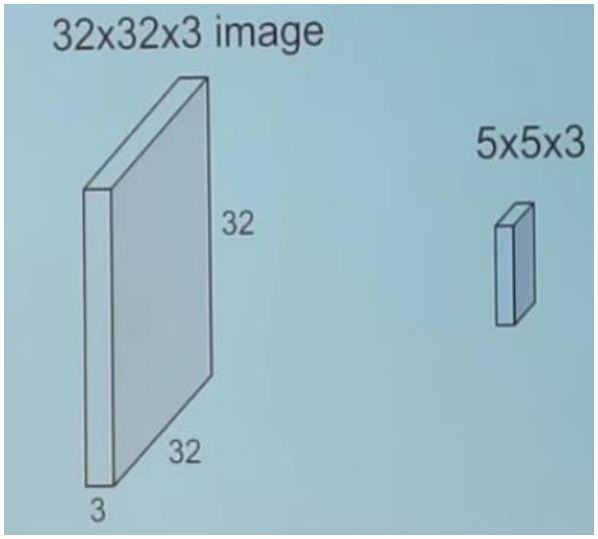

Fig 4.4: Convolving an Image with a Filter [14]

We take the 5*5*3 filter and slide it over the complete image and along the way take the dot product between the filter and chunks of the input image.

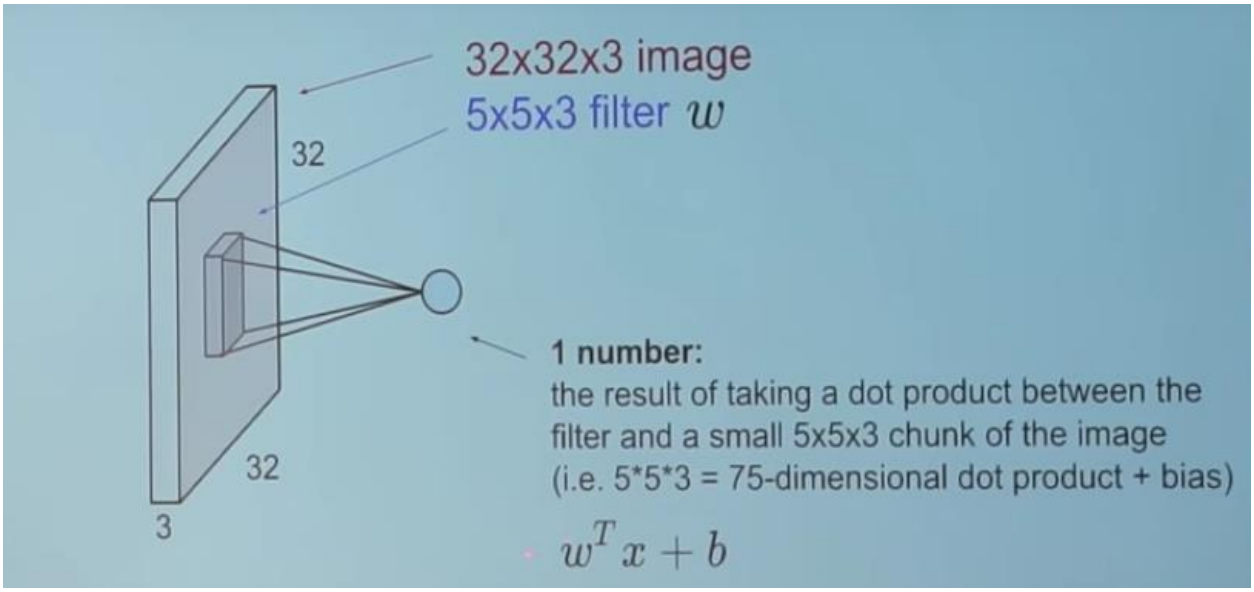

Fig 4.5: How Convolution Looks [14]

For every dot product taken, the result is a scalar. when we convolve the complete image with the filter we will get 28*28 unique positions where the filter can be put on the image.

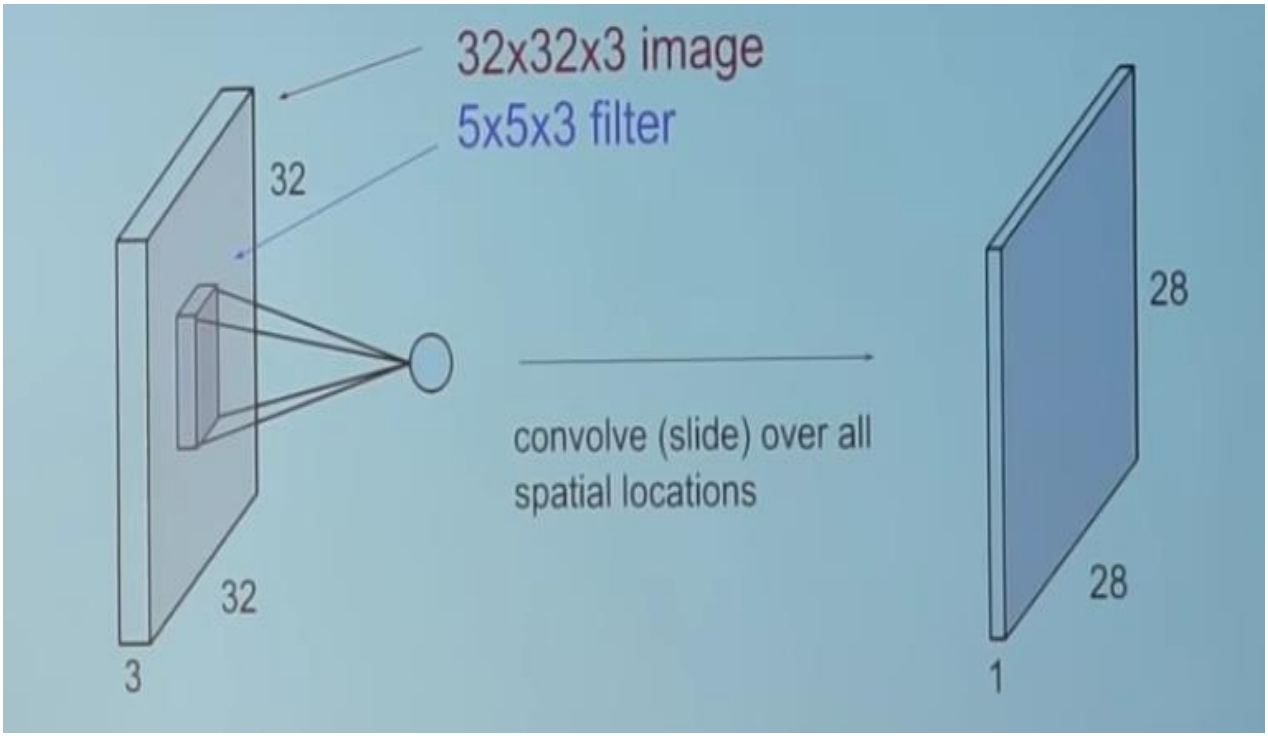

Fig 4.6: Convolved Image [14]

Now, if we back to CNNs the convolution layer is the main building block of a convolutional neural network.

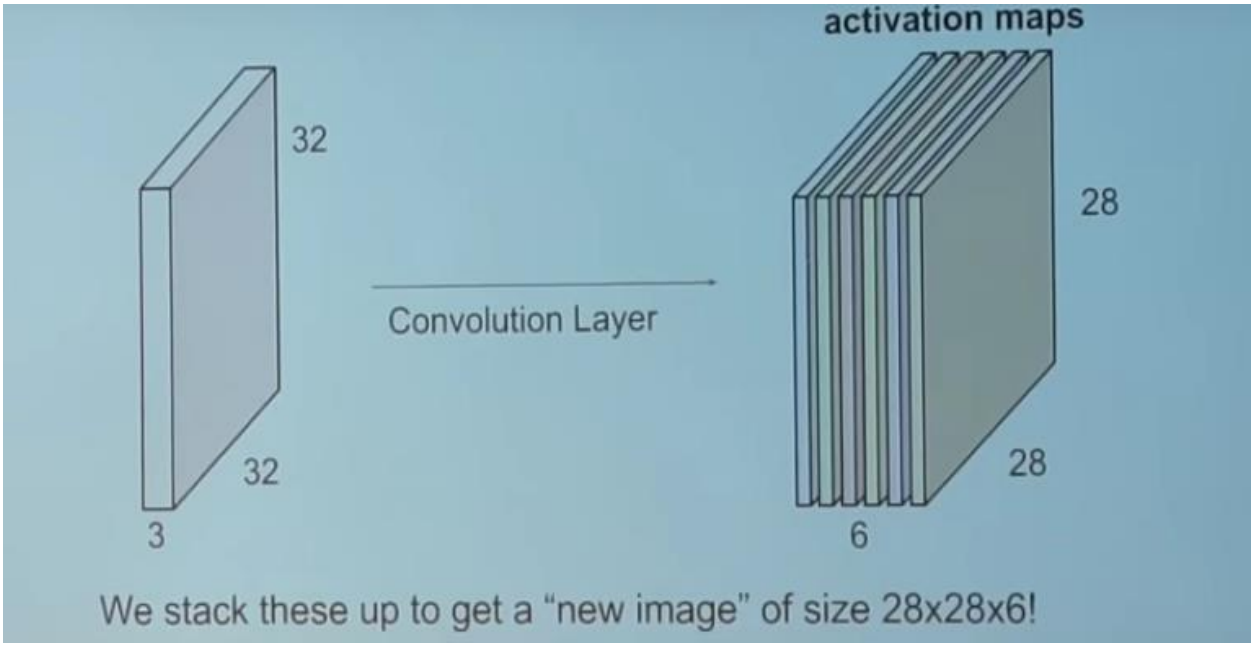

Fig 4.7: Convolution Layer [14]

The convolution layer comprises of a set of independent filters. In this example there are 6 independent filters. Each filter is independently convolved with the image and we end up with 6 feature maps of shape 28*28*1. Suppose we have a number of convolution layers in sequence. Then we will get convolution layers in sequence like below.

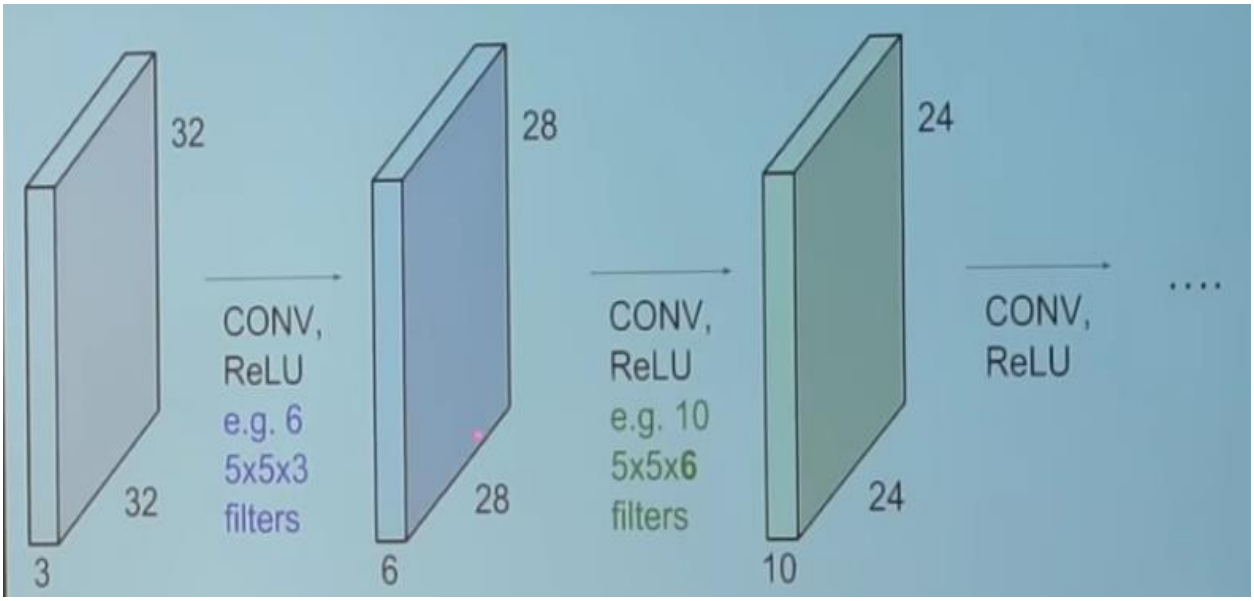

Fig 4.8: Convolution Layers in Sequence [14]

All these filters are initialized randomly and become our parameters which will be learned by the network subsequently. An example of a trained network of face images is look like below.

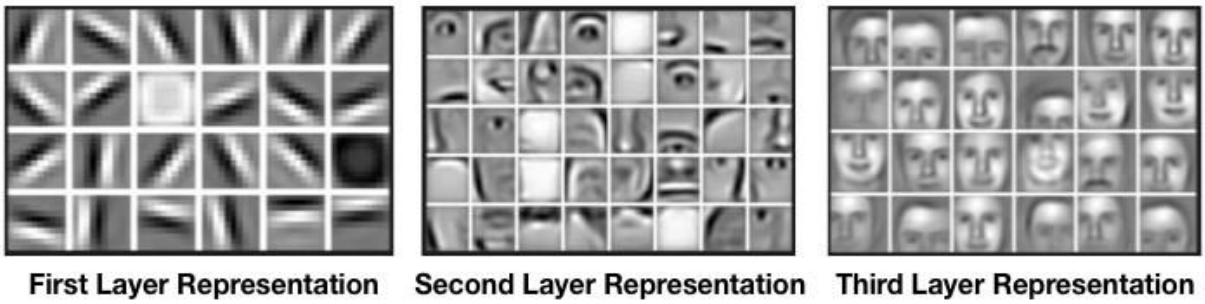

Fig 4.9: Filters in a Trained Network [14]

Take a look at the filters in the very first layer. These are our 5*5*3 filters. Through back propagation, they have tuned themselves to become blobs of colored pieces and edges. As we go deeper to other convolution layers, the filters are doing dot products to the input of the previous convolution layers. So, they are taking the smaller colored pieces or edges and making larger pieces out of them.

Take a look and imagine the 28*28*1 grid as a grid of 28*28 neurons. For a particular feature map the output received on convolving the image with a particular filter is called a feature map, each neuron is connected only to a small chunk of the input image and all the neurons have the same connection weights. So again coming back to the differences between CNN and a neural network.[14]

4.4.4 CNNs Parameter and Connectivity

Parameter sharing is sharing of weights by all neurons in a particular feature map. Local connectivity is the concept of each neural connected only to a subset of the input image unlike a neural network where all the neurons are fully connected. This helps to reduce the number of parameters in the whole system and makes the computation more efficient.[14]

4.4.5 Pooling Layers

A pooling layer is another building block of a CNN. Its function is to progressively reduce the spatial size of the representation to reduce the number of parameters and computation in the network. Pooling layer operates on each feature map independently.

The most common approach used in pooling is max pooling. Where the maximum value of a chunk is selected for the next pooling layers. Further all the selected maximum values are together to make the next stage of pooling layer.[14]

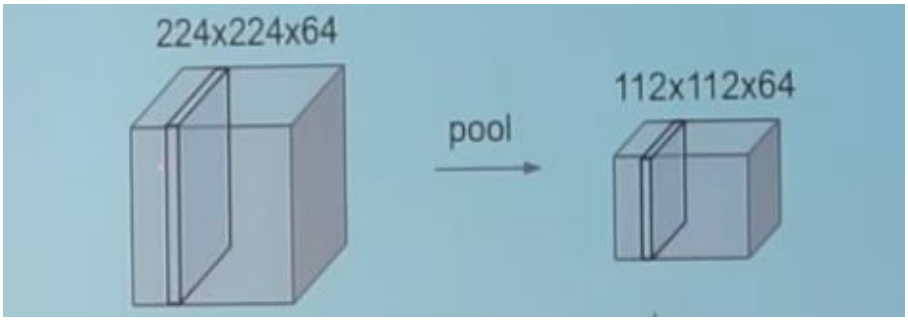

Fig 4.10: Pooling [14]

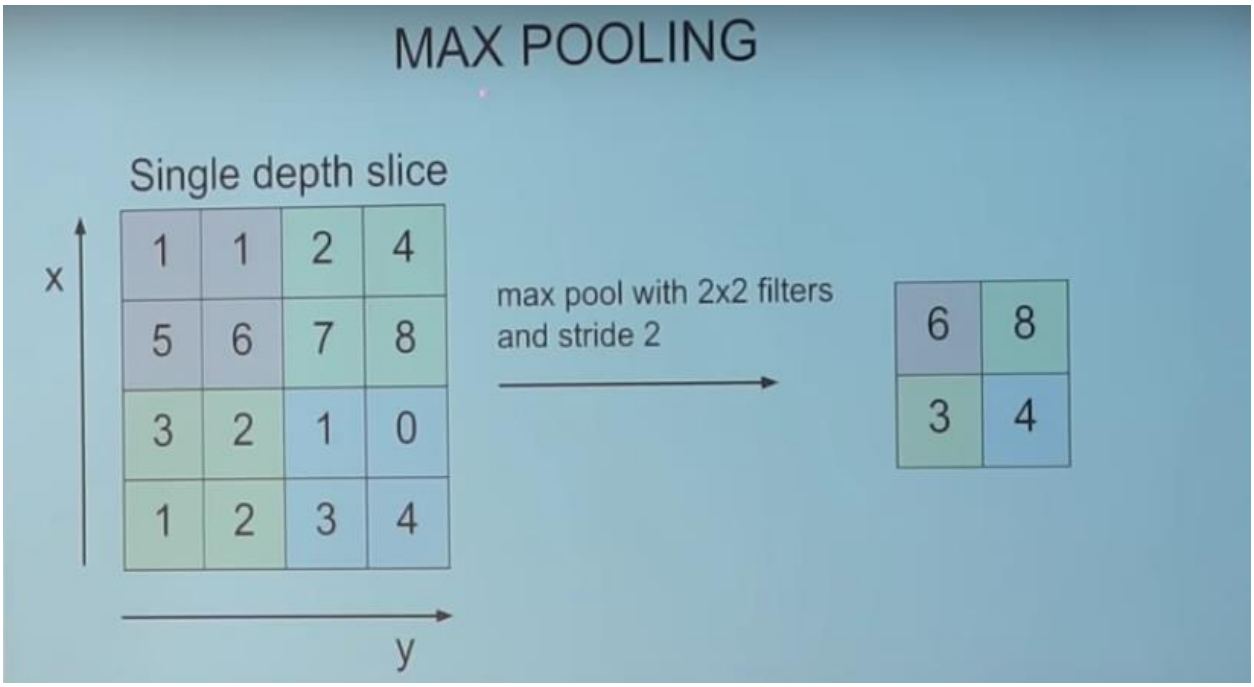

Fig 4.11: Max Pooling [14]

4.4.6 Typical Architecture of CNN

We have already discussed about convolution layers denoted by CONV and pooling layers denoted by POOL. Rectified Linear Unit also denoted as RELU is just a non-linearity which is applied similar to neural networks.

The FC is the fully connected layer of neurons at the end of CNN. Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks and work in a similar way.[14]

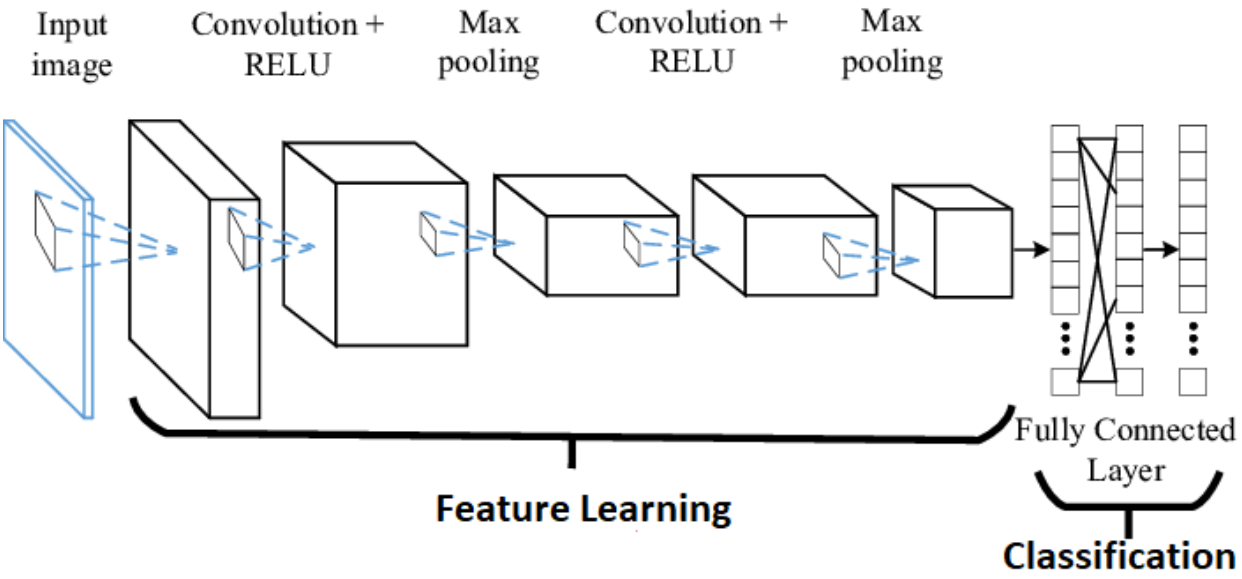

Fig 4.12: Typical Architecture of CNN [14]

4.5 Summary

In this chapter proposed system is discussed in details. A diagram is also provided that depicts how the proposed system will work and its description provided as well. A user view workflow, features and merits of the proposed system is also given. Finally, required tools for the proposed system is also mentioned.

Chapter 5

Implementation and Testing

5.1 Platform

5.1.1 Jupyter Notebook

The Jupyter Notebook is an incredibly powerful tool for interactively developing and presenting data science projects. This article will walk you through how to set up Jupyter Notebooks on your local machine and how to start using it to do data science projects.

Firstly, a notebook integrates code and its output into a single document that combines visualizations, narrative text, mathematical equations, and other rich media. This intuitive workflow promotes iterative and rapid development, making notebooks an increasingly popular choice at the heart of contemporary data science, analysis, and increasingly science at large.

On Windows, you can run Jupyter via the shortcut Anaconda adds to your start menu, which will open a new tab in your default web browser that should look something like the following screenshot.[15]

Fig 5.1: Jupyter Notebook on Windows

5.1.2 Tensorflow

TensorFlow is a Python-friendly open source library for numerical computation that makes machine learning faster and easier.

Machine learning is a complex discipline. But implementing machine learning models is far less daunting and difficult than it used to be, thanks to machine learning frameworks such as Google's TensorFlow that ease the process of acquiring data, training models, serving predictions, and refining future results.

Created by the Google Brain team, TensorFlow is an open source library for numerical computation and large-scale machine learning. TensorFlow bundles together a slew of machine learning and deep learning models and algorithms and makes them useful by way of a common metaphor. It uses Python to provide a convenient front-end API for building applications with the framework, while executing those applications in high-performance C++.[16]

5.2 Dataset

Each keypoints is specified by an (x, y) real-valued pair in the space of pixel indices. There are 15 keypoints, which represent the following elements of the face:

Left eye center, right eye center, left eye inner corner, left eye outer corner, right eye inner corner, right eye outer corner, left eyebrow inner end, left eyebrow outer end, right eyebrow inner end, right eyebrow outer end, nose tip, mouth left corner, mouth right corner, mouth center top lip, mouth center bottom lip. Left and right here refers to the point of view of the subject.

In some examples, some of the target keypoint positions are missing such as encoded as missing entries in the csv, i.e., with nothing between two commas.

The input image is given in the last field of the data files, and consists of a list of pixels ordered by row, as integers in (0,255). The images are 96x96 pixels.

Table 5.1: Dataset Indicating Several Keypoints

	0	1	2	3
left_eye_center_x	66.0336	64.3329	65.0571	65.2257
left_eye_center_y	39.0023	34.9701	34.9096	37.2618
right_eye_center_x	30.227	29.9493	30.9038	32.0231
right_eye_center_y	36.4217	33.4487	34.9096	37.2618
left_eye_inner_corner_x	59.5821	58.8562	59.412	60.0033
left_eye_inner_corner_y	39.6474	35.2743	36.321	39.1272
left_eye_outer_corner_x	73.1303	70.7227	70.9844	72.3147
left_eye_outer_corner_y	39.97	36.1872	36.321	38.381
right_eye_inner_corner_x	36.3566	36.0347	37.6781	37.6186
right_eye_inner_corner_y	37.3894	34.3615	36.321	38.7541
right_eye_outer_corner_x	23.4529	24.4725	24.9764	25.3073
right_eye_outer_corner_y	37.3894	33.1444	36.6032	38.0079
left_eyebrow_inner_end_x	56.9533	53.9874	55.7425	56.4338
left_eyebrow_inner_end_y	29.0336	28.2759	27.5709	30.9299
left_eyebrow_outer_end_x	80.2271	78.6342	78.8874	77.9103
left_eyebrow_outer_end_y	32.2281	30.4059	32.6516	31.6657
right_eyebrow_inner_end_x	40.2276	42.7289	42.1939	41.6715
right_eyebrow_inner_end_y	29.0023	26.146	28.1355	31.05
right_eyebrow_outer_end_x	16.3564	16.8654	16.7912	20.458
right_eyebrow_outer_end_y	29.6475	27.0589	32.0871	29.9093

After loading and visualizing data, missing data in columns are found using isnull() method of pandas library. And then missing data are filled using fillna() method. This method take a parameter which indicate the way how the missing data are filled. This either could be forward fill which means that cell will be filled using above cell data of that column. Otherwise it could be backward fill which means that the cell will be filled using below cell data of that column.

Table 5.2: Dataset Indicating Several Keypoints and Image

nose_tip_x	44.4206	48.2063	47.5573	51.8851
nose_tip_y	57.0668	55.6609	53.5389	54.1665
mouth_left_corner_x	61.1953	56.4214	60.8229	65.5989
mouth_left_corner_y	79.9702	76.352	73.0143	72.7037
mouth_right_corner_x	28.6145	35.1224	33.7263	37.2455
mouth_right_corner_y	77.389	76.0477	72.732	74.1955
mouth_center_top_lip_x	43.3126	46.6846	47.2749	50.3032
mouth_center_top_lip_y	72.9355	70.2666	70.1918	70.0917
mouth_center_bottom_lip_x	43.1307	45.4679	47.2749	51.5612
mouth_center_bottom_lip_y	84.4858	85.4802	78.6594	78.2684
Image	238 236 237 238 240 240 239 241 241 243 240 23...	219 215 204 196 204 211 212 200 180 168 178 19...	144 142 159 180 188 188 184 180 167 132 84 59 ...	193 192 193 194 194 194 193 192 168 111 50 12 ...

Then each image is splitted as a list of numbers and then push it into another list of images. Then the list of images is converted into numpy array and reshaped to 96x96 pixel shape. A random image that is drawn is look like an image like Fig: 5.2.

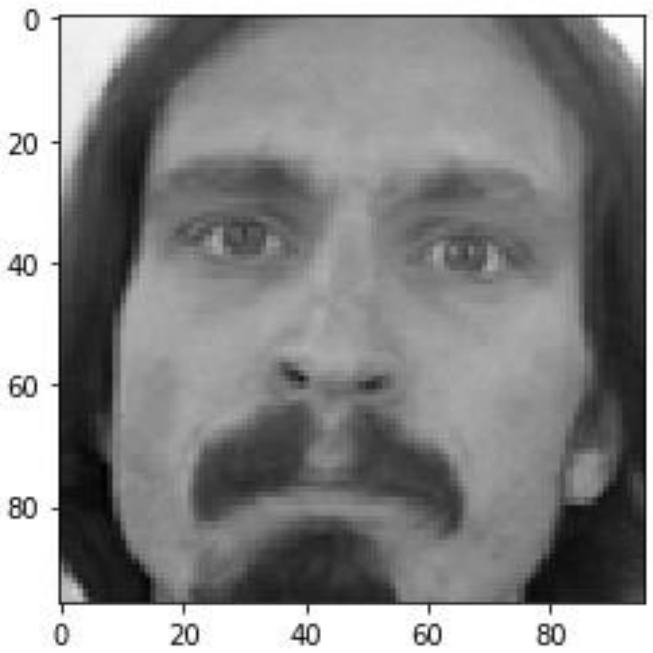

Fig 5.2: Image Data

Then a generic method is implemented to draw some number of images in left column and their corresponding images with labelled key points in the right column. A quick overview then can be done where non keypoints labelled images in the left side of the figure and image with keypoints labelled is in the right side of the figure.

Fig 5.3: Image Data without and with Labelling

5.3 2D Convolution

From the training dataset, we may denote images as X and keypoints labelling as Y. Then feature scaling is done by dividing images by 255 and model is to be prepared.

The sequential API allows you to create models layer-by-layer for most problems. It is limited in that it does not allow to create models that share layers or have multiple inputs or outputs.

Conv2D is a 2D Convolution Layer, this layer creates a convolution kernel that is wind with layers input which helps produce a tensor of outputs.

```
import tensorflow as tf
model= tf.keras.models.Sequential(

    layers=[

         #convolution 1st time
        tf.keras.layers.Conv2D(filters=32,kernel_size=(3,3),activation=tf.nn.relu,input_shape=(96,96,1)),
        tf.keras.layers.MaxPool2D(2,2),

         #convolution 2nd time
        tf.keras.layers.Conv2D(filters=32,kernel_size=(3,3),activation=tf.nn.relu,input_shape=(96,96,1)),
        tf.keras.layers.MaxPool2D(2,2),

         #convolution 3rd time
        tf.keras.layers.Conv2D(filters=64,kernel_size=(3,3),activation=tf.nn.relu,input_shape=(96,96,1)),
        tf.keras.layers.MaxPool2D(2,2),

        #input layer
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(units=526,activation='relu'),
        tf.keras.layers.Dense(units=526,activation='relu'),
        tf.keras.layers.Dropout(0.3),


        # number of keypoint
        tf.keras.layers.Dense(units=30,activation='relu')
    ]
)
```

Fig 5.4: Model

In image processing kernel is a convolution matrix or masks which can be used for blurring, sharpening, embossing, edge detection and more by doing a convolution between a kernel and an image.

Mandatory Conv2D parameter is the numbers of filters that convolutional layers will learn from. It is an integer value and also determines the number of output filters in the convolution.

Here we are learning a total of 32 filters and then we use Max Pooling to reduce the spatial dimensions of the output volume. As far as choosing the appropriate value for no. of filters, it is always recommended to use powers of 2 as the values.

This parameter kernel_size determines the dimensions of the kernel. Common dimensions include 1×1, 3×3, 5×5, and 7×7 which can be passed as (1, 1), (3, 3), (5, 5), or (7, 7) tuples. It is an integer or tuple/list of 2 integers, specifying the height and width of the 2D convolution window. This parameter must be an odd integer.[17]

ReLU stands for rectified linear unit, and is a type of activation function. Mathematically, it is defined as y = max(0, x). ReLU is the most commonly used activation function in neural networks, especially in CNNs. If you are unsure what activation function to use in your network, ReLU is usually a good first choice.[18]

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 94, 94, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 47, 47, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 45, 45, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 22, 22, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 20, 20, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 10, 10, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 6400)              0
_________________________________________________________________
dense (Dense)                (None, 526)               3366926
_________________________________________________________________
dense_1 (Dense)              (None, 526)               277202
_________________________________________________________________
dropout (Dropout)            (None, 526)               0
_________________________________________________________________
dense_2 (Dense)              (None, 30)                15810
=================================================================
Total params: 3,688,002
Trainable params: 3,688,002
Non-trainable params: 0
_________________________________________________________________
```

Fig 5.5: Model Summary

A dense layer represents a matrix vector multiplication. The values in the matrix are the trainable parameters which get updated during backpropagation. So, you get a m dimensional vector as output. A dense layer thus is used to change the dimensions of your vector. Mathematically speaking, it applies a rotation, scaling, translation transform to your vector.

A dropout layer is used for regularization where you randomly set some of the dimensions of your input vector to be zero with probability keep_prob. A dropout layer does not have any trainable parameters. To ensure that expected sum of vectors fed to this layer remains the same if

no dropout was applied, the remaining dimensions which are not set to zero are scaled by 1/keep_prob.

Flattening a tensor means to remove all of the dimensions except for one. A Flatten layer in Keras reshapes the tensor to have a shape that is equal to the number of elements contained in the tensor.[19]

Then we compile our model using adam optimizer with mean square error loss and accuracy metrics. The choice of optimization algorithm for your deep learning model can mean the difference between good results in minutes, hours, and days.

The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing.[20]

```
model.compile(optimizer='adam',
              loss='mse',
               metrics=['acc'])
```

```
hist=model.fit(x=X_train,y=y_train,batch_size=128,epochs=200,verbose=2,validation_split=0.2)
hist
```

```
Train on 5639 samples, validate on 1410 samples
Epoch 1/200
5639/5639 - 58s - loss: 607.3474 - acc: 0.2690 - val_loss: 92.0384 - val_acc: 0.6872
Epoch 2/200
5639/5639 - 59s - loss: 77.9744 - acc: 0.3160 - val_loss: 15.1014 - val_acc: 0.6872
Epoch 3/200
5639/5639 - 62s - loss: 47.7054 - acc: 0.3506 - val_loss: 19.7581 - val_acc: 0.6872
Epoch 4/200
5639/5639 - 64s - loss: 39.9947 - acc: 0.3600 - val_loss: 11.0960 - val_acc: 0.6872
Epoch 5/200
5639/5639 - 66s - loss: 35.6645 - acc: 0.3799 - val_loss: 6.4131 - val_acc: 0.6872
Epoch 6/200
5639/5639 - 65s - loss: 32.5133 - acc: 0.4006 - val_loss: 6.7715 - val_acc: 0.6872
Epoch 7/200
5639/5639 - 77s - loss: 29.6928 - acc: 0.4205 - val_loss: 9.2704 - val_acc: 0.6872
Epoch 8/200
5639/5639 - 65s - loss: 27.9307 - acc: 0.4205 - val_loss: 6.7373 - val_acc: 0.6872
Epoch 9/200
5639/5639 - 65s - loss: 27.5974 - acc: 0.4306 - val_loss: 15.1525 - val_acc: 0.6872
Epoch 10/200
```

Fig 5.6: Model Compilation

Mean Squared Error (MSE) is the workhorse of basic loss functions: it’s easy to understand and implement and generally works pretty well. To calculate MSE, you take the difference between your predictions and the ground truth, square it, and average it out across the whole dataset.

Regardless of whether your problem is a binary or multi-class classification problem, you can specify the ‘acc‘ metric to report on accuracy.[21]

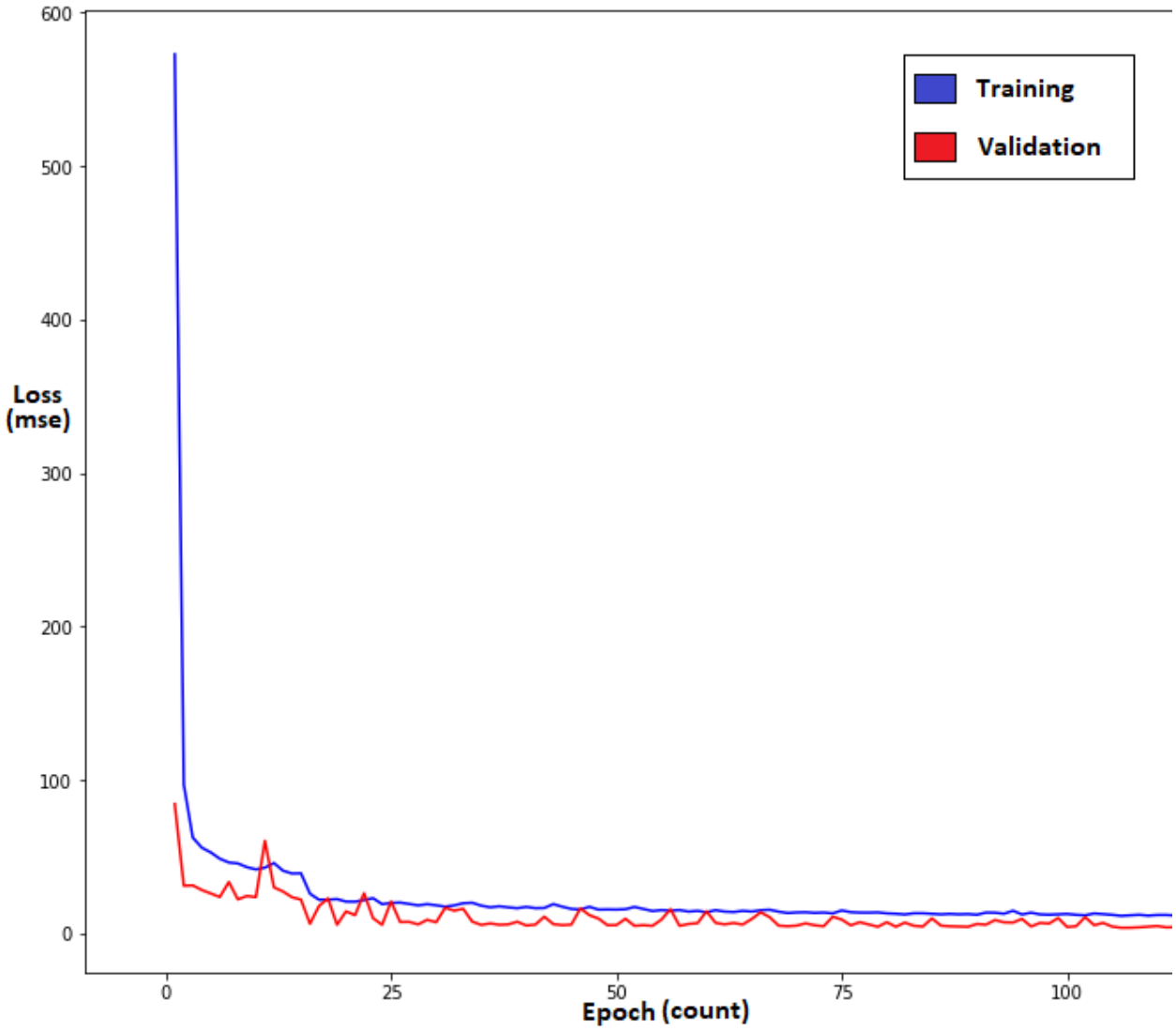

Fig 5.7: Loss per Epoch in Training and Validation

One Epoch is when an entire dataset is passed forward and backward through the neural network only once. Since one epoch is too big to feed to the computer at once we divide it in several smaller batches. The above chart indicating the loss per epoch on both Training dataset and Cross Validation dataset. The more the epochs are performed, the less the loss in both dataset.

5.4 Testing

The residual sum of squares, also known as the sum of squared residuals or the sum of squared errors, is the sum of the squares of residuals. It is a measure of the discrepancy between the data and an estimation model. A small residual sum of squares indicates a tight fit of the model to the data.[22]

It is used to measure performance of a model. sklearn r2_score module is equivalent of it.

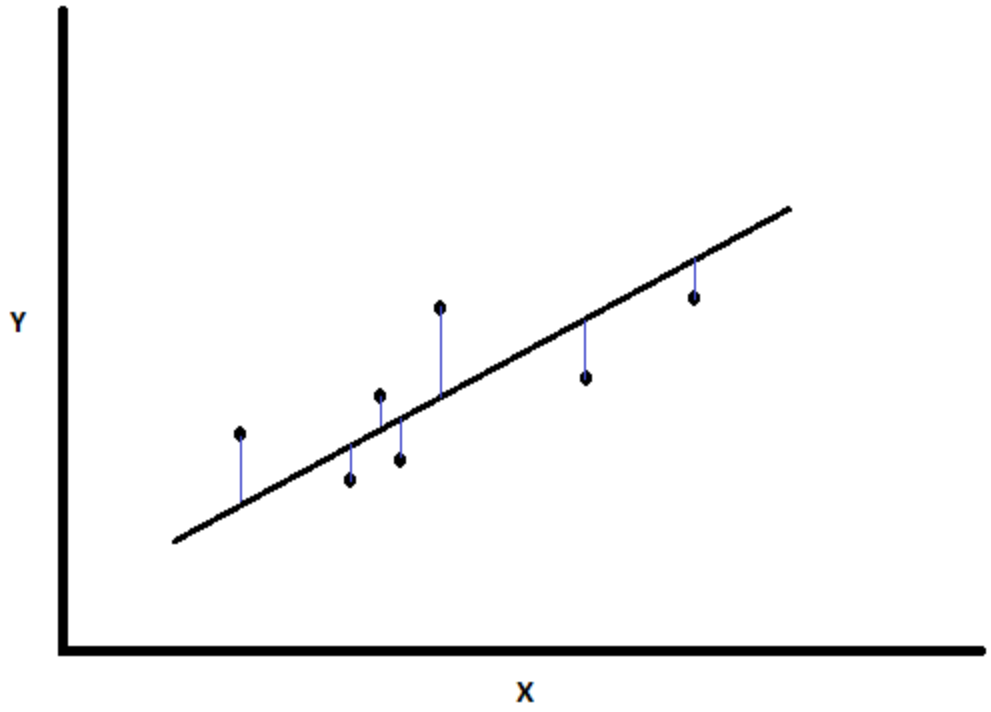

Fig 5.8: Residual Sum of Square

Hence, r2_score module from sklearn metrics package is imported to measure performance of the model. Prediction on training data is performed and y_pred is found which represents predicted different key points x and y value. Then it is compared with actual different key points x and y value using r2_score method. The less the score, the more the accurate model on data.

```
from sklearn.metrics import r2_score
```

```
y_pred = model.predict(X_train)

score = r2_score(y_train,y_pred)
print(score)
```

```
0.13463084013414037
```

Fig 5.9: Score of Residual Sum of Square

Then a python script is run to draw face images with actual keypoints which are given with the dataset and the face images with predicted keypoints which are predicted with our convolutional neural network algorithm. In that plot, actual keypoints are depicted in the left column of the plot and predicted keypoints are depicted in the right column of the plot. The plot looks like below figure Fig 5.11.

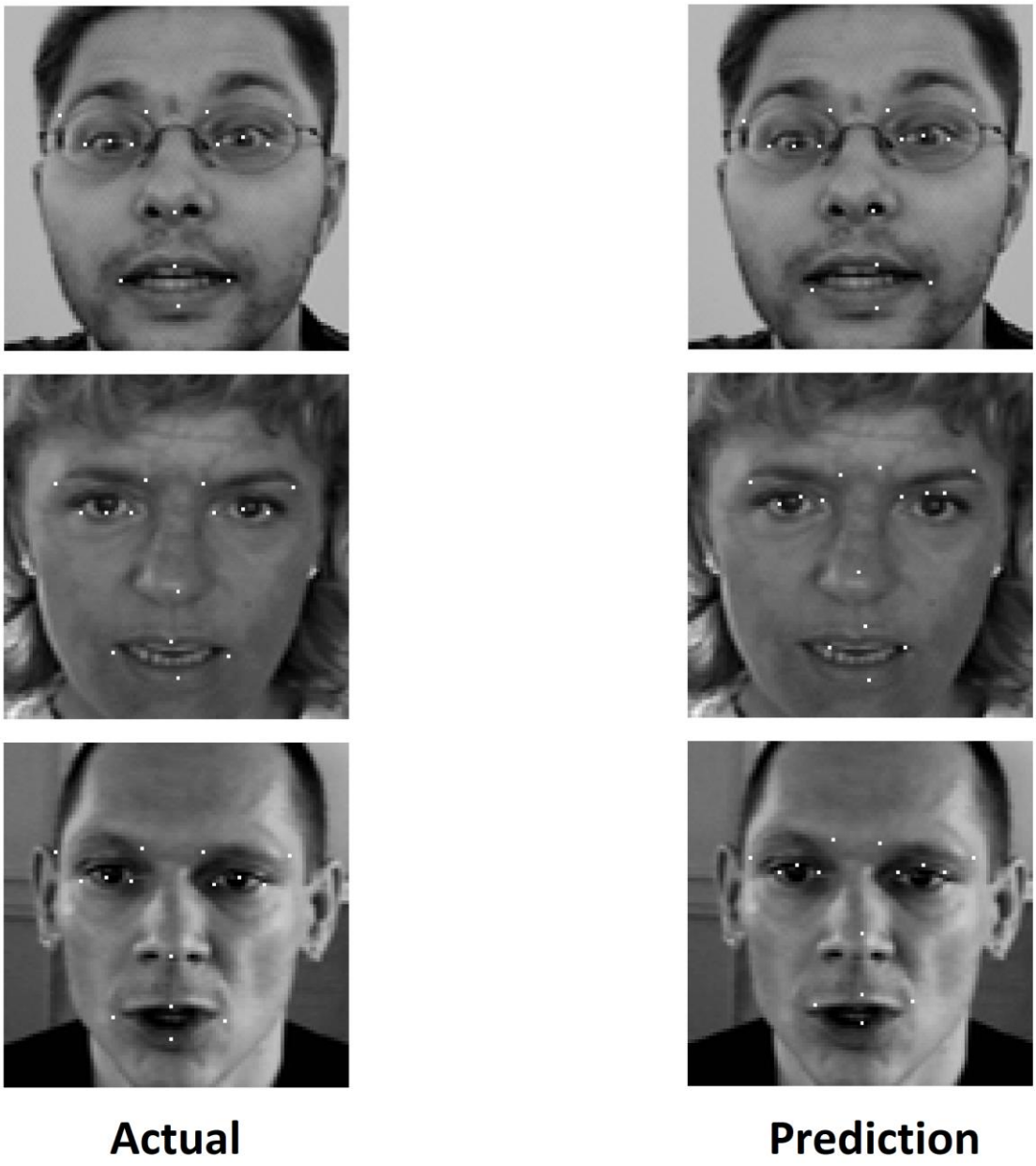

Fig 5.10: Actual Keypoints vs Prediction

It can be said that our model's performance is good enough to predict facial keypoints as it got 0.13(approximate) r2_score. Accuracy of predicted facial keypoints and differences with actual facial keypoints also can be visualized from above figure.

5.5 Features

Some features of the proposed system are:

- This system is a knowledge-based system
- It can be used in online learning system
- It is easy to evolve the model
- It is possible to automate the learning system

5.6 Merits of the System

Some merits of the proposed system are:

- Detect facial key points with high accuracy
- Easy to use in time series problem
- Low false positive rate
- Improved computing performance with short training time

5.7 Required Tools

5.7.1 Hardware Tools

Required hardware tools for the proposed system are:

- CPU: Core i7 at least 3.0 GHz and quad core
- RAM: 16 GB
- GPU: Nvidia graphics at least 3 GB
- Auxiliary Storage: 512 GB SSD with 1 TB HDD

5.7.2 Software Tools

Required software tools for the proposed system are:

- Python 3.7
- Anaconda 2019.07
- Jupyter Notebook 6.0.1
- Numpy 1.17
- Matplotlib 3.1.1
- Pandas 0.25.1
- Tensorflow 1.14
- Keras 2.2.5

5.8 Summary

In this chapter implementation and result are discussed in details. Platform of this system are Jupyter notebook and Tensorflow are discussed. Data is collected from google AI platform which contains facial image and keypoints. A convolutional neural network model is prepared to fit the training dataset and predict keypoints for test dataset. And finally, result is depicted using residual sum of square metrics and visual result is also provided using a plot.

Chapter 6

Conclusion and Future Works

6.1 Conclusion

We conclude that CNN, which is indeed a faster and more accurate method of deep learning, when applied facial keypoints detection gives result far better than simple networks. Also that the approach is not affected by imaging factors. Detecting facial keypoints is a very challenging problem. Facial features vary greatly from one individual to another due to position, viewing angle, and illumination conditions. Computer vision research has come a long way in addressing these difficulties, but there remain many opportunities for improvement. This proposed system will provide high accuracy performance to detect facial key points using latest trending technologies. Adjusting filter sizes, keep larger filters for data input layers and decreasing the size in subsequent layers, produces better result. With further tuning of networks and large data size, the network is bound to improve.

6.2 Future Works

Future work of this thesis is to build more accurate machine learning model so that it will be able to determine facial key points better. Suitable, advanced and new machine learning methods will be applied to enhance the system. The portable binary version of the mini project will be made and package will also be available so that other developers can include it easily. Also there is a lot that can decrease the complexity of processing such as: Simply try changing nodes in the hidden layer of the new improved structure to reach an optimal value, With better computing resources trying to train data with more epochs then the root square mean equation will be better, Adding some layers of preprocessing just before of two ending layers can help reduce the complexity of the problem together with gaining accuracy.

References

[1] A. Chandra, "Facial Key Points Detection", *Towards Data Science*, 2018.

[2] S. Shi, "Machine Learning", *Cornell University*, 2017.

[3] K. Corporation, "Facial Keypoints Detection", Kaggle.com, 2016. Available: https://www.kaggle.com/c/facial-keypoints-detection.

[4] A. Sharma, "Face detection: Artificial Intelligence", *IIT Kanpur*, 2015.

[5] J. P Dandale, "Face Detection and Security", *International Journal of Computer Science and Information Security*, 2010.

[6] O. Corporation, "Face Recognition with OpenCV — OpenCV 2.4.13.7 documentation", *Docs.opencv.org*, 2013. Available: https://docs.opencv.org/2.4.13.7/modules/contrib/doc/facerec/facerec_tutorial.html.

[7] K. McNulty, "Getting Started to Machine Learning", *Towards Data Science*, 2018.

[8] N. Kumar, "Getting started with Machine Learning", *GeeksforGeeks*, 2017. Available: https://www.geeksforgeeks.org/getting-started-machine-learning.

[9] D. Karunakaran, "Intro to deep learning", *Medium*, 2018. Available: https://medium.com/intro-to-artificial-intelligence/deep-learning-series-1-intro-to-deep-learning-abb1780ee20.

[10] S. Saha, "Convolutional Neural Network", *Towards Data Science*, 2018.

[11] Lopez & Ruiz, "Facial Key Points Detection", *Super Data Science*, 2017.

[12] S. Raval, "Machine Learning System", *The School of AI*, 2017.

[13] A. Géron, *Hands-on machine learning with Scikit-Learn and TensorFlow*, 1st ed. 2017.

[14] H. Pokharna, "Convolutional Neural Network Explained", *Technology Made Easy*, 2016.

[15] B. Pryke, "Intro to Jupyter Notebook", *Data Quest*, 2019.

[16] C. Kozyrkov, "Tensorflow for Beginners", *Hackernoon*, 2018.

[17] M. Walia, "Keras Conv2D Class", *GeeksforGeeks*, 2016. Available: https://www.geeksforgeeks.org/keras-conv2d-class/.

[18] D. Liu, "A Practical Guide to ReLU", *Super Data Science*, 2017.

[19] K. Gak, "Tensorflow Layers", *Data Science Stack Exchange*, 2019.

[20] J. Brownlee, "Adam Optimization", *Data Science Mastery*, 2017.

[21] J. Brownlee, "Error Measurement Metrics", *Data Science Mastery*, 2019.

[22] S. Glen, "Residual Sum of Squares", *Data Science Central*, 2014.