IMAGE ENHANCEMENT USING AUTOENCODERS

A
MINI PROJECT REPORT

Submitted in the partial Fulfillment of the requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY
IN
ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted By

1.CH.PremChand - 20R21A0412

UNDER THE GUIDANCE OF

DR K. Nishanth Rao
Associate Professor

MARRI LAXMAN REDDY
GROUP OF INSTITUTIONS

MLR Institute of Technology
(Autonomous)
(Affiliated to JNTUH, Hyderabad)
Dundigal, Hyderabad-500043
2020-2024

MARRI LAXMAN REDDY GROUP OF INSTITUTIONS

MLR Institute of Technology

(Autonomous)

(Affiliated to JNTUH, Hyderabad)

Dundigal, Hyderabad-500043

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the project *entitled "**IMAGE ENHANCEMENT USING AUTOENCODER**" is the bonafied work done by, CH.PremChand(20R21A0412)* in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering from MLR Institute of Technology, Hyderabad, during the academic year 2021-22.

Internal Guide
DR K.NISHANTH RAO

Head of the Department
DR S.V.S PRASAD

External Examiner

ACKNOWLEDGEMENT

We express our profound thanks to the management of **MLR Institute of Technology**, Dundigal, Hyderabad, for supporting us to complete this project.

We take immense pleasure in expressing our sincere thanks to **Dr.K.Srinivasa Rao**, Principal, MLR Institute of Technology, for his kind support and encouragement.

We are very much grateful to **Dr S.V.S Prasad**, Professor & Head of the Department, MLR Institute of Technology, for encouraging us with his valuable suggestions.

We are very much grateful to **Dr.Nishanth Rao, Associate Professor** for his unflinching cooperation throughout the project.

We would like to express our sincere thanks to the teaching and non-teaching faculty members of ECE Dept., MLR Institute of Technology, who extended their help to us in making our project work successful. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

Project associates:

CH.SVS.Srikar	20R21A0411
CH.PremChand	20R21A0412
P.Ganesh	20R21A0440
P.Bharath Reddy	20R21A0446

ABSTRACT

Image Enhancement, is the act of denoising the low resolution images. With the explosion in the number of digital images taken every day, the demand for more accurate and visually pleasing images is increasing. However, the images captured by modern CCTV cameras are degraded by noise, which leads to distorted visual image quality. Therefore, work is required to reduce noise without losing image features (edges, corners, and other sharp structures). In the criminal investigation field, images are the principal forms for investigation and for probing crime detection. The imaging science applied in criminal investigation is face detection, surveillance camera imaging, and crime scene analysis.

This problem statement is quite familiar. You may all have faced problems with distorted images at some point and hence would have tried to enhance the image quality. Well, due to advances in deep learning techniques, we'll try to enhance the resolution of images by training a auto-encoder convolution neural network using LFW dataset. Denoising auto-encoders, consists of an encoder, decoder and several hidden layers. In this paper the quality of image is improved using deep learning methods such as auto-encoder. Reconstruction of input image by passing through the neural network and validating with the ground truth image, non-linear sigmoid function is used as activation function for dealing with complex weights. Suspect identification across different blurred images is made simple.

One of the fundamental challenges in the field of image processing and computer vision is image denoising, where the underlying goal is to estimate the original image by suppressing noise from a noise-contaminated version of the image. Image noise may be caused by different intrinsic (i.e., sensor) and extrinsic (i.e., environment) conditions which are often not possible to avoid in practical situations. Therefore, image denoising plays an important role in a wide range of applications such as image restoration, visual tracking, image registration, image segmentation, and image classification, where obtaining the original image content is crucial for strong performance. While many algorithms have been proposed for the purpose of image denoising, the problem of image noise suppression remains an open challenge, especially in situations where the images are acquired under poor conditions where the noise level is very high.

Our project aims to solve this challenge, we investigate an alternate approach to the problem of Image Denoising based on data-adaptive optimization via Autoencoders.

Disadvantages of Gaussian and wiener filters is they degrade image details and the edges of the image. Therefore ,de-noised image would be blurred. Coming to Markov Random Field (MRF) -based methods ,it preserve fine structure to some extend.Our Auto-encoder model reduces noise and preserves edges to a higher extend. The noise degrades performance of image processing algorithms. Image denoising methods are important image processing algorithms which are used to reduce the noise. Auto-encoder denoising is one of the most important parts of crime detection tools.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER-1
INTRODUCTION

1.1Overview

As images give as more information about the certain situation or person. In any organization a record of the activities is maintained in the form of books, databases, surveillance to incline with an able management of the organization. These records help trace and track any of the investigation within the organization during occasions of crime.

Problem Statement – Enhance Image Resolution using Autoencoder

You'll be quite familiar with the problem statement here. most of us have struggled with clicking blurred images and struggling to enhance their resolution. Well, we'll solve that problem using autoencoders here!

Let's say we have a set of images of people's faces in low resolution. Our task is to enhance the resolution of these images. It can be done with the help of photo editing tools such as Photoshop. However, when there are thousands of images at hand, we need a much smarter way to do this task.

Image enhancement tools are often classified into point operations and spatial operations. Point operations include contrast stretching, noise clipping, histogram modification, and pseudo-coloring. Point operations are, in general, simple nonlinear operations that are well known in the image processing literature and are covered elsewhere. Spatial operations used in image processing today are, on the other hand, typically linear operations. The reason for this is that spatial linear operations are simple and easily implemented. Although linear image enhancement tools are often adequate in many applications, significant advantages in image enhancement can be attained if nonlinear techniques are applied. Nonlinear methods effectively preserve edges and details of images while methods using linear operators tend to blur and distort them. Additionally, nonlinear

image enhancement tools are less susceptible to noise. Noise is always present due to the physical randomness of image acquisition systems. For example, under-exposure and low-

light conditions in analog photography conditions lead to images with film-grain noise, which, together with the image signal itself, are captured during the digitization process.

Surveillance cameras are used to record clippings of the activities surrounding any of the public places around the organization. The police make use of the cameras installed by organization to investigate any crime related activities. As images play a vital role in Investigation, the detection and analysis of the same is of vital importance in today's digital world of Internet. Image enhancement is a challenging task, because images are complex 2dimensional data. There is no common pattern on how to analyze them. We define a DAE system as a collection of methodologies that process and improve the quality in them. Such a System can find use in a wide variety of application areas like Image search engine, Crime analysis. In this study we attempt to enhance underlying features in input image by using auto-encoders.

Image is a medium by which one expresses how a person looks in real life without his presence. Images play an important factor in sensitive job areas like crime analysis, social media applications and many others where one has to store their image. Enhancing images is a tough task as every individual has a different feature and intensity. To enhance the quality of the image, is the goal of this paper.

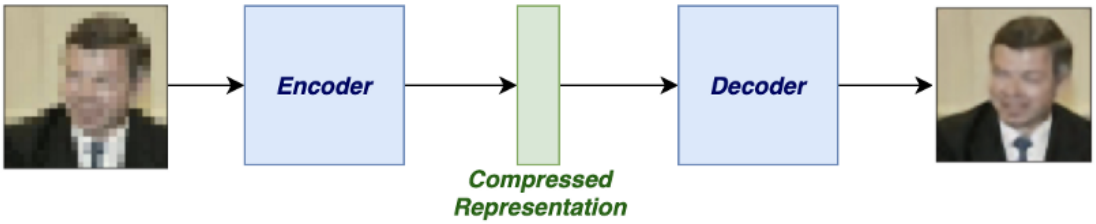

Fig-1.1.1(ARCHITECTURE OF AUTOENCODER)

Now we can try to use this autoencoder to enhance the image quality, but if we understood the architecture correctly, It will try to reconstruct the original input we feed to the network. So if we send a low-resolution image then it will only reconstruct the noisy image.

To achieve the image enhancement trough encoders, we can do a minor change in the way we calculate the loss function. instead of calculating the loss using input(low-resolution

image) and reconstructed image we can use the clear image and reconstructed image. So the model will learn to reconstruct the clear image instead of the low-resolution input image.

Image enhancement plays an important role in image processing. There are many techniques available to remove different types of noises from the image. However, the best technique that able to remove the noise and keep image details. The enhancement of an image is divided in two categories which are linear and nonlinear model. Generally linear model is fast in removing noise but unable to preserve image details. Whereas, the nonlinear model can remove the noise and preserve the details of image but with more time [3]. The noise domination or noise removal is considering as an essential task in images processing that the researcher are focused on.

Generally, the most used methods for implementing the noise in images include a Gaussian, uniform, or salt-and-pepper distribution. The Gaussian Noise is used to model natural noise processes that happen from electronic noise in the image acquisition system. Figure 2 presents an example of spin image and the effects of Gaussian noise (AF: Average Filter; AAF: Adaptive Average Filter ; MF: Median Filter ; WMF; Weighted Median Filter ; WCMF: Weighted Coefficients Median Filter). Uniform Noise is beneficial that because it uses to generate any other type of noise distributed. Also, it applies to corrupt images in the restoration algorithms. And Salt and Pepper's noise is usually produced by errors in the data transmission. Also, Salt and pepper noise it can cause as result of an error in data transmission or a defect pixel elements in camera sensors, etc. Usually, the noise appears in the image that creates using a scanner because of darker dots and disturbances. Sometimes, the image interpreted wrongly because the noise cannot overcome in some cases. Removal of noise is considered one of the significant tasks in computer vision and image enhancement processing since unwanted noise leads to the erroneousness in the image. Filtering techniques are used to reduce the noise for enhancing images and sharpening them as illustrated in Figure1.1.2.

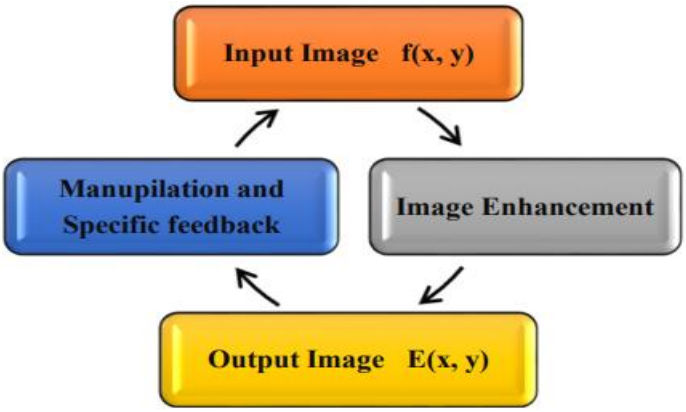

Fig-1.1.2(Filtering techniques)

Image Super Resolution

Image Super Resolution refers to the task of enhancing the resolution of an image from low-resolution (LR) to high (HR). It is popularly used in the following applications: Surveillance: to detect, identify, and perform facial recognition on low-resolution images obtained from security cameras.

Super resolution image reconstruction is to obtain a high-resolution image from a sequence of low-resolution images which are degraded by unknown blur, noise, and down sample.

II. TYPES OF DIGITAL IMAGE

In the field of image processing, it is useful to include both types of image analogue and digital. The mathematical model of computing the value of image depends on the calculation of two variables (X direction and Y direction). The digital image is represented in a two-dimensional array of discrete values of the image. Several techniques are used to enhance and process an image. Therefore, it is recommended to define the type of image first and then choose the suitable method to enhance it. An image is defined as 2D function f(x, y). Where x and y is spatial coordinate and f is amplitude of any pair (x, y) is called intensity level of the image at any point. Image has a finite number of elements, which has a specific value and location. Each element called a pixel. The digital image is a numeric representation of predefined pixels. The digital image can be classified into three categories as shown in Figure 1.1.2.

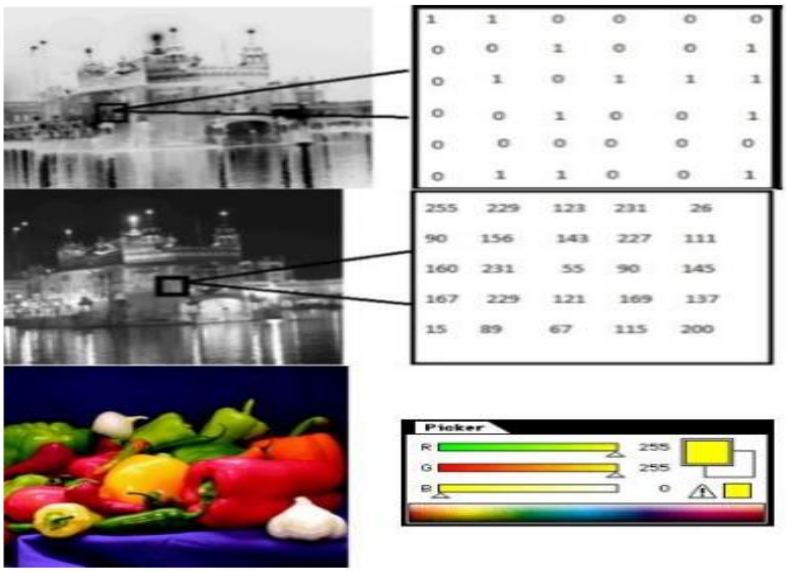

Fig-1.1.3(IMAGE IN 2D ARRAY)

The four main type of image which area, Binary image, Black & White Image, 8-bit image and last one is 64-bit image.

Image formed by Binary elements: Binary elements images can understand as image which has pixels of 0 & 1 element only and where 0 refers as black color and 1 refers as white color. You also have heard the name Monochrome image. This is also a binary elements image.

Image formed by Black and White elements: These types of images only have two colors that are black & white and whole pixel contains only these two colors for formation of digital image.

Image formed by 8-bit color elements: This is gray scale elements which formed image by using 256 different types of color shades. We can extract color from this number as 0, 255, 127 which means 0 give black color, 255 used for white color and 127 used for gray.

Image formed by 16-bit elements: This formation comes in colored image category and has 65, 536 colors in its range. You can understand it as high format color range. You have heard a color mode name which is RGB that means Red, Green and Blue. This comes in 16-bit elements image. These were some of mostly discussed image format and it think now you have a little bit idea about image formation. Now let us move the fundaments of processing of digital image. For doing anything in proper way there must by a fundament steps and by following that fundament steps result of processing of that thing will be accurate.

1.2Previous Case Study

1.2.1 Case study 1: The Boston marathon bombings, at 2:49 PM on April 15, 2013 two bombs exploded near the finish line of the annual Boston Marathon killing three people and injuring 264. The first reports about the about the terrorist attack were spread through Twitter.

Technology certainly played a crucial role in the hunt for the two Boston Marathon bombers, whether it was important smartphone pictures from the public or the forward-looking infrared cameras that spotted suspect No. 2 in the backyard boat.

But the best piece of evidence may have come from a Lord and Taylor department store camera, which provided video of one of the suspects dropping his backpack at the spot of the bombing.

Yes, we like our privacy. To some of us, it’s uncomfortable to think that cameras seem to be always watching our every move. But the fact is that surveillance cameras helped the FBI solve this complicated crime and allowed the city of Boston to live in peace.

1.2.2 Case study 2: CCTV helped snare failed terrorists

On Thursday, 21 July 2005, four attempted bomb attacks by Islamist extremists disrupted part of London's public transport system as a follow up attack from the 7 July 2005 London bombings that occurred two weeks earlier. The explosions occurred around midday at Shepherd's Bush, Warren Street and Oval stations on the London Underground, and on London Buses route 26 in Bethal Green on Hackney Road. A fifth bomber dumped his device without attempting to set it off.

Soon after the failed July 21 bombings, grainy CCTV images of those suspected of carrying out the attacks were released by police. In each case, they caught a moment of escape and, along with eyewitness reports by commuters, provided crucial identification evidence that would lead to the would-be bombers' capture.

1.2.3 Case Study 3: Conviction Through Enhanced Fingerprint Identification

A short time later, investigators assigned to the case witnessed a demonstration of fingerprint image enhancement at a forensic conference. Faced with a dead-end murder investigation, they decided to try the technique on the unidentifiable pillowcase fingerprint from the crime scene. Investigators took the best DFO photograph and shipped it to a facility with the capability to perform image enhancement. Throughout the enhancement process,the accuracy of the print was documented through photographic records of each stage. Within 4 hours, the enhancement yielded an identifiable print.

1.3 Purpose of the project

Image Enhancement is the part of image processing which is gaining more popularity ad need for it increases enormously. Although there are methods to enhance images using machine learning techniques, this project attempts to use deep learning to enhance the quality of input image.

DAE is used in crime investigation according to the image, identifying the suspect in a location is made easy. A blurred image of the suspect is denoised using this neural network to get the quality image from it.

Image search engine perform dimensional reduction to find similar images across Internet. Auto-encoder extracts useful features to compare across different images and return equivalent images.

Image enhancement is used when we need to focus or pick out some important features of an image. Certain spectral components of an image may need to be enhanced in images obtained from telescopes or space probes. In some cases, the contrast may need to be enhanced.

1.4 Motivation

Image enhancement is the process of adjusting digital images so that the results are more suitable for display or further image analysis. For example, you can remove noise, sharpen, or brighten an image, making it easier to identify key features. Image enhancement is an essential technique used in many imaging applications. The main motivation of image enhancement is processing an image to be more suitable for specific utilization. Inference of the input image helps in further crime investigation. This helps in various aspects, and also can be used in many different applications.

CHAPTER-2

Literature Survey

2.1 Video surveillance image enhancement via a convolutional neural network and stacked denoising auto encoder

Authors: Muhamad Faris Che Aminuddin & Shahrel Azmin Suandi

In an extensive-scale surveillance system, the quality of the surveillance camera installed varies. This variation of surveillance camera produces different image quality in terms of resolution, illumination, and noise. The quality of the captured image depends on the surveillance camera hardware, placement and orientation, and the surrounding light. A pixelated, low illumination and noisy image produced by a low-quality surveillance camera causes critical issues for video surveillance face recognition systems. To address these issues, a deep learning image enhancement (DLIE) model is proposed. By utilizing a deep learning architecture such as a convolutional neural network (CNN) and a denoising auto encoder, the image quality can be enhanced.

2.2 Image Resolution Enhancement Using Convolutional Auto encoders

Authors: Pawan Kumar and Nikita Goel

Resolution is an important characteristic to determine the nature and features of the image. Enhancing the resolution strengthens the features hidden within the image, and make the image sharper and more informative. The image quality is improved when noise is removed/suppressed from it. The proposed model provides a technique to enhance the resolution of different types of images, obtained from imaging devices, using an auto encoder.

2.3 Enhancing Image Resolution and Denoising Using Auto encoder

Authors: Ch. Sekhar, M. Srinivasa Rao & K. Venkata Rao, A. S. Keerthi Nayani

Nowadays, in real time, image processing is involved in various sectors like security, health care, banking, and face recognition. While capturing an image, there is more chance of noise engaged with multiple aspects of the surroundings. To improve the quality of the image and to get better classification results, we need to clean the picture, which is called pre-processing of the image.

2.4 Image Processing Operations Identification via Convolutional Neural Network

Authors: Bolin Chen, Haodong Li

In recent years, image forensics has attracted more and more attention, and many forensic methods have been proposed for identifying image processing operations. Up t o now, most existing methods are based on hand crafted features, and just one specific operation is considered in their methods. In many forensic scenarios, however, multiple classification for various image processing operations is more practical. Besides, it is difficult to obtain effective features by hand for some image processing operations. In this paper, therefore, we propose a new convolutional neural network (CNN) based method to adaptively learn discriminative features for identifying typical image processing operations.

CHAPTER-3
METHODOLOGY

3.1 PROPOSED SYSTEM

Therefore, this paper proposes a deep learning-based method to enhance the input image at a satisfactory accuracy. Our method chooses Auto-encoder as the baseline, latent representation of the features in the image are encoded into the feature map. Reconstruct of the original image is performed by decoding the feature map. Improvements caused by every modification are discussed in this paper

3.2ADVANTAGES OF PROPOSED SYSTEM

1.These results demonstrate the robustness and feasibility of our model.

2.At last, after obtaining a satisfactory model, a graphical user interface (GUI) is designed to make our model more user-friendly.

3.Inference of the image is performed very fast.

3.3 SYSTEM REQUIREMENTS

3.3.1 HARDWARE REQUIREMENTS

Minimum hardware requirements are very dependent on the particular software being developed by a given though Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

- Processor : Intel i5
- Ram : 8 GB
- GPU : 16GB

3.3.2 SOFTWARE REQUIREMENTS

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regards to what the areas of strength and deficit are and how to tackle them.

- Operating System: Windows/Linux
- Programming Language: Python 3.6/ Python 3.7

3.3.3 FUNCTIONAL REQUIREMENTS

Functional requirements are represented or stated in the form of input to be given to the system, the operation performed and the output expected. System should collect the data from any resources. All the collected data should be processed for proper use, some analysis should be done for understanding the data properly.

1. Upload LFW Dataset
2. Image Pre-Processing
3. Model Generation
4. Upload Test Image
5. Enhance the Image

3.3.4 Applications of image enhancement

Here are some examples of image enhancement:

- Deblur images
- Contrast adjustment
- Brighten an image

- Smooth and sharpen**:-**Image smoothing is a digital image processing technique that reduces and suppresses image noises. Commonly seen smoothing filters include average smoothing, Gaussian smoothing, and adaptive smoothing.

  Image sharpening filters highlight edges by removing blur.

- **Noise removal**

  Noises are introduced to images at the point of capture from cameras, printing, or during transmission. In terms of image processing, noises can be identified with a variance of intensity from its neighbor pixels. There are various types of noises. For example, Gaussian noise changes each pixel by a (usually) small amount, and salt-and-pepper noise (impulse noise) randomly scatters white or black pixels over the image. Noise removal techniques reduce the visibility of noises by smoothing the image using linear or non-linear filters.

- **Grayscale image histogram equalization**

  Histogram equalization refers to the transformation where an output image has approximately the same number of pixels at each gray level, i.e., the histogram of the output is uniformly distributed.

3.4 TECHNOLOGIES USED

3.4.1 PYTHON

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on.

3.4.2 MACHINE LEARNING

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy. Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, uncovering key insights within data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, the market demand for data scientists will increase, requiring them to assist in the identification of the most relevant business questions and subsequently the data to answer them.

Machine learning is a growing technology which enables computers to learn automatically from past data. Machine learning uses various algorithms for building mathematical models and making predictions using historical data or information**.** Currently, it is being used for various tasks such as image recognition**,** speech recognition**,** email filtering**,** Facebook auto-tagging**,** recommender system, and many more.

This machine learning tutorial gives you an introduction to machine learning along with the wide range of machine learning techniques such as Supervised**,** Unsupervised**,** and Reinforcement learning. You will learn about regression and classification models, clustering methods, hidden Markov models, and various sequential models.

3.5 DEEP LEARNING

Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability—allowing it to “learn” from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy. Deep learning drives many artificial intelligence (AI) applications and services that improve automation, performing analytical and physical tasks without human intervention. Deep learning technology lies behind everyday products and services (such as digital assistants, voice-enabled TV remotes, and credit card fraud detection) as well as emerging technologies (such as self-driving cars).

Working:

First, we need to identify the actual problem in order to get the right solution and it should be understood, the feasibility of the Deep Learning should also be checked (whether it should fit Deep Learning or not). Second, we need to identify the relevant data which should correspond to the actual problem and should be prepared accordingly. Third, Choose the Deep Learning Algorithm appropriately. Fourth, Algorithm should be used while training the dataset. Fifth, Final testing should be done on the dataset.

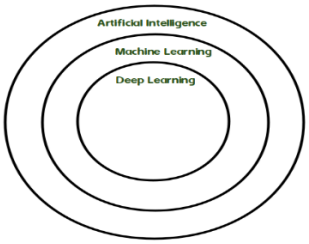

Fig-3.5.1(Machine Learning REPRESENTATION)

3.5 MODULES USED IN PROJECT

3.5.1 NUMPY

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 Nd array object. This encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance.

- In NumPy, the array object is commonly known as ndarray. Numpy provides a lot of supporting functions for performing operations on its array object and with these functions, working with ndarray becomes very easy.
- Also, the NumPy arrays are more compact than Python Lists in terms of the size.
- NumPy uses much less memory in order to store data and it provides an easy mechanism of specifying the data types. Thus code can be optimized easily.

Now you must be thinking, that how NumPy works faster than lists. Don't worry, we have an answer for your question.

NumPy arrays are mainly stored at one continuous place in memory contrary to lists. Thus you can access and manipulate them very efficiently and this behavior is commonly known as locality of reference. Due to this reason, Numpy is faster than lists. Numpy is optimized to work with latest CPU architecture.

Like we mentioned above, NumPy is also used along with packages like SciPy (Scientific Python) and Matplotlib (plotting library in python).

This combination is mainly a replacement for MatLab(which is a popular platform for technical computing). Also, Python is an alternative to MatLab and is now seen as a modern and complete programming language.

3.5.2 MATPLOTLIB

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open-source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications. Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats.
- Embed in JupyterLab and Graphical User Interfaces.

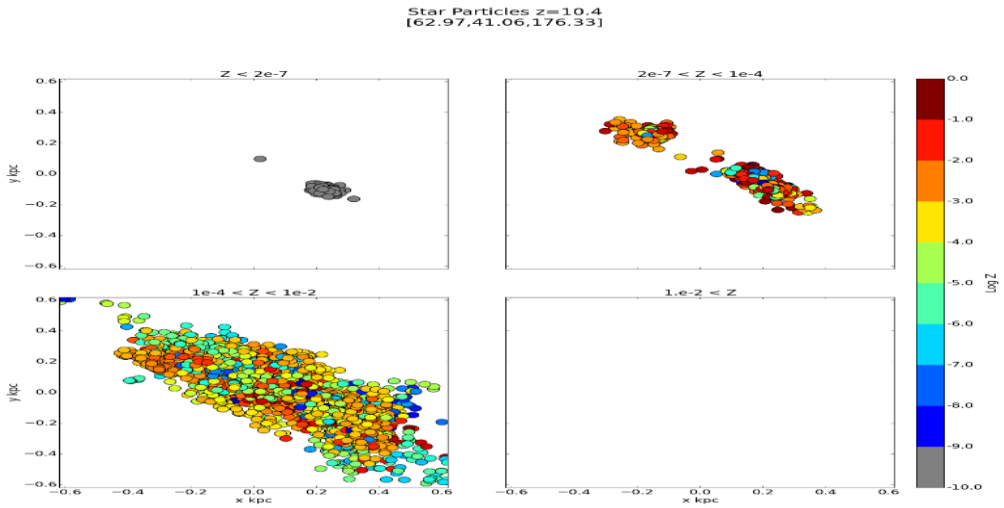

Fig-3.5.2.1(MATPLOTLIB)

3.5.3 SKLEARN

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Implementation of Sklearn

Scikit-learn is mainly coded in Python and heavily utilizes the NumPy library for highly efficient array and linear algebra computations. Some fundamental algorithms are also built in Cython to enhance the efficiency of this library. Support vector machines, logistic regression, and linear SVMs are performed using wrappers coded in Cython for LIBSVM and LIBLINEAR, respectively. Expanding these routines with Python might not be viable in such circumstances.

Scikit-learn works nicely with numerous other Python packages, including SciPy, Pandas data frames, NumPy for array vectorization, Matplotlib, seaborn and plotly for plotting graphs, and many more.

Components of scikit-learn:

Scikit-learn comes loaded with a lot of features. Here are a few of them to help you understand the spread:

- Supervised learning algorithms: Think of any supervised machine learning algorithm you might have heard about and there is a very high chance that it is part of scikit-learn. Starting from Generalized linear models (e.g Linear Regression), Support Vector Machines (SVM), Decision Trees to Bayesian methods – all of them are part of scikit-learn toolbox. The spread of machine learning algorithms is one of the big reasons for the high usage of scikit-learn.
- Cross-validation: There are various methods to check the accuracy of supervised models on unseen data using sklearn.
- Unsupervised learning algorithms: Again there is a large spread of machine learning algorithms in the offering – starting from clustering, factor analysis, principal component analysis to unsupervised neural networks.
- Various toy datasets: This came in handy while learning scikit-learn. I had learned SAS using various academic datasets (e.g. IRIS dataset, Boston House prices dataset).

- Feature extraction: Scikit-learn for extracting features from images and text (e.g., Bag of words)

3.5.4 TENSORFLOW

TensorFlow allows developers to create *dataflow graphs*—structures that describe how data moves through a graph, or a series of processing nodes. Each node in the graph represents a mathematical operation, and each connection or edge between nodes is a multidimensional data array, or *tensor*.

TensorFlow applications can be run on most any target that's convenient: a local machine, a cluster in the cloud, iOS and Android devices, CPUs or GPUs. If you use Google's own cloud, you can run TensorFlow on Google's custom TensorFlow Processing Unit (TPU) silicon for further acceleration. The resulting models created by TensorFlow, though, can be deployed on most any device where they will be used to serve predictions.

TensorFlow provides multiple APIs (Application Programming Interfaces). These can be classified into 2 major categories:

1. Low level API:
   - complete programming control
   - recommended for machine learning researchers
   - provides fine levels of control over the models
   - TensorFlow Core is the low-level API of TensorFlow.
2. High level API:
   - built on top of TensorFlow Core
   - easier to learn and use than TensorFlow Core
   - make repetitive tasks easier and more consistent between different users
   - tf.contrib.learn is an example of a high level API.

3.5.5 KERAS

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result as fast as possible is key to doing good research.*

Keras is:

- **Simple** -- but not simplistic. Keras reduces developer *cognitive load* to free you to focus on the parts of the problem that really matter.

- **Flexible** -- Keras adopts the principle of *progressive disclosure of complexity*: simple workflows should be quick and easy, while arbitrarily advanced workflows should be *possible* via a clear path that builds upon what you've already learned.
- **Powerful** -- Keras provides industry-strength performance and scalability: it is used by organizations and companies including NASA, YouTube, or Waymo.

Keras is the high-level API of TensorFlow 2: an approachable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning. It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity.

Keras empowers engineers and researchers to take full advantage of the scalability and cross-platform capabilities of TensorFlow 2: you can run Keras on TPU or on large clusters of GPUs, and you can export your Keras models to run in the browser or on a mobile device.

Fig-3.5.5.1(KERAS)

3.5.6 OPENCV

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being an Apache 2 licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people

of user community and estimated number of downloads exceeding 18 million. the library is used extensively in companies, research groups and by governmental bodies.

The steps to read and display an image in OpenCV are:

1. Read an image using imread() function.

2. Create a GUI window and display image using imshow() function.

3. Use function waitkey(0) to hold the image window on the screen by the specified number of seconds, o means till the user closes it, it will hold GUI window on the screen.

4. Delete image window from the memory after displaying using destroyAllWindows() function.

Computer vision allows the computer to perform the same kind of tasks as humans with the same efficiency. There are a two main task which are defined below:

- **Object Classification -** In the object classification, we train a model on a dataset of particular objects, and the model classifies new objects as belonging to one or more of your training categories.
- **Object Identification -** In the object identification, our model will identify a particular instance of an object - for example, parsing two faces in an image and tagging one as Virat Kohli and other one as Rohit Sharma.

OpenCV

Functionality

- Image/video I/O, processing, display (core, improc, high Gui)
- Object/feature detection (obj detect, features2d)
- Geometry-based monocular or stereo computer vision (calib3d, stitching, video stab)
- Computational photography (photo, videos)
- Machine learning & clustering (ML)
- CUDA acceleration (GPU)

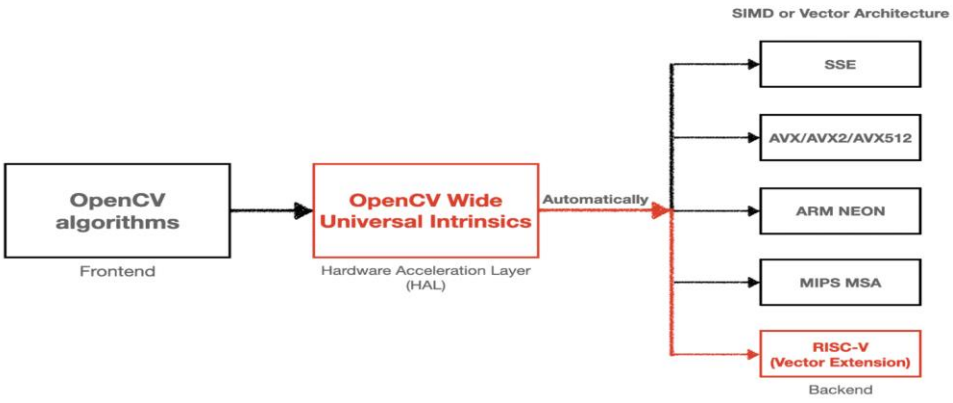

Fig-3.5.6.1(OPENCV)

3.5.7 GLOB

The glob module finds all the pathnames matching a specified pattern according to the rules used by the Unix shell, although results are retuned in arbitrary order. No tilde expansion is done, but *,?, and character ranges expressed with [] will be correctly matched.

glob.**glob**(*pathname*, *, *root_dir=None*, *dir_fd=None*, *recursive=False*, *include_hidden=False*)

Return a possibly empty list of path names that match *pathname*, which must be a string containing a path specification.

With the help of the Python glob module, we can search for all the path names which are looking for files matching a specific pattern (which is defined by us). The specified pattern for file matching is defined according to the rules dictated by the Unix shell. The result obtained by following these rules for a specific pattern file matching is returned in the arbitrary order in the output of the program. While using the file matching pattern, we have to fulfil some requirements of the glob module because the module can travel through the list of the files at some location in our local disk. The module will mostly go through those lists of the files in the disk that follow a specific pattern only.

3.5.8 TQDM

tqdm is a library in Python which is used for creating Progress Meters or Progress Bars. tqdm got its name from the Arabic name **taqaddum** which means ‘progress’.

Implementing tqdm can be done effortlessly in our loops, functions or even Pandas. Progress bars are pretty useful in Python.

How are the tqdm progress bars used?

Using Python for processes like data scraping and training machine learning modules is often all about running excessively large loops behind the scenes and scraping through huge datasets. The processes can take a long time and there is no way of knowing whether the kernel is at all working. This is where the progress bars come in.

You can use a progress bar to

- Monitor the progress of a process through visual representation
- Determine whether the process is at all working
- Get an estimated time of completion

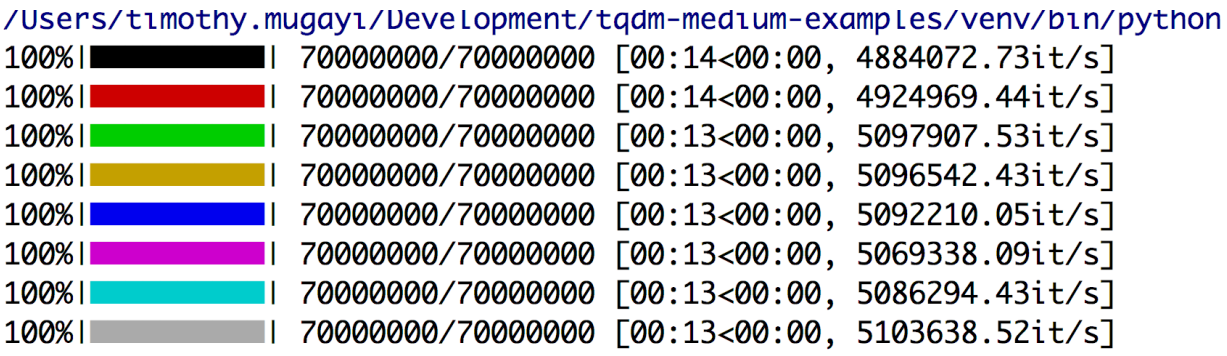

Fig-3.5.8.1(TQDM OUTPUT)

3.6 SOFTWARE DEVELOPMENT LIFE CYCLE

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process

Stage 1 : Planning and Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical

feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

Stage 2 : Defining Requirements

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an SRS (Software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle.

Stage 3 : Designing the Product Architecture

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

Stage 4 : Building or Developing the Product

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

Stage 5 : Testing the Product

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

Stage 6 : Deployment in the Market and Maintenance

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization.The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

3.7 PROPOSED SYSTEM ARCHITECTURE

AUTO ENCODERS

Autoencoders are an unsupervised learning technique in which we leverage neural networks for the task of representation learning. Specifically, we'll design a neural network architecture such that we impose a bottleneck in the network which forces a compressed knowledge representation of the original input.

As visualized below, we can take an unlabeled dataset and frame it as a supervised learning problem tasked with outputting x_hat, a reconstruction of the original input x.

“Auto encoders are essentially neural network architectures built with the objective of learning the lower-dimensional feature representations of the input data.”

Autoencoders are comprised of two connected networks – encoder and decoder. The aim of an encoder is to take an input (x) and produce a feature map (z):

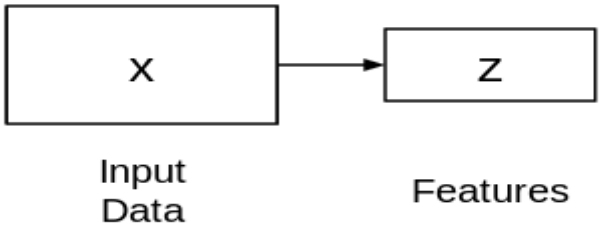

Fig-3.7.1

Since we want z to capture only the meaningful factors of variations that can describe the input data, the shape of z is usually smaller than x.

Now, the question is how do we learn this feature representation (z)? How do we train this model? For that, we can add a decoder network on top of the extracted features and then train the model:

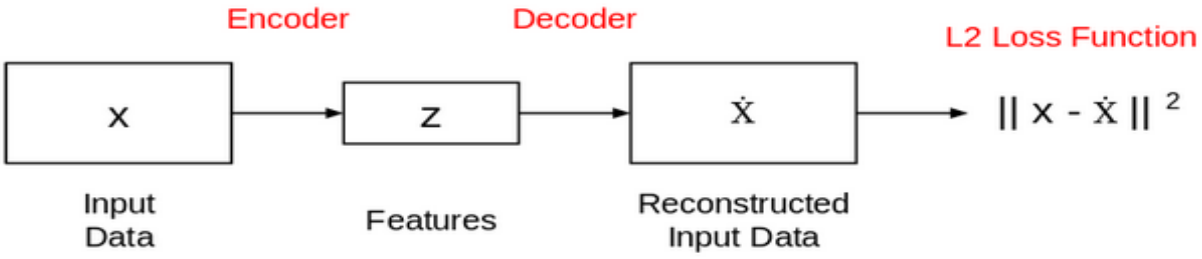

Fig-3.7.2(AUTO ENCODER)

This is what a typical autoencoder network looks like. This network is trained in such a way that the features (z) can be used to reconstruct the original input data (x). If the output (Ẋ) is different from the input (x), the loss penalizes it and helps to reconstruct the input data.

The data that moves through an autoencoder isn't just mapped straight from input to output, meaning that the network doesn't just copy the input data. There are three components to an autoencoder: an encoding (input) portion that compresses the data, a component that handles the compressed data (or bottleneck), and a decoder (output) portion. When data is fed into an autoencoder, it is encoded and then compressed down to a smaller size. The network is then trained on the encoded/compressed data and it outputs a recreation of that data.

So why would you want to train a network to just reconstruct the data that is given to it? The reason is that the network learns the "essence", or most important features of the input data. After you have trained the network, a model can be created that can synthesize similar

data, with the addition or subtraction of certain target features. For instance, you could train an autoencoder on grainy images and then use the trained model to remove the grain/noise from the image.

A small tweak is all that is required here. Instead of using the input and the reconstructed output to compute the loss, we can calculate the loss by using the ground truth image and the reconstructed image. This diagram illustrates my point wonderfully:

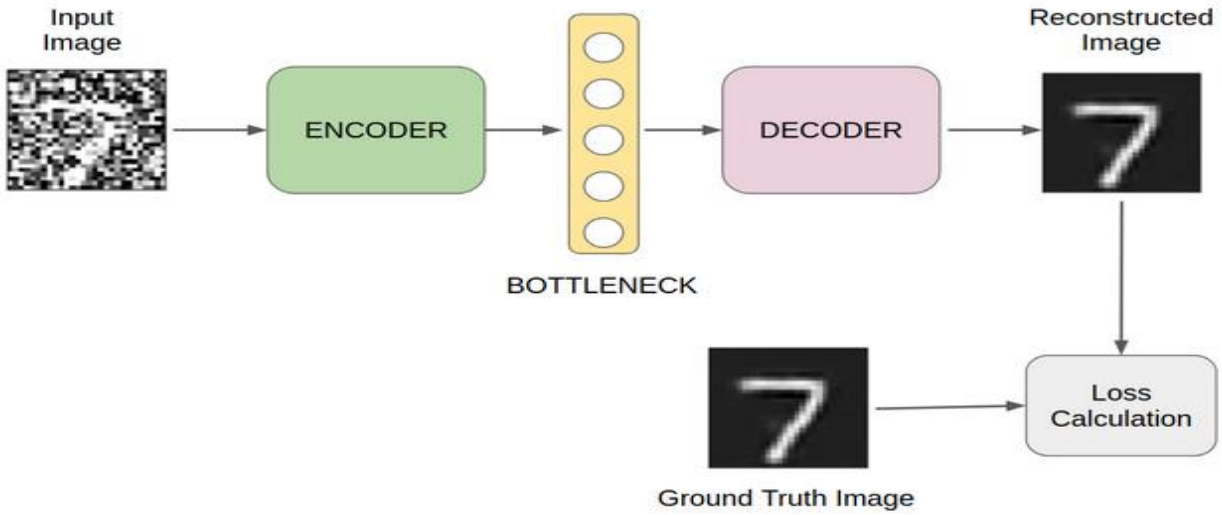

Fig-3.7.3(Model training)

An autoencoder consists of three components:

- **Encoder**: An encoder is a feedforward, fully connected neural network that compresses the input into a latent space representation and encodes the input image as a compressed representation in a reduced dimension. The compressed image is the distorted version of the original image.
- **Code**: This part of the network contains the reduced representation of the input that is fed into the decoder.
- **Decoder**: Decoder is also a feedforward network like the encoder and has a similar structure to the encoder. This network is responsible for reconstructing the input back to the original dimensions from the code.

Application of autoencoders

So far we have seen a variety of autoencoders and each of them is good at a specific task. Let's find out some of the tasks they can do

Data Compression

Although auto encoders are designed for data compression yet they are hardly used for this purpose in practical situations. The reasons are:

- **Lossy compression**: The output of the auto encoder is not exactly the same as the input, it is a close but degraded representation. For lossless compression, they are not the way to go.
- **Data-specific**: Auto encoders are only able to meaningfully compress data similar to what they have been trained on. Since they learn features specific for the given training data, they are different from a standard data compression algorithm like jpeg or gzip. Hence, we can't expect an auto encoder trained on handwritten digits to compress landscape photos.

Since we have more efficient and simple algorithms like jpeg, LZMA, LZSS(used in WinRAR in tandem with Huffman coding), auto encoders are not generally used for compression. Although auto encoders have seen their use for image denoising and dimensionality reduction in recent years.

Image Denoising

Auto encoders are very good at denoising images. When an image gets corrupted or there is a bit of noise in it, we call this image a noisy image.

To obtain proper information about the content of the image, we perform image denoising.

It is a process to reserve the details of an image while removing the random noise from the image as far as possible.

We classify the image denoising filters into 2 broad categories -
1). Traditional Filters - Filters which are traditionally used to remove noise from images. These filters are further divided into Spatial domain filters and Transform domain filters.
2). Fuzzy based Filters - Filters which include the concept of fuzzy logic in their filtering procedure.

Image Restoration

The process of recovering degraded or corrupted image by removing the noise or blur, to improve the appearance of the image is called image restoration. The degraded image is the convolution of the original image, degraded function, and additive noise. Restoration of the image is done with the help of prior knowledge of the noise or the disturbance that causes the degradation in the image. It can be done in two domains: spatial domain and frequency domain. In spatial domain the filtering action for restoring the images is done directly on the operating pixels of the digital image and in frequency domain the filtering action is done by mapping the spatial domain into the frequency domain, by fourier transform. After the filtering, the image is remapped by inverse fourier transform into spatial domain, to obtain the restored image. We

can choose any of the domains based on the applications required.

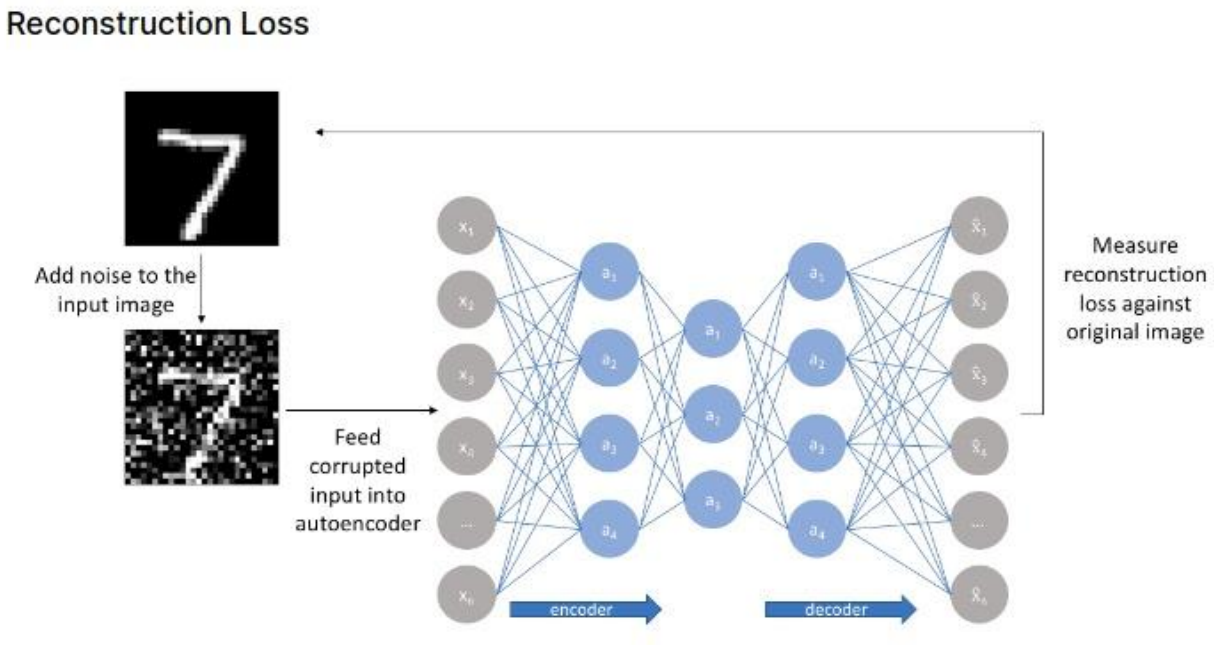

Fig-3.7.4(Reconstruction loss)

Image Data Compression

Compression, as the name implies, deals with techniques for reducing the storage required to save an image, or the bandwidth required to transmit it, without degrading the quality of an image to an unacceptable level. Compression allows to store more images in a given amount disk or memory space. It also reduces the time required for images to be transmitted over the Internet or downloaded from web pages. Compression techniques can be of two types:-lossy and lossless compression. Lossy compressions are irreversible, some data from the original image file is lost. Lossy methods are especially suitable for natural images such as photographs in applications where minor loss of fidelity is acceptable to achieve a substantial reduction in bit rate. In lossless compression, we can reduce the size of an image without any quality loss. Lossless compression is preferred for archival purposes and often for medical imaging, technical drawings, clip art, or comics.

Some of the common image compression techniques are:

• Fractal • Wavelets • Chroma sub sampling • Transform coding • Run-length encoding

Image Segmentation

Image Segmentation is the process of partitioning a digital image into multiple segments, to simplify and/or change the representation of an image into something that is more

meaningful and easier to analyze. It may make use of statistical classification, thresholding , edge detection, region detection, or any combination of these techniques. Usually a set of classified elements is obtained as the output of the segmentation step. Segmentation techniques can be classified as either region-based or edge-based. The former techniques rely on common patterns in intensity values within a cluster of neighboring pixels, and the goal of the segmentation algorithm is to group regions according to their anatomical or functional roles. The edge-based techniques rely on discontinuities in image values between distinct regions, and the goal of the segmentation algorithm is to accurately demarcate the boundary separating these regions.

Morphological Processing

Morphological processing is a collection of linear operations that is used for extracting image components that are useful in the representation and description of shape. Structuring element is a small set to probe an image under study. Structuring element is compared with the corresponding neighborhood of pixels by positioning it, at all possible locations in the image. Some operations check whether the element "fits" within the neighborhood, while others check whether it "hits" or intersects the neighborhood. The basic morphological operations are dilation, erosion and their combinations. Erosion is useful for the removal of structures of certain shape and size ,which is given by the structuring element while dilation is useful in filling of holes of certain size and shape given by the structuring element.

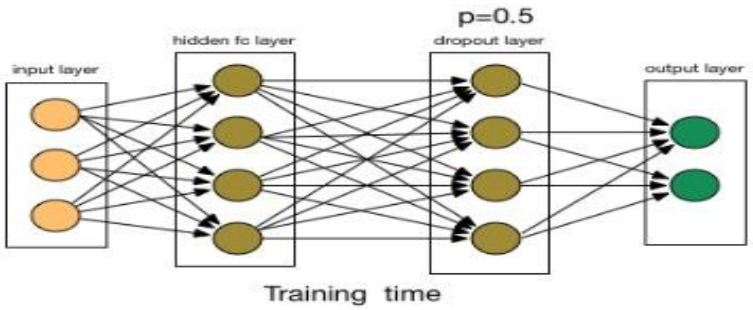

Fig-3.7.5(Neural network)

3.8 FLOW CHART

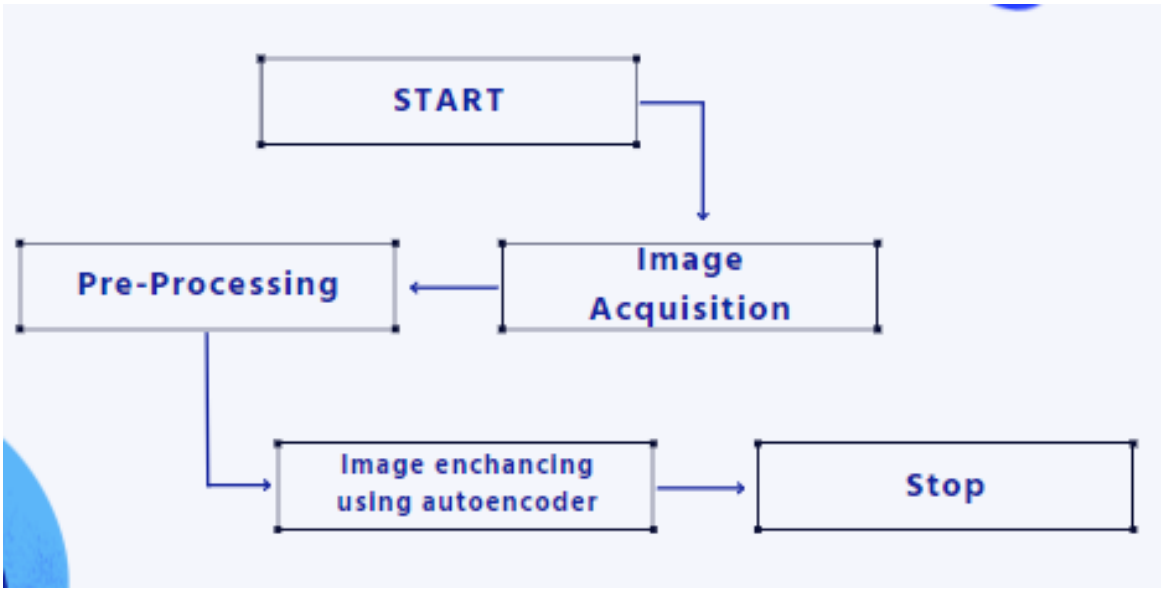

Figure 3.8.1(FlowChart)

The model starts taking input image of people from ‘labeled faces in wild’ dataset and then preprocessing of trained dataset it will identify the feature of images.The model is trained on the dataset.And download the model in the folder

3.9 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML. The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

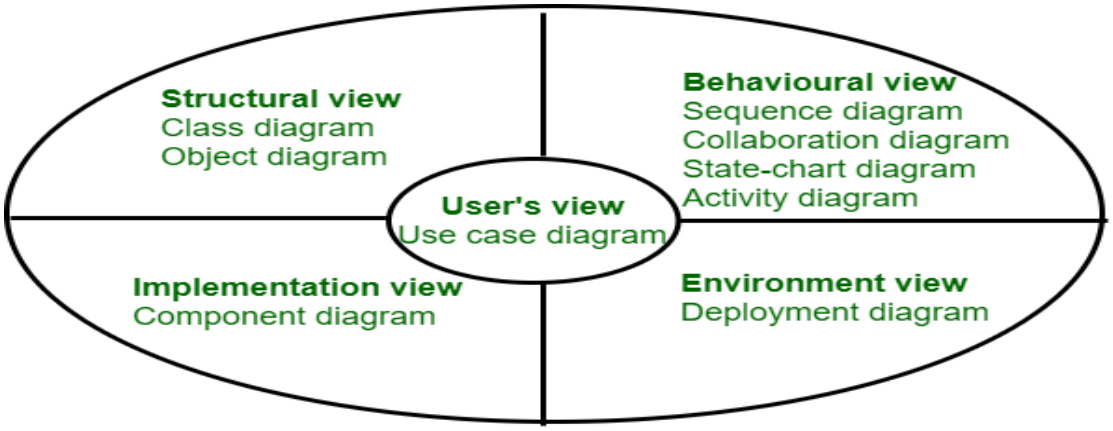

Fig-3.9.1(UML diagrams)

GOALS :

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

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.

2. Provide extendibility and specialization mechanisms to extend the core concepts.

3. Be independent of particular programming languages and development process.

4. Provide a formal basis for understanding the modeling language.

3.9.1 CLASS DIAGRAM

Class diagrams are the most common diagrams used in UML. Class diagram consists of classes, interfaces, associations, and collaboration. Class diagrams basically represent the object-oriented view of a system, which is static in nature.

Active class is used in a class diagram to represent the concurrency of the system.

Class diagram represents the object orientation of a system. Hence, it is generally used for development purpose. This is the most widely used diagram at the time of system construction.

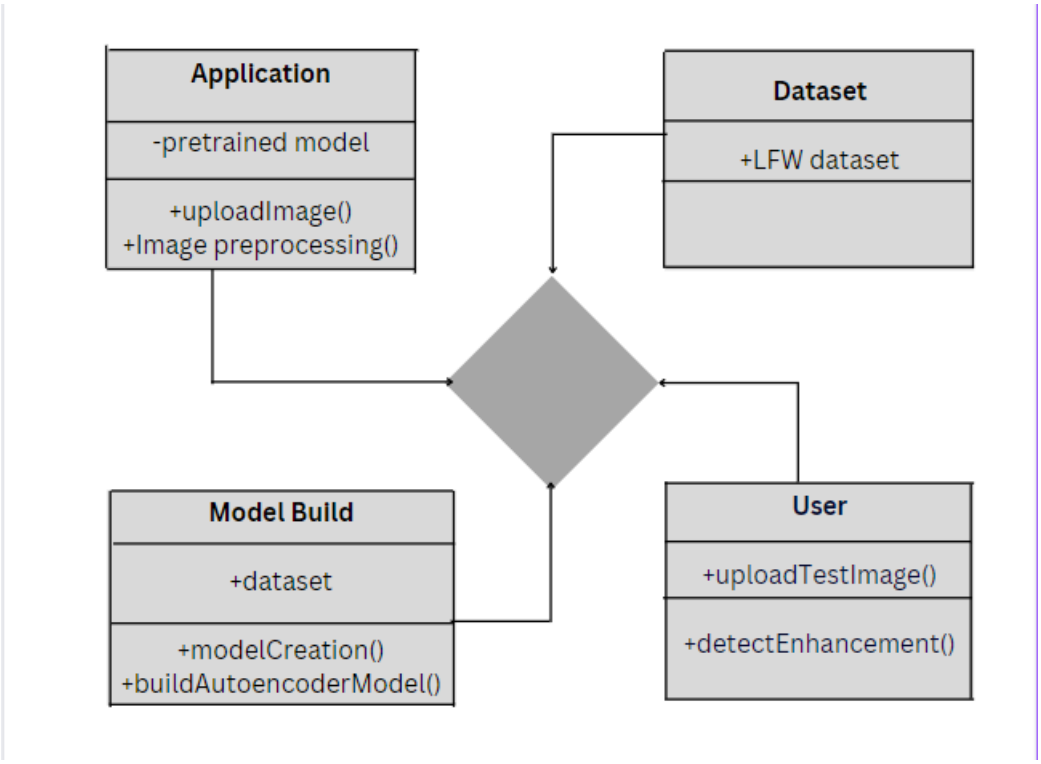

FIGURE 3.9.1.1(Class diagram)

The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities.

3.9.2 USE CASE DIAGRAM

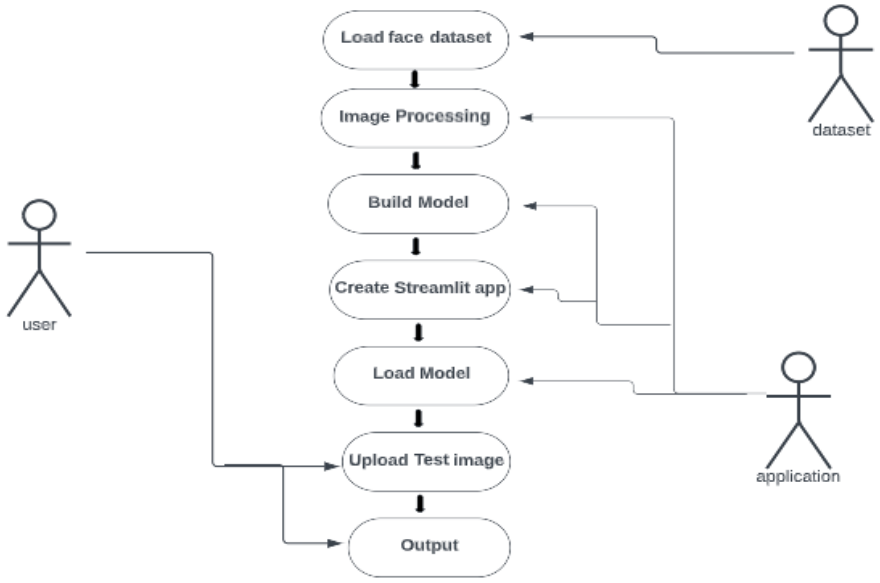

FIGURE 3.9.2.1(Use Case Diagram)

Use case diagrams are a set of use cases, actors, and their relationships. They represent the use case view of a system.

A use case represents a particular functionality of a system. Hence, use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as actors.

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

3.9.3 ACTIVITY DIAGRAM

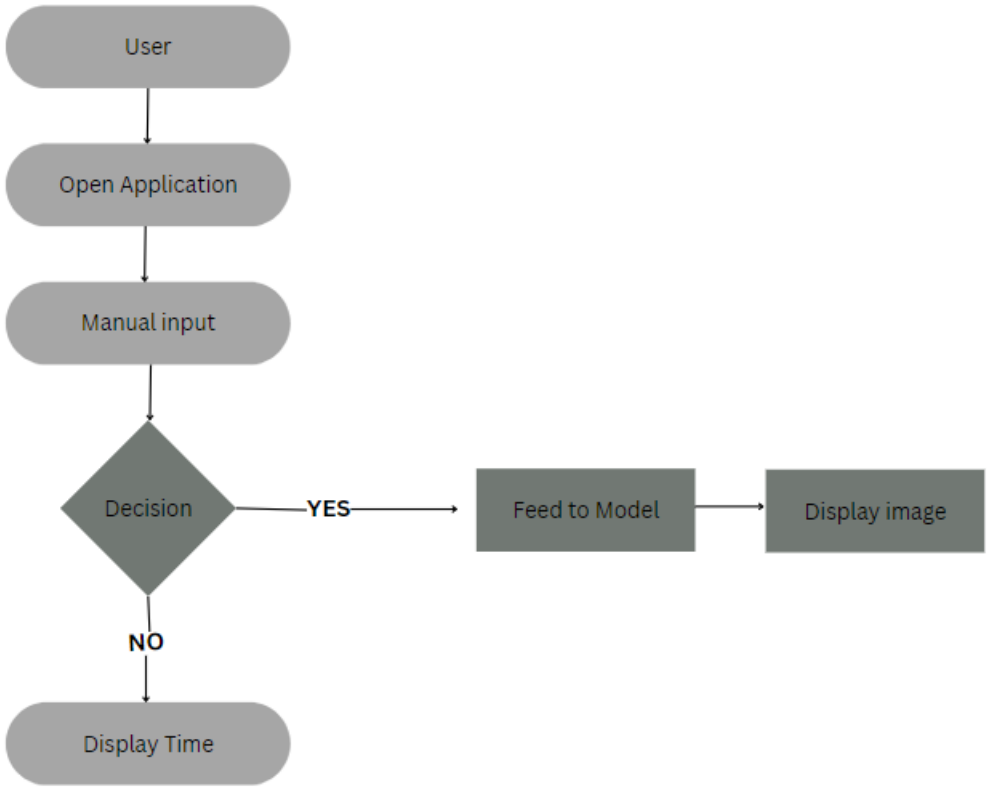

FIGURE 3.9.3.1(Activity Diagram)

Activity diagram describes the flow of control in a system. It consists of activities and links. The flow can be sequential, concurrent, or branched.Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a system.

Activity diagrams are used to visualize the flow of controls in a system. This is prepared to have an idea of how the system will work when executed.

Note − Dynamic nature of a system is very difficult to capture. UML has provided features to capture the dynamics of a system from different angles.

3.10 IMPLEMENTATION

3.10.1 CODING PART(COLAB NOTEBOOK)

```
import cv2

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

 Import pyngrok

from tensorflow.keras import Model, Input, regularizers

from tensorflow.keras.layers import Dense, Conv2D, MaxPool2D, UpSampling2D

from tensorflow.keras.callbacks import EarlyStopping

from keras.preprocessing import image

 import glob

from tqdm import tqdm

import warnings;

warnings.filterwarnings('ignore')
```

PREPARE THE DATASET

```
! wget http://vis-www.cs.umass.edu/lfw/lfw.tgz

! tar -xvzf lfw.tgz

face = glob.glob('lfw/**/*.jpg')

face=face[:6000]
```

```
all = []
 for i in tqdm(face):
  img = image.load_img(i, target_size=(100,100,3))
  img = image.img_to_array(img)
  img = img/255
  all.append(img)
all_images = np.array(all)
 # split data into train and validation data
train_x, val_x = train_test_split(all_images, random_state=32, test_size=0.1)
```

IMAGE PREPROCESSING

```
def reduce_image(image, percent = 40):
  width = int(image.shape[1] * percent / 100)
  height = int(image.shape[0] * percent / 100)
  dim = (width, height)
  small_image = cv2.resize(image, dim, interpolation = cv2.INTER_AREA)
  # scale back to original size
  width = int(small_image.shape[1] * 100 / percent)
  height = int(small_image.shape[0] * 100 / percent)
  dim = (width, height)
   low_image = cv2.resize(small_image, dim, interpolation = cv2.INTER_AREA)
 return low_image
train_x_px = []
 for i in range(train_x.shape[0]):
```

```
 temp = reduce_image(train_x[i,:,:,:])

 train_x_px.append(temp)

train_x_px = np.array(train_x_px)

val_x_px = []

for i in range(val_x.shape[0]):

 temp = reduce_image(val_x[i,:,:,:])

 val_x_px.append(temp)

val_x_px = np.array(val_x_px)
```

MODEL BUILDING

```
Input_img = Input(shape=(100, 100, 3))

x1 = Conv2D(256, (3, 3), activation='sigmoid', padding='same',kernel_regularizer=regularizers.l1(10e-10))(Input_img)

x2 = Conv2D(128, (3, 3), activation='sigmoid', padding='same',kernel_regularizer=regularizers.l1(10e-10))(x1)

x2 = MaxPool2D( (2, 2))(x2)

encoded = Conv2D(64, (3, 3), activation='sigmoid', padding='same',kernel_regularizer=regularizers.l1(10e-10))(x2)

# decoding architecture

x3 = Conv2D(64, (3, 3), activation='sigmoid', padding='same',kernel_regularizer=regularizers.l1(10e-10))(encoded)

x3 = UpSampling2D((2, 2))(x3)

x2 = Conv2D(128,(3, 3), activation='sigmoid', padding='same',kernel_regularizer=regularizers.l1(10e-10))(x3)

x1 = Conv2D(256, (3, 3), activation='sigmoid', padding='same',kernel_regularizer=regularizers.l1(10e-10))(x2)
```

```
decoded = Conv2D(3, (3, 3), padding='same',kernel_regularizer=regularizers.l1(10e-10))(x1)

autoencoder = Model(Input_img, decoded)

autoencoder.compile(optimizer='adam', loss='mse',metrics=['accuracy'])

autoencoder.summary()
```

MODEL TRAINING

```
early_stopper = EarlyStopping(monitor = 'val_loss',min_delta = 0, patience = 9,verbose = 1,restore_best_weights = True)

hist = autoencoder.fit(train_x_px, train_x,

        epochs=50,

        batch_size=1,

        shuffle=True,

        validation_data=(val_x_px, val_x),

        callbacks=[early_stopper])
```

SAVING VALIDATION IMAGES

```
from PIL import Image

for i in range(180,200):

   im = Image.fromarray((val_x_px[i] * 255).astype(np.uint8))

   im.save(str(i)+".jpg")
```

MODEL VALIDATION

```
predictions = autoencoder.predict(val_x_px)

plt.figure(figsize=(20,8))

plt.plot(hist.history['loss'])

plt.plot(hist.history['val_loss'])

plt.title('model loss')
```

```
plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['train', 'val'], loc='upper left')

plt.show()
```

TESTING

```
n = 0

for i,m in zip(val_x_px,val_x):

    img,mask = i,m

    if n < 20:

        fig, axs = plt.subplots(1 , 2, figsize=(20,4))

        axs[0].imshow(img)

        axs[0].set_title('Low Resolution Image')

        axs[0].get_xaxis().set_visible(False)

        axs[0].get_yaxis().set_visible(False)

        axs[1].imshow(predictions[n])

        axs[1].set_title('Predicted High Resolution Image')

        axs[1].get_xaxis().set_visible(False)

        axs[1].get_yaxis().set_visible(False)

        plt.show()

        n+=1

    else:

break
```

APPLICATION DEPLOYMENT

```
%%writefile app.py

import numpy as np

import streamlit as st

from tensorflow import keras

from tensorflow.keras.preprocessing.image import load_img, img_to_array

import matplotlib.pyplot as plt

from PIL import Image

model = keras.models.load_model("./simple31.h5")

def welcome():

return 'welcome all'

 def prediction(image):

    #file = load_img(image, target_size=(800,1200))

    file=Image.open(image)

    file=file.resize((80,80))

    #input_arr = np.array(file)

    #print(input_arr.shape)

    #input_arr = input_arr.astype('float32') / 255

    input_arr = img_to_array(file)

    input_arr = np.array([input_arr])

    input_arr = input_arr.astype('int') / 255

    k=model.predict(input_arr)

    return k[0],file
```

```
 # this is the main function in which we define our webpage
def main():
# giving the webpage a title
st.title("Imagno")
# here we define some of the front end elements of the web page like
# the font and background color, the padding and the text to be displayed
html_temp = """
<div style ="background-color:yellow;padding:13px">
<h1 style ="color:black;text-align:center;">Streamlit image quality ML App </h1>
</div>"""
# this line allows us to display the front end aspects we have
# defined in the above code
st.markdown(html_temp, unsafe_allow_html = True)
# the following lines create text boxes in which the user can enter
# the data required to make the prediction
img_input = st.file_uploader("upload image",type=["png","jpg","jpeg"])
# the prediction function defined above is called to make the prediction
# and store it in the variable result
if st.button("Infer"):
result,img = prediction(img_input)
st.image(img,width=500)
st.image(result,clamp=True,width=500)
if __name__=='__main__':
```

```
main()

!streamlit run /kaggle/working/app.py & npx localtunnel --port 8501

#The above given line is the app interface of the Image enchanement using Auto
```

CHAPTER-4

RESULTS AND DISCUSSION

4.1 RESULT

Our project is succesfully executed by enchancing the quality of image with 90% accuracy.

1.Accuracy

```
results = autoencoder.evaluate(val_x_px, val_x)
print('val_loss,accuracy', results)
```

```
19/19 [==============================] - 1s 49ms/step - loss: 0.0018 - accuracy: 0.9091
val_loss,accuracy [0.0018028658814728 26, 0.9090916514396667]
```

Figure 4.1.1(accuracy)

2.Model Loss Chart

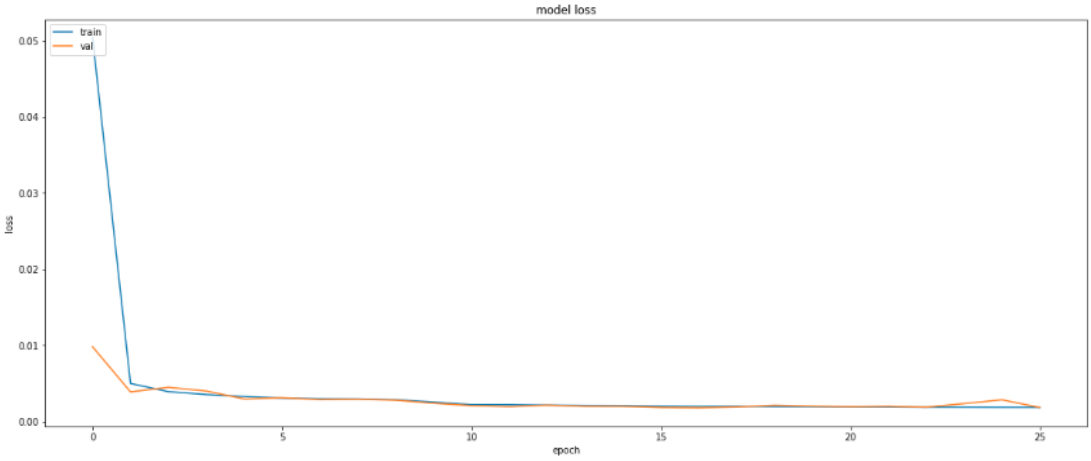

Figure 4.1.2(loss chart)

The loss of validation images with number of epochs is shown in the above chart

3.Enhancement done using the auto encoder model

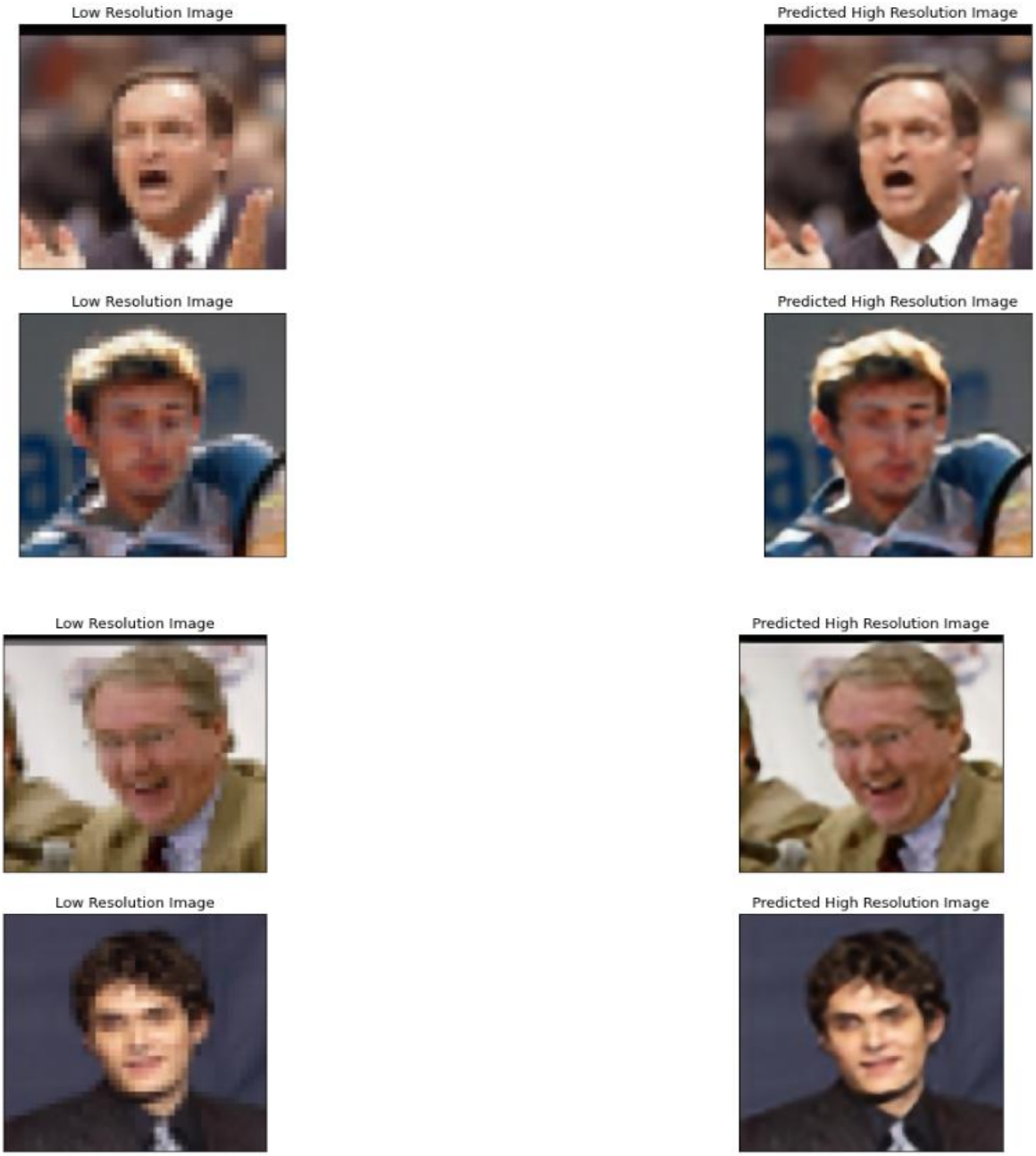

Figure:4.1.3(test output 1)

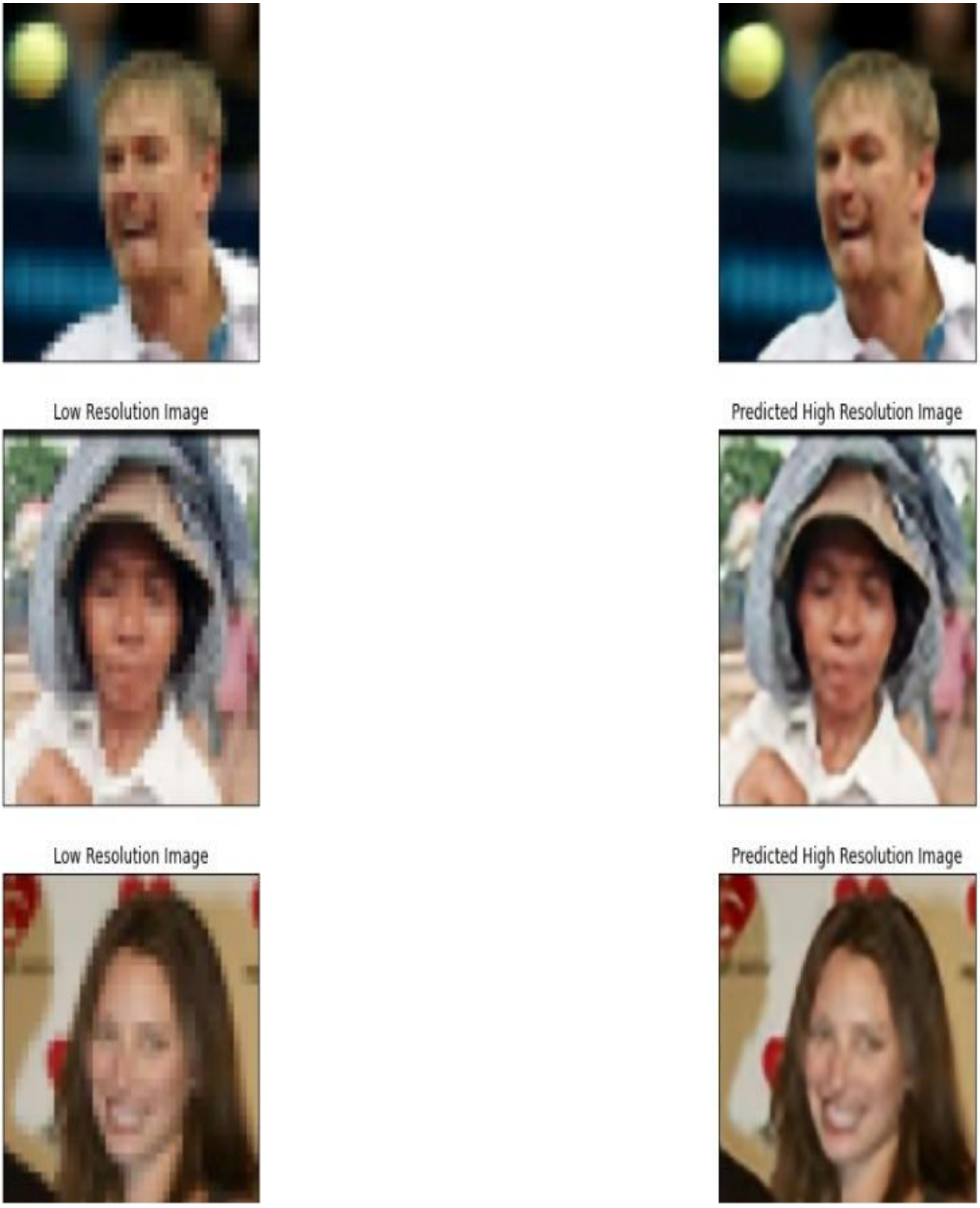

Figure:4.1.3(test output 2)

CHAPTER-5

CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

In this paper, we presented a fast, accurate and easy-to-use method to enhance the images. And the main contributions are as follows:

(a) We designed the denoising autoencoder model. This modification can help reduce the noise from the image.

(b) non-linear function was added in the backbone of the modified model to further improve accuracy at negligible additional computational cost.

(c) A GUI is designed using streamlit to make our method more user-friendly after obtaining the satisfactory model , which means our model can be applied to practical engineering easily. The method in this research differs from more conventional methods like manual enhancing.

(d) Our technology enhances the quality of images in many low resolution images

(e) This technology can be inherited to medical and campus institutions...etc.

5.2 FUTURE SCOPE

Some of the drawbacks can be resolved in the future to make the model more accurate and

Efficient.

- We can improve the model by training the model a variety of datasets which increases the accuracy of the model.
- Use of high GPU power and RAM can increase the quality of images.
- The current model works with (100 X 100) resolution image, but to work with high resolution images. we should use high compute power.
- Increasing the number of epochs will help in extracting more features from the image.
- GAN can also be used

REFERENCES

1. A review of Image Enhancement Systems and a case study of Salt &pepper noise removing Muna O. Al-Hatmi & Jabar H. Yousif. International Journal of Computation and Applied Sciences IJOCAAS, Volume 3, Issue 2, October 2017.
2. Z.-S. Liu, W.-C. Siu, L.-W. Wang, C.-T. Li, M.-P. Cani and Y.-L. Chan, *Unsupervised Real Image Super-Resolution via Generative Variational AutoEncoder*, 2020.
3. W. Dong, L. Zhang, G. Shi and X. Li, "Nonlocally centralized sparse representation for image restoration", *IEEE Trans. Image Process.*, vol. 22, no. 4, pp. 1620-1630, 2013.
4. J. Mairal, F. R. Bach, J. Ponce, G. Sapiro and A. Zisserman, "Nonlocal sparse models for image restoration", *Proc. IEEE Int. Conf. Comp. Vis.*, pp. 2272-2279, 2009.
5. X.-J. Mao, C. Shen and Y.-B. Yang, *Image restoration using convolutional auto-encoders with symmetric skip connections*, 2016.
6. S. Pal, S. Jana and R. Parekh, "Super-Resolution of Textual Images using Autoencoders for Text Identification", *2018 IEEE Applied Signal Processing Conference (ASPCON)*, pp. 153-157, 2018.
7. C. E. Duchon, "Lanczos Filtering in One and Two Dimensions", *Journal of Applied Meteorology*, vol. 18, pp. 1016-1022, 1979.
8. W. Ruangsang and Supavadee Aramvith, "Efficient Super-Resolution Algorithm using Overlapping Bicubic Interpolation", *IEEE 6th Global Conference on Consumer Electronics (GCCE 2017)*, 2017.
9. H.S. Hou and H.C. Andrews, "Cubic Splines for Image Interpolation and Digital Filtering", *IEEE Transactions on Acoustics Speech and Signal Processing ASSP-26*, pp. 508-517, 1978.
10. H. Chang, D.Y Yeung and Y. Xiong, "Super-resolution through neighbor embedding", *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. I-I, 2004.
11. Z. Wang, D. Liu, J. Yang, W. Han and T. Huang, "Deep networks for image super-resolution with sparse prior", *IEEE International Conference on Computer Vision (ICCV)*, pp. 370-378, 2015.
12. H. Zhang, J. Yang, Y. Zhang and T. S. Huang, "Image and video restorations via nonlocal kernel regression", *IEEE Trans. Cybern.*, vol. 43, no. 3, pp. 1035-1046, Jun. 2013.
13. C. Dong, C.C. Loy, K. He and X. Tang, "Learning a deep convolutional network for image super-resolution", *Proceedings of 13th European Conference on Computer Vision*, pp. 184-199, 2014.

14. Gabriel B. Cavallari, Leonardo SF Ribeiro and Moacir A. Ponti, "Unsupervised representation learning using convolutional and stacked auto-encoders: a domain and cross-domain feature space analysis", *2018 31st SIBGRAPI Conference on Graphics Patterns and Images (SIBGRAPI)*, pp. 440-446, 2018.
15. A. Badola, V. P. Nair and R. P. Lal, "An Analysis of Regularization Methods in Deep Neural Networks", *2020 IEEE 17th India Council International Conference (INDICON)*, pp. 1-6, 2020.
16. A. Rahangdale and S. Raut, "Deep Neural Network Regularization for Feature Selection in Learning-to-Rank", *IEEE Access*, vol. 7, pp. 53988-54006, 2019.
17. M. Akbari, A. H. Foruzan and Y.-W. Chen, "A Multi-Cluster Random Forests-Based Approach to Super-Resolution of Abdominal CT Images Using Deep Neural Networks", *2020 28th Iranian Conference on Electrical Engineering (ICEE)*, pp. 1-5, 2020.
18. C. Ledig, L. Theis, F. Husz'ar, J. Caballero, A. Cunningham, A. Acosta, et al., "PhotoRealistic Single Image Super-Resolution Using a Generative Adversarial Network", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 105-114, 2017.
19. Beniwal, P., & Singh, T. Image Enhancement by Hybrid Filter. International Journal of scientific research and management, 1(5),2013