**TABLES OF CONTENTS**

**S.No**  **Chapter** **Page.No**

1. Introduction

1.1 Objective

2. Literature Survey

3. System Requirement

4. Modules

4.1 Data gathering

4.2 Data annotation

4.3 Data preprocessing

4.4 CNN training

4.5 Prediction

5. System Architecture

5.1 What is pooling layer

5.2 Flow chart

5.3 Class Diagram

5.4 Sequence Diagram

5.5 Use case Diagram

6. Implementation

6.1 Data gathering code

6.2 Data annotation code

6.3 Data preprocesing code

6.4 CNN training code

6.5 Prediction

7. Testing

8. Screenshot

8.1 A person without mask

8.2 A person with mask

9. Conclusion

10. Reference

Abbreviations

AAM Active Appearance Models

AI Artificial Intelligence

ANN Artificial Neural Network

AFW Annotated Face in-the-Wild dataset

CNN Convolutional Neural Network

ConvL Convolutional layer

ConvNet Convolutional Network

CPU Central Processing Unit

DFW Disguised Faces in the Wild dataset

DL Deep Learning

DNN Deep Neural Network

EBGM Elastic Bunch Graph Matching dataset

FERET Facial Recognition Technology dataset

FR Facial Recognition

GPU Graphics Processing Unit

HMM Hidden Markov Models

HPO Head Poses and Occlusion

IBUG Tight face bounding box dataset

IOU Intersection over Union

JFA Joint Head Pose Estimation and Face Alignment Framework

L1-SVM Linear norm Support Vector Machine

L2-SVM Square norm Support Vector Machine

LAG Large Age-Gap Database

LBF Local Binary Features

LDA Linear Discriminant Analysis

LFPW Labelled Face Parts in the Wild

ML Machine Learning

OFR Occluded Face Recognition

PCA Principal component analysis

R-CNN Region Based Convolutional Neural Networks

ReLU Rectified linear unit

ROI Regions of Interest

RPN Region proposal network

SCfaceDB surveillance cameras face database

SIFT Scale-Invariant feature transform

SOF Specs on Faces Dataset

SVM Support Vector Machine

XM2VTS Extended Multi-modal face database

YOLO You only look once

# CHAPTER 1

# INTRODUCTION

Artificial Intelligence and Neural networks have already transformed internet technologies across the globe from something useful to something significant in our lives. The artificial intelligence innovations have intervened in all fields ranging from improved health care, where these machines are better suited to cancer detection than any doctor, to self-driving cars that are considered safer than human driving. Without ignoring the practical assistance of AI in simulations to measuring, monitoring and resource management on climate change, conservation and environment.

Artificial neural networks are one of the most important technologies ever developed that can operate without human intervention. In the programming conventional techniques, we tell the machine the steps to breaking big problems down to small ones in different scenarios. Next, the device can use its computational capabilities to help users manage data more rapidly and effectively. On the contrary, machine learning uses a massive amount of data to develop a pattern that can classify and predict to solve complicated problems without human interferes. Therefore, we train the machine and create various models that predict the outcome with high accuracy, which will be a significant aid in achieving a complicated mission. The computer can find a solution to any problems by supplying an appropriate amount of labelled data and using supervised learning techniques. However, had its portion of problems not in space but on the ground.

The last twenty years, we have been facing an increase in global terrorism. Nonetheless, the modern world has had its share of problems not in space but on the ground. The last twenty years, we have been faced with a spike in global terrorism. This problem affected law-enforcement agencies in airports, border crossings and ports across the globe on millions of people travelling daily.

According to the United States Department of Transportation around 900 million passengers have been transported in 2019 using airports, hence managing the traffic

flow by a human was complicated task to accomplish. To avoid security breaches and human mistakes. Governments began funding many agencies and companies to develop these technologies to enhance human work and fulfil the grown needs to improve artificial intelligence. Since the Iris and fingerprint recognition are a slow way to achieve the task of recognition and not safe, authorities start to use a new biometric recognition called "Face Recognition", and these techniques become so popular and ubiquitous last five years.

At the end of 2019, the world was confronted with a widespread problem in the history of humanity Coronavirus pandemic (COVID-19). The COVID-19 virus affects the respiratory system and spreads through close contact. As a result, most people began to use face masks for protection; thus, facial recognition systems struggle to identify faces wearing medical masks.

The purpose of this thesis aims to explore and test the new detection model and evaluation algorithms and their efficacy on masked faces. The author will create a masked faces dataset by collecting images from the internet to use it for the recognition models, also we will use a masked face detection dataset to trained different models to detect masked faces and recognize the faces. It was a complicated problem since there were no labelled datasets for masked faces to train our recognition model; nor do we have enough earlier research on this subject for comparison and evaluation.

The methodology of developing Artificial intelligence systems are correlated to full- stack developments approaches through different aspects such as, provide the front end applications to the end-users and host the AI models on the backend which are built to be running on a server. In general, FR systems use many software development methodologies to improve the system's performance, such as using the agile approach to deploy facial recognition systems in easily manageable environments, with the agile approach the FR systems result delivered in quick, constant control. The agile software developments are appropriate to computer vision systems because the end-users need to be involved earlier to further examination, adjustments, improvements, and

finally evaluate the facial recognition models. Many developers use facial recognition technologies nowadays to grant access to various applications.

**1.1 Objective**

Face recognition is one of the problems of computer vision that has gained a lot of attention in the previous decade. Many researchers have been contributed to develop and innovate new theories to improve computer vision and applied these theories in real applications. Various studies have published on face recognition, that focuses on developing methods to ascertain the presence of face and recognize it. The study conduces to clarify the obstacles of detection and identification and introduce an explication that able to develop a new recognition system method for both circumstances ( full-visible features and half-visible features ).

The thesis aims to develop a new facial recognition system capable of recognizing an individual wearing a medical mask. To understand how the facial recognition system functions, the author divides the issue into four stages. The purpose of the division is to make the problem understandable and experimental by taking each stage on its own and clarifying the characteristics and the necessity for experiments to be carried out. The thesis, therefore, presents these steps, beginning by presenting some popular algorithms used previously to implement facial recognition systems alongside datasets used to build and evaluate these algorithms. Next step is to research the detection and alignment processes, to understand the mechanism behind them and how they can apply them in the system According to Dawson R. Hancock and Bob Algozzine (2017, 3), Various types of questions interpretable. (What?, How?, Why?) that have driven scholars to explore the causes why things have happened and to create more specific approaches. Usually, while specialists are analysing a topic, this implies that they are seeking to obtain feedbacks for a better understanding of the subject, alternative scenarios for analysis, and possible explanations for review. These feedbacks and questions have driven the study to draw conclusions that are reliable, practical, and

According to Dawson H. et al. (2017,4), A research effort should not be carried out by a researcher without an organizational paradigm. This paradigm lays out for the researcher the distinguishing characteristics of the study and the possibilities for obtaining answers to the questions. Therefore, the author determines a paradigm to conduct systematic research that investigates the study's topic by identifying three critical pillars to design the research process

* Study's methods
* Gathering information
* Results' confirmation

These three essential pillars drove the study to comprehend research methods, data, and analysis forms to illustrate the OFR stages and to understand the data required to build a very accurate model. By following the previous steps, the research questions based on the triple paradigm are:

* 1. What are the best approaches used to develop an OFR system?
  2. How does the quantity and quality of data affect the OFR system?
  3. How does the OFR system detect and extract visual facial features?
  4. What is the evaluation of the OFR system’s performance

**CHAPTER 2**

**LITERATURE SURVEY**

As one of the most successful applications of image analysis and understanding, face recognition has recently received significant attention, especially during the past few years. This is evidenced by the emergence of face recognition conferences such as the International Conference on Audioand Video-Based Authentication (AVBPA) since 1997 and the International Conference on Automatic Face and Gesture Recognition (AFGR) since 1995, systematic empirical evaluations of face recognition techniques (FRT), including theFERET [Phillips et al. 1998b, 2000; Rizvi et al. 1998], FRVT 2000 [Blackburn et al.2001], FRVT 2002 [Phillips et al. 2003], and XM2VTS [Messer et al. 1999] protocols,and many commercially available systems (Table II). There are at least two reasons for this trend; the first is the wide range of commercial and law enforcement applications and the second is the availability of feasible technologies after 30 years of research. In addition, the problem of machine recognition of human faces continues to attract researchers from disciplines such as image processing, pattern recognition, neural networks, computer vision, computer graphics, and psychology.The strong need for user-friendly systems that can secure our assets and protect our privacy without losing our identity in a sea of numbers is obvious. At present, one needs a PIN to get cash from an ATM, a password for a computer, a dozen others to access the internet, andso on. Although very reliable methods of biometric personal identification exist, forexample, fingerprint analysis and retinal or iris scans, these methods rely on the cooperation of the participants, whereas a personal identification system based on analysis of frontal or profile images of the face is often effective without the participant’s cooperation or knowledge. Some ofthe advantages/disadvantages of different biometrics are described in Phillips et al.[1998]. Table I lists some of the applications of face recognition. Commercial and law enforcement applications of FRT range from static, controlled-format photographs to uncontrolled video images, posing a wide range of technical challenges and requiring anequally wide range of techniques from image processing, analysis, understanding,and pattern recognition. One can broadly classify FRT systems into two groups depending on whether they make use of static images or of video. Within these groups, significant differences exist, dependingon the specific application. The differences are in terms of image quality, amount of background clutter (posing challenges to segmentation algorithms), variability of the images of a particular individual that must be recognized, availability of a well-defined recognition or matching criterion, and the nature, type, and amount of input from a user. A list of some commercial systems is given in. A general statement of the problem of machine recognition of faces can be formulated as follows: given still or video images of a scene, identify or verify one or more persons in the scene using a stored database of faces. Available collateral information such as race, age, gender, facial expression, or speech may be used in narrowing the search (enhancing recognition). The solution to the problem involves segmentation of faces (face de- tection) from cluttered scenes, feature ex- traction from the face regions, recognition, or verification (Figure 1). In identification problems, the input to the system is an un- known face, and the system reports back the determined identity from a database of known individuals, whereas in verifica- tion problems, the system needs to confirm or reject the claimed identity of the input face.Face perception is an important part of the capability of human perception sys- tem and is a routine task for humans, while building a similar computer sys- tem is still an on-going research area. The earliest work on face recognition can be traced back at least to the 1950s in psy- chology [Bruner and Tagiuri 1954] and to the 1960s in the engineering literature [Bledsoe 1964]. Some of the earliest stud- ies include work on facial expression of emotions by Darwin [1972] (see also Ekman [1998]) and on facial profile-based biometrics by Galton [1888]). But re- search on automatic machine recogni- tion of faces really started in the 1970s [Kelly 1970] and after the seminal work of Kanade [1973]. Over the past 30 years extensive research has been con- ducted by psychophysicists, neuroscien- tists, and engineers on various aspects of face recognition by humans and ma- chines. Psychophysicists and neuroscien- tists have been concerned with issues such as whether face perception is a dedicated process (this issue is still be- ing debated in the psychology community [Biederman and Kalocsai 1998; Ellis 1986;

|  |  |
| --- | --- |
| Areas | Specific applications |
| Entertainment | Video game, virtual reality, training programs |
| Human-robot-interaction, human-computer-interaction |
| Smart cards | Drivers’ licenses, entitlement programs |
| Immigration, national ID, passports, voter registration |
| Welfare fraud |
| Information security | TV Parental control, personal device logon, desktop logon |
| Application security, database security, file encryption |
| Intranet security, internet access, medical records |
| Secure trading terminals |
| Law enforcement  and surveillance | Advanced video surveillance, CCTV control |
| Portal control, postevent analysis |
| Shoplifting, suspect tracking and investigation |

Gauthier et al. 1999.

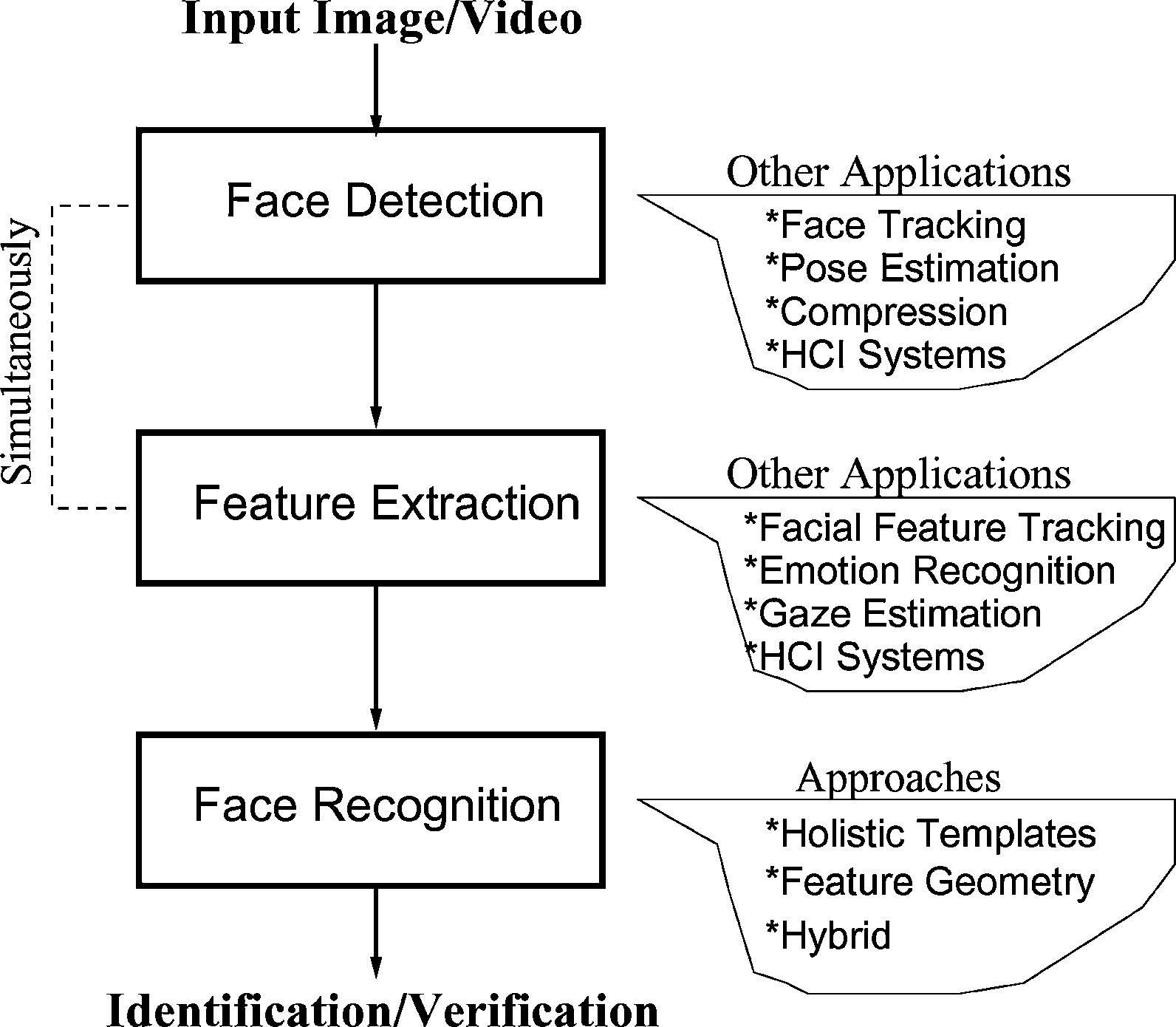
**Fig 1. Typical Applications of Face Recognition**

findings have important consequences for engineers who design algorithms and sys- tems for machine recognition of human faces. Section 2 will present a concise re- view of these findings.

Barring a few exceptions that use range data [Gordon 1991], the face recognition problem has been formulated as recogniz- ing three-dimensional (3D) objects from two-dimensional (2D) images. Earlier ap- proaches treated it as a 2D pattern recog- nition problem. As a result, during the early and mid-1970s, typical pattern clas- sification techniques, which use measured attributes of features (e.g., the distances between important points) in faces or face profiles, were used [Bledsoe 1964; Kanade 1973; Kelly 1970]. During the 1980s, work on face recognition remained largely dor- mant.

and extraction of features such as eyes, mouth, etc. Meanwhile, significant advances have been made in the design of classifiers for successful face recognition. Among appearance-based holistic approaches, eigenfaces [Kirby and Sirovich 1990; Turk and Pentland 1991] and Fisher- faces [Belhumeur et al. 1997; Etemad and Chellappa 1997; Zhao et al. 1998] have proved to be effective in experiments with large databases. Feature-based graph matching approaches [Wiskott et al. 1997] have also been quite suc- cessful. Compared to holistic Since the early 1990s, research in- terest in FRT has grown significantly. One can attribute this to several reasons: an in- crease in interest in commercial opportu- nities; the availability of real-time hard- ware; and the increasing importance of surveillance-related applications.

Over the past 15 years, research has focused on how to make face recognition systems fully automatic by tackling prob- lems such as localization of a face in a given image or video clip approaches, feature-based methods are less sensi- tive to variations in illumination and viewpoint and to inaccuracy in face local-



**Fig 2. Configuration of a generic face recognition system**

**CHAPTER 3**

**SYSTEM REQUIREMENTS**

**HARDWARE REQUIREMENT**

* RAM : 4GB
* HARD DISK : 8GB
* OPERATING SYSTEM: 64 bit

**SOFTWARE REQUIREMENT**

* LANGUAGE: 1.Python

2.JavaScripts

* SOFTARE: Anaconda

**CHAPTER 4**

**MODULES**

There are four modules:

1.Data gathering

2.Data annotation

3.Data preprocessing

4.CNN Training

5.Prediction

**4.1 Data gathering:**

Data collection is defined as the procedure of collecting, measuring and analyzing accurate insights for research using standard validated techniques. A researcher can evaluate their hypothesis on the basis of collected data. In most cases, data collection is the primary and most important step for research, irrespective of the field of research. The approach of data collection is different for different fields of study, depending on the required information.

The most critical objective of data collection is ensuring that information-rich and reliable data is collected for  so that data-driven decisions can be made for research.

Essentially there are four choices for data collection – in-person interviews, mail, phone and online. There are pros and cons to each of these modes.

Pros: In-depth and a high degree of confidence on the data  
Cons: Time consuming, expensive and can be dismissed as anecdotal

Pros: Can reach anyone and everyone – no barrier Cons: Expensive, data collection errors, lag time

Pros: High degree of confidence in the data collected, reach almost anyone  
Cons: Expensive, cannot self-administer, need to hire an agency

Pros: Cheap, can self-administer, very low probability of data errors  
Cons: Not all your customers might have an email address/be on the internet, customers may be wary of divulging information online.

In-person  always are better, but the big drawback is the trap you might fall into if you don’t do them regularly. It is expensive to regularly conduct interviews and not conducting enough interviews might give you false positives. Validating your research is almost as important as designing and conducting it. We’ve seen many instances where after the research is conducted – if the results do not match up with the “gut-feel” of upper management, it has been dismissed off as anecdotal and a “one-time” phenomenon. To avoid such traps, we strongly recommend that data-collection be done on an “ongoing and regular” basis. This will help you in comparing and analyzing the change in perceptions according to marketing done for your products/services. The other issue here is sample size. To be confident with your research you have to interview enough people to weed out the fringe elements.

A couple of years ago there was quite a lot of discussion about online surveys and their statistical validity. The fact that not every customer had internet connectivity was one of the main concerns. Although some of the discussions are still valid, the reach of the internet as a means of communication has become vital in the majority of customer interactions. According to the US Census Bureau, the number of households with computers has doubled between 1997 and 2001. Keep in mind, the reach here is defined as “All U.S. Households.” In most cases, you need to take a look at how many of your customers are online and make a determination. If all your customers have email addresses, you have a 100% reach of your customers.

Another important thing to keep in mind is the ever-increasing dominance of cellular phones over landline phones. United States FCC rules prevent automated dialing and calling cellular phone numbers and there is a noticeable trend towards people having cellular phones as the only voice communication device. This introduces the inability to reach cellular phone customers who are dropping home phone lines in favor of going entirely wireless. Even if automated dialing is not used, another FCC rule prohibits from phoning anyone who would have to pay for the call

**Web Scrapping:**

Web Scripting is an automatic method to obtain large amounts of data from websites. Most of this data is unstructured data in an HTML format which is then converted into structured data in a spreadsheet or a database so that it can be used in various applications. There are many different ways to perform web scraping to obtain data from websites. these include using online services, particular API’s or even creating your code for web scraping from scratch. Many large websites like Google, Twitter, Facebook, StackOverflow, etc. have API’s that allow you to access their data in a structured format. This is the best option but there are other sites that don’t allow users to access large amounts of data in a structured form or they are simply not that technologically advanced. In that situation, it’s best to use Web Scraping to scrape the website for data.

Web scraping requires two parts namely the **crawler** and the **scraper**. The crawler is an artificial intelligence algorithm that browses the web to search the particular data required by following the links across the internet. The scraper, on the other hand, is a specific tool created to extract the data from the website. The design of the scraper can vary greatly according to the complexity and scope of the project so that it can quickly and accurately extract the data.

Web Scrapers can extract all the data on particular sites or the specific data that a user wants. Ideally, it’s best if you specify the data you want so that the web scraper only extracts that data quickly. For example, You might want to scrape an Amazon page for the types of juicers available, but you might only want the data about the models of different juicers and not the customer reviews.

So when a web scraper needs to scrape a site, first it is provided the URL’s of the required sites. Then it loads all the HTML code for those sites and a more advanced scraper might even extract all the CSS and Javascript elements as well. Then the scraper obtains the required data from this HTML code and outputs this data in the format specified by the user. Mostly, this is in the form of an Excel spreadsheet or a CSV file but the data can also be saved in other formats such as a JSON file.

**Types of Scrapping:**

Web Scrapers can be divided on the basis of many different criteria including Self-built or Pre-built Web Scrapers, Browser extension or Software Web Scrapers, and Cloud or Local Web Scrapers.

You can have **Self-built Web Scrapers** but that requires advanced knowledge of programming. And if you want more features in your Web Scraper, then you need even more knowledge. On the other hand, **Pre-built Web Scrapers** are previously created scrapers that you can download and run easily. These also have more advanced options that you can customize.

**Browser extension Web Scrapers** are extensions that can be added to your browser. These are easy to run as they are integrated with your browser but at the same time, they are also limited because of this. Any advanced features that are outside the scope of your browser are impossible to run on Browser extension Web Scrapers. But **Software Web Scrapers** don’t have these limitations as they can be downloaded and installed on your computer. These are more complex than Browser extension Web Scrapers but they also have advanced features that are not limited by the scope of your browser.

**Cloud Web Scrapers** run on the cloud which is an off-site server mostly provided by the company that you buy the scraper from. These allow your computer to focus on other tasks as the computer resources are not required to scrape data from websites. **Local Web Scrapers**, on the other hand, run on your computer using local resources. So if the Web Scrapers require more CPU or RAM, then your computer will become slow and not be able to perform other tasks.

It seems to be in fashion these days! It is the most popular language for web scraping as it can handle most of the processes easily. It also has a variety of libraries that were created specifically for Web Scraping.  is a very popular open-source web crawling framework that is written in Python. It is ideal for web scraping as well as extracting data using APIs.  is another Python library that is highly suitable for Web Scraping. It creates a parse tree that can be used to extract data from HTML on a website. Beautiful soup also has multiple features for navigation, searching, and modifying these parse trees.

import cv2,os

import numpy as np

from keras.utils import np\_utils

data\_path='C:\\Users\\MYPC\\Projects\\face-mask-detector\\dataset'

categories=os.listdir(data\_path)

labels=[i for i in range(len(categories))]

label\_dict=dict(zip(categories,labels))

print(label\_dict)

print(categories)

print(labels)

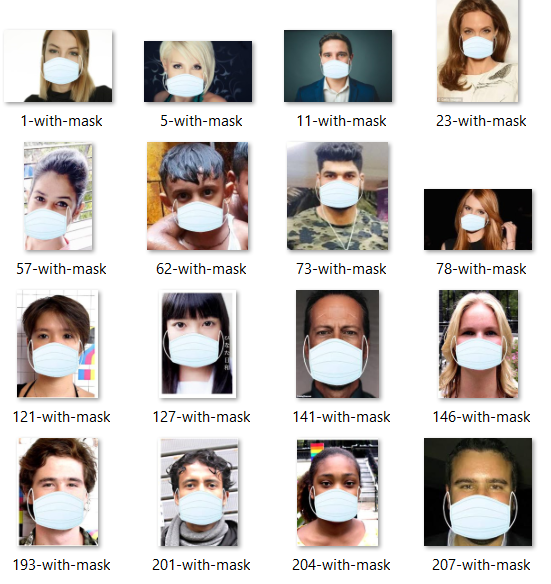
data=[] target=[]

**About the Dataset:**

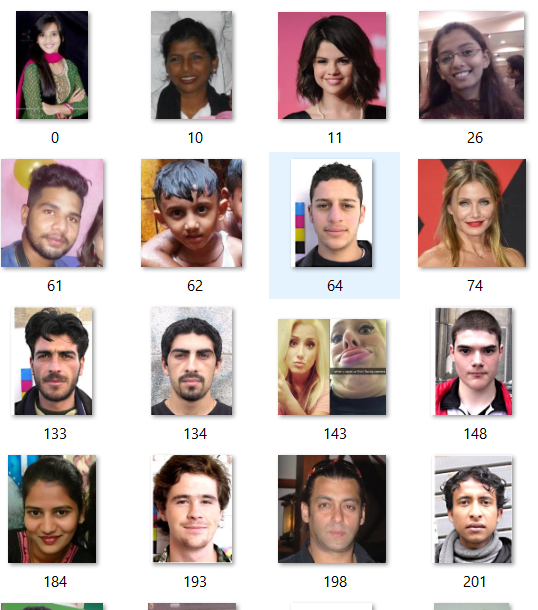
Mask play a crucial role in protecting the health of individuals against respiratory diseases, as is one of the few precaution available for covid-19 in the absence of immunization with the dataset it is possible to create a model to detect people wearing mask improperly this dataset contain 800 images.

Belonging to the 3 classses as well as their bounding boxes in the PASCAL VOC. Format.

**A person with mask:**



**A person without mask:**

****

**Tensorflow:**

TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

TensorFlow is an open source machine learning framework for all developers. It is used for implementing machine learning and deep learning applications. To develop and research on fascinating ideas on artificial intelligence, Google team created TensorFlow. TensorFlow is designed in Python programming language, hence it is considered an easy to understand framework.

This tutorial has been prepared for python developers who focus on research and development with various machine learning and deep learning algorithms. The aim of this tutorial is to describe all TensorFlow objects and methods.

Before proceeding with this tutorial, you need to have a basic knowledge of any Python programming language. Knowledge of artificial intelligence concepts will be a plus point.

In this chapter, we will learn about the basics of TensorFlow. We will begin by understanding the data structure of tensor.

Tensors are used as the basic data structures in TensorFlow language. Tensors represent the connecting edges in any flow diagram called the Data Flow Graph. Tensors are defined as multidimensional array or list.

Tensors are identified by the following three parameters −

Unit of dimensionality described within tensor is called rank. It identifies the number of dimensions of the tensor. A rank of a tensor can be described as the order or n-dimensions of a tensor defined.

The number of rows and columns together define the shape of Tensor.Type describes the data type assigned to Tensor’s elements.

A user needs to consider the following activities for building a Tensor −

* Build an n-dimensional array

Convert the n-dimensional array.

being required to wear facemasks for the prevention of virus spreading, healthy persons also need towear facemasks in order to protect themselves from infection [1]. Facemasks, when fitted properly,

e\_ectively disrupt the forward momentum of particles expelled from a cough or sneeze, preventingdisease transmission [5]. However, the e\_ectiveness of facemasks in containing the spread of airbornediseases in the general public has been diminished, mostly due to improper wearing [8]. Therefore,it is necessary to develop an automatic detection approach for facemask-wearing condition, which cancontribute to personal protection and public epidemic prevention.The distinctive facial characteristics in facemask-wearing conditions provide an opportunityfor automatic identification. Recent rapid technological innovations in deep learning and computervision have presented opportunities for development in many fields [9,10]. As the main componentof deep learning methods, deep neural networks (DNNs) have demonstrated superior performancein many fields, including object detection, image classification, image segmentation, and distancingdetection [11–16]. One primary model of DNNs is convolutional neural networks (CNNs), which havebeen widely used in the field of computer vision tasks. After training, CNNs can recognize andclassify facial images—even with slight di\_erences—due to their powerful feature extraction capability.As one of the CNNs, image super-resolution (SR) networks can restore image details. Recently,SR networks have become more in-depth, and the ideas of auto-encoder and residual learninghave been integrated for performance improvement [17,18]. SR networks have also been appliedfor image processing before image segmentation or classification, reconstructing images for higherresolution and restoring details [19–23]. Moreover, SR networks can improve the classification accuracysignificantly, especially when using a dataset with low-quality images, and provide a feasible solutionto improve facemask-wearing condition identification performance. Therefore, the combination of anSR network with a classification network (SRCNet) could be utilized in facial image classification for accuracy improvement.To our knowledge, there have not been any published reports related to SR networks combined withclassification networks for accuracy improvement in facial image classification, especially regardingthe automatic detection of facemask-wearing conditions. Therefore, we intend to develop a novelmethod combining an SR network with a classification network (SRCNet) to identify facemask-wearingconditions, in order to improve classification accuracy with low-quality facial images.Our main contributions can be summarized as follows.(1) Development of a new face accessory identification method that combines an SR network with aclassification network (SRCNet) for facial image classification(2) Utilization of a deep learning method for automatic identification of facemask-wearing conditions To our knowledge, this is the first time a deep learning method has been applied to identifyingfacemask-wearing condition.(3) Improving the SR network structure, including activation functions and the density of skipconnections, which outperforms previous state-of-the-art methods.The idea of reconstructing high-quality images from low-resolution images has a long history.Bicubic was one of the most widely used methods, which up-sampled low-resolution images by linearinterpolation in both the x-axis and y-axis. However, the reconstructed images using the bicubic methodwere blurred, due to the loss of high-frequency information. Hence, high-performance algorithms havebeen introduced. Yang, et al. [24] presented an SR method based on sparse representation, which usedsparse representations for each patch of the low-resolution input and then calculated the coe\_cients togenerate a high-resolution output. The example-based SR method was introduced by Timofte, et al. [25],which reconstructs images based on a dictionary of low-resolution and high-resolution exemplars.Recently, deep learning methods have also been introduced for SR [26–34]. Dong, et all.

**Keras:**

Keras is a deep learning API written in Python, running on top of the machine learning platform  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.

is an end-to-end, open-source machine learning platform. You can think of it as an infrastructure layer for It combines four key abilities:

* Efficiently executing low-level tensor operations on CPU, GPU, or TPU.
* Computing the gradient of arbitrary differentiable expressions.
* Scaling computation to many devices (e.g. the at Oak Ridge National Lab, which spans 27,000 GPUs).
* Exporting programs ("graphs") to external runtimes such as servers, browsers, mobile and embedded devices.

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.

What you just saw is the most elementary way to use Keras: it mirrors the Scikit-Learn API.

However, Keras is also a highly-flexible framework suitable to iterate on state-of-the-art research ideas. Keras follows the principle of **progressive disclosure of complexity**: it makes it easy to get started, yet it makes it possible to handle arbitrarily advanced use cases, only requiring incremental learning at each step.

In much the same way that you were able to train & evaluate a simple neural network above in a few lines, you can use Keras to quickly develop new training procedures or exotic model architectures. Here's a low-level training loop example, combining Keras functionality with the TensorFlow

Keras (κέρας) means horn in Greek. It is a reference to a literary image from ancient Greek and Latin literature, first found in the Odyssey, where dream spirits (Oneiroi, singular Oneiros) are divided between those who deceive dreamers with false visions, who arrive to Earth through a gate of ivory, and those who announce a future that will come to pass, who arrive through a gate of horn. It's a play on the words κέρας (horn) / κραίνω (fulfill), and ἐλέφας (ivory) / ἐλεφαίρομαι (deceive).

Given that the TensorFlow project has adopted Keras as the high-level API for the upcoming TensorFlow 2.0 release, Keras looks to be a winner, if not necessarily the winner. In this article, we'll explore the principles and implementation of Keras, with an eye towards understanding why it’s an improvement over low-level deep learning APIs.

Even in TensorFlow 1.12, the official  uses the  embedded in TensorFlow, **tf.keras**. By contrast, the requires working with TensorFlow computational graphs, tensors, operations, and sessions, some of which can be hard to understand when you're just beginning to work with TensorFlow. There are some advantages to using the low-level TensorFlow Core API, mostly when debugging, but fortunately you can mix the high-level and low-level TensorFlow APIs as needed.

Keras was created to be user friendly, modular, easy to extend, and to work with Python. The API was “designed for human beings, not machines,” and “follows best practices for reducing cognitive load.”

Neural layers, cost functions, optimizers, initialization schemes, activation functions, and regularization schemes are all standalone modules that you can combine to create new models. New modules are simple to add, as new classes and functions. Models are defined in Python code, not separate model configuration files.

The biggest reasons to use Keras stem from its guiding principles, primarily the one about being user friendly. Beyond ease of learning and ease of model building, Keras offers the advantages of broad adoption, support for a wide range of production deployment options, integration with at least five back-end engines (TensorFlow, CNTK, Theano, MXNet, and PlaidML), and strong support for multiple GPUs and distributed training. Plus, Keras is backed by Google, Microsoft, Amazon, Apple, Nvidia, Uber, and others.

Keras proper does not do its own low-level operations, such as tensor products and convolutions; it relies on a back-end engine for that. Even though Keras supports multiple back-end engines, its primary (and default) back end is TensorFlow, and its primary supporter is Google. The Keras API comes packaged in TensorFlow as  which as mentioned earlier will become the primary TensorFlow API as of TensorFlow 2.0.

The comments in the code above are worth reading. It’s also worth noting how little cruft there is in the actual code compared to, say, the low-level TensorFlow APIs. Each layer definition requires one line of code, the compilation (learning process definition) takes one line of code, and fitting (training), evaluating (calculating the losses and metrics), and predicting outputs from the trained model each take one line of code.

**Open-cv:**

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 a BSD-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 The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV’s deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux,  and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured and  interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

OpenCV is a cross-platform library using which we can develop real-time computer visionapplications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection. In this tutorial, we explain how you can use OpenCV in your applications.

This tutorial has been prepared for beginners to make them understand the basics of OpenCV library. We have used the Java programming language in all the examples, therefore you should have a basic exposure to Java in order to benefit from this tutorial.

For this tutorial, it is assumed that the readers have a prior knowledge of Java programming language. In some of the programs of this tutorial, we have used JavaFX for GUI purpose. So, it is recommended that you go through our JavaFX tutorial before proceeding further.

**OpenCV** is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e whatever operations one can do in Numpy can be combined with OpenCV.This OpenCV tutorial will help you learn the Image-processing from Basics to Advance, like operations on Images, Videos using a huge set of Opencv-programs and projects.

**Computer vision** is a process by which we can understand the images and videos how they are stored and how we can manipulate and retrieve data from them. Computer Vision is the base or mostly used for Artificial Intelligence. Computer-Vision is playing a major role in self-driving cars, robotics as well as in photo correction apps.

## **OpenCV**

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today’s systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as Numpuy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

The first OpenCV version was 1.0. OpenCV is released under a BSD license and hence it’s free for both **academic** and **commercial** use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. When OpenCV was designed the main focus was real-time applications for computational efficiency. All things are written in optimized C/C++ to take advantage of multi-core processing.

**4.2 Data Annotation:**

Data annotation is the task of labeling data with metadata in preparation for training a machine learning model. Both data and metadata come in many forms, including content types such as text, audio, images, and video. These  can be used to train autonomous vehicles, chatbots, and translation systems.

In this article, we’ll explore six different types of data annotation and their most common uses in machine learning.

Data annotation is the process of adding metadata to a dataset. This metadata usually takes the form of tags, which can be added to any type of data, including text, images, and video. Adding comprehensive and consistent tags is a key part of developing a training dataset for machine learning.

Data annotation is an indispensable stage of data preprocessing because supervised machine learning models learn to recognize recurring patterns in annotated data. After an algorithm has processed enough annotated data, it can start to recognize the same patterns when presented with new, unannotated data. As a result, data scientists need to use clean, annotated data to train machine learning models.

**Data Annotation type:**

There are many different types of data annotation, all of which suit different use cases. In the section below, we run through a few of the more common annotation types that are used for popular machine learning projects. It’s by no means an exhaustive list, but should give you some idea of the breadth of the field of data annotation. Let’s dive in:

### **Semantic Annotation**

Semantic annotation is the task of annotating various concepts within text, such as people, objects, or company names. Machine learning models use semantically annotated data to learn how to categorize new concepts in new texts. This can help to improve search relevance and train chatbots.

### **Image and Video Annotation**

Image annotation comes in a variety of forms, from bounding boxes, which are imaginary boxes drawn on images, to semantic segmentation, where every pixel in an image is assigned a meaning. This label usually helps a machine learning model to recognize the annotated area as a distinct type of object. This type of data often serves as a ground truth for image recognition models that can recognize and block sensitive content, guide autonomous vehicles, or perform facial recognition tasks.

Similar to image annotation, video annotation often involves adding bounding boxes, polygons, or keypoints to content. This can be done on a frame-by-frame basis, with these frames then stitched together to help track the movement of the annotated object, or in the video itself using a video annotation tool. This type of data also plays an important role in the development of computer vision models for tasks like object tracking and localization.

### **Text Categorization**

Text categorization and content categorization refer to the task of assigning predefined categories to documents. For example, you can tag sentences or paragraphs within a document by topic, or organizing news articles by subject such as domestic, international, sports, or entertainment.

### **Entity Annotation**

Entity annotation is the process of labeling unstructured sentences with information so that a machine can read them.

Within entity annotation, there are a multitude of processes that can be layered to create an understanding of language. An exhaustive list would be too long to reproduce here, but these examples should give a sense of the broad possibilities on offer:

**Named entity recognition:** Named entity recognition (NER) refers to the classification of named entities present in a body of text. These entities are labeled based on predefined categories such as person, organization, and place. Named entity recognition models add semantic knowledge to your content, making it easy for individuals and systems to quickly identify and understand the subject of any given text.

**Entity linking:** This is the process of annotating the relationship between two parts of a text. For example, you can tag the company and employee, or person and their hometown as related concepts.

## **Performing Data Annotation**

Annotating your data can be a significant undertaking, but you don’t need to spend hours on data annotation by yourself. There are many third party companies and individuals who can help you. For example, Lionbridge AI can help you with data annotation tasks for text, image, video, and audio datasets. We’ll help you to define the kind of data annotation services you’re looking for, develop a clear gold standard for your workers, and build a comprehensive dataset that’s perfect for training your machine learning model. Get in touch below to learn more about how we can help.

**Converting image into grayscale:**

''' The dataset we are using consists of images with different colors, different sizes, and different orientations. Therefore, we need to convert all the images into grayscale because we need to be sure that color should not be a critical point for detecting mask. After that, we need to have all the images in the same size (150x150) before applying it to the neural network. ''

for category in categories:

folder\_path=os.path.join(data\_path,category)

img\_names=os.listdir(folder\_path)

for img\_name in img\_names:

mg\_path=os.path.join(folder\_path,img\_name)

img=cv2.imread(img\_path)

try:

gray=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

#Coverting the image into gray scale

resized=cv2.resize(gray,(100,100))

#resizing the gray scale into 100x100, since we need a fixed common size for all the images in the dataset

data.append(resized)

target.append(label\_dict[category])

#appending the image and the label(categorized) into the list (dataset)

except Exception as e:

print('Exception:',e)

#if any exception rasied, the exception will be printed here. And pass to the next image.

This reads the image in and converts it into a Numpy array. For a detailed description of what this does and why, check out the prequel post to this one For grayscale images, the result is a two-dimensional array with the number of rows and columns equal to the number of pixel rows and columns in the image. Low numeric values indicate darker shades and higher values lighter shades. The range of pixel values is often 0 to 255. We divide by 255 to get a range of 0 to.

Color images are represented as three-dimensional Numpy arrays - a collection of three two-dimensional arrays, one each for red, green, and blue channels. Each one, like grayscale arrays, has one value per pixel and their ranges are identical.

**4.3 Data pre processing:**

The dataset we are using consists of images with different colors, different sizes, and different orientations. Therefore, we need to convert all the images into grayscale because we need to be sure that color should not be a critical point for detecting mask. After that, we need to have all the images in the same size (100x100) before applying it to the neural network.

import cv2,os

import numpy as np

from keras.utils import np\_utils

data\_path='dataset'

categories=os.listdir(data\_path)

labels=[i for i in range(len(categories))]

label\_dict=dict(zip(categories,labels))

data=[]

target=[]

for category in categories:

folder\_path=os.path.join(data\_path,category)

img\_names=os.listdir(folder\_path)

for img\_name in img\_names:

img\_path=os.path.join(folder\_path,img\_name)

img=cv2.imread(img\_path)

try:

gray=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

resized=cv2.resize(gray,(100, 100))dataset

data.append(resized)

target.append(label\_dict[category])

except Exception as e:

print('Exception:',e)

data=np.array(data)/255.0

data=np.reshape(data,(data.shape[0],100, 100,1))

target=np.array(target)

new\_target=np\_utils.to\_categorical(target)

np.save('data',data)

np.save('target',new\_target)

The goal of face detection is to determine if there are any faces in the image or video. If multiple faces are present, each face is enclosed by a bounding box and thus we know the location of the faces

Human faces are difficult to model as there are many variables that can change for example facial expression, orientation, lighting conditions and partial occlusions such as sunglasses, scarf, mask etc. The result of the detection gives the face location parameters and it could be required in various forms, for instance, a rectangle covering the central part of the face, eye centers or landmarks including eyes, nose and mouth corners, eyebrows, nostrils, etc.

One of the popular algorithms that use a feature-based approach is the Viola-Jones algorithm and here I am briefly going to discuss it. If you want to know about it in detail, I would suggest going through this article, [Face Detection using Viola Jones Algorithm](https://www.mygreatlearning.com/blog/viola-jones-algorithm).

**Viola-Jones** algorithm is named after two computer vision researchers who proposed the method in 2001, Paul **Viola** and Michael **Jones** in their paper, “Rapid Object Detection using a Boosted Cascade of Simple Features”. Despite being an outdated framework, Viola-Jones is quite powerful, and its application has proven to be exceptionally notable in real-time face detection. This algorithm is painfully slow to train but can detect faces in real-time with impressive speed.

Given an image(this algorithm works on grayscale image), the algorithm looks at many smaller subregions and tries to find a face by looking for specific features in each subregion. It needs to check many different positions and scales because an image can contain many faces of various sizes. Viola and Jones used Haar-like features to detect faces in this algorithm.

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, lacking in certain behaviors or trends, and is likely to contain many errors.Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing.  
Data preprocessing is used in database-driven applications such as customer relationship management and rule-based applications (like neural networks).

In Machine Learning (ML) processes, data preprocessing is critical to encode the dataset in a form that could be interpreted and parsed by the algorithm.

**4.4 CNN Training:**

Deep learning refers to a subfield of machine learning that is based on learning levels of representations, corresponding to a hierarchy of features, factors or concepts, where higher-lever concepts are defined from lower-lever ones, and the same lower-lever concepts can help to define many higher-lever concepts. Deep learning is learning multiple levels of representation and abstraction, helps to understand the data such as images, audio and text. The concept of Deep Learning comes from the study of Artificial Neural Network, Multilayer Perceptron which contains more hidden layers is a Deep Learning structure. In the late 1980s, the invention of Back Propagation algorithm used in Artificial Neural Network brings hope to machine learning and creates a trend of machine learning based on statistical models. In the 1990s, a variety of Shallow Learning models have been proposed such as Support Vector Machines (SVM), Boosting, Logistic Regression (LR). The structure of these models can be seen as one hidden node (SVM, Boosting), or no hidden nodes (LR). These models gained a great success both in theoretical analysis and applications. In 2006, Geoffrey Hinton who is the professor of University of Toronto, Canada and the dean of machine learning and his students Ruslan Salakhutdinov published an article in “Science”, led to a trend of machine learning in academia and industry. The article had two points:

1) Artificial Neural Network with multiple hidden layers has an excellent ability of characteristic learning. The characteristics obtained from learning have an essential description to data, then facilitate visualization or classification.

2) The difficulties of deep neural network in training can overcome by layer-wise pre-training. In this article, the implementation of layer-wise pre-training is achieved through unsupervised learning. Feedforward neural network or Multilayer Perceptron with multiple hidden layers in artificial neural networks is usually known as Deep Neural Networks (DNNs). Convolutional Neural Networks (CNN) is one kind of feedforward neural network.

In 1960s, when Hubel and Wiesel researched the neurons used for local sensitive orientation-selective in the cat’s visual system, they found the special network structure can effectively reduce the complexity of Feedback Neural Networks and then proposed Convolution Neural Network. CNN is an efficient recognition algorithm which is widely used in pattern recognition and image processing. It has many features such as simple structure, less training parameters and adaptability. It has become a hot topic in voice analysis and image recognition

Its weight shared network structure make it more similar to biological neural networks. It reduces the complexity of the network model and the number of weights. Generally, the structure of CNN includes two layers one is feature extraction layer, the input of each neuron is connected to the local receptive fields of the previous layer, and extracts the local feature. Once the local features is extracted, the positional relationship between it and other features also will be determined. The other is feature map layer, each computing layer of the network is composed of a plurality of feature map.

Every feature map is a plane, the weight of the neurons in the plane are equal. The structure of feature map uses the sigmoid function as activation function of the convolution network, which makes the feature map have shift invariance. Besides, since the neurons in the same mapping plane share weight, the number of free parameters of the network is reduced. Each convolution layer in the convolution neural network is followed by a computing layer which is used to calculate the local average and the second extract, this unique two feature extraction structure reduces the resolution. CNN is mainly used to identify displacement, zoom and other forms of distorting invariance of two-dimensional graphics. Since the feature detection layer of CNN learns by training data, it avoids explicit feature extraction and implicitly learns from the training data when we use CNN. Furthermore, the neurons in the same feature map plane have the identical weight, so the network can study concurrently.

This is a major advantage of the convolution network with respect to the neuronal network connected to each other. Because of the special structure of the CNN’s local shared weights makes it have a unique advantage in speech recognition and image processing. Its layout is closer to the actual biological neural network. Shared weights reduces the complexity of the network. In particular multi-dimensional input vector image can directly enter the network, which avoids the complexity of data reconstruction in feature extraction and classification process. Face recognition is a biometric identification technology based on the facial features of persons. The study of face recognition system began in the 1960s, in the late 1980s with the development of computer technology and optical imaging techniques it has been improved; in the late 1990s it truly entered the stages of initial applications.

In practical applications, such as monitoring system, the collected face images captured by cameras are often low resolution and with great pose variations. Affected by pose variation and low resolution, the performance of face recognition degrades sharply. And pose variations bring great challenge to face recognition. They bring nonlinear factors into face recognition. And some of the existing machine learning method mostly use shallow structure. Deep learning can achieve the approximation of complex function by a deep nonlinear network structure. In this article, we use convolution neural network to solve face recognition. It can overcome the influence of pose or resolution in face recognition. Due to the long training time and the large recognition computing, it is difficult to meet the real-time requirements, or the delay exceeds the range of tolerance. So we use the cloud platform.

**BACKGROUND AND RELATED WORK**:

Convolutional Neural Networks can be applied in many different areas. Yann LeCun and his team specially designed Convoutional Neural Networks to deal with the variability of 2D shapes, which are shown to outperform all other techniques.[1] Dan C and his team present a fast, fully parameterizable GPU implementation of Convolutional Neural Network variants for Image Classification. [2] Another team proposes two novel frontends for robust language identification (LID) using a convolutional neural network (CNN) trained for automatic speech recognition (ASR). [5] What’s more, Convolutional Neural Networks are used in Visual Recognition[9] and many other areas, such as Facial Point Detection[6], House Numbers Digit Classification[10], Multi-digit Number Recognition from Street View Imagery[11]. Besides these works, many teams are focusing on the speed up of ConvNets. For example, Multi-GPU Training of ConvNets. In this work , Facebook AI Group consider a standard architecture [1] trained on the Imagenet dataset [2] for classification and investigate methods to speed convergence by parallelizing training across multiple GPUs.

Convolution neural network algorithm is a multilayer perceptron that is the special design for identification of two-dimensional image information . Always has more layers: input layer, convolution layer, sample layer and output layer. In addition, in a deep network architecturethe convolution layer and sample layer can have multiple. CNN is not as restricted boltzmann machine, need to be before and after the layer of neurons in the adjacent layer for all connections, convolution neural network algorithms, each neuron don't need to do feel global image, just feel the local area of the image. In addition, each neuron parameter is set to the same, namely, the sharing of weights , namely each neuron with the same convolution kernels to deconvolution image.

CNN algorithm has two main processes: convolution and sampling . Convolution process: use a trainable filter Fx, deconvolution of the input image (the first stage is the input image, the input of the after convolution is the feature image of each layer, namely Feature Map), then add a bias bx, we can get convolution layer Cx. A sampling process: n pixels of each neighborhood through pooling steps, become a pixel, and then by scalar weighting Wx + 1 weighted, add bias bx + 1, and then by an activation function, produce a narrow n times feature map Sx + 1.

The key technology of CNN is the local receptive field, sharing of weights , sub sampling by time or space, so as to extract feature and reduce the size of the training parameters.The advantage of CNN algorithm is that to avoid the explicit feature extraction, and implicitly to learn from the training data;The same neuron weights on the surface of the feature mapping, thus network can learn parallelly , reduce the complexity of the network;Adopting sub sampling structure by time or space, can achieve some degree of robustness, scale and deformation displacement;Input information and network topology can be a very good match, It has unique advantages in speech recognition and image processing.

In view of the 32 × 32 input after preprocessing, There is a total of 17 different pictures. C1 layer for convolution, convolution layer adopts 6 convolution kernels, each the size of the convolution kernels is 5 × 5, can produce six feature map, each feature map contains (32-5 + 1) × (32-5 + 1) = 28 × 28 = 784 neurons. At this point, a total of 6 × (5 × 5 + 1) = 156 parameters to be trained . S1 layer for sub sampling, contains six feature map, each feature map contains 14 \* 14 = 196 neurons. the sub sampling window is 2 × 2 matrix, sub sampling step size is 1, so the S1 layer contains 6 × 196 × (2 × 2 + 1) = 5880 connections. Every feature map in the S1 layer contains a weights and bias, so a total of 12 parameters can be trained in S1 layer . C2 is convolution layer, containing 16 feature graph, each feature graph contains (14-5 + 1) (14-5 + 1) = 100 neurons and adopts full connection, namely each characteristic figure used to belong to own 6 convolution kernels with six characteristics of the sample layer S1 convolution and figure. Each feature graph contains 6 × 5 × 5 = 150 weights and a bias. So, C2 layer contains a total of 16 × (150 + 1) = 150 parameters to be trained. S2 is sub sampling layer, containing 16 feature map, each feature map contains 5 × 5 neurons, S2 total containing 25 × 16 = 400 neurons. S2 on characteristic figure of sub sampling window for 2 × 2, so there is 32 trainable S2 parameters. As a whole connection layer, hidden layer H contains 170 neurons, each neuron is connected to 400 neurons on S2. So H layer contains 170 × (400 + 1) = 48120 parameters feature map. Output layer F for all connections, including 17 neurons. A total of 17 × (170 + 1) = 2907 parameters to be trained.

**CNN Algorithm and Back propagation algorithm:**

output of neuron of row k , column y in the l th convolution layer and k th feature pattern：

( ) tanh( ) ( 1, ) ( , ) ( , ) 1 0 0 0 ( , ) ( , ) , , W O Bias l t l k x r x c f t r c k t r c l k x y k k O h w = + − + + − = = = ∑∑∑ （3.1） among them, f is the number of convolution cores in a feature pattern。 output of neuron of row x , column y in the l th sub sample layer and k th feature pattern： ( ) tanh( ) ( , ) 0 0 ( 1, ) ( , ) ( ) , , W O Biasl k r c l k x r y c k l k x y s s s s O h w h w = ∑∑ + = = − × + × + （3.2） the output of the j th neuron in l th hide layer H ： tanh( ) ( 1, ) ( , ) ( , ) 1 0 0 0 ( , ) O( , ) W( , )O Bias l k l j x y sk x y j k l j x y sh sw = + − − = = = ∑∑∑ （3.3） among them, sis the number of feature patterns in sample layer. output of the The second stage of SRCNet is facemask-wearing condition identification. As CNNs are one of

the most common types of network for image classification, which perform well in facial recognition,

a CNN was adopted for the facemask-wearing condition identification network in the second stage

of SRCNet. The goal was to form a function G(FI), where FI is the input face image, which outputsthe probabilities of the three categories (i.e., NFW, IFW, and CFW). The classifier then outputs theclassification result based on the output possibilities.

MobileNet-v2 was applied as the facemask-wearing condition identification network, which is

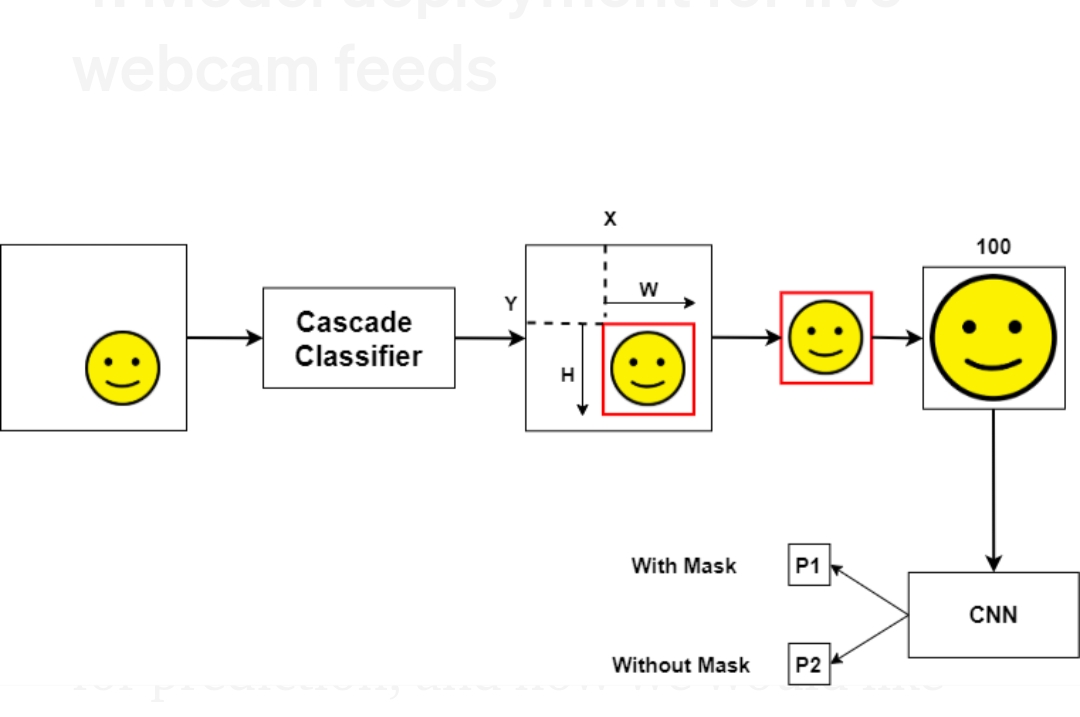
a lightweight CNN that can achieve high accuracy in image classification. The main features ofMobileNet-v2 are residual blocks and depthwise separable convolution [42,43]. The residual blockscontribute to the training of the deep network, addressing the gradient vanishing problem andachieving benefits by back-propagating the gradient to the bottom layers. As for facemask-wearingcondition identification, there are slight di\_erences between IFW and CFW. Hence, the capability off eature extraction or the depth of the network are essential, contributing to the final identification accuracy. Depthwise separable convolution is applied for the reduction of computation and model size while maintaining the final classification accuracy, which separable convolution splits into two layers: One layer for filtering and another layer for combining. Transfer learning is applied in the network training procedure, which is a kind of knowledge migration between the source and target domains. The network is trained in three steps: Initialization,forming a general facial recognition model, and knowledge transfer to facemask-wearing condition identification. The first step is initialization, which contributes to the final identification accuracy and training speed Then, a general facial recognition model is formed using a large facial image dataset, where the network gains the capability of facial feature extraction. After watching millions of faces, the network then concentrates on facial information, rather than the interference frombackgrounds and the di\_erences caused by image shooting parameters. The final step is knowledge transfer between facial recognition and facemask-wearing condition identification. The final fully connected layer is modified to meet with category requirements of facemask-wearing condition identification. The reason for adopting transfer learning was the considerable di\_erences in data volumes and their consequences. The facemask-wearing condition identification dataset is relatively small, compared to general facial recognition datasets, which may cause overfitting problems and a reduction in identification accuracy during the training process. Hence, the network gains knowledge about faces in the general facial recognition model training process for the reduction in overfitting and the improvement in accuracy. The final stage of the classifier is the softmax function, which calculates the probabilities of all classes using the outputs of its direct ancestor (i.e., fully connected layer neurons).

**4.5 Prediction:**

“Prediction” refers to the output of an after it has been  on a historical dataset and applied to new data when forecasting the likelihood of a particular outcome, such as whether or not a customer will churn in 30 days. The algorithm will generate probable values for an unknown variable for each record in the new data, allowing the model builder to identify what that value will most likely be.

The word “prediction” can be misleading. In some cases, it really does mean that you are predicting a future outcome, such as when you’re using machine learning to determine the  in a marketing campaign. Other times, though, the “prediction” has to do with, for example, whether or not a transaction that already occurred was fraudulent. In that case, the transaction already happened, but you’re making an educated guess about whether or not it was legitimate, allowing you to take the appropriate action.

Machine learning  predictions allow businesses to make highly accurate guesses as to the likely outcomes of a question based on historical data, which can be about all kinds of things – customer churn likelihood, possible fraudulent activity, and more. These provide the business with  that result in tangible business value. For example, if a model predicts a customer is likely to churn, the business can target them with specific communications and outreach that will prevent the loss of that customer.



**Fig 4.5 Model deployment**

**CHAPTER 5**

**SYSTEM ARCHITECTURE**

In the past few decades, Deep Learning has proved to be a very powerful tool because of its ability to handle large amounts of data. The interest to use hidden layers has surpassed traditional techniques, especially in pattern recognition. One of the most popular deep neural networks is Convolutional Neural Networks.

Since the 1950s, the early days of AI, researchers have struggled to make a system that can understand visual data. In the following years, this field came to be known as Computer Vision. In 2012, computer vision took a quantum leap when a group of researchers from the University of Toronto developed an AI model that surpassed the best image recognition algorithms and that too by a large margin.

The AI system, which became known as AlexNet (named after its main creator, Alex Krizhevsky), won the 2012 ImageNet computer vision contest with an amazing 85 percent accuracy. The runner-up scored a modest 74 percent on the test.

At the heart of AlexNet was Convolutional Neural Networks a special type of neural network that roughly imitates human vision. Over the years CNNs have become a very important part of many Computer Vision applications. So let’s take a look at the workings of CNNs.

CNN’s were first developed and used around the 1980s. The most that a CNN could do at that time was recognize handwritten digits. It was mostly used in the postal sectors to read zip codes, pin codes, etc. The important thing to remember about any deep learning model is that it requires a large amount of data to train and also requires a lot of computing resources. This was a major drawback for CNNs at that period and hence CNNs were only limited to the postal sectors and it failed to enter the world of machine learning.

In 2012 Alex Krizhevsky realized that it was time to bring back the branch of deep learning that uses multi-layered neural networks. The availability of large sets of data, to be more specific ImageNet datasets with millions of labeled images and an abundance of computing resources enabled researchers to revive CNNs.

In  a **convolutional neural network** (**CNN/ConvNet**) is a class of most commonly applied to analyze visual imagery. Now when we think of a neural network we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Now in mathematics **convolution** is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other.

But we don’t really need to go behind the mathematics part to understand what a CNN is or how it works.

Bottom line is that the role of the ConvNet is toreduce the images into a form that is easier to process, without losing features that are critical for getting a good prediction.

Before we go to the working of CNN’s let’s cover the basics such as what is an image and how is it represented. An RGB image is nothing but a matrix of pixel values having three planes whereas a grayscale image is the same but it has a single plane. Take a look at this image to understand more.

Convolutional neural networks are composed of multiple layers of artificial neurons. Artificial neurons, a rough imitation of their biological counterparts, are mathematical functions that calculate the weighted sum of multiple inputs and outputs an activation value. When you input an image in a ConvNet, each layer generates several activation functions that are passed on to the next layer.

The first layer usually extracts basic features such as horizontal or diagonal edges. This output is passed on to the next layer which detects more complex features such as corners or combinational edges. As we move deeper into the network it can identify even more complex features such as objects, faces, etc.

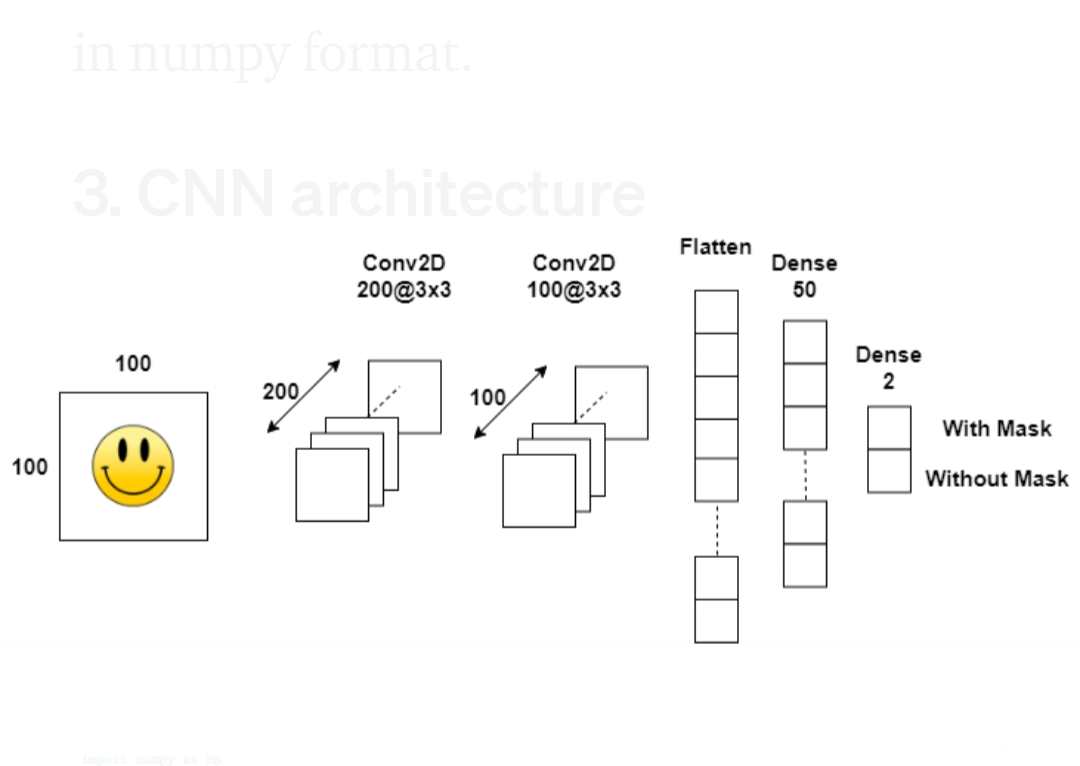
Based on the activation map of the final convolution layer, the classification layer outputs a set of confidence scores (values between 0 and 1) that specify how likely the image is to belong to a “class.” For instance, if you have a ConvNet that detects cats, dogs, and horses, the output of the final layer is the possibility that the input image contains any of those animals.

## **5.1`What’s a pooling layer:**

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to **decrease the computational power required to process the data** by reducing the dimensions. There are two types of pooling average pooling and max pooling. I’ve only had experience with Max Pooling so far I haven’t faced any difficulties.

So what we do in Max Pooling is we find the maximum value of a pixel from a portion of the image covered by the kernel. Max Pooling also performs as a**Noise Suppressant**. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction.

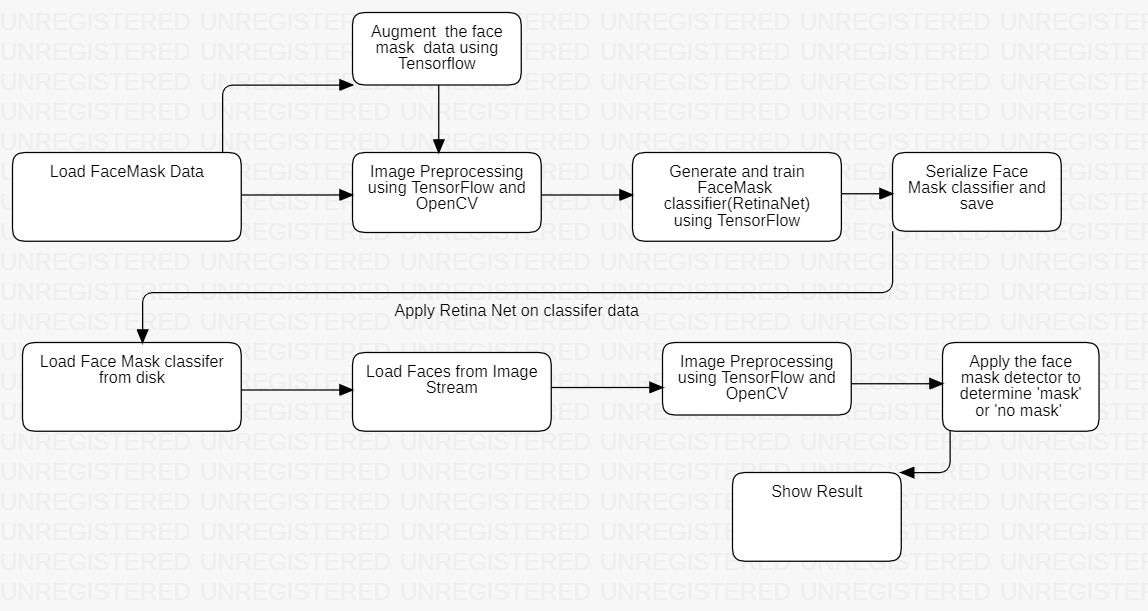
On the other hand**, Average Pooling**returns **the average of all the values**from the portion of the image covered by the Kernel. Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that **Max Pooling performs a lot better than Average Pooling.**

**Fig 5.1.1 CNN Model**

Despite the power and resource complexity of CNNs, they provide in-depth results. At the root of it all, it is just recognizing patterns and details that are so minute and inconspicuous that it goes unnoticed to the human eye. But when it comes to **understanding**the contents of an image it fails.

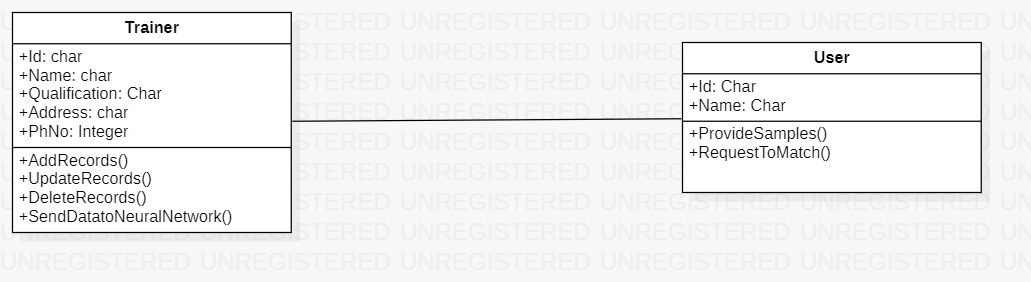
Let’s take a look at this example. When we pass the below image to a CNN it detects a person in their mid-30s and a child probably around 10 years. But when we look at the same image we start thinking of multiple different scenarios. Maybe it’s a father and son day out, a picnic or maybe they are camping. Maybe it is a school ground and the child scored a goal and his dad is happy so he lifts him.

**5.2. Flow chart:**

****

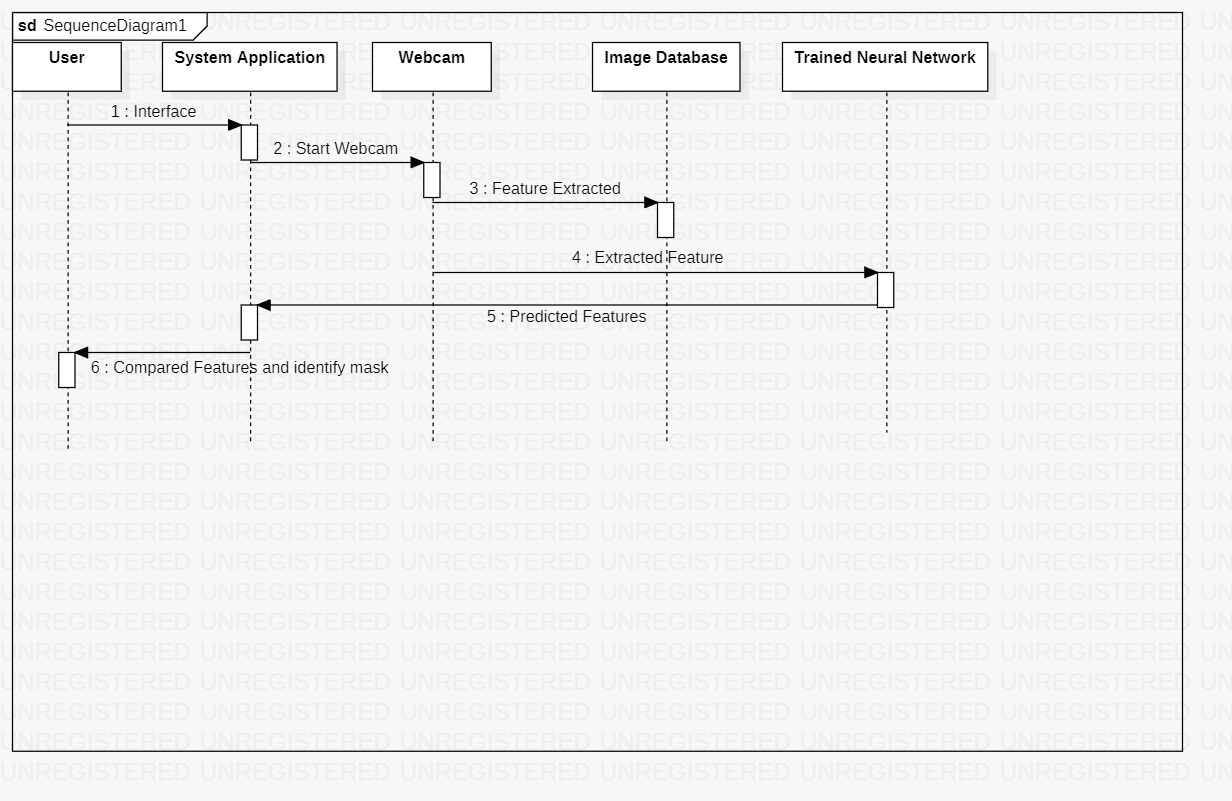
**Fig 5.2 Flow chart**

**5.3 Class Diagram:**

****

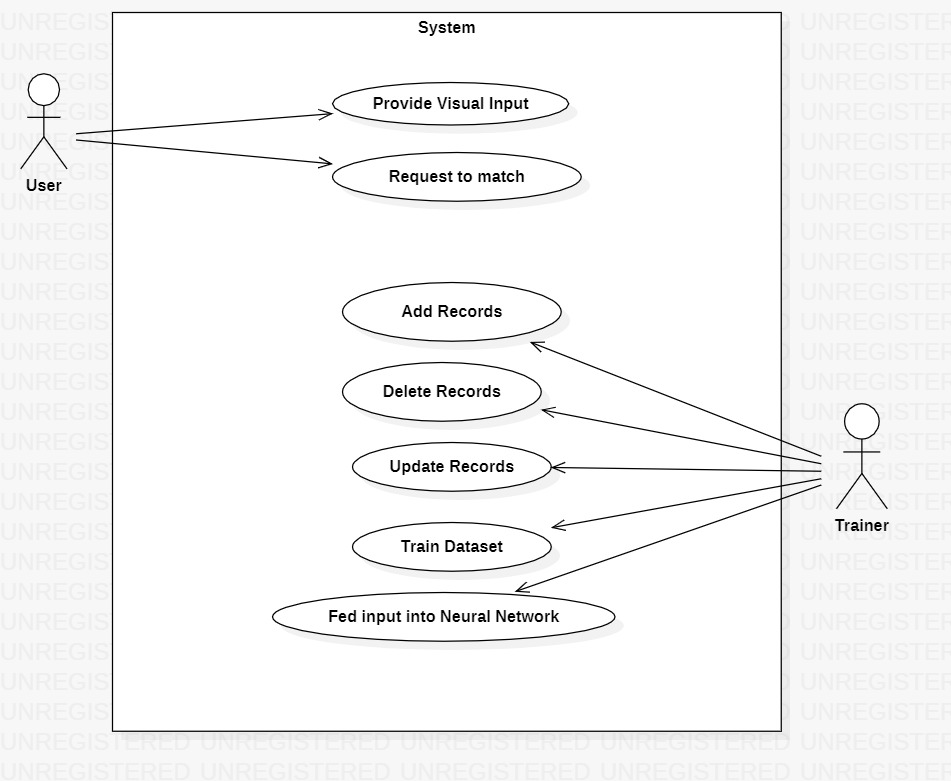
**Fig 5.3 Class diagram**

**5.4 Sequence Diagram:**

****

**Fig 5.4 Sequence Diagram**

**5.5 Use Case Diagram:**

****

**Fig 5.5 Use Case Diagram**

**CHAPTER 6**

**IMPLEMENTATION**

**6.1 Data pre processing code:**

import cv2,os

import numpy as np

from keras.utils import np\_utils

data\_path='C:\\Users\\MYPC\\Projects\\face-mask-detector\\dataset'

categories=os.listdir(data\_path)

labels=[i for i in range(len(categories))]

label\_dict=dict(zip(categories,labels))

print(label\_dict)

print(categories)

print(labels)

data=[]

target=[]

**6.2 Converting images into grayscale image:**

''' The dataset we are using consists of images with different colors, different sizes, and different orientations. Therefore, we need to convert all the images into grayscale because we need to be sure that color should not be a critical point for detecting mask. After that, we need to have all the images in the same size (150x150) before applying it to the neural network. '''

for category in categories:

folder\_path=os.path.join(data\_path,category)

img\_names=os.listdir(folder\_path)

for img\_name in img\_names:

mg\_path=os.path.join(folder\_path,img\_name)

img=cv2.imread(img\_path)

try:

gray=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

#Coverting the image into gray scale

resized=cv2.resize(gray,(100,100))

#resizing the gray scale into 100x100, since we need a fixed common size for all the images in the dataset

data.append(resized)

target.append(label\_dict[category])

#appending the image and the label(categorized) into the list (dataset)

except Exception as e:

print('Exception:',e)

#if any exception rasied, the exception will be printed here. And pass to the next image

**6.3 Loading the saved numpy arrays:**

import numpy as np

data=np.array(data)/255.0

data=np.reshape(data,(data.shape[0],60,60,1))

target=np.array(target)

new\_target=np\_utils.to\_categorical(target)

np.save('data',data)

np.save('target',new\_target)

data=np.load('data.npy')

target=np.load('target.npy')

**6.4 Building Sequencial Model:**

'''Here,we build our Sequential CNN model with various layers such as Conv2D,MaxPooling2D, Flatten, Dropout and Dense. In the last Dense layer, we use the ‘softmax’ function to output a vector that gives the probability of each of the two classes.Since we have two categories (with mask and without mask) we can use binary\_crossentropy.'''

from keras.models import Sequential

from keras.layers import Dense,Activation,Flatten,Dropout

from keras.layers import Conv2D,MaxPooling2D

from keras.callbacks import ModelCheckpoint

model=Sequential()

model.add(Conv2D(100,(3,3),input\_shape=data.shape[1:]))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2,2)))

#The first CNN layer followed by Relu and MaxPooling layers

model.add(Conv2D(100,(3,3)))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2,2)))

#The second convolution layer followed by Relu and MaxPooling layers

model.add(Dropout(0.5))

#Flatten layer to stack the output convolutions from second convolution layer

model.add(Dense(50,activation='relu'))

#Dense layer of 64 neurons

model.add(Dense(2,activation='softmax'))

#The Final layer with two outputs for two categoriesmodel.compile(loss='binary\_crossentropy',optimizer='adam',metrics=['accuracy'])

**6.5 Training the CNN Model:**

'''In this step we fit our images in the training set and the test set to our Sequential model we built using keras library.I have trained the model for 20 epochs (iterations). However, we can train for more number of epochs to attain higher accuracy lest there occurs over-fitting.'''

from sklearn.model\_selection import train\_test\_split

train\_data,test\_data,train\_target,test\_target=train\_test\_split(data,target,test\_size=0.1)

checkpoint=ModelCheckpoint('model{epoch:03d}.model', monitor='val\_loss',verbose=0,save\_best\_only=True,mode='auto')

history=model.fit(train\_data,train\_target,epochs=20, callbacks=[checkpoint], validation\_split=0.2)

**6.6 Deployment:**

import numpy as np

import keras

import keras.backend as k

from keras.layers import Conv2D,MaxPooling2D,SpatialDropout2D,Flatten,Dropout,Dense

from keras.models import Sequential,load\_model

#from keras.optimizers import adam

from tensorflow.python.keras.optimizers import \*

#from keras\_radam import RAdam

from keras.preprocessing import image

import cv2

import datetime

**# IMPLEMENTING LIVE DETECTION OF FACE MASK**

mymodel=load\_model('leo.h5')

cap=cv2.VideoCapture(0)

face\_cascade=cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

while cap.isOpened():

\_,img=cap.read()

face=face\_cascade.detectMultiScale(img,scaleFactor=1.1,minNeighbors=4)

for(x,y,w,h) in face:

face\_img = img[y:y+h, x:x+w]

cv2.imwrite('temp.jpg',face\_img)

test\_image=image.load\_img('temp.jpg',target\_size=(150,150,3))

test\_image=image.img\_to\_array(test\_image)

test\_image=np.expand\_dims(test\_image,axis=0)

pred=mymodel.predict\_classes(test\_image)[0][0]

if pred==1:

cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),3)

cv2.putText(img,'NO MASK',((x+w)//2,y+h+20),cv2.FONT\_HERSHEY\_SIMPLEX,1,(0,0,255),3)

else:

cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),3)

cv2.putText(img,'MASK',((x+w)//2,y+h+20),cv2.FONT\_HERSHEY\_SIMPLEX,1,(0,255,0),3)

datet=str(datetime.datetime.now())

cv2.putText(img,datet,(400,450),cv2.FONT\_HERSHEY\_SIMPLEX,0.5,(255,255,255),1)

cv2.imshow('img',img)

if cv2.waitKey(1)==ord('q'):

break

cap.release()

cv2.destroyAllWindows()

**CHAPTER 7**

**TESTING**

Software testing is the process of evaluation a software item to detect differences between given input and expected output. Also to assess the feature of A software item. Testing assesses the quality of the product. Software testing is a process that should be done during the development process. In other words software testing is a verification and validation process.

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

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

## **Why Software Testing is Important:**

**Software Testing is Important** because if there are any bugs or errors in the software, it can be identified early and can be solved before delivery of the software product. Properly tested software product ensures reliability, security and high performance which further results in time saving, cost effectiveness and customer satisfaction.

Testing is important because software bugs could be expensive or even dangerous. Software bugs can potentially cause monetary and human loss, and history is full of such examples.

* In April 2015, Bloomberg terminal in London crashed due to software glitch affected more than 300,000 traders on financial markets. It forced the government to postpone a 3bn pound debt sale.
* Nissan cars recalled over 1 million cars from the market due to software failure in the airbag sensory detectors. There has been reported two accident due to this software failure.
* Starbucks was forced to close about 60 percent of stores in the U.S and Canada due to software failure in its POS system. At one point, the store served coffee for free as they were unable to process the transaction.
* Some of Amazon's third-party retailers saw their product price is reduced to 1p due to a software glitch. They were left with heavy losses.
* Vulnerability in Windows 10. This bug enables users to escape from security sandboxes through a flaw in the win32k system.
* In 2015 fighter plane F-35 fell victim to a software bug, making it unable to detect targets correctly.
* China Airlines Airbus A300 crashed due to a software bug on April 26, 1994, killing 264 innocents live
* In 1985, Canada's Therac-25 radiation therapy machine malfunctioned due to software bug and delivered lethal radiation doses to patients, leaving 3 people dead and critically injuring 3 others.
* In April of 1999, a software bug caused the failure of a $1.2 billion military satellite launch, the costliest accident in history
* In May of 1996, a software bug caused the bank accounts of 823 customers of a major U.S. bank to be credited with 920 million US dollars.

## **What are the benefits of Software Testing:**

Here are the benefits of using software testing:

* **Cost-Effective:**It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.
* **Security:**It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.
* **Product quality:**It is an essential requirement of any software product. Testing ensures a quality product is delivered to customers.
* **Customer Satisfaction:**The main aim of any product is to give satisfaction to their customers. UI/UX Testing ensures the best user experience.

#### Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to.

validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements.

#### **Basics of software testing:**

There are two basics of software testing: blackbox testing and whitebox testing.

##### **7.3.1Blackbox Testing:**

Black box testing is a testing technique that ignores the internal mechanism of the system and focuses on the output generated against any input and execution of the system. It is also called functional testing.

##### **7.3.2White**box**Testing**

White box testing is a testing technique that takes into account the internal mechanism of a system. It is also called structural testing and glass box testing.

Black box testing is often used for validation and white box testing is often used for verification.

**Type of testing:**

there are many types of testing like

1. Unit Testing
2. Integration Testing
3. Functional Testing
4. System Testing
5. Stress Testing
6. Performance Testing
7. Usability Testing
8. Acceptance Testing
9. Regression Testing
10. Beta Testing

**Unit Testing:**

Unit testing is the testing of an individual unit or group of related units. It falls under the classof white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input.

**Integration Testing:**

Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing.

**Functional Testing:**

Functional testing is the testing to ensure that the specified functionality required in the system requirements works. It falls under the class of black box testing.

**System Testing:**

System testing is the testing to ensure that by putting the software in different environments (e.g., Operating Systems) it still works. System testing is done with full system implementation and environment. It falls under the class of black box testing.

**Stress Testing:**

Stress testing is the testing to evaluate how system behaves under unfavorable conditions. Testing is conducted at beyond limits of the specifications. It falls under the class of black box testing.

**Performance Testing:**

Performance testing is the testing to assess the speed and effectiveness of the system and to make sure it is generating results within a specified time as in performance requirements. It falls under the class of black box testing.

**Usability Testing:**

Usability testing is performed to the perspective of the client, to evaluate how the GUI is user-friendly? How easily can the client learn? After learning how to use, how proficiently can the client perform? How pleasing is it to use its design? This falls under the class of black box testing.

**Acceptance Testing:**

Acceptance testing is often done by the customer to ensure that the delivered product meets the requirements and works as the customer expected. It falls under the class of black box testing.

**Regression Testing:**

Regression testing is the testing after modification of a system, component, or a group of related units to ensure that the modification is working correctly and is not damaging or imposing other modules to produce unexpected results. It falls under the class of black box testing.

**Beta Testing:**

Beta testing is the testing which is done by end users, a team outside development, or publicly releasing full pre-version of the product which is known as beta version. The aim of beta testing is to cover unexpected errors. It falls under the class of black box testing.

**CHAPTER 8**

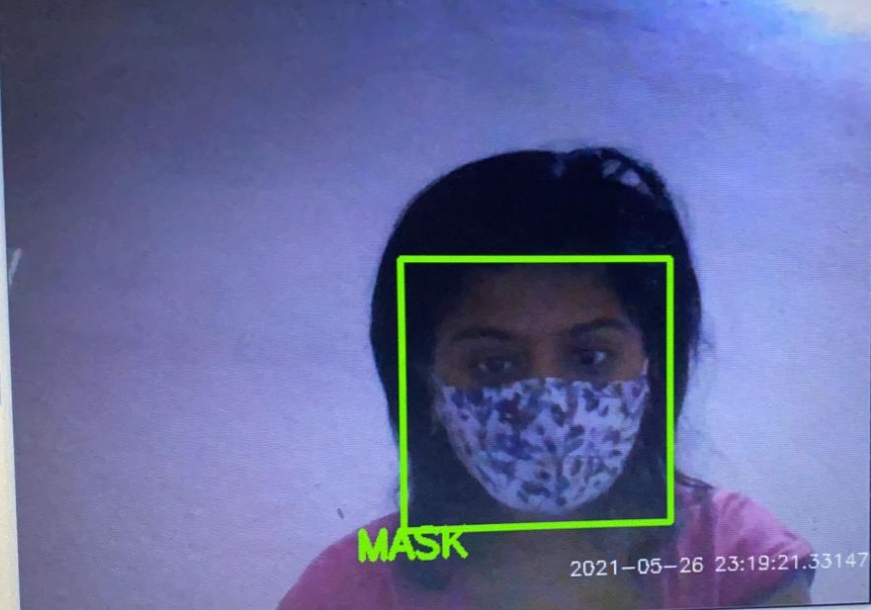
**SCREEN SHOT**

**8.1 A person without mask:**

****

**Fig 8.1 A person without mask**

**8.2 A person with mask:**

****

**Fig 8.2 A person with mask**

**CHAPTER 9**

**CONCLUSION**

* As the technology are blooming with emerging trends the availability so we have novel face mask detector which can possibly contribute to public healthcare.
* The architecture consists of Retina Net as the backbone it can be used for high and low computation scenarios. In order to extract more robust features, we utilize transfer learning to adopt weights from a similar task face detection, which is trained on a very large dataset.

**CHAPTER 10**

**REFERENCES**

* [1] P. A. Rota, M. S. Oberste, S. S. Monroe, W. A. Nix, R. Campagnoli, J. P. Icenogle, S. Penaranda, B. Bankamp,

K. Maher, M.-h. Chenet al., “Characterization of a novel coronavirus associated with severe acute

respiratorysyndrome,”science, vol. 300, no. 5624, pp. 1394–1399, 2003.

* [2] Z. A. Memish, A. I. Zumla, R. F. Al-Hakeem, A. A. AlRabeeah, and G. M. Stephens, “Family cluster of

middleeast respiratory syndrome coronavirus infections,”New England Journal of Medicine, vol. 368, no. 26,

pp.2487–2494, 2013.

* [3] Y. Liu, A. A. Gayle, A. Wilder-Smith, and J. Rocklöv, “The reproductive number of covid-19 is higher

comparedto sars coronavirus,”Journal of travel medicine, 2020.

* [4] Y. Fang, Y. Nie, and M. Penny, “Transmission dynamics of the covid-19 outbreak and effectiveness of

governmentinterventions:

* A datadriven analysis,”Journal of medical virology, vol. 92, no. 6, pp. 645–659, 2020. Chavez, S.; Long, B.; Koyfman, A.; Liang, S.Y. Coronavirus Disease (COVID-19): A primer for emergency
* physicians. Am. J. Emerg. Med. **2020**. [CrossRef] [PubMed]
* Cowling, B.J.; Chan, K.H.; Fang, V.J.; Cheng, C.K.; Fung, R.O.;Wai,W.; Sin, J.; Seto,W.H.; Yung, R.; Chu, D.W.;
* et al. Facemasks and hand hygiene to prevent influenza transmission in households: A cluster randomized
* trial. Ann. Intern. Med. **2009**, 151, 437–446. [CrossRef] [PubMed]
* Tracht, S.M.; Del Valle, S.Y.; Hyman, J.M. Mathematical modeling of the e\_ectiveness of facemasks in reducing
* the spread of novel influenza A (H1N1). PLoS ONE **2010**, 5, e9018. [CrossRef] [PubMed]
* Je\_erson, T.; Del Mar, C.B.; Dooley, L.; Ferroni, E.; Al-Ansary, L.A.; Bawazeer, G.A.; van Driel, M.L.; Nair, S.;
* Jones, M.A.; Thorning, S.; et al. Physical interventions to interrupt or reduce the spread of respiratory viruses.
* Cochrane Database Syst. Rev. **2011**, CD006207. [CrossRef] [PubMed]
* Sim, S.W.; Moey, K.S.; Tan, N.C. The use of facemasks to prevent respiratory infection: A literature review in
* the context of the Health Belief Model. Singap. Med. J. **2014**, 55, 160–167. [CrossRef]
* Lai, A.C.; Poon, C.K.; Cheung, A.C. E\_ectiveness of facemasks to reduce exposure hazards for airborne
* infections among general populations. J. R. Soc. Interface **2012**, 9, 938–948. [CrossRef] [PubMed]
* Elachola, H.; Ebrahim, S.H.; Gozzer, E. COVID-19: Facemask use prevalence in international airports in Asia,
* Europe and the Americas, March 2020. Travel Med. Infect. Dis. **2020**, 101637. [CrossRef]
* Je\_erson, T.; Foxlee, R.; Del Mar, C.; Dooley, L.; Ferroni, E.; Hewak, B.; Prabhala, A.; Nair, S.; Rivetti, A.
* Physical interventions to interrupt or reduce the spread of respiratory viruses: Systematic review. BMJ **2008**,
* 336, 77–80. [CrossRef]
* LeCun, Y.; Kavukcuoglu, K.; Farabet, C. Convolutional Networks and Applications in Vision. In Proceedings
* of the 2010 IEEE International Symposium on Circuits and Systems, Paris, France, 30 May–2 June 2010;
* pp. 253–256.
* Qin, B.; Liang, L.; Wu, J.; Quan, Q.; Wang, Z.; Li, D. Automatic Identification of Down Syndrome Using
* Facial Images with Deep Convolutional Neural Network. Diagnostics **2020**, 10, 487. [CrossRef]
* Cristani, M.; Bue, A.D.; Murino, V.; Setti, F.; Vinciarelli, A. The Visual Social Distancing Problem. IEEE Access
* **2020**, 8, 126876–126886. [CrossRef]
* Zhang, K.; Zhang, Z.; Li, Z.; Qiao, Y. Joint Face Detection and Alignment Using Multitask Cascaded
* Convolutional Networks. IEEE Signal Process. Lett. **2016**, 23, 1499–1503. [CrossRef]
* Sensors **2020**, 20, 5236 20 of 23
* Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks.
* Commun. ACM **2017**, 60, 84–90. [CrossRef]
* Ning, F.; Delhomme, D.; LeCun, Y.; Piano, F.; Bottou, L.; Barbano, P.E. Toward automatic phenotyping of
* developing embryos from videos. IEEE Trans. Image Process. **2005**, 14, 1360–1371. [CrossRef] [PubMed]
* Koo, J.H.; Cho, S.W.; Baek, N.R.; Kim, M.C.; Park, K.R. CNN-Based Multimodal Human Recognition in
* Surveillance Environments. Sensors **2018**, 18, 3040. [CrossRef]
* Liu, S.; Song, Y.; Zhang, M.; Zhao, J.; Yang, S.; Hou, K. An Identity Authentication Method Combining
* Liveness Detection and Face Recognition. Sensors **2019**, 19, 4733. [CrossRef] [PubMed]
* Mao, X.-J.; Shen, C.; Yang, Y.-B. Image Restoration Using Convolutional Auto-encoders with Symmetric Skip
* Connections. arXiv **2016**, arXiv:1606.08921.
* Zhang, Y.; Tian, Y.; Kong, Y.; Zhong, B.; Fu, Y. Residual Dense Network for Image Super-Resolution.
* In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City,
* UT, USA, 18–22 June 2018; pp. 2472–2481.
* Yang, K.; Hu, X.; Chen, H.; Xiang, K.; Wang, K.; Stiefelhagen, R. DS-PASS: Detail-Sensitive Panoramic
* Annular Semantic Segmentation through SwaftNet for Surrounding Sensing. arXiv **2019**, arXiv:1909.07721.
* Bruzzone, L.; Benediktsson, J.A.; Bovolo, F.; Gyftakis, S.; Tsenoglou, T.; Bratsolis, E.; Charou, E.; Vassilas, N.
* Fusion of aerial images with mean shift-based upsampled elevation data for improved building block
* classification. In Proceedings of the Image and Signal Processing for Remote Sensing XX, Amsterdam,
* The Netherlands, 22–24 September 2014.
* Thomas, R.; Rangachar, M.J.S. Fractional Bat and Multi-Kernel-Based Spherical SVM for Low Resolution
* Face Recognition. Int. J. Pattern Recognit. Artif. Intell. **2017**, 31. [CrossRef]
* Liang, M.; Jiao, L.; Yang, S.; Liu, F.; Hou, B.; Chen, H. Deep Multiscale Spectral-Spatial Feature Fusion for
* Hyperspectral Images Classification. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. **2018**, 11, 2911–2924.
* [CrossRef]
* Zhu, X.; Li, Z.; Li, X.; Li, S.; Dai, F. Attention-aware perceptual enhancement nets for low-resolution image
* classification. Inform. Sci. **2020**, 515, 233–247. [CrossRef]
* Yang, J.; Wright, J.; Huang, T.S.; Ma, Y. Image super-resolution via sparse representation. IEEE Trans.
* Image Process. **2010**, 19, 2861–2873. [CrossRef] [PubMed]
* Timofte, R.; De Smet, V.; Van Gool, L. A+: Adjusted Anchored Neighborhood Regression for Fast
* Super-Resolution. In Computer Vision—ACCV 2014; Cremers, D., Reid, I., Saito, H., Yang, M.H., Eds.;
* Springer: Cham, Switzerland, 2015; Volume 9006, pp. 111–126.
* Kim, J.; Lee, J.K.; Lee, K.M. Deeply-Recursive Convolutional Network for Image Super-Resolution. arXiv
* **2015**, arXiv:1511.04491.
* Dong, C.; Change Loy, C.; Tang, X. Accelerating the Super-Resolution Convolutional Neural Network. arXiv
* **2016**, arXiv:1608.00367.
* Johnson, J.; Alahi, A.; Fei-Fei, L. Perceptual Losses for Real-Time Style Transfer and Super-Resolution.
* In Computer Vision—ECCV 2016; Leibe, B., Matas, J., Sebe, N.,Welling, M., Eds.; Springer: Cham, Switzerland,
* 2016; Volume 9906, pp. 694–711.
* Kim, J.; Lee, J.K.; Lee, K.M. Accurate Image Super-Resolution Using Very Deep Convolutional Networks.
* In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas,
* NV, USA, 27–30 June 2016; pp. 1646–1654.
* Shi, W.; Caballero, J.; Huszar, F.; Totz, J.; Aitken, A.P.; Bishop, R.; Rueckert, D.; Wang, Z. Real-Time
* Single Image and Video Super-Resolution Using an E\_cient Sub-Pixel Convolutional Neural Network.
* In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas,
* NV, USA, 27–30 June 2016; pp. 1874–1883.
* Chen, Y.; Tai, Y.; Liu, X.; Shen, C.; Yang, J. FSRNet: End-to-End Learning Face Super-Resolution with Facial
* Priors. arXiv **2017**, arXiv:1711.10703.
* Lai,W.-S.; Huang, J.-B.; Ahuja, N.; Yang, M.-H. Deep Laplacian Pyramid Networks for Fast and Accurate
* Super-Resolution. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition
* (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 5835–5843.
* Lim, B.; Son, S.; Kim, H.; Nah, S.; Lee, K.M. Enhanced Deep Residual Networks for Single Image
* Super-Resolution. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition
* Workshops (CVPRW), Honolulu, HI, USA, 21–26 July 2017; pp. 1132–1140.