

# **AN EFFICIENT APPROACH TOWARDS IDENTIFYING AND RECOGNIZING FACES IN REAL TIME EMPLOYING DEEP LEARNING**

A DISSERTATION

Submitted in partial fulfilment of the requirements for the award of the  
degree  
of  
Master of Technology (M.Tech)  
in  
Computer Science and Engineering (CSE)

Submitted by  
**Lt Col Vivek B Sharma** (Roll No: **2K20/CSE/27**)

under the guidance of  
**Prof. Anil Singh Parihar**  
Supervisor  
Dept. of Computer Science and Engineering



DEPT. OF COMPUTER SCIENCE AND ENGINEERING  
DELHI TECHNOLOGICAL UNIVERSITY, DELHI  
MAY 22

## **DECLARATION**

I, the undersigned solemnly declare that the project dissertation titled “**AN EFFICIENT APPROACH TOWARDS IDENTIFYING AND RECOGNIZING FACES IN REAL TIME EMPLOYING DEEP LEARNING**” is my own work, carried out under the supervision of **Prof. Anil Singh Parihar**. The work submitted is original and has not been plagiarised from any source without appropriate reference. Also, this report is being submitted for the first time and never been submitted earlier for any certification/Degree purpose. In drafting the report, I adhered to the requirements supplied by the institution.

Lt Col Vivek B Sharma  
2K20/CSE/27

# **CERTIFICATE**

I certify that, the report "**AN EFFICIENT APPROACH TOWARDS IDENTIFYING AND RECOGNIZING FACES IN REAL TIME EMPLOYING DEEP LEARNING**" submitted to the Department of Computer Science and Engineering, Delhi Technological University (DTU), in partial fulfilment of the requirement for the award of the degree of Masters of Technology in Computer Science and Engineering, is work carried out by Lt Col Vivek B Sharma. All of the assistance he got from different sources has been recognised. To the best of my knowledge, no section of this paper has ever been presented for any other degree.

**Prof. ANIL SINGH PARIHAR**  
**SUPERVISOR**  
**Department of Computer Engineering**  
**Delhi Technological University**

# ABSTRACT

Face recognition can be termed as one of the most common uses of computer vision and image processing, in which a computerised system automatically recognises a person's face from a big image collection or even a live video. This thesis focuses on facial recognition, a topic that has received a lot of attention due to its usefulness in a variety of civilian and military applications. These systems are being used for numerous objectives like fraud management, security etc. and improving user experience. Face recognition algorithms have been developed in a variety of ways, with varying degrees of success. Due to the dynamic nature of the human face and the various stances it might adopt, this challenge is difficult to solve. In the present project, we propose to use YOLO which utilizes fewer samples as compared to CNN as the initial object detection method. We have used transfer learning on YOLO to use it for detecting faces. For finetuning face recognition models, we suggest a transfer learning (TL) method which combines TL techniques with CNN. Transfer learning may be used to train long-lasting, top-performance ML models that need less time and resources as compared to models learnt from the ground up. We executed transfer learning on a pretrained face recognition model to create a network capable of generating correct predictions on considerably smaller datasets. To achieve acceptable accuracy, convolutional neural networks are known to require big datasets. This project presents a solution to this problem by minimising the number of people involved, thereby reducing the number of training examples to one while maintaining near-perfect accuracy applying the transfer learning principle.

## **ACKNOWLEDGEMENT**

I extend my thanks to Prof. Anil Singh Parihar, Dept of Computer Science and Engineering, Delhi Technological University, Delhi, for his persistent supervision during the course of my work. His earnestness, diligence, and dedication are inspirational and motivating.

Lt Col Vivek B Sharma  
2K20/CSE/27

# CONTENTS

**List of tables**

**List of figures**

**Abbreviations, Symbols and Nomenclature**

## **1. Introduction**

1.1. Brief	9
1.2. Background and Motivation	10
1.3. Aim	10
1.4. Scope	11

## **2. Literature Review**

2.1. CNN	12
2.2. MT CNN	15
2.3. DNN	15
2.4. One Shot Learning	17
2.5. Transfer Learning	17
2.6. Siamese NN	17
2.7. Face Detection	18
2.8. Face Recognition	19
2.9. Open CV Haar Cascade	19
2.10 SVM	19
2.11 Gaussian Naïve Bayes	20
2.12 kNN	20
2.13 FaceNet	21
2.14 YOLO	21

3 Technology Stack	22
--------------------	----

4 Previous Work	24
-----------------	----

5 Implementation	25
------------------	----

6 Conclusion	35
--------------	----

7 References	36
--------------	----

8 List of Publications (I & II)	38
---------------------------------	----

# LIST OF FIGURES

2.1 CNN	12
2.2 MT CNN	13
2.3 Triplet Loss	14
5.1 Training the Model	15
5.2 Localizing Face	16
5.3 Encoding Data	18
5.4 Model Training Code	19

# ABBREVIATIONS, SYMBOLS AND NOMENCLATURE

CNN	Convolutional Neural Networks
DL	Deep Learning
YOLO	You Only Look Once
SSD	Single Shot Detector
PCA	Principal Component Analysis
SVM	Support Vector Machine
CPU	Central Processing Unit
GPU	Graphics Processing Unit
MTCNN	Multi-Task Cascaded Convolutional Neural Network
FPS	Frame Per Second



# CHAPTER 1: INTRODUCTION

## 1.1 Brief

The human face is a one-of-a-kind reflection of one's unique personality. Face recognition is a method of identifying a person by comparing a captured picture to previously saved images in a database. It is a technique that let computers to recognise or verify an individual from an image or video frame. It's been extensively used in various security systems to safeguard access control as a biometrics solution. Despite its lesser accuracy than other biometrics technologies such as fingerprint or iris recognition systems, face recognition has the inherent benefits of being easy to acquire and non-invasive in nature. New applications have emerged in recent years. Face recognition is in great demand since it is utilised in a variety of applications like video surveillance, human-computer interaction, and video indexing.

In Military, security is the most important factor. The topic selected is selected keeping in view, the importance of continuous surveillance by employing various software and hardware. In recent years Face Recognition has been proved to be of great utility. It can help various Military establishments to monitor their incoming and outgoing traffic. In the present project, Deep Learning techniques are being exploited for real time identification and recognition of faces.

## **1.2 Background & Problem Motivation**

Since the 1960s, face recognition has been a study topic. Theoretical interests of cognitive scientists, face recognition is a technique for verifying or identifying someone's identification by glancing at their face from the front or behind. A single photograph or a video feed Software for facial recognition is used. It's used for a multitude of things, security and healthcare as well. To maintain track of the patient's medication consumption and to give control measures. Researchers have increasingly paid enhanced time to this area. Experimenting with a huge variety of models and continually refining those that currently exist. Building a face recognizer is tough, especially if the dataset is small, as in practical scenarios. The main challenges when the dataset is limited is if someone's face may seem different from usual, yet various persons might have similar appearances. Consider creating a recognizer for unlocking mobile IDs. To construct a face recognition system, it would be unfeasible to need the user to submit lakhs of images.

## **1.3 Aim**

The motive behind this research is to use one-shot learning to construct a real-time facial recognition system.

It should be simple to use and maintain up with the system. As a result, a web service will be built as well. If a user wishes to add additional individuals to a system eg; when a corporation employs new staff, this makes it easier. Because this is an actual-time facial recognition system, it is also critical to consider the system's speed. All algorithms have the same goal: to locate an individual in a video and categorise them accordingly.

The system must be adaptable to different situations. Installing this at a company's entryway, where it should be able to distinguish workers approaching the doorway, is one scenario.

Other uses for the technology include security applications and self-driving automobiles, in which the system detects the driver and adjusts the driving parameters appropriately.

## **1.4 Scope**

This thesis is constrained in a number of ways. A face recognition system may be utilized in a number of unique ways as open to user's imagination. It can be used by any industry may it be defence, retail, pharma, academia, corporates, or any industry or field for that matter. In fact, in future, each corner of the planet would be populated with the high-end face recognition systems. If implemented thoroughly, the terrorist attacks can be completely avoided, the criminals won't be able to find hiding places. There would be no safe shelters for the anti-social elements as surveillance would be carried out at all places all the time. Many practical situations like crowd control and nabbing of the miscreants can be done. Although, the technology is still very far from the perfect system wherein the masked faces also can be identified from far distances, but even today, the system provides many practical solutions which can be implemented in almost all spheres of life.

## **CHAPTER 2: LITERATURE REVIEW**

The chapter starts with an explanation of the methodologies utilised by a short review of CNNs and their typical topologies. The chapter then moves on to discuss face detection, identification, and picture categorization.

### **2.1 Convolutional Neural Network**

Convolutional neural network or CNN is a well-known and widely used approach for image categorization (CNN). There are numerous CNNs, all of which have the same basic structure but vary in construction. All CNNs have the same fundamental structure. The convolutional layers include numerous kernels, each of which is responsible for calculating 13 feature presentations of the picture. The feature presentation is known as feature maps, which include face values. Following that, a pool layer attempts to subsequently reduce the resolution that are of the feature maps. The process flow comprises of certain steps, plus there are usually many convolutional plus pooling layers. To recap, the first convolutional layer detects edges in the image, which is referred to as low level features. Following that, the following convolutional layer is in charge of extracting more abstract characteristics. One function that may be utilised in the last layer is the Soft max function.

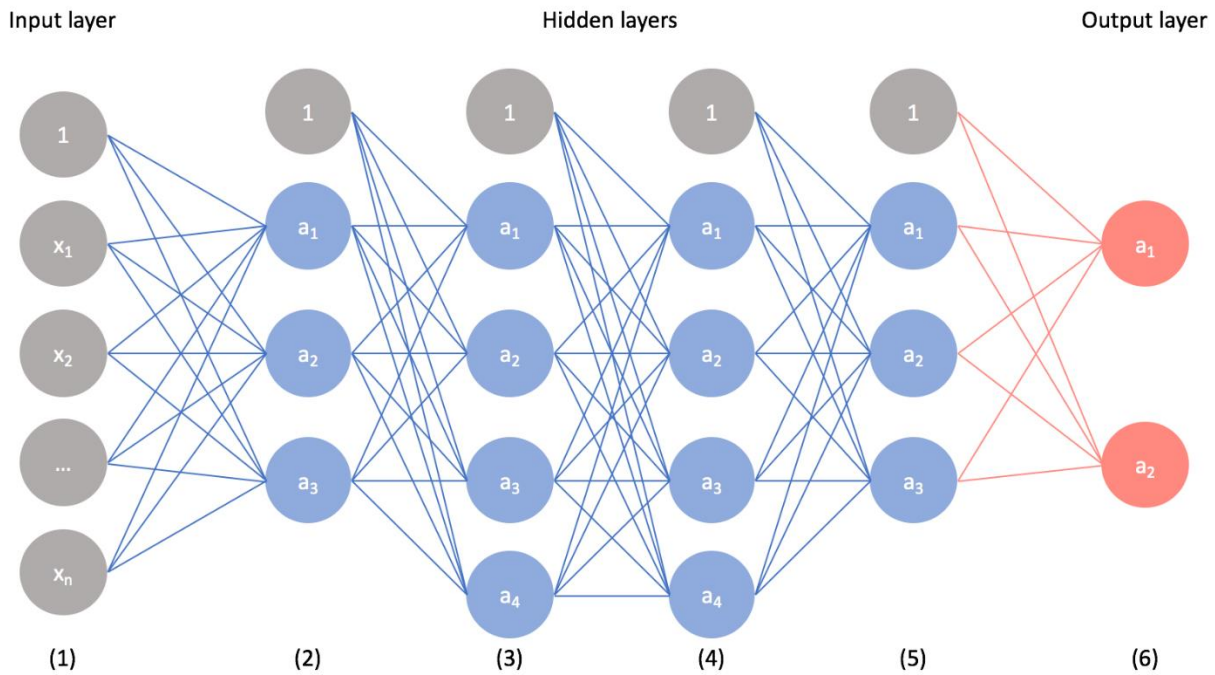


Fig 2.1: CNN architecture. (Jeremy Jordan, Convolutional Neural Networks, 26 July 2017)

### 2.1.1 Common CNN Architectures

There are several CNNs, each with its own architecture. The section that follows will provide information of typical CNN architectures.

InceptionResnetv1 can be defined as a hybrid Inception module with a convolutional neural network. InceptionResnetv1 has a comparable computational cost as Inception-v3. The computational cost specifies the amount of processing power required by the network. InceptionResnetv1 adds a residual connection to the output of the convolutional procedures.

As a result, the original convolutional is supplemented by a  $1 \times 1$  convolutional. The residual connection replaces the pooling procedures of the Inception module in InceptionResnetv1. [1]

Inception-v3 is a 42-layer deep. There is a pre-trained model available for usage which is trained on lakhs of photos. Because it has fewer parameters, this architecture is more efficient than previous Inception designs. Inception-v1 and Inception-v2 are the two prior versions. It had a low incidence of errors and performed well in the competition. ImageNet offers 15 million high-resolution photographs that have been tagged and organised into 22,000 categories. [2]

Another network is Resnet18. The size of the dataset may influence which networks are chosen. A situation where dataset is very small, say in case of a college or a class or a small company, it is desirable that very few layers are there, therefore network with subsequently fewer layers may be desirable when dataset is small. [3]

Alexnet, a convolutional neural network, is introduced in ImageNet Classification using Deep Convolutional Neural Networks. Alexnet is the very first successful neural network, as per the ImageNet dataset. Eight layers make up the network, 5 of them can be termed as convolutional and 3 of which are completely connected. [4]

Squeezenet achieves AlexNet-level correctness with 50% less attributes and a model size of 0.5MB, as shown in the article. This network has a total of 18 layers. Squeezenet is a network created specifically for small datasets. [5]

### **2.1.2 Multi Task Cascade Convolutional Neural Network**

Mtcnn, is a cutting-edge facial recognition system. At the same time, the network may provide bounding box frames, 5-point facial landmarks, with detection probability. It is a deep multi-task 15 framework that improves working by using the inherent relationship between them. The work is divided into three phases. In P-Net, the candidate window is generated using a shallow convolutional network. R-purpose Net's is to remove number of non-face windows as possible. Using a more sophisticated network, O-Net refines R-output Nets. The model, has a layered structure with three tiers of carefully constructed deep convolutional networks that categorise face with landmark locations. The network outperformed the competitors in the FDDB as well as WIDER FACE face detection benchmarks. It keeps the network running in practical time, enabling it to be utilised in a practical system. The structure is shown in Figure 2.2.

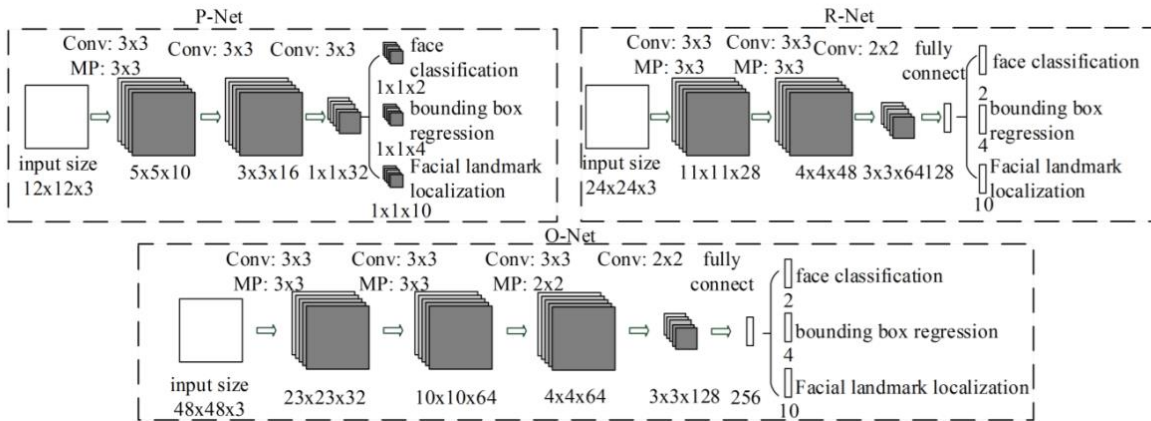


Fig 2.2: MTCNN (Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, MTCNN, 24 July 2018)

## 2.2 One-shot learning

This is an object classification problem, and machine learning algorithms often need thousands of samples to train, forming a large dataset. On the other hand, one-shot learning aims to learn knowledge from only one or a few instances. In machine learning, learning from tiny samples is still a key challenge. As a consequence, the method is complicated since each class may have a limited number of examples, implying a fixed instances of training instances and, in certain cases, just one pic for each one of them. This problem arises in a variety of real-life situations.

## 2.3 Transfer learning

Transfer learning (TL) stores knowledge from one problem & apply it to a different but related issue. Assume a system is designed to recognise human faces in a photograph. We have cases wherein we have models which we have trained or the system has been on lakhs of faces, also known as pre-trained models as we call them, may be used to address related difficulties without the requirement for large quantities of data.

Although traditional ML technology has had a great deal of success and has been utilised in a number of real-world applications, it still has certain limitations in specific situations. A large number of labelled training instances with the same frequency as the test data is the best machine

learning scenario. However, acquiring adequate training data might be excessively expensive, time-consuming, or even impossible in many circumstances. Semi-supervised learning may be able to assist solve this problem by reducing the need for great amounts of labelled data. To improve learning accuracy, semi-supervised techniques frequently use a less amount of labelled data and a big amount of unlabelled data. Unlabelled samples, on the other hand, might be difficult to come by, rendering the resultant standard models inadequate.

A possible machine learning paradigm for tackling the aforementioned difficulty is transfer learning, which emphasizes on knowledge transfer between domains. It's possible that the concept of transfer learning came from educational psychology. Learning to transfer is the result of experience generalisation, according to generalisation theory of transfer by psychologist C. H. Judd's. It is possible to transfer from one circumstance to another as long as a person extends his experience. The link between two learning activities, according to this idea, is a prerequisite for transfer.

Transfer learning, also recognised as inductive transfer, is the way of exchanging knowledge and broad feature extraction skills from a model that has now been designed to recognize features in one domain to a new model that will learn more particular characteristics appropriate to a different domain. This method to machine learning offers a number of solutions and may help decrease the entry barrier to developing good-performance models. When DL is applied to data that is very small to gain substantial features, transfer learning, with its ability to draw on previous information from models that completed lengthy time of training on greatly large datasets, may deliver improved performance.

To put it differently, a model created from scratch may extend more correctly with less training examples than a model built from scratch. This gives up a number of ways for promptly fine-tuning models to a particular dataset and training with alternative training parameters that would be impossible to change on a model that is being trained from nick.



## **2.4 Siamese Neural Network**

It is a kind of ANN that uses the ditto weights to calculate comparable output vectors while working in parallel on two unique input vectors. Two input photos are communicated to neural networks as part of the network's architecture. After that, the model creates different feature maps, or vectors, containing the face representations. After that, a distance function is used to generate these vectors in order to identify the resemblances between the two feature maps. Why exactly do we train the system, basically, the purpose of training is to decrease the distance function for similar classes while growing the distance function for uncorrelated classes. [6]

## **2.5 Face Detection**

A technique for recognising and searching a face in an image or video is called as face detection. There are various nodal points on anyone's face, somewhere around 80 plus, like distance between eyes, jaw line width, forehead size etc. Now, these nodal points are different for each individual. The algorithm calculates differences among these factors and compare them with the images stored in the system and generates an output based on the comparison and calculation. There are a few options for accomplishing the tasks. It is possible to build a convolutional neural network (CNN) from scratch, which would need a large amount of data, or to use a pre-trained model that has already been trained on lakhs of faces. When the dataset is small or the issue to be addressed is connected to the problem of the pretrained model, the model which has already been made to learn by feeding it with various labelled and unlabelled data, the latter technique is helpful. All algorithms have different sizes. As a consequence, certain algorithms work better on websites and mobile phones.

Due to variable postures, illuminations, and occlusions, face detection in unconstrained situations may be problematic. Photographs with human faces are easily recognised by humans. This is, however, a challenging task for computers. Deep learning, on the other hand, has been shown through study to provide good results in this discipline.

## 2.6 Face Recognition

A face recognition system can identify a person based on an image or video frame by carrying out calculations by virtue of its training by the supervised & unsupervised learning with images. Face recognition systems use many approaches, but normally in usual sense, they function by comparing photos from a provided image inside a database. The model recognises all photos in the database given. Euclidean distance is a standard function that may be used to categorise a given picture from a database of images. A facial recognition system may be utilised in a variety of situations. They are often employed in security applications, for example. They may also be utilised in self-driving automobiles to forecast the person and modify all car settings for them.

## 2.7 FaceNet

FaceNet offers an unvarying embedding for applications like as face recognition, confirmation, and clustering. Another word for embedding is feature map, which refers to a vector representation of values pertaining to the face. It maps each face picture into a Euclidean space, where distance correlates to facial similarities. More precisely, a picture of person X will be put near to other photographs of person X, but another person that can be named as Y will be positioned far away from person X.

Now, how Facenet carry out its learning of the mapping from the photos and builds embeddings. To conclude, the embeddings obtained may be utilised straight away for face recognition, authentication, and clustering using, for example, Support Vector Machine (SVM) or K-Nearest Neighbors (K-NN). Facenet learns using the Triplet loss function.

## 2.8 Triplet Loss

When an anchor and a positive in a given dataset reflect that same person, triplet loss reduces the distance between them. As a consequence, the other distance i.e. between the anchor and a negative that displays a unique individuality is optimised. As a result, a photograph of an anchor (person A) must be closer to positive pictures (all pictures of person A) and farther away from negative images.



Fig 2.3: Triplet Loss

## 2.9 OpenCV Haar Cascades

Haar Cascade is a classifier that is not a convolutional neural network but is linked to convolutional neural networks. A Haar Feature is comparable to a CNN kernel. Thus, in a CNN, the kernel values are confirmed by training, while the Haar-Feature is calculated manually.

## 2.10 OpenCV DNN

This is an Opencv module, as stated in Section 3.5. It is possible to employ a pre-trained Tensorflow model, which is also detailed in section 3.5. This is, nevertheless, a deep neural network that may be utilised for inference using a pre-trained model. Caffe, Tensorflow, Darknet, and PyTorch are among the frameworks supported by OpenCV DNN. This module may be used to construct a variety of applications, including face detection and object identification. [7]

## 2.11 Support Vector Machine

Support SVM, or Vector Machine, is a learning model which has been supervised and is data-driven. This model may be used for both classification as well as regression. SVM employs a linear dividing hyperplane to divide data into two groups. Two classes are shown in the image above, with grey dots indicating one and blue squares indicating the other. A hyperplane which is there in the middle separates these two groups. The optimum hyperplane for this separation must be chosen carefully. Kernels are often employed in SVM, and several kernels may be utilised depending on the data and application situation. A kernel approach in machine learning is referred to as a kernel in general. It's a linear classifier-based technique for tackling non-linear problems [8].

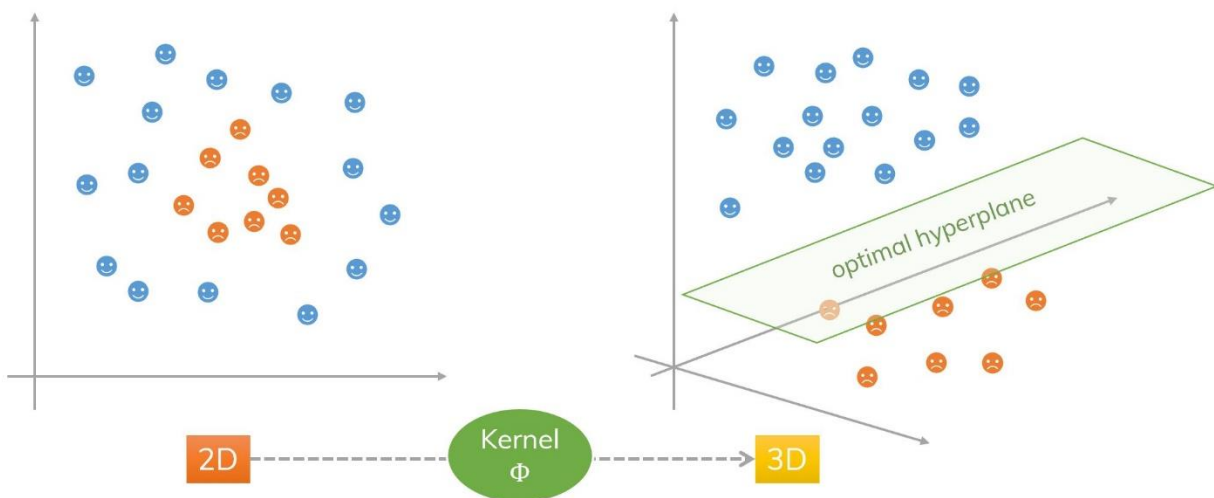


Fig 2.4: SVM

## 2.12 Gaussian Naïve Bayes

The Gaussian Naive Bayes (GNB) approach builds a linear classifier using supervised learning. The GNB i.e. Gaussian Naive Bayes is parted into three parts. First, it assumes the data has a normal distribution when dealing with real-time data with continuous distribution. Next, Multinomial Naive Bayes may be employed in multinomial distributions with frequencies as the features. Finally, if the features are autonomous or Boolean, they were formed by a Bernoulli process, thus a BNB classifier might be utilised. [9]

## 2.13 K-Nearest Neighbours

K-Nearest Neighbours is a supervised learning approach that may be used to carry out the solution of regression and classification issues. A classification difficulty results in class membership. As a result, the categorization of an item is decided by the votes of its closest neighbours. For example, if  $k = 1$ , the object is then provided to the class that is adjacent to the neighbour. If there is a regression issue, the resulting output is the object's property value. The worth of an item is the mean of its  $k$  closest neighbours' values. [10]

## 2.14 YOLO

R-CNN is the first to introduce deep learning-based object identification methods. However, the detecting technique for a single picture takes around 40 seconds. SPP-Net suggested spatial pyramid pooling to improve detection speed and alleviate the issue of fixed input picture size (SPP). The detection speed is substantially quicker than R-CNN due to this structure. Fast R-CNN demonstrates ROI pooling, which is quicker in training and detection than R-CNN. Furthermore, the approach uses Softmax as a classifier rather than SVM [11]. Faster RCNN offered region proposal network (RPN) and delegated region proposal selection to RPN. R-FCN provides the principles of position-sensitive score maps, employs deeper shared network layers, and then significantly increases detection speed.

YOLO is the first CNN-based one-stage detector. The method separates the input picture into grid cells and then predicts the coordinates and classification for each cell directly. Although the detection accuracy is lower than that of two-stage detectors, the speed is many times quicker. The updated versions of YOLO demonstrated excellent performance on PASCAL VOC [13] and remained quick, allowing the approach to satisfy the demand for real-time detection in practise.

On the COCO dataset, YOLOv3 used a novel network backbone dubbed darknet-53 and obtained promising results. We utilised the YOLOv3 architecture as our basic network structure and enhanced in numerous ways; I evaluated the approach on the WIDER FACE data set and the Fddb dataset and obtained noticeably higher performance. The suggested approach may be employed in real-time face detection applications at various sizes.

# CHAPTER 3: TECHNOLOGY STACK

## 3.1 The Text Editor

As an IDE, Visual Studio Code content management was employed. This Integrated Development Environment (IDE) is compatible with a wide range of programming languages. It comes with a connected code editorial management, compiler, CLI (command line interface), and breakpoint debugger that can debug both machine and source code. It has been flattened down for the goal of creating and investigating modern online and cloud App development.

## 3.2 Language of Programming

We use Python 2.7 and Python 3.6 as programming languages. Python allows for fast system integration and a basic working environment that even inexperienced users can learn and use quickly and effectively. Because it is an interpretative, general-purpose, and may be used to develop networks, servers, and a number of other applications. It is based on object-oriented programming. It also has a straightforward syntax that allows developers to easily write accurate, logical code for both large and small projects.

## 3.3 Google Collaboration

Google Colab, similar to Jupyter Notebooks, is a free cloud administration for artificial intelligence and research. It provides an enhanced platform for running fully configured deep learning applications, as well as additional access to a powerful GPU. We may utilise Google Collaboratory to build a wide range of data-intensive apps utilising the low-cost Tesla K80 GPU with Keras, Machine Learning, and TensorFlow. Google Colab benefits the project.

GPU support is completely free.

1. It enables remote users and developers to exchange Jupyter Notebooks, other files, and Google Docs.
2. Python's most crucial libraries are already installed.
3. It is designed and built using the Jupyter Notebook platform.

4. It allows for the free training of deep learning models.

The operating systems used in our experiments were Windows 11 and Ubuntu 18.04 LTS.

## CHAPTER 4: PREVIOUS WORKS

Lei Zhang & Yandong Guo describe the challenge of one-shot learning in the context of facial recognition in [14]. Their goal is to develop a vast-scale face acknowledgement system which can recognise several identities. In order to overcome the issue of imbalanced data, this strategy sets the norms of given weight vectors. Because it promotes unrepresented classes in the learn model, the new loss term (UP) is efficient. This increases system's performance.

They employed a dataset consisting of more than 20000 classes to train a classifier during the creation of the one-shot learning phase.

The method is a 34-layer residual network-based multinomial logistic regression. This analysis found that 99 percent of underrepresented courses and 99.8 percent of typical classes were represented. [14]

Zhao et al. present the results of yet another face recognition study. In this paper, they present a face recognition system that uses PCA and LDA. The strategy is split into two parts. They begin by projecting the source vector of the face photo to a face subspace using PCA. After that, they utilise LDA 21, which is a linear classifier. The convergence of these tactics improves a model's capacity to categorise classes after it has been trained on a small dataset.

A face recognition system that is based on a CNN and principal component analysis is described by Edy et al (CNN-PCA). Edy and colleagues talk about their face recognition system, which uses a mix feature extraction technique (CNN-PCA). The solution is to integrate these two approaches to build a more efficient feature extractor technique that will provide a more precise estimation. Their system's objective is to develop a system that recognizes human faces in real time while also being reliable and strong. This method, according to the conclusions of their analysis, provides precise data processing and great accuracy. They show how incorporating a CNN into PCA enhances accuracy. When they use only PCA, they have 98 percent reliability in 50 items rather than 96 percent.



# CHAPTER 5: IMPLEMENTATION

## 5.1 Libraries

Various libraries were used to construct the ML algorithms throughout the project. Python libraries, which are usually used in machine learning, are utilised for the bulk of the algorithms. Machine learning libraries, on the other hand, aren't the only ones used. Split-folders, for example, is a package that divides data into percentages. In the next part, we'll talk about libraries.

Tensorflow is a library which is Python-based and open-source. It's easy and very well-written, and it lets programmers create vast neural networks with several layers. It may be used with either a CPU or a GPU. This library is being used for the purpose of loading the inception models.

Numpy is a Python module that provides support for many dimensional arrays and matrices of enormous size. Array operations are performed by this library.

Matplotlib is a Python tool that lets you create fixed, dynamic, and interactive graphs. It's a fantastic tool.

The precision of machine learning techniques is assessed using this package.

PyTorch is a DL library akin to TensorFlow. This library is compatible with both the CPU and GPU. In this project, the library is used to create the dataset, which includes shaping and resizing the images.

OpenCV is a zero cost & open domain available library. It's a large library that includes over thousands of efficient machine learning algorithms. For example, C++ and Python are supported.

This library was heavily utilized in this project for image reading, resizing, RGB conversion, image storage, and enabling the face recognition system perform in practical time limits.

SKlearn is again an open-source machine learning approach that incorporates multi-dimensional arrays among its characteristics. For high-level mathematical functions, this library also supports array support. Algorithms for machine learning are also supported.

Albumentations is a free Python module that lets you to expand your dataset. This library was utilised in this study to increase the amount of data samples in the dataset.

Pytube is a Python library that allows you to download online videos without having to install any other software which is being greatly utilized by many developers to download online video. The face recognition algorithm categorises the persons in the video feed using this video stream.

## 5.2 Experimentation

There was a lot of trial and error so without getting into the individual variations of each hyperparameter the values of important features which give the best accuracy are:

- 32 kernels in the first 2 convolutional layers and 64 in the following 2.
- 4 pooling layers with a pool size of (2,2) which gives approximately 590,000 trainable parameters. This works well with the amount of data trained.
- ReLU activation function does the calculation using the function  $f(x)=\max(0,x)$ . This makes the solution non-linear
- Learning rate is set at 0.0001
- Softmax function is used in the output layer as it gives us a probability array of each class which is what we require in our classification problem.
- Adam optimizer is used as it yielded better results than SGD.
- High dropout is used in dense layer to prevent overfitting.

## 5.3 Training Data

Objective is to minimise the amount of data required while maintaining maximum accuracy.

20 images per known person and 500 images for unknown persons are used for optimum speed and accuracy. This works out to about 1:1 ratio for each class after data augmentation.

Type of Data Needed

- For known person they must be the only person in the image and their face must be clearly visible.

- For unknown persons we need diverse data so any images with different people clearly visible.

```
... train : 1/140 0.67% done
    eval : 3/60 2.62% done
    eval : 4/60 4.64% done
    train : 2/140 1.39% done
    eval : 5/60 5.92% done
    eval : 6/60 6.81% done
    train : 3/140 2.03% done
    eval : 7/60 7.79% done
    train : 2/140 2.57% done
    eval : 8/60 10.79% done
    train : 3/140 3.23% done
    eval : 9/60 11.50% done
    eval : 10/60 13.41% done
    train : 4/140 4.16% done
    train : 5/140 4.56% done
    eval : 11/60 15.00% done
    eval : 12/60 16.72% done
```

Fig 5.1 Training of the System on Google Colab

## 5.4 Localization

Pattern recognition problems rely on the features inherent in the pattern of images. So, to obtain accurate results we need to localize the faces in the image so the model can learn the features of the face correctly.

## 5.5 Data Pre-processing

We need to artificially extend our dataset since our goal is to reduce the quantity of training data necessary and we are using just 20 photos per class, which is extremely little data to train our model. To supplement our data, we utilise Keras' Image Data Generator class. The changes we make must not jeopardise the class's integrity. In our example, a vertical flip destroys the integrity of the face, but a horizontal flip does not and gives us with important new information. Consider the test data that the model may face and attempt to simulate it through augmentation.

### 5.5.1 Augmentation

Because our objective is to minimise the amount of training data required and we are using only 20 images per class which is very little data to train our model, we need to artificially enlarge our dataset. We use Keras' ImageDataGenerator class to augment our data. The transformations we make must not ruin the integrity of the class. For instance, in our case, a vertical flip ruins the integrity of the face but a horizontal flip does not and provides us with useful new information. It is good to think about the test data the model may encounter and try to mimic that using augmentation

### 5.5.2 Encoding

We need to label the pictures that have been trained, and we utilise a one-hot encoding for this classification task. The phrase "one hot" derives from digital circuits, where one-hot refers to a set of bits where the only permissible values are those that have a single high (1) bit and all others are low (0). A one-hot encoding, on the other hand, is a binary vector representation of category data. For example:

```

def autocrop(directory, face_cascade):
    for people in os.listdir(directory):
        for images in os.listdir(directory+"/"+people):
            img = cv2.imread(directory+"/"+people+"/"+ images)
            height = img.shape[0]
            width = img.shape[1]
            size = height * width

            # For very large images
            if size > (500^2):
                r = 500.0 / img.shape[1]
                dim = (500, int(img.shape[0] * r))
                img = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)

            # Don't need RGB values for face detection so changed colour scheme
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            # Parameters: image; scaleFactor - scaling image size so 1.3 reduces image by 30%
            #minNeighbours - higher value means less detections but more quality
            faces = face_cascade.detectMultiScale(gray, 1.3, 5)

            for (x,y,w,h) in faces:
                imgCrop = img[y:y+h,x:x+w]
                # save cropped images in our database
                cv2.imwrite("training_dataset_cropped"+people+"/"+str(images),imgCrop)
                print(str(images)+" has been processed and cropped")
            print("All images have been processed")

```

## 5.2 Fig Function that localizes the face for training data

```

def load_data():
    data = []
    list_names = []
    IMG_SIZE = 106
    # parameters for the data generator that will augment our data
    gen = ImageDataGenerator(rotation_range=10, shear_range=2, channel_shift_range=2,
horizontal_flip=True, brightness_range=(0.001,2),width_shift_range=10,
height_shift_range=10)
    num_classes = 7
    with tf.device('/gpu:1'):
        for people in os.listdir("training_dataset_cropped"):
            for images in os.listdir("training_dataset_cropped/" + people):
                img = np.expand_dims(cv2.imread("training_dataset_cropped/" + people + "/" +
images),0)
                aug_iter = gen.flow(img)
                if people == "UNKNOWN":
                    # 2 augmentations for each unknown face
                    aug_images = [next(aug_iter)[0].astype(np.uint8) for i in range(2)]
                else:
                    # 25 augmentations for each known face
                    aug_images = [next(aug_iter)[0].astype(np.uint8) for i in range(25)]
                for images in aug_images:
                    img = cv2.resize(images, (IMG_SIZE, IMG_SIZE))
                    # creating label data for images(from 0 - no. of classes-1)
                    if people == "Himanshu":
                        list_names.append(0)
                    elif people == "Vivek Sharma":
                        list_names.append(1)
                    elif people == "Aditya":
                        list_names.append(2)
                    elif people == "Mehul":
                        list_names.append(3)
                    elif people == "Ankita":
                        list_names.append(4)
                    elif people == "Ritwik":
                        list_names.append(5)
                    elif people == "UNKNOWN":
                        list_names.append(6)
                    else:
                        print("Invalid")
                data.append(np.array(img))
    # takes our label data and converts into one-hot vector
    encoding = to_categorical(list_names)
    data = list(zip(data, encoding))
    shuffle(data)
    np.save("train_data.npy", data)

```

Fig 5.3: Function that encodes the data

```

def train_model(x,y):
    with tf.device('/gpu:1'):
        model = Sequential()
        model.add(Conv2D(32, kernel_size=(3, 3),
            activation='relu',
            input_shape=(IMG_SIZE, IMG_SIZE, 3)))
        model.add(MaxPool2D(pool_size=(2,2)))
        model.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
        model.add(MaxPool2D(pool_size=(2,2)))
        model.add(Dropout(0.1))
        model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
        model.add(MaxPool2D(pool_size=(2,2)))
        model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
        model.add(MaxPool2D(pool_size=(2,2)))
        model.add(Dropout(0.1))
        model.add(Flatten())
        model.add(Dense(512, activation='relu'))
        model.add(Dropout(0.5))
        model.add(Dense(num_classes, activation='softmax'))
        print(model.summary())
        opt = optimizers.Adam(lr=0.0001)
        model.compile(loss=keras.losses.categorical_crossentropy,
            optimizer=opt,
            metrics=['accuracy'])
        model.fit(x, y,
            batch_size=128,
            epochs=40,
            validation_split = 0.2)
    return model

```

Fig 5.4: Training the Model Code

### **5.5.3 Data Post Processing**

#### **Face detection**

We must first detect faces and localize them for our model to make predictions. So we through frame by frame and use haar cascades again to obtain cropped face images.

#### **Prediction**

We then make predictions on each detected face and the class with the highest probability is saved as the result.

#### **Display**

The model is not perfect and is bound to make mistakes, so to ensure accuracy we only confirm a result, i.e., a confirmed face if it is recognized for 10 consecutive frames.



```

def predict_faces(faces, model, img):
    for (x,y,w,h) in faces:
        imgCrop = img[y:y+h,x:x+w]
        image = np.expand_dims(cv2.resize(imgCrop, (IMG_SIZE, IMG_SIZE)),0)
        # obtain index of max probability
        pred = np.argmax(model.predict_on_batch(image))
        if pred == 0:
            result = "Himanshu"
            color = (255,255,255)
        elif pred == 1:
            result = ("Vivek Sharma")
            color = (0,0,0)
        elif pred == 2:
            result = ("Aditya")
            color = (0,0,0)
        elif pred == 3:
            result = ("Mehul")
            color = (50,100,150)
        elif pred == 4:
            result = ("Ankita")
            color = (0,0,0)
        elif pred == 5:
            result = ("Ritwik")
            color = (0,255,255)
        elif pred == 6:
            result = ("Unknown")
            color = (255,0,0)
        else:
            break
        # update counter of specific class

        thing[result]+=1
        # display result if it is in more than 10 consecutive frames
        if thing[result]>10:
            cv2.putText(img,result,(x+w+10,y+h),0,0.8,color)
            cv2.rectangle(img,(x,y),(x+w,y+h),color,2)
            cv2.imshow('Window', img)
    return thing, prev_thing

```

Fig 5.5: Function that makes predictions on detected faces

```

def video(model):
    result = ""
    # counters for each specific class
    thing = {"Himanshu":0,"Vivek
Sharma":0,"Aditya":0,"Mehul":0,"Ankita":0,"Ritwik":0,"Unknown":0}
    prev_thing = {"Himanshu":0,"Vivek
Sharma":0,"Aditya":0,"Mehul":0,"Ankita":0,"Ritwik":0,"Unknown":0}
    cam = cv2.VideoCapture(0)
    with tf.device('/gpu:1'):
        while cv2.waitKey(1) != 27:
            ret_val, img = cam.read()
            height = img.shape[0]
            width = img.shape[1]
            size = height * width

            if size > (500^2):
                r = 500.0 / img.shape[1]
                dim = (500, int(img.shape[0] * r))
                img = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)

            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            faces = face_cascade.detectMultiScale(
                gray,
                scaleFactor=1.3,
                minNeighbors=5,
            )

            if type(faces) == np.ndarray:
                predict_faces(faces,model,img)
            elif type(faces) == tuple:
                cv2.imshow('Window', img)
                if type(faces) == np.ndarray:
                    predict_faces(faces,model,img)
            # resetting counters if face is absent
            for key in thing:
                if thing[key] == prev_thing[key]:
                    thing[key] = 0

```

5.6 Fig: Function that captures and displays video after post-processing

## **CONCLUSION**

In terms of accuracy, our findings reveal that CNNs have a distinct edge over the traditional approaches we evaluated. Furthermore, CNNs often have the advantage of being simpler to employ in more realistic settings. They are especially useful for studying photos in more natural situations, such as outdoor shots. This enables improved processing of pictures from various data sources, as well as the integration of data augmentation and other approaches that may improve the performance of training the network by avoiding it from receiving identical input photos during training.

## REFERENCES

- [1] Guo, Y., & Zhang, L. (2017). One-shot face recognition by promoting underrepresented classes. arXiv preprint arXiv:1707.05574.
- [2] Sharma, R., Kumar, D., Puranik, V., & Gautham, K. (2019, April). Performance Analysis of Human Face Recognition Techniques In 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU) (pp. 1-4). IEEE
- [3] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp.4700-4708).
- [4] Guo, Y., & Zhang, L. (2017). One-shot face recognition by promoting underrepresented classes. arXiv preprint arXiv:1707.05574. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016).
- [5] Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Processing Letters, 23(10), 1499-1503.
- [6] Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2016). Inceptionv4, inception-resnet and the impact of residual connections on learning. arXiv preprint arXiv:1602.07261.
- [7] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., ... & Chen, T. (2018). Recent advances in convolutional neural networks. Pattern Recognition, 77, 354-377.
- [8] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
- [9] Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size. arXiv preprint arXiv:1602.07360.

- [10] Md. Asif Anjum Akash, M A. H. Akhand, N. Siddique, “Robust Face Detection Using Hybrid Skin Color Matching under Different Illumination”, International Conference on Electrical, Computer and Communication Engineering (ECCE), 2019.
- [11] RuWang, XinShi He, “Face Detection on Template Matching and Neural Network”, International Conference on Communications, Information System and Computer Engineering (CISCE) , 2019.
- [12] Chien-Yu Chen, Jian-Jiun Ding, Hung-Wei Hsu, Yih- Cherng Lee, “Advanced Orientation Robust Face Detection Algorithm Using Prominent Features and Machine Learning Techniques”, IEEE Visual Communications and Image Processing (VCIP) , 2018.
- [13] Rong Qi, Rui-Sheng Jia, Qi-Chao, Hong-Mei Sun, Ling- Qun Zuo, “Face Detection Method Based on Cascaded Convolutional Networks”, Digital Object Identifier 10.1109/ACCESS.2019.2934563, IEEE Access, 2019.
- [14] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 815-823).

## LIST OF PUBLICATIONS

[1] Vivek Sharma, “Analysis of Contemporary Face Recognition Techniques”. Accepted at the International Conference on Advances in Computing, Communication Control and Networking (ICAC3N–22) Accepted in IEEE Explore

INDEXED BY – Scopus, IEEE and Google Scholar.

Paper Id:-878

**Abstract-** Face recognition can be termed as one of the most common uses of computer vision and image processing, in which a computerised system automatically recognises a person’s face from a big image collection or even a live video. This thesis focuses on facial recognition, a topic that has received a lot of attention due to its usefulness in a variety of civilian and military applications. These systems are being used for numerous objectives like fraud management, security etc. and improving user experience. Face recognition algorithms have been developed in a variety of ways, with varying degrees of success. Due to the dynamic nature of the human face and the various stances it might adopt, this challenge is difficult to solve. In the present project, we propose to use YOLO which utilizes fewer samples as compared to CNN as the initial object detection method. We have used transfer learning on YOLO to use it for detecting faces. For finetuning face recognition models, we suggest a transfer learning (TL) method which combines TL techniques with CNN. Transfer learning may be used to train long-lasting, top-performance ML models that need less time and resources as compared to models learnt from the ground up. We executed transfer learning on a pretrained face recognition model to create a network capable of generating correct predictions on considerably smaller datasets. To achieve acceptable accuracy, convolutional neural networks are known to require big datasets. This project presents a solution to this problem by minimizing the number of people involved, thereby reducing the number of training examples to one while maintaining near-perfect accuracy applying the transfer learning principle.

## LIST OF PUBLICATIONS

[2] Vivek Sharma, “An Efficient Approach Towards Identifying and Recognizing Faces in Real Time Employing Deep Learning”.

- Accepted at the International Conference on Advances in Computing, Communication Control and Networking (ICAC3N-2022)
  - Indexed by – Scopus, IEEE and Google Scholar.
  - Paper Id:-879
  
- Accepted at the International Conference on IoT Based Control Networks and Intelligent Systems (ICICNIS - 2022)
  - Indexed by – Scopus
  - Paper Id:-ICICNIS070

**Abstract-** The humanistic face is a distinct characteristic of the human body. This uniqueness varies from person to person since each human face is distinct in several ways and hence is a unique biometric of an individual person. Face detection is a subfield of artificial intelligence that detects faces in images, video recordings, and so on. Detecting human faces is important in a variety of applications such as facial animation, face recognition, human image database management and human computer interface, face verification and validation, and as a pre-processing step for many high-end and complex computer vision tasks for which a number of state-of-the-art detection methods have been developed. However, the complexity of the human face and the changes caused by various influences like as lighting, stance, and angle of the photo shot will also affect the identification accuracy of the algorithms, which is where advances in deep learning come in handy. With the progress of deep learning, we are now able to train artificial neural networks for face identification and recognition, which in some cases outperform convectional techniques in terms of accuracy and performance. The primary goal of this study is to analyze contemporary face recognition algorithms.



**Microsoft CMT** <email@msr-cmt.org>  
to Vivek ▾

Sat, Jun 4, 8:43 AM (6 days ago)



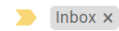
Dear Vivek Sharma,  
DTU

Greetings from ICAC3N-22 ...!!!

Congratulations....!!!!

On behalf of the 4th ICAC3N-22 Program Committee, we are delighted to inform you that the submission of "Paper ID- 878 " titled " Analysis of Contemporary Face Recognition Techniques " has been accepted for presentation at the ICAC3N- 22 and will be sent for the submission in the conference proceedings to be published by the IEEE.

## Notification 4th IEEE ICAC3N-22 & Registration: Paper ID 879



**Microsoft CMT** <email@msr-cmt.org>  
to Vivek ▾

Sat, Jun 4, 8:43 AM (6 days ago)



Dear Vivek Sharma,  
DTU

Greetings from ICAC3N-22 ...!!!

Congratulations....!!!!

On behalf of the 4th ICAC3N-22 Program Committee, we are delighted to inform you that the submission of "Paper ID- 879 " titled " An Efficient Approach Towards Identifying and Recognizing Faces in Real Time Employing Deep Learning " has been accepted for presentation at the ICAC3N- 22 and will be sent for the submission in the conference proceedings to be published by the IEEE.



*Acceptance Letter*

To

**Vivek Sharma**  
Delhi Technological University, Delhi ,India.

**Paper ID: ICICNIS070**

**Dear Authors,**

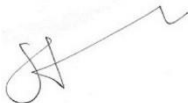
Hearty Congratulations!

This invitation letter is to confirm that your peer-reviewed & referred full paper entitled **“An Efficient Approach Towards Identifying and Recognizing Faces in Real Time Employing Deep Learning”** is **accepted** for oral presentation at the ICICNIS 2022: International Conference on IoT Based Control Networks and Intelligent Systems to be held in Bengaluru, India during 01-02, July 2022.

In this regard, we appreciate if you could send the final paper, publishing agreement form and other necessary documents to the conference at the earliest, to ensure a timely publication of your research paper. When submitting your final paper, please highlight the changes made to the research paper according to the specified reviewer comments.

All registered and presented papers will be recommended for inclusion in Springer - Lecture Notes in Networks and Systems

Yours sincerely,



Conference Chair  
ICICNIS – 2022