Face Recognition Using Convolutional Neural Networks

Anshid Babu T M[1], Muhammed Haneesh K P[2]
[1]*Assistant Professor Department of Computer Applications, SAFI Institute of Advanced Study (Autonomous), Malappuram, 673633, India*
[2]*Assistant professor Department of Computer Science and Artificial Intelligence, SAFI Institute of Advanced Study (Autonomous), Malappuram, 673633, India.*

***Abstract*—Reliable and precise facial identification is highly valuable in numerous law enforcement scenarios, including locating missing persons, tracking fugitives, monitoring suspicious individuals, and identifying unknown deceased individuals. To address these needs efficiently, we propose Original Face (Face Recognition) , a system that incorporates advanced algorithms like CNN for face detection and recognition. This system can be integrated with existing law enforcement databases, allowing it to utilize extensive data repositories for improved accuracy and performance. As more information is added, the system becomes increasingly robust and effective, providing faster and more reliable support for various investigative processes.**

***Keywords*—CNN (Convolutional neural network), Face Recognition.**

I. INTRODUCTION

With the explosion of deep learning in recent years, the field of computer vision has seen rapid development and progress. One of the longest standing problems of computer vision remains accurate and reliable facial recognition. Nowadays, deep learning methods are being employed to achieve state-of-the- art results in terms of accuracy and reliability.

We propose a pipeline for facial recognition which we believe will be of benefit to the police and other law enforcement agencies. Currently our solution is deployed as a web service and is a part of the Crime Drive application. The goal of this project is to provide a system that will allow the police the ability to easily identify criminals, missing cases, dead bodies, etc. Our system is rotation and scale invariant and can detect partially occluded faces. It consists of three main modules, the detector, encoder and the classifier.

1.1. Problem Statement:

Our goal is the help improve the efficiency of law enforcement by enabling fast and accurate facial detection and recognition. Face detection would be a useful technology as there are many cases where it would prove helpful, such as passport verification, verification of missing and/or suspicious individuals, dead body identification and so on.

1.2. Design and Implementation Constraints:

Implementing a robust face recognition system involves several practical limitations and technical demands. The core of this application relies on Convolutional Neural Networks (CNNs), which are well-suited for extracting detailed features from facial images. However, running CNN-based models requires substantial computational resources to process image data efficiently and to deliver prompt results, especially when operating on video feeds or large image sets. To maintain high accuracy, the system must handle variations in lighting, facial orientation, and partial occlusion, which can affect recognition performance. Furthermore, managing and storing facial data and feature representations securely calls for a reliable and scalable database, along with strict security measures to safeguard sensitive biometric information.

High-performance hardware such as GPUs or specialized AI processors is essential to support the computational load of deep learning models. In addition, a user-friendly interface is necessary to enable easy interaction with the system, allowing users to perform searches, view results, and manage records seamlessly. These constraints must be carefully considered to ensure that the face recognition system remains fast, accurate, and secure under real-world conditions. Nvidia GPU that has a compute score of 3.7 or higher and at-least a high-end quad-core intel/AMD CPU.

II. LITERATURE REVIEW

2.1 History of Face Detection

Face recognition and detection traces back to 1977 with the research of Takeo Kanade and the introduction of the Lucas-Kanade-Tomasi method for face detection. In 1987, Sirovich and Kirby developed a new method called Eigenfaces, which quickly became the most reliable way to identify faces form an image. In 1996, Fisher face was developed using Linear Discriminant Analysis (LDA) for dimensionality reduction. It could identify faces from variable lighting conditions, which was difficult for Eigenfaces [6]. In 2001, Viola and Jones developed a method, known as the Viola-Jones Harr Cascade algorithm, which used the AdaBoost ensemble learning method to learn the features of a face from a series of weak classifiers which detect harr-like features on faces [4].

Viola-Jones was a breakthrough as it was not only more accurate at detecting faces than other methods but it was also significantly faster. Although fast, Viola-Jones suffers when faces are in unfavorable lighting conditions or partially occluded [4]. In 2007, a new method was proposed called Histogram of Oriented Gradients. The idea was based on the assumption that objects, given most lighting conditions, have relatively the same areas where they are dark or light. Using this information one can calculate a gradient for each pixel based upon the horizontal and vertical directions in which pixels go from dark to light. Using a histogram of all such gradient values we can form a general pattern for matching most faces. Combining this with a linear SVM allows us to rapidly detected faces in most condition [5]. This method is fast and works well on faces from various different lighting conditions.

2.2 Residual Networks

Residual Networks were first proposed by He, Kaiming, et al. at Microsoft Research as a method for allowing the training of very deep neural networks. ResNets utilize the use of skip, or 'highway', nets to overcome the problem exploding and vanishing gradients [1]. ResNets have become the de facto structure for problems where very deep networks are beneficial, such as the field of image analysis, object detection and facial recognition.

2.3 Convolutional Neural Networks

The idea of Convolutional Neural Networks was first proposed by Yann LeCun [8]. These initial networks were called LeNets. The paper outlines various gradient-based approaches for learning and classifying 2D data such as images. It highlights that traditional learning methods, such as backpropagation based dense networks are inefficient at classifying and dealing with image data due to the inherent spatial complexity of images in and the inter-relationship between adjacent pixels [8].

III. METHODOLOGY

3.1 Pipeline Architecture

The True-Face face detection pipeline consists of 3 models, two classifiers and one encoder. In both training and testing, images are fed to the Detector, which identifies faces and returns their locations. These locations and their respective images are passed to the ResNet based CNN encoder, which outputs a 128-d array of encodings related to various parts of the faces. These encodings are then passed to a KNN classifier which outputs an array of closest neighbors based on Euclidean distance. The results that cross a fixed threshold boundary are assumed to be the correct classification.

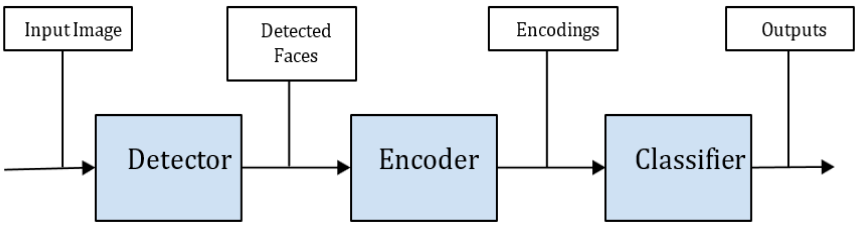

Figure 1: Pipeline Architecture

3.1.1 Detector

True-Face makes use of the Dlib C++ library for fast machine learning and deep learning problem solving. Dlib comes equipped with various unique methods for face detection. In our solution we utilize Histogram of Oriented Gradients (HOG) as well as a Convolutional Neural Network trained using the Max Margin Object Detection (MMOG) method [2].Computationally, HOG is much faster than the CNN and is capable of detecting faces in real time on

60fps video streams on CPU. However, it struggles to detect faces that are not straight-facing, occluded or in poor lighting conditions. In other words, HOG trades accuracy for speed and is a good method for lower end machines. However, when it comes to detecting faces in real world scenarios, more robust solutions are required.

Dlib's CNN solution is far more capable at detecting faces that are not straight- facing. In fact, it is capable of detecting faces irrespective of scale, orientation or partial occlusion with a high degree of accuracy. However, the high accuracy of the model comes at the cost of extremely long processing time on CPU cores. For this reason, a CUDA compatible GPU from Nvidia is necessary for fast execution of the CNN solution. The detector returns a 4-tuple consisting of the coordinate points where the face begins and ends. This is passed to the Encoder.

3.1.2 Encoder

The encoder takes the face locations identified by the Detector and the original image. The Encoder is a deep convolutional network based upon the ResNet-34 architecture, with 29 convolutional layers. The network is trained using triplet-loss optimization over 3 million image of faces. This means that the network has developed the ability to generate reliably unique encodings for individual faces. The encodings are a 128-dimensional array unique to each instance of a face. The encodings and their respective labels are passed to a K- Nearest Neighbors classifier.

3.1.3 Classifier

The encodings are passed to a K-Nearest Neighbors classifier. The 128 valued encodings can be represented as a point in 128-d space. Each point then would represent a single instance of a face, with the same faces clustered together at maximum distance from all other faces. When a set of faces is to be trained, the labelled encodings are mapped to their respective positions in space. During detection, the encodings of the party in question are passed to the KNN classifier, which measures the distance from the input to its closest neighbors. The classification is done based upon the majority vote of k neighbors and whether the closest neighbor is under a specific distance threshold. If these conditions are not satisfied the person is assumed to be ‘unknown’.

3.2 Model Architectures

This pipeline employs two major kinds of models, deep neural networks and a nearest neighbor classifier. Both our neural network implementations utilize Residual Networks, which our nearest neighbor classifier is a K-nearest neighbor implementation.

3.2.1 Residual Networks

One of the longest standing issues facing deep artificial neural networks is that of diminishing returns with an increase in depth. Ideally, the deeper a network is, the more generalized its understanding should be of whatever it is trying to learn. A more generalized model proves to be more robust when solving real- world problems. However, in reality, classical deep neural network architectures do not provide a significant advantage over shallow counterpartsin terms of accuracy. This is because, classical deep networks increase in accuracy until a certain point, beyond which their accuracy plateaus and beyond that, begins to decrease. In fact, shallow networks achieve better results than their deep counterparts with lower error rates on both validation and test data [1].This is especially true in the domain of object detection and recognition, where the inputs are images. One of the main reasons for the loss of performance at higher depth levels is due to vanishing and exploding gradients.

Vanishing and exploding gradients are two phenomena that can arise in very deep networks due to the nature of backpropagation. Because backpropagation in deep networks results in the multiplication of derivatives (chain rule), if the majority of the derivatives are less than 1, then by the time the weights are updated towards the input layer, the change in weights becomes arbitrarily small. This means that anything learned at the more general end of the network won’t transfer to the more specific layers. The opposite happens with exploding gradient, when the derivatives are greater than 1. Then the weight changes towards the input layer become exponentially bigger. This means small changes in the general end causes huge changes in the specific end.

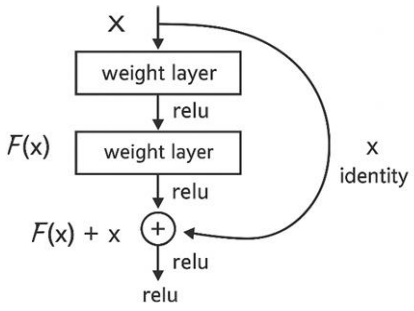

Figure 2: Skip connections

Residual networks overcome this by utilizing "skip-connections". Skip connections ensure that for a set of layers, the minimum output will always be equal to the input x. Therefore, the output of the layers will always be of the form $F(x) + x$. This eliminates the exploding and vanishing gradient problem because the minimum value of outputs is no longer 0, this ensure that at the very least, input data is preserved from layer to layer. Such layers are called residual layers.

Residual layers ensure that small changes in the data (difference between a few pixels) does not cause large changes in the network's understanding. According to the research done by Veit, Andreas et al, they were able to show that a deep residual network was similar to chaining together several shallow networks. They showed that removing a layer from a ResNet did not drastically increase the error rate, unlike regular networks, where removing even one layer completely destroyed the accuracy [7]. Thus, ResNets allow for very deep neural networks to utilize the benefit of adding depth without the adverse effects.

3.2.2 Convolutional Neural Networks

Convolutional Neural Networks were first proposed as an effective method for learning from image data. It's first use was in recognizing handwritten digits [3]. Today, CNNs are the most popular deep learning algorithm for learning from images and videos. They are effective at learning data with spatial properties, such as 2D arrays of pixels. Convolutional networks consist of several kinds of layers stacked in successive order. These include convolution layers, max-pooling layer and fully connected layers.

The following are the various aspects of a ConvNet:

3.2.2.1 Convolution:

The convolution layer accepts a 2D input array. Each node in the convolution layer represents an [m x m] portion of an image, called a 'feature'. The process of convolution involves comparing the current feature with every other [m x m] portion of the image, to calculate the similarity of the feature to all other parts of the image. Once a feature is fully compared with an image the feature matrix is shifted using a sliding window protocol.

This produces a likeness matrix of values between -1 and 1. Where -1 indicates no likeness and 1 indicates strong likeness.

3.2.2.2 Normalization (ReLU):

Next, each node is passed through a rectified linearity (ReLU) function, of the form:

$F(x) = 0 \; if \; x < 0, x \; if \; x \geq 0$

This ensures that none of the values are negative. The existence of negative values can affect the learning as it can increase the complexity of the network.

3.2.2.3 Max Pooling:

The max pooling layer reduces the overall complexity of the image and generalizes it. This is done by taking an [n x n] region of the feature matrix and extracting the maximum value in that region.

3.2.2.4 Fully Connected Layers:

Fully connected layers are typical dense layers, they are used in a CNN to decompose 2D data down to an 1D array of values. In our encoder network, the output layer is a fully connected layer of 128 nodes, where each output value ranges from -1 to 1.

3.2.3 K-Nearest Neighbors:

K nearest neighbor algorithm works based on minimum distance from the query instance to the training samples. After we gather K nearest neighbors, we calculate the prediction based on the majority class. First, we determine the value of k as a parameter of this algorithm. Then we calculate the distance between the query instance and all the training samples, here we use Euclidean distance for this. The next step is to find the K-nearest neighbors. We include training sample as nearest neighbor if the distance of this training sample to the query instance is less than or equal to the K-th smallest distance.

In our solution we chose to go with a 'ball-tree' implementation of KNN. Ball- tree is a data structure used to split and organize points in high-dimensional space. It allows for fast and efficient training and searching. Ball-tree is a modification of the K-D tree algorithm, with the added benefit that ball-tree supports many more distance metrics. Ball-tree constructs a binary-tree of all

the points based upon their dimensionality. A KNN using ball-tree implementation for organizing data is fast to train and search with the help of highly efficient binary tree building and searching algorithms.

3.3Structure of Data:

Throughout the pipeline the data takes many forms, the input data is an image, therefore it is represented as a 2D array. An RGB image comprises of an [n x n] array where each element is a [1 x 3] array of values representing the intensities of red, green and blue channels respectively.

These arrays are converted to a grayscale image where each value of the image matrix is a number ranging from 0 to 255 indicating the black-to-white

intensity, with 0 being black and 255 being white. This matrix is passed to the detector CNN.

The detector returns a set of 4-tuples of the form (x1, y1, x2, y2) indicating the regions of the image where it identifies faces.This 4-tuple and the original image is passed to the encoder. The encoder uses the 4-tuples to locate the regions of the image with faces and calculates their encodings. The encodings returned is a 128-d array.

These arrays are passed to the KNN classifier, which calculates the distance between the input point and N nearest neighbours and checks if that distance is below an accepted threshold. This is done for all detected faces. The final output is a name if the person is identified or 'Unknown' and their respective probabilities.

3.4 Pipeline Deployment:

Our pipeline is deployed as a Django REST API accessible to any application with the proper API key. This allows our application to be deployed independently while still remaining compatible with applications that which to integrate with it.

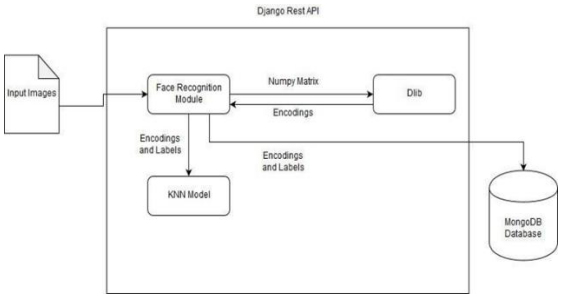

Figure 3: REST API Structure

IV. RESULT AND DISCUSSION

The graph illustrates the relationship between the number of samples per person and the corresponding accuracy of the face recognition system. As shown, an increase in the number of samples per individual leads to an improvement in recognition accuracy. When only 5 samples are used, the system achieves an accuracy of 92.1%. As the number of samples increases to 10, the accuracy rises to 96.05%. With 15 samples, a slight increase to 95.4% is observed, followed by a marginal improvement to 96.05% when the sample size reaches 20.These results demonstrate that having more samples per person contributes to better recognition performance, although the improvement plateaus beyond a certain point. This indicates that while a minimum threshold of samples is essential for reliable identification, additional samples beyond this threshold yield only minor gains in accuracy.

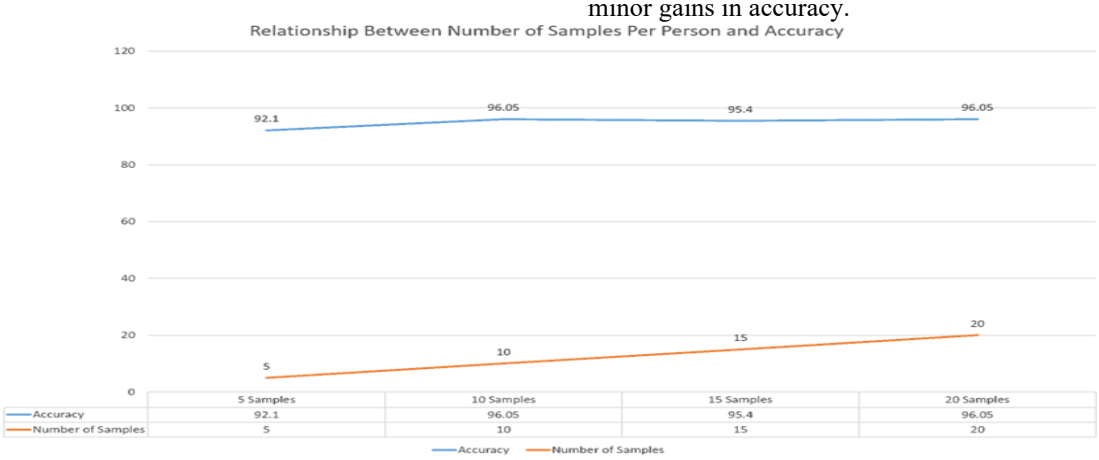

Figure 4: Relationship Between Samples and Accuracy

There are still many improvements that could be made to our pipeline. The following are a list of improvements we wish to make in the future:

- Identification from videos
- Liveness Detection (detecting attempts to spoof faces)
- Emotion Detection
- Posture Prediction

REFERENCES

[1] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[2] King, Davis E. "Max-margin object detection." arXiv preprint arXiv:1502.00046 (2015).

[3] LeCun, Yann, et al. "Comparison of learning algorithms for handwritten digit recognition." International conference on artificial neural networks. Vol. 60. 1995.
[4] Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." CVPR (1) 1 (2001): 511-518.
[5] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." International Conference on computer vision & Pattern Recognition (CVPR'05). Vol. 1. IEEE Computer Society, 2005.
[6] Turk, Matthew, and Alex Pentland. "Eigenfaces for recognition." Journal of cognitive neuroscience 3.1 (1991): 71-86.
[7] Veit, Andreas, Michael J. Wilber, and Serge Belongie. "Residual networks behave like ensembles of relatively shallow networks." Advances in neural information processing systems. 2016.
[8] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.