Proceedings of the 4th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/45/20241677

Real-time face recognition method based on MTCNN-Inception-ResNet-v2-SVM model

Hongyi Zhang

Department of Mathematics, Southern University of Science and Technology, Shenzhen, China

zhy20010127@outlook.com

Abstract. With the widespread use of smart devices such as smartphones, facial recognition applications have experienced rapid growth. Additionally, breakthroughs in deep learning algorithms have led to the development of facial recognition technology based on deep convolutional networks, greatly improving recognition speed and accuracy. Therefore, this study proposes a real-time face recognition method based on the MTCNN-Inception-ResNet-v2-SVM model. In the facial detection phase, the MTCNN algorithm, which offers better overall performance compared to face detection algorithms such as OpenCV and Dlib, is utilized. Data augmentation and other methods are employed in the image preprocessing phase to enhance data diversity. The Inception-ResNet-v2 deep convolutional neural network is used as the backbone network for feature extraction in the facial recognition backbone network section. In the final classification stage, an SVM classifier is employed for the ultimate facial classification. Comparative analyses with models such as Inception-ResNet-v1, Inception-v3, and Inception-v4 are conducted in the backbone network section, determining that the Inception-ResNet-v2 model exhibits the best overall performance. The final result is a real-time face recognition model, MTCNN-Inception-ResNet-v2-SVM, with a high accuracy of 98.79% and fast processing speed.

Keywords: Deep Learning, Facial Recognition, Convolutional Neural Network, MTCNN, Inception-ResNet-v2.

1. Introduction

Face recognition is one of the most common biometric features, widely employed in various software applications for facial recognition due to its proactive recognition capabilities and high-precision features, surpassing other biometric technologies. However, face recognition has several drawbacks, including susceptibility to lighting variations, facial angles, and obstructions such as sunglasses and masks.

Traditional face recognition methods primarily extract facial features through techniques like PCA [1] and LBP [2], followed by the establishment of a comparative model based on these features to construct a face recognition system. However, these models often exhibit suboptimal performance and are inadequate for large-scale applications. With the development of deep learning, convolutional neural networks, advancements in GPU technology, and the construction of large-scale databases, the performance of face recognition systems has improved.

DOI: 10.54254/2755-2721/45/20241677

This study aims to address issues related to the size of training models and the computational time of face recognition. By employing deep learning, the goal is to reduce the impact of factors such as facial localization, feature detection, and environmental changes that may lead to misjudgments. The ultimate objective is to enable rapid loading of the face recognition system on computers and ensure real-time computation and recognition.

Referring to the challenges mentioned in "Face recognition using eigenfaces" [3], such as facial localization, feature detection, and facial orientation, deep learning, through its neural network approach and sufficient training data, along with data augmentation methods, can alleviate the difficulty of these issues.

Currently, the industry extensively employs deep learning methods for face recognition, exemplified by Google's FaceNet [4]. The concept involves obtaining feature vectors for each facial image through a neural network. After establishing a database of these feature vectors, the similarity between two facial images is represented by the Euclidean distance of their feature vectors, with smaller distances indicating higher similarity.

The deep convolutional neural network in this study utilizes Inception-ResNet-v2 [5, 6]. The loss function incorporates both Softmax Loss and Center Loss [7] to increase inter-class distance while reducing intra-class distance, thereby enhancing the accuracy of face classification. The training model structure is based on FaceNet, which employs two distinct deep convolutional neural networks [8, 9]. In comparison to FaceNet, the Inception-ResNet-v2 used in this study has fewer parameters and lower computational complexity.

The research framework of this study is mainly divided into four parts: face detection, face image processing, face recognition, and feature-based classification comparison. In the training phase, facial image data is loaded into the MTCNN [10] face detection system, followed by data processing for facial images, such as data augmentation and face alignment. The processed image data is then input into a deep convolutional neural network for training, resulting in the extraction of facial feature vectors. The trained model is saved for future use. In the prediction phase, the face to be tested undergoes feature extraction using a pre-trained facial model structure. Subsequently, the two embedded feature vectors are compared to determine if they are the same. However, having discriminative features obtained from the trained model alone is not sufficient for effective face recognition. Therefore, it is necessary to complement the trained model with an SVM classifier to achieve the desired face recognition effect.

The study flow framework is shown in Figure 2.

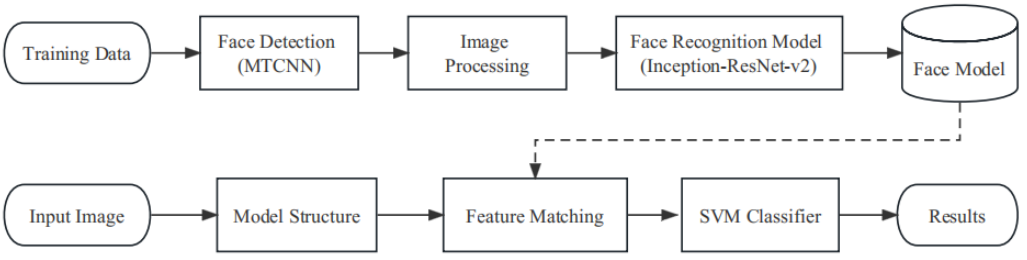

Figure 1. Flowchart of Face Recognition.

2. Models and Methods

2.1. Face Detection Face Recognition

Face detection is a critical task in the field of computer vision, aiming to accurately locate and identify the positions of faces in images or videos. In real-world scenarios, the presence of abundant and complex lighting variations often leads to a high error rate in face detection. Therefore, it is crucial to choose methods that exhibit high tolerance to environmental changes for effective face detection. In this chapter, MTCNN is selected as the face detection method for this study. A comparative analysis is conducted

DOI: 10.54254/2755-2721/45/20241677

against traditional face detection methods such as OpenCV and Dlib, with an examination of their respective strengths and weaknesses.

2.1.1. OpenCV and Dlib Face Detection

OpenCV's face detection utilizes the Haar Feature-based Cascade Classifier method [11]. This method boasts an extremely fast computation speed but suffers from a notable drawback of high error rates. Its detection accuracy is susceptible to the influence of training data or variations in lighting conditions, making it more suitable for simple and less dynamic environments. By sliding windows across the image to compute feature values at each position, different masks form weak classifiers, which are then combined through cascading and adaptive boosting (AdaBoost) to create a robust classifier. These weak classifiers quantify image features, such as the difference in eye color relative to the cheeks, using pixel values in rectangular blocks. The final robust classifier is generated through the combination of these weak classifiers using AdaBoost.

Dlib's face detection employs the Histogram of Oriented Gradients (HOG) method [12]. This method is characterized by high accuracy but slower computation speed. In real-world scenarios, it is less susceptible to environmental disturbances and provides more accurate facial annotations. However, under real-time conditions, additional techniques or suitable hardware may be required for face detection. The core idea of HOG is to use histograms to describe the direction, intensity, and distribution density of gradients, making it particularly suitable for pedestrian detection. To address limb movement issues, the HOG model is divided into a main model and multiple sub-models, cross-verified to enhance overall recognition rates during limb motion changes.

The selection of OpenCV and Dlib for comparison with MTCNN in face detection is primarily driven by a comprehensive evaluation of traditional feature-based methods and deep learning methods. OpenCV and Dlib, representing widely used traditional approaches, exhibit a certain level of accuracy and real-time performance. Comparing them with MTCNN aids in assessing the strengths and weaknesses of deep learning methods in face detection tasks. Moreover, the choice is influenced by the fact that OpenCV and Dlib are open-source libraries with extensive user and community support, making the comparison more practically meaningful.

2.1.2. MTCNN Face Detection

Due to the abundant and complex variations in lighting conditions in real-world scenarios, the error rate of face detection increases. Therefore, it is necessary to choose a method with a high tolerance to environmental image changes for face detection. In this study, considering both accuracy and computational speed, we utilized MTCNN (Multi-Task Cascaded Convolutional Networks) as the face detection method, which achieved the best overall performance.

MTCNN, a Multi-task Convolutional Neural Network, integrates face region detection and facial key point detection. Its overall structure is similar to a cascade and consists of three network layers: P-Net, R-Net, and O-Net.

Firstly, the image is transformed into different scales to build an image pyramid, adapting to the detection of faces of various sizes.

1. P-Net (Proposal Network): P-Net is a fully convolutional network responsible for initial feature extraction and bounding box calibration. It operates on the image pyramid, and its Fully Convolutional Network (FCN) conducts preliminary feature extraction and bounding-box regression. Bounding boxes are adjusted, and Non-Maximum Suppression (NMS) [13] is applied to filter out most windows.
2. R-Net (Refine Network): R-Net is a convolutional neural network that refines the candidate boxes. It introduces a fully connected layer, making the screening of input data more stringent. After P-Net, numerous candidate windows are retained. These windows are fed into R-Net, which filters out many poorly performing candidates. Finally, Bounding-Box Regression and NMS are applied to optimize the predictions further.

DOI: 10.54254/2755-2721/45/20241677

3. O-Net (Output Net): O-Net is a more complex convolutional neural network than R-Net, with an additional convolutional layer. Unlike R-Net, O-Net employs more supervision to recognize facial regions and regress facial key points. It ultimately outputs five facial feature points.

MTCNN balances performance and accuracy by avoiding the significant computational cost associated with traditional approaches like sliding windows and classifiers. It first uses a small model (P-Net) to generate candidate boxes for potential targets. Then, a more complex model (R-Net) performs detailed classification and higher-precision bounding box regression. This process is recursively executed through three layers (P-Net, R-Net, O-Net) to achieve rapid and efficient face detection.

In summary, MTCNN employs an image pyramid for initial scale transformation, utilizes P-Net to generate a large number of candidate target boxes, applies R-Net for the first round of selection and bounding box regression, and finally uses the more complex and accurate O-Net for discrimination and fine-grained bounding box regression on the remaining target boxes.

2.2. Face Image Processing

After completing face detection, image processing serves not only to remove interference but also to transform the image into an ideal format for subsequent tasks such as feature extraction, aiming to enhance the overall recognition performance. This section mainly explores image processing techniques related to this study, including data augmentation, image normalization, and facial alignment.

2.2.1. Data Augmentation

In real-world scenarios, people may have a set of image data captured in limited scenes. However, our target application may involve different conditions, such as varying orientations, positions, scaling ratios, and brightness levels. Therefore, it is necessary to train neural networks to handle these diverse situations by incorporating synthetically generated data. Data augmentation is a technique that artificially expands limited data to generate more valuable data, overcoming issues of insufficient and uniform training data. Currently, this method finds widespread application in the field of deep learning. However, due to differences between generated and real data, it inevitably introduces some challenges. The essence of data augmentation is to enhance the learning model's adaptability to unknown data. In this study, five data augmentation techniques are employed, including flipping, rotation, cropping, scaling, and shifting.

2.2.2. Image Normalization

The image normalization process consists of two parts. The first part is grayscale conversion, primarily aimed at reducing image information. The second part is image extraction, which aims to rapidly obtain facial key points, remove unnecessary background images, and normalize the image size to enhance the model's generalization performance.

1. Grayscale Conversion: In the RGB color space, each pixel contains three components—red, green, and blue—with values ranging from 0 to 255. In grayscale images, the brightness of each pixel is represented by a value ranging from 0 to 255. To convert a color image to grayscale, a common formula is used, as follows.

$$Gray(R, G, B) = 0.299R + 0.587G + 0.144B \quad (1)$$

   This method enhances contrast, with brightness in the image highlighting edge features, making facial features more effectively captured during feature extraction.
2. Image Extraction: Image extraction is primarily aimed at removing unnecessary information from facial images. Since each facial image has a different size, standardizing facial edges requires extracting the desired portions using image region extraction. The irrelevant external parts of facial images need to be removed during this process, which is also a part of normalization.

DOI: 10.54254/2755-2721/45/20241677

2.2.3. Face Alignment

Face alignment refers to adjusting the detected face to a standardized pose for better adaptation to subsequent face recognition or analysis tasks. The goal of face alignment is to ensure consistent facial poses and appearances, thereby enhancing the robustness and accuracy of subsequent tasks.

After performing facial key point detection using the MTCNN facial detection algorithm mentioned above, the detected key point information can be utilized to apply an affine transformation for rotation and scaling, ultimately achieving the effect of face alignment.

Affine transformation is a type of linear transformation. For a two-dimensional affine transformation, it can be represented as a combination of matrix multiplication and translation. Assuming a two-dimensional point (x, y), the resulting point after an affine transformation is denoted as (x', y'). If the image undergoes a rotation angle θ and scaling factors S_x and S_y in the x and y directions, the affine transformation can be expressed by the following formula.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} S_x cos(\theta) & -S_x cos(\theta) & c \\ S_x sin(\theta) & S_y cos(\theta) & f \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{2}$$

In this study, face alignment is achieved by detecting facial key points to obtain the coordinates of these key points. Subsequently, the facial orientation is adjusted based on the coordinates of the facial key points, accomplishing the process of face alignment. The approach involves first determining the coordinates of key points using MTCNN and then employing an affine transformation in OpenCV for alignment.

2.3. Face Recognition Algorithms

Traditional face feature algorithms include PCA (Principal Component Analysis) and LBP (Local Binary Pattern). With the development of convolutional neural networks (CNNs), facial feature extraction is now predominantly handled by CNNs. On one hand, CNNs enable the training of large batches of data, and on the other hand, they significantly improve the precision and accuracy of feature extraction, yielding more satisfactory results.

CNNs process image data through components such as convolutional layers, pooling layers, and fully connected layers. Local features are extracted through convolutional operations, data dimensions are reduced through pooling operations, and final classification is performed using fully connected layers. The core idea is to extract high-level abstract features of images through layer-by-layer stacking, allowing the network to automatically learn patterns and features in the data. In contrast, ResNet [14] introduces the concept of residual learning. By designing residual blocks and introducing skip connections in the network, gradients can propagate directly through these skip connections, effectively addressing the issues of gradient vanishing and exploding during deep neural network training. The design of ResNet makes the network easier to train, allowing the construction of deeper network structures, and leading to significant achievements in practical applications. This introduces the main focus of the current research, the network architectures of Inception and Inception-ResNet.

2.3.1. Inception Architecture

1. Inception-v3

From the input image, passing through the Stem, and then to the Inception Module, there are 3 Inception-A Modules, 5 Inception-B Modules, and 2 Inception-C Modules. The Inception modules are carefully designed convolutional modules that can generate discriminative features while reducing the number of parameters. Each Inception module consists of parallel convolutional and pooling layers.

After the convolutional layers and Inception modules, the feature map size is 5x5, with a 2048-dimensional vector. Feature extraction utilizes the Stride, Padding, and MaxPooling operations within the convolution to compute the feature vector of the facial image. Finally, the feature vector undergoes processing through Average Pooling, Dropout, Fully Connected Layer, and L2 Normalization Layer.

2. Inception-v4

DOI: 10.54254/2755-2721/45/20241677

From the input image, passing through the Stem and then to the Inception Module, there are 4 Inception-A Modules, 7 Inception-B Modules, and 3 Inception-C Modules, along with Reduction-A Module and Reduction-B Module. Each Inception module consists of parallel convolutional and pooling layers.

After the convolutional layers and Inception modules, the feature map size is 5x5, with a 1536-dimensional vector. Feature extraction utilizes the Stride, Padding, and MaxPooling operations within the convolution to compute the feature vector of the facial image. Finally, the feature vector undergoes processing through Average Pooling, Dropout, Fully Connected Layer, and L2 Normalization Layer.

In contrast, Inception-v4 further increases the depth from 48 layers to 73 layers, incorporating 4 Inception modules with a total parameter count of 42 million. Inception-v4 eliminates the auxiliary classifiers present in Inception-v3, opting for a simplified and optimized structure. It also introduces techniques such as residual connections from ResNet. These enhancements are aimed at improving model performance, including higher accuracy and better generalization capabilities. The choice of which model to use often depends on the complexity of the task and the availability of computational resources. Inception-v4 is designed to adapt to more intricate computer vision tasks, providing robust support for the advancement of deep learning in the field of image processing.

2.3.2. Inception-ResNet Architecture

Use 20-point type for the title, aligned to the center, linespace exactly at 20-point with a bold and italic font style and initial letters capitalized. No formulas or special characters of any form or language are allowed in the title.

1. Inception-ResNet-v1

From the input image, passing through the Stem, and then to the Inception-ResNet Module, there are 5 Inception-ResNet-A Modules, 10 Inception-ResNet-B Modules, and 5 Inception-ResNet-C Modules, along with Reduction-A Module and Reduction-B Module. Feature extraction utilizes operations within the convolution, including Stride, Padding, and MaxPooling, to compute the feature vector of the facial image. Finally, the feature vector undergoes processing through AveragePooling, Dropout, Fully Connected Layer, and L2 Normalization Layer.

2. Inception-ResNet-v2

The architecture of Inception-ResNet-v2 is not significantly different from Inception-ResNet-v1. The distinctions lie in the output dimensions after computation through different Stem structures, with Inception-ResNet-v2 having higher dimensions than Inception-ResNet-v1. Additionally, each module has more parameters. Feature extraction similarly involves operations within convolution, including Stride, Padding, and MaxPooling, to compute the feature vector of the facial image. Finally, the feature vector undergoes processing through AveragePooling, Dropout, Fully Connected Layer, and L2 Normalization Layer.

In the Residual-Inception network, the Inception-ResNet module utilizes an Inception module with lower computational complexity than the original Inception module. Notably, the term "Inception" implies the incorporation of ResNet's residual structure, where the output of each Inception-ResNet layer is added to its input, deepening the network structure. The Inception-ResNet module is a meticulously designed convolutional module capable of generating distinctive features while reducing the number of parameters.

In terms of structure, Inception-ResNet-v1 is similar to Inception-ResNet-v2, with the main difference being that the latter is deeper and more complex, resulting in more parameters and achieving higher accuracy. A significant difference lies in the preprocessing part, where Inception-ResNet-v2 adopts a more complex Stem structure. The output dimension of the Stem in Inception-ResNet-v2 is a 384-dimensional vector, larger than the 256-dimensional vector of Inception-ResNet-v1. The intricate Stem structure leads to a slightly slower training speed for Inception-ResNet-v2, but it delivers better overall performance.

DOI: 10.54254/2755-2721/45/20241677

2.4. SVM Classifier

SVM is a supervised learning algorithm primarily used for classification and regression tasks. In classification problems, the goal of SVM is to find an optimal hyperplane that separates the dataset into two classes, maximizing the margin between the two classes. Support vectors are the training samples closest to the hyperplane, playing a crucial role in defining the hyperplane.

The core idea of SVM is to map the raw data into a high-dimensional space, finding a hyperplane in this space to achieve linear separation of the data. Commonly used kernel functions include linear, polynomial, RBF, and others. By selecting different kernel functions, SVM can adapt to various types of data distributions.

In deep learning, backbone networks such as Inception-ResNet-v2 are often employed to learn high-level feature representations that better capture information in images. However, deep learning models may suffer from overfitting or insufficient generalization, especially when trained on limited data or dealing with complex tasks.

To address this issue, traditional machine learning classifiers like SVM can be used to classify the high-level features extracted by deep learning models. This approach combines the feature learning capability of deep learning with the generalization ability of traditional machine learning.

Therefore, by using an SVM classifier after Inception-ResNet-v2, the accuracy of facial classification can be further improved on the foundation of a deep learning model, enhancing robustness to small-scale data. This combination approach has demonstrated excellent results in many facial recognition tasks.

3. Experimental Information and Results

3.1. Experimental Datasets

The facial image training dataset is divided into training, testing, and validation sets. In this study, the CASIA-WebFace [15] database is utilized, with 90% of individual facial images allocated to the training set and 10% to the validation set for model training. The model learns to compare facial feature vectors in the feature space to determine whether they possess similar characteristics. Finally, the LFW [16] database is employed for testing, serving as a benchmark to evaluate the model's accuracy.

The CASIA-WebFace database comprises 10,575 individuals, totaling 494,414 images, provided by the Chinese Academy of Sciences Institute of Automation (CASIA). It includes a diverse collection of facial images from various individuals, exhibiting complexity with different poses, expressions, and lighting conditions. This makes the CASIA-WebFace database a challenging dataset valuable for testing the robustness and generalization ability of facial recognition algorithms.

The LFW database is a standard evaluation dataset for face recognition. It contains over 13,000 facial images gathered from the internet, representing approximately 1,680 different identities of 5,749 individuals. The database features real-world facial images captured in diverse environments, encompassing different ages, ethnicities, genders, and expressions. This establishes the LFW database as a standard benchmark for evaluating the performance of facial recognition algorithms.

3.2. Experimental Process

In this paper, the real-time facial recognition system is primarily divided into two components. The first component is the facial training model architecture, and the second component is the SVM classifier. In the facial training model architecture, we input facial images, and then, after training with Inception-ResNet-v2, output facial image feature vectors. The distance between vectors represents the similarity of images, which is further classified through the SVM classifier. The following image illustrates the architecture of the facial training model.

DOI: 10.54254/2755-2721/45/20241677

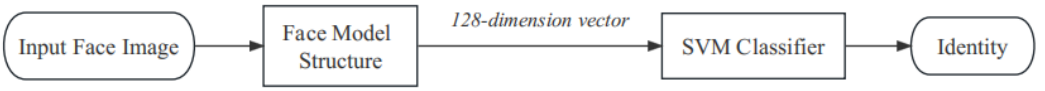

Figure 2. Real-time Face Recognition System.

The performance of the facial training model architecture directly impacts the overall accuracy of the entire facial recognition system. The facial training model structure in this paper is based on FaceNet, with a distinction lying in the utilization of the Inception-ResNet-v2 deep convolutional neural network and the Center Loss + Softmax Loss as the loss function, as opposed to FaceNet.

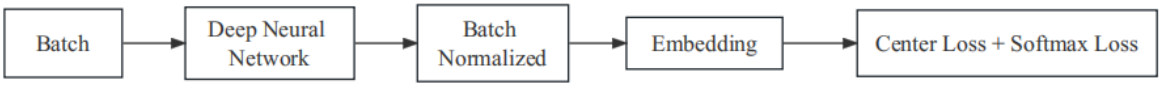

Figure 3. Training Structure of Face Recognition Model.

Batch refers to inputting pre-processed facial images into the model. Deep Architecture is based on different CNN architectures, such as Inception-v3, Inception-v4, Inception-ResNet-v1, Inception-ResNet-v2, etc. Batch Normalization involves normalizing features to a consistent range after preprocessing to obtain more discriminative features. Embedding is the process of inputting the 128-dimensional vector generated by the trained Inception-ResNet-v2 and Batch Normalized features into the loss function for feature vector differentiation. Center Loss + Softmax Loss obtains an embedding function from the vector, where features between the same faces are closer, and features between different faces are as separate as possible, making the determination of whether two facial images belong to the same person more effectively.

After training the model, the 128-dimensional feature vectors are obtained. Using the pre-processed data, a hyperplane is found to separate the two parts of the categories. Once the SVM classifier is trained, this method can be used for fast and accurate classification. Finally, by combining the facial features with the trained model and SVM classifier, the facial recognition system is enhanced.

3.3. Experimental Hyperparameter Settings

The facial recognition method in this study combines the Inception-ResNet-v2 deep convolutional neural network. Images of size 160×160, obtained after face detection and alignment using MTCNN, are used as input. Dimension vectors of 256 and 128 are tested, and results indicate that the 128-dimensional vector input performs the best, consistent with FaceNet. Three optimizers, RMSPROP, ADAM, and ADAGRAD, are compared, with ADAM showing the best performance. The learning rate is set to -1, allowing the model to converge to a local minimum. Dropout is experimented with values of 0.2, 0.4, 0.6, and 0.8, with 0.8 being the optimal choice during training. The optimal value for weight decay is 5e-4. Unlike FaceNet, this study utilizes ADAM as the optimizer.

During data training, a batch size of 90 is chosen, indicating that 90 images are used in each training batch. The number of epochs is set to 100, representing 100 training cycles. The epoch size is set to 1000, meaning that each training cycle consists of 1000 iterations. The study employs the ADAM optimizer to minimize the loss value, and the accuracy of the neural network is calculated after every 1000 steps, guiding the training until reaching 100 epochs.

3.4. Experimental Results

This study initially conducted a comparison of face detection algorithms. MTCNN was utilized for face detection, and it exhibited excellent performance for single-person image detection. The study tested three face detection methods, MTCNN, Dlib, and OpenCV, by measuring the average computation time for processing a single-person image at a resolution of 1080×720. The time taken for each of the mentioned face detection methods is presented in the following table.

DOI: 10.54254/2755-2721/45/20241677

Table 1. Average processing time for a single image.

Model	Average processing time
OpenCV	1.1ms
Dlib	72ms
MTCNN	2.4ms

When comparing processing times, OpenCV exhibits the fastest processing speed at 1.1ms, followed by MTCNN with a slightly slower speed than OpenCV. Dlib, on the other hand, has a significantly slower processing speed compared to the other two detection algorithms. However, when considering the overall performance, which combines both algorithm speed and success rate in face detection, there is a substantial gap. In a complex scenario involving the multi-face recognition of 28 individuals, the success rates of face detection for the three algorithms are shown in the following table.

Table 2. Success rate of facial detection for 28 people.

Model	Number of detection	Detection success rate
OpenCV	21	75.0%
Dlib	24	85.7%
MTCNN	27	96.4%

Consequently, it can be concluded that OpenCV has a fast detection speed but a lower accuracy rate. Dlib, despite its slow processing speed, exhibits an improved success rate in comparison to OpenCV. Meanwhile, MTCNN outperforms both detection methods not only in terms of detection speed but also in detection success rate. In summary, MTCNN demonstrates the best overall performance.

This study also includes a comparison of the accuracy and speed of four convolutional neural networks—Inception-v3, Inception-v4, Inception-ResNet-v1, and Inception-ResNet-v2—on the LFM dataset. The table is provided below.

Table 3. Regression performance for each model.

Model	Accuracy	Speed
Inception-ResNet-v1	97.98%	0.34s
Inception-ResNet-v2	98.79%	0.47s
Inception-v3	96.64%	0.46s
Inception-v4	92.53%	0.74s

According to the table above, considering both training speed and accuracy, the optimal model is determined to be Inception-ResNet-v2. Therefore, it is chosen as the backbone for the facial training model in this study.

In conclusion, the best facial recognition model obtained in this study involves using MTCNN for face detection, inputting processed data into the backbone network of Inception-ResNet-v2, and finally classifying faces using an SVM model. The resulting final facial recognition accuracy is 98.79%.

4. Conclusion and Prospects

The research conducted a comparative validation in the aspect of face detection using MTCNN. Despite its slightly slower detection speed compared to OpenCV, MTCNN demonstrated the best detection recall. Considering both time and accuracy, MTCNN stands out as the optimal choice for face detection. Factors influencing image facial recognition speed and accuracy in this study include the number of recognized faces, neural network principles, and architectures, as well as hyperparameter settings. To increase facial recognition speed, methods such as reducing the number of neural network layers and shrinking the convolutional kernel can be employed, but this might lead to a decrease in recognition accuracy. On the other hand, enhancing accuracy can be achieved by increasing convolutional layers

DOI: 10.54254/2755-2721/45/20241677

and incorporating residual structures. The study compared four different deep convolutional neural networks and concluded that Inception-ResNet-v2 exhibited the best overall performance. Ultimately, the MTCNN-Inception-ResNet-v2-SVM facial recognition model achieved the highest accuracy of 98.79%.

Due to variations in facial recognition time resulting from different combinations of datasets and convolutional network architectures, it is challenging to make direct time comparisons. Given that facial recognition prioritizes accuracy, accuracy serves as a better metric to evaluate a model's quality. Ideally, the goal is to achieve the highest recognition rate in the shortest time possible. Recognition speed and accuracy are interrelated, and the improvement goal is to enhance facial recognition speed without compromising accuracy. Therefore, to implement the facial recognition system using the approach in this study, aside from deepening the depth of the convolutional neural network to improve recognition accuracy, enhancements can be made by adjusting network hyperparameters, utilizing faster GPU acceleration, or adopting more advanced network architectures to improve both accuracy and speed of facial recognition.

Acknowledgments

I would like to express my deepest gratitude to my parents and my girlfriend for their unwavering support and encouragement throughout this academic journey. Their love and belief in me have been my constant motivation, and I am truly thankful for their understanding during the demanding times of my research.

I am also immensely grateful to Assistant Professor Rabih Younes at Duke University for his invaluable guidance, mentorship, and support. His expertise, encouragement, and insightful feedback significantly contributed to the success of this research. I am fortunate to have had the opportunity to learn from such a dedicated and knowledgeable mentor.

Furthermore, I extend my appreciation to all the teachers who have imparted their knowledge and wisdom during my academic pursuits. Each instructor has played a vital role in shaping my understanding and passion for the subject matter.

This research would not have been possible without the support and contributions of everyone mentioned above. Their influence has left an indelible mark on both my academic and personal development.

References

[1] Turk, M., & Pentland, A. (1991). Eigenfaces for recognition. Journal of cognitive neuroscience, 3(1), 71-86.

[2] Ojala, T., Pietikainen, M., & Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Transactions on pattern analysis and machine intelligence, 24(7), 971-987.

[3] Turk, M. A., & Pentland, A. P. (1991, January). Face recognition using eigenfaces. In Proceedings. 1991 IEEE computer society conference on computer vision and pattern recognition (pp. 586-587). IEEE Computer Society.

[4] 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).

[5] Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2017, February). Inception-v4, inception-resnet and the impact of residual connections on learning. In Proceedings of the AAAI conference on artificial intelligence (Vol. 31, No. 1).

[6] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Identity mappings in deep residual networks. In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14 (pp. 630-645). Springer International Publishing.

DOI: 10.54254/2755-2721/45/20241677

[7] Wen, Y., Zhang, K., Li, Z., & Qiao, Y. (2016). A discriminative feature learning approach for deep face recognition. In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VII 14 (pp. 499-515). Springer International Publishing.

[8] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

[9] Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13 (pp. 818-833). Springer International Publishing.

[10] 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.

[11] Viola, P., & Jones, M. J. (2004). Robust real-time face detection. International journal of computer vision, 57, 137-154.

[12] Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05) (Vol. 1, pp. 886-893). IEEE.

[13] Neubeck, A., & Van Gool, L. (2006, August). Efficient non-maximum suppression. In 18th international conference on pattern recognition (ICPR'06) (Vol. 3, pp. 850-855). IEEE.

[14] 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).

[15] Yi, D., Lei, Z., Liao, S., & Li, S. Z. (2014). Learning face representation from scratch. arXiv preprint arXiv:1411.7923.

[16] Huang, G. B., Mattar, M., Berg, T., & Learned-Miller, E. (2008, October). Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In Workshop on faces in'Real-Life'Images: detection, alignment, and recognition.