Ingeniería y Ciencia
ISSN:1794-9165 | ISSN-e: 2256-4314
ing. cienc., vol. 17, no. 34, pp. 77–95, julio-diciembre. 2021.
http://www.eafit.edu.co/ingciencia

A Low-Cost Raspberry Pi-based System for Facial Recognition

Cristian Miranda Orostegui[1], Alejandro Navarro Luna[2] , Andrés Manjarrés García[3] and Carlos Augusto Fajardo Ariza[4]

Received: 21-06-2021 - Accepted: 05-10-2021 - Online: 01-12-2021

MSC: 68Wxx - PACS: 84.35.+i, 07.05.Mh

doi:10.17230/ingciencia.17.34.4

Abstract

Deep learning has become increasingly popular and widely applied to computer vision systems. Over the years, researchers have developed various deep learning architectures to solve different kinds of problems. However, these networks are power-hungry and require high-performance computing (i.e., GPU, TPU, etc.) to run appropriately. Moving computation to the cloud may result in traffic, latency, and privacy issues. Edge computing can solve these challenges by moving the computing closer to the edge where the data is generated. One major challenge is to fit the high resource demands of deep learning in less powerful edge computing devices. In this research, we present an implementation of an embedded facial recognition system on a low cost Raspberry Pi, which is based on the FaceNet architecture. For this implementation it was required

[1] Universidad Industrial de Santander, cristianmirandaororstegui@gmail.com, Bucaramanga, Colombia.
[2] Universidad Industrial de Santander, alnavluna@gmail.com, Bucaramanga, Colombia.
[3] Instituto Nacional de Astrofísica, Óptica y Electrónica, manjarres@inaoep.mx, Puebla, México.
[4] Universidad Industrial de Santander, cafajar@uis.edu.co, Bucaramanga, Colombia.

the development of a library in C++, which allows the deployment of the inference of the Neural Network Architecture. The system had an accuracy and precision of 77.38% and 81.25%, respectively. The time of execution of the program is 11 seconds and it consumes 46 [kB] of RAM. The resulting system could be utilized as a stand-alone access control system. The implemented model and library are released at `https://github.com/cristianMiranda-Oro/FaceNet_EmbeddedSystem`.

Keywords: Deep learning; facial recognition; embedded systems; faceNet; googLeNet; labeled faces in the wild.

Sistema de reconocimiento facial sin reentrenamiento para nuevos usuarios

Resumen

El aprendizaje profundo se ha vuelto cada vez más popular y se aplica ampliamente a los sistemas de visión por computadora. A lo largo de los años, los investigadores han desarrollado varias arquitecturas de aprendizaje profundo para resolver diferentes tipos de problemas. Sin embargo, estas redes consumen mucha energía y requieren computación de alto rendimiento (es decir, GPU, TPU, etc.) para funcionar correctamente. Mover la computación a la nube puede resultar en problemas de tráfico, latencia y privacidad. La computación en el borde puede resolver estos desafíos, pues permite acercar el proceso de computación al lugar donde se generan los datos. Un desafío importante es adaptar las altas demandas de recursos del aprendizaje profundo a dispositivos de computación de borde menos potentes. En esta investigación, presentamos una implementación de un sistema de reconocimiento facial integrado en una Raspberry Pi de bajo costo, la cual está basada en la red FaceNet. Esta implementación requirió el desarrollo de una biblioteca en C++ que puede describir la inferencia de la arquitectura de la red neuronal FaceNet. El sistema tuvo una exactitud y precisión de 77.38 % y del 81.25 %, respectivamente. El tiempo de ejecución de cada inferencia es de 11 segundos y consume 46 [kB] de RAM. El sistema resultante podría utilizarse como un sistema de control de acceso independiente. El modelo y la librería implementados están disponibles en `https://github.com/cristianMiranda-Oro/FaceNet_EmbeddedSystem`.

Palabras clave: Deep learning; reconocimiento facial; sistemas embebidos; faceNet; googLeNet; labeled faces in the wild.

1 Introduction

A person's face contains physical information that can be used for security and access control applications. The main motivation for facial recognition

is because it is considered a passive and non-intrusive system. Most of the biometric data needs to be collected by special hardware such as a fingerprint scanner, a palm print scanner, a DNA analyzer, etc [1]. Face recognition does not require physical touch, it is less intrusive than other biometric systems.

A vast amount of work has been done to make facial recognition algorithms more reliable and accurate. In recent years, deep learning approaches have dominated the facial recognition field due to their high performance in learning discriminative features. As an example, the solution proposed in [2] achieved a precision of 99.63 % in the Labeled Faces in the Wild (LFW) [3] dataset using a deep learning system called FaceNet with almost 7.5M parameters. This architecture learns a mapping of facial images to a compact Euclidean space where the distances correspond directly to a measure of facial similarity. Another deep learning solution is [4] that attains an accuracy of 99.52% in the same LFW dataset using a VGGNet-16 neural network architecture, with 138M parameters. This work implements a new loss function called range loss, designed to decrease intra-personal variations while increasing inter-personal differences in extremely unbalanced data. Also, the authors in [5] propose a new loss function called Additive Angular Margin Loss (ArcFace), which incorporates margins in a well-established loss function to maximize face class separability. They use the ResNet100 neural network with 65M parameters and obtain an accuracy of 99.83% in the LFW dataset.

The implementation of these state-of-the-art neural networks need the leverage of high-performance and power-hungry hardware [6]. These huge demands of computational and memory resources impedes their deployment in edge devices (e.g. Microcontrollers, SoC, etc.).

Bringing the computation closer to the location where it is needed (computation at the egde) can improve the response times, save bandwidth and minimize the data transmission time. Processing data at the edge also preserves the privacy of the users, since there is no need to upload the data to the cloud. This means that the data is processed at the source. Cameras, speakers, microphones, and multiple sensors are all located at the edge of the network, which provides a great opportunity of running deep learning algorithms here [7]. Edge devices are inexpensive, small, and flexible hardware devices. They are characterized by their low energy consumption and

reduced cost.

In this work, we present an implementation of an embedded facial recognition system on a Raspberry Pi. The model is based on the FaceNet architecture. The system achieved an accuracy and precision of 77.38% and 81.25%, respectively. The time of execution of each inference is around 11 seconds and only 46 [kB] of RAM are required.

The rest of this paper is structured as follows. Section 2 describes the related work. Section 3 shows the FaceNet generalities and the structure of a facial recognition system. In section 4, we present the various methods used to construct the system. The metrics utilized and their experimental results are presented in section 5. Finally, section 6 draws conclusions of our work and indicates future studies.

2 Related work

Generally, a facial recognition system is composed of three basic steps: (1) face detection, (2) feature extraction, and (3) face recognition [8]. The face detection step locates the face that appears in the image. The feature extraction step extracts a feature vector from the detected face. This feature vector is obtained by $v = f(x)$ where x is the image of the detected face and $f(\cdot)$ is the deep neural network. Finally, the face recognition step compares the extracted features with all registered faces and verifies if it is part of the database [9].

Annalakshmi et al., [10] introduced algorithms using the enhanced local binary pattern (SLBP) and histogram of oriented gradients (HOG) to classify the human gender with a SVM classifier. Over the LFW database, their proposed hybrid method achieved an accuracy of 95.7%. They attained an accuracy of 99.1% with the FERET dataset. Xi et al. [11] have introduced a new unsupervised deep learning-based technique, called local binary pattern network (LBPNet), to extract hierarchical representations of data. The LBPNet maintains the same topology as the convolutional neural network (CNN). With an accuracy of 94.04% on LFW, it shows that LBPNet is comparable to other unsupervised techniques. Arigbabu et al. [12] proposed a novel face recognition system based on the Laplacian filter

and the pyramid histogram of gradient (PHOG) descriptor. They reached an accuracy of 88.50% on LFW. In addition, a support vector machine (SVM) was used with different kernel functions as the recognition step in the system.

To achieve better results, computer vision has moved towards Convolution Neural Networks (CNN), which is a deep learning approach and state-of-the-art in computer vision. The authors in [13] introduced a new approach using texture analysis and CNNs to detect face liveness, a technique that is used for face spoofing attacks. Their enhanced architecture based on the inception version 4 network obtained 100% accuracy on the NUAA Photograph Impostor dataset for face liveness detection. An pairwise differential siamese network for occluded face recognition is proposed by Song et al. [14]. The AR dataset that contains images with natural occlusions was used for evaluation. Their proposed method outperformed the state-of-the-art algorithms with an accuracy of 99.72% and 100% on the scarf and sunglass subsets of the AR dataset, respectively. Their method also achieved an accuraccy of 99.2% on the LFW dataset.

3 FaceNet

FaceNet is a face recognition, verification and clustering neural network [2]. The authors of this model presented several models with the same overarching name called FaceNet. They discussed two different deep network architectures: The Zeiler&Fergus style networks [15] and the Inception type networks [16]. This last model is based on the on GoogLeNet and has 20x fewer parameters and 5x fewer FLOPS when compared to other proposed models in [2]. We used this model architecture for our study because of its reduction in size.

FaceNet maps a face or image to a 128-dimensional feature vector in a Euclidean space. Let x be an image, the mapping is represented by $f(x) \in \mathbb{R}^{128}$, where f is the embedding function, i.e., the GoogLeNet based neural network. The mapping can be of any dimension, but in this system, we used a length of 128 as recommended in [2]. Additionally, the embedding is an element of a 128-dimensional hypersphere, i.e., $\|f(x)\|_2 = 1$. This is beneficial in the context of nearest-neighbor classification. The distance

between each mapping is correlated to a measure of face similarity. In other words, the distance between each feature vector can be used to determine the identity of a person.

The network is trained using the triplet loss function [17, 18]. This triplet consists of 3 images: an anchor (x_i^a), a positive (x_i^p) and a negative (x_i^n). The anchor and the positive images correspond to the same identity. The negative image has a different identity than the anchor image. The triplet loss function, equation 1, tries to enforce a margin between each pair of faces from one person to all other faces in the embedding space. Equation 1 tries to bring the term, $\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha$, close to zero. This means the distances between the embeddings of the anchor images and the positive images will tend to be smaller than distances between the embeddings of the anchor images and the negative images by a margin of α, i.e. $\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha = \|f(x_i^a) - f(x_i^n)\|_2^2$. The parameter α is a margin that is enforced between positive and negative pairs. This process happens during training and can be observed in Figure 1. In hard triplets, the negative image is very close to the anchor, and the positive image is very far from it. We used these hard triplets during the training process. Figure 2 shows the block diagram of FaceNet's architecture during the training process which involves the triplet loss step.

$$\sum_i^N \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+ \tag{1}$$

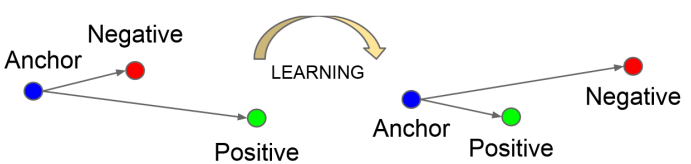

Figure 1: The result of the triplet loss training for the FaceNet models. The distance between the anchor and positive embeddings reduce while the distance between the anchor and positive embeddings maintain a distance determined by the parameter α. [2]

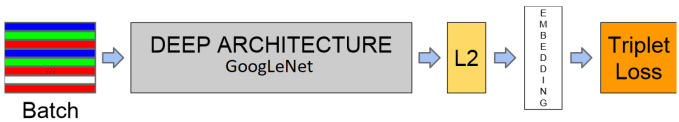

Figure 2: FaceNet's overall architecture for training. The triplet loss uses the embeddings for training. The inference of the FaceNet models doesn't include the triplet loss step. [2]

4 Methods

4.1 Dataset

There are several datasets for face recognition. The Labeled Faces in the Wild (LFW) dataset [3] consists of 13,233 images of unique 5,749 persons of different ages. This dataset is widely used for training the various methodologies of face recognition. It is also used for the evaluation of the proposed solutions. Youtube Faces DB [19] dataset has gained popularity in the face recognition community. It is made up of face images that are found from frames of videos. The data set contains 3,425 videos of 1,595 different people.

CelebFaces Attributes Dataset (CelebA) [20] is a large-scale face attributes dataset. It contains 202,599 face images and 10,177 identities. The images in this dataset cover large pose variations and background clutter. The dataset can be employed as the training and test sets for computer vision tasks, such as face recognition, face attribute recognition, and face detection. We chose the CelebA dataset for the training of the neural network because of the higher amount of unique images that each identity had as compared to other facial recognition datasets.

Our system uses the Viola-Jones algorithm [21] in the detection step. Retraining was done to adjust the network to these characteristics. The images were processed using the OpenCV libraries [22], which allow us to locate the person's face in the image and crop it out. We retrained the model with a training set of 4150 images. In the training set, each identity had an average of 26 photos. Figure 3 shows a sample of the training set. We use a GoogLeNet [16] pretrained model, which was downloaded from [23]. This model receives as input an image of $96 \times 96 \times 3$ pixels.

Figure 3: Sample from the CelebA Dataset that was used for retraining. [20]

The size of the retrained GoogLeNet model is around 15$[MB]$. The model was trained using the triple loss function (Equation 1). Hard triplets were used. We used an Adam optimizer with a learning rate of 10^{-3} and a batch of 325 images. The number of epochs was 50. Table 1 shows each layer of the implemented model on the microcontroller. The table also shows each layer's respective output size and number of parameters.

4.2 Deep learning library

We created our own library to implement the model on the device. One motivation for creating this library is to offer the possibility to deploy Deep Learning Models on any edge device that supports the C language. Our library accepts the parameters of the model in a header file (.h). The retrained model is converted to a .h5 file using TensorFlow. Since the weights come from a .h5 file, a python script converts them to a .h file (for C language execution). The number of decimals for each parameter in the network is truncated to six.

Once we have the weights, we build the neural network shown in Table 1 in C. The library developed in C is used to describe each layer in the model. This process converts the model as a function using C++. The library uses the NHWC format [24] for deep learning volumes. Table 2 shows all the functions created with their corresponding description and input parameters. The library dynamically manages RAM used in the process.

Table 1: Details of the GoogLeNet Based Architecture

Type	**Output**	**Parameters**
input	96x96x3	0
Zero_Padding2d	102x102x3	0
Conv2D	48x48x64	9472
BatchNormalization	48x48x64	256
Activation	48x48x64	0
Zero_padding2d	50x50x64	0
max_pooling2d	24x24x64	0
Conv2D	24x24x64	4160
BatchNormalization	24x24x64	256
Activation	24x24x64	0
zero_padding2d	26x26x64	0
Conv2D	24x24x192	110784
BatchNormalization	24x24x192	768
Activation	24x24x192	0
zero_padding2d	26x26x192	0
max_pooling2d	12x12x192	0
Inception_3a	12x12x256	165168
inception_3b	12x12x320	229568
inception_3c	6x6x640	399712
inception_4a	6x6x640	548608
inception_4e	3x3x1024	719840
inception_5a	3x3x96	794688
inception_5b	3x3x736	665664
average_pooling2d	1x1x736	0
flatten	736	0
dense_layer	128	94336
L2 Normalization	128	0
Total		**3743280**

The "free memory pointer" parameter in Table 2 gives the corresponding function the signal to clear or not clear the memory it creates. This parameter takes the values of 1 or 0. This library and the steps on how to use it are available on GitHub [25].

4.3 Back-end

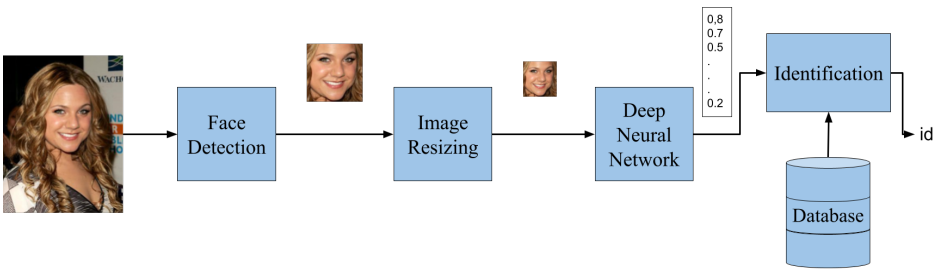

Figure 4: Diagram of the facial recognition system. The steps for face detection and image resizing are followed by the inference of the DNN which generates a 128-dimensional embedding. This vector is finally compared with the database (embeddings of the anchor images) to conclude the identity of the input image.

The back-end part of the system was implemented in C++. Figure 4 shows a block diagram of our facial recognition system. The explanation of each block is described in the following. First, we use a detection algorithm from OpenCV to obtain the face of the person. This algorithm uses the Viola-Jones method [21]. The output of this block is the image of the detected face. The dimensions of this image are variable due to the multiple windows that the Viola-Jones algorithm uses. The next step is the resizing of the detected face. The input of this algorithm is an image of any dimension, and the output is an image of size $96 \times 96 \times 3$ pixels. After the image has been resized, it passes through the deep neural network model (Table 1). Its output is a 128-dimensional feature vector. Finally, the identification of this encoding uses a database that has other feature vectors stored inside of it.

Table 2: Description and parameters of the functions in the deep learning library

Function	Description and Parameters
matrix_NHWC_alloc	Allocate memory for a NHWC data structure
	kernel or filter, width, height, and channels
cmo_lib_free	Free up memory
	Shape pointer
cmo_NHWC_l2_normalize	Apply L2 normalization on a volume
	Input volume
cmo_NHWC_MaxPooling	Apply max pooling
	Input volume, window width, window height, horizontal stride, vertical stride, and free memory pointer
cmo_NHWC_AveragePooling	Apply average pooling on a volume
	Input volume, window width, window height, horizontal stride, vertical stride, free memory pointer
cmo_NHWC_conv	Convolution between two volumes
	Input volume, input kernel, input bias, horizontal stride, vertical stride, padding type, and free memory pointer
cmo_NHWC_dense	Fully connected layer
	Input volume, kernel, bias, and free memory pointer
cmo_NHWC_batch_normalize	Apply batch normalization on a volume
	Input volume, scale, and offset
cmo_NHWC_concat	Concatenates 2 volumes in order
	Volumes, free memory pointer
cmo_NHWC_padding	Adds padding to the volume
	Input image, padding left, padding right, padding up, padding down, and free memory pointer
cmo_NHWC_ActivationRelu	Apply Relu activation to layer function
	Input volume

This Identification step consists of comparing the Euclidean distance between the generated encoding and the database's encodings. The system identifies the person when the smallest distance found is within a threshold. The identity corresponding to this encoding is the output of the system.

4.4 Front-end

The Front-End consists of an interface between the user and the facial recognition system (Listing 4.4).

The algorithm was developed in C++ and has two processes. The first one allows the addition of a person into the database. This process starts by taking a picture of the person and storing the corresponding feature vector into the database (see Section 3). The second process allows the recognition of the face. This process also starts by taking a picture of the person, which passes through the system, then the generated feature vector goes through the identification step described in section 4.3. Finally, the name of the identified person is displayed on the monitor. The database mentioned above consists of a list of feature vectors (embeddings) corresponding to the anchor images of the identities registered. By running the first process an image is taken (anchor image) and the feature vector corresponding to that anchor image, which is generated by running the inference of the system, is added to the database.

Listing 1: Front-end Pseudo code

```
while (True)
    Ask the user to choose an option
    Read option
    while (option == 1)||(option == 2):
        switch option
            1: Enroll face:
               Request name.
               Take the photo.
               Generate the encoding.
               Save the encoding in
               the database.

            2: Detect face:
               Take the photo.
```

```
                Generate the encoding.
                Calculate L2 distance
                of all registered users.
                Choose the user with the
                smallest distance within
                the threshold.
                Present the identified
                user.
            end switch
            Ask the user to choose an option
            Read option
        end while
end while
```

5 Results

In this work, we used the Raspberry Pi 3B + model, which is a credit-card-sized single-board computer. This model comprises a 1 GB of RAM memory with four USB ports and a 10/100 Ethernet port. A 5MP Raspberry Pi camera module was used.

5.1 Test set

To evaluate the performance of the system, we collected 103 images from 25 different individuals. The test set was distributed as follows, 82 images belonged to registered users and 21 images were imposters or unregistered users. The 82 images belonged to 15 individuals and each identity had between 3 and 7 images. The database of the system consisted of the feature vectors for the anchor images of these 15 individuals.

The pictures were taken from a distance of around 30[cm] from the camera. The dataset is formed by males and females with ages ranging between 23-85 years.

5.2 Model performance evaluation

The performance of a facial recognition system for identification scenarios can be evaluated based on the results obtained by the identification step

(see section 4.3). Let n be the number of identities and $sample_i$ be the number of face samples of an identity i, the total number of samples is $Total = \sum_{i=1}^{n} sample_i$.

5.2.1 Confusion matrix We use a confusion matrix to measure the model performance. We evaluate the system for different values of the threshold (see section 4.3). This parameter represents a distance threshold that determines whether an embedding is close enough to the anchor embedding to conclude that the embedding in question corresponds to the identity of the anchor embedding. Table 3 shows the results found for each case. In Table 3, the row in bold corresponds to the model performance for a threshold of 0.55. This threshold value achieved the highest F1 score of 0.793 for the system. For a threshold value of 0.55, the accuracy and precision were 77.38% and 81.25%, respectively. As the threshold increased, the values of the true positive rate (TPR) also increased. This behavior is because there was a less strict requirement on deciding a match for the person. Similarly, the number of claimed matches and the value of the false positive rate (FPR) increased as the threshold increased .

5.2.2 ROC curve The machine learning community often uses the ROC area under the curve AUC statistic for model comparison [26]. This practice is questioned because AUC estimates are noisy and suffer from other problems [27]. Nonetheless, the coherence of AUC is a respected measure of classification performance.

Figure 5 shows the values of the TPR and the FPR for different values of the threshold parameter. The AUC of the ROC curve is 0.8. The blue dashed line in Figure 5 is an approximated curve for the values. The blue squares represent the performance of the system found for a value of the threshold. The threshold used for Figure 5 ranges between 0.2 and 1. Due to the size of the dataset, the blue squares do not form a smooth curve.

5.2.3 Processing Time of the System We present the average time that the system spent on various tasks. Table 4 shows the average execution time of both processes (see section 4.4) and the inference of the GoogLeNet based model on the Raspberry Pi. The table shows that most of the run time of both processes is spent by the inference of the model.

Table 3: Results of the classification for the facial recognition system.

Threshold	TPR %	FPR %	Pre. %	Acc. %	F1
0.20	4.69	0.00	100.00	27.38	0.43
0.25	15.63	0.00	100.00	35.71	0.53
0.30	31.75	4.76	95.24	47.62	0.63
0.35	50.79	4.76	96.97	61.90	0.76
0.40	59.68	18.18	90.24	65.48	0.76
0.45	68.33	33.33	83.67	67.86	0.75
0.50	79.66	48.00	79.66	71.43	0.75
0.55	**88.14**	**48.00**	**81.25**	**77.38**	**0.79**
0.60	93.10	53.85	79.41	78.57	0.79
0.65	94.83	65.38	76.39	76.19	0.76
0.70	100.00	65.38	77.33	79.76	0.79
0.75	100.00	69.23	76.32	78.57	0.77
0.80	100.00	73.08	75.32	77.38	0.76
0.85	100.00	80.77	73.42	75.00	0.74
0.90	100.00	84.62	72.50	73.81	0.73
0.95	100.00	88.46	71.60	72.62	0.72
1.00	100.00	88.46	71.60	72.62	0.72
1.05	100.00	92.31	70.73	71.43	0.71
1.10	100.00	96.15	69.88	70.24	0.70
1.15	100.00	100.00	69.05	69.05	0.69

Table 4: Average time for the various tasks that the facial recognition system is capable of carrying out.

	Avg. Time [s]
Enrollment Process	10.78
Recognition Process	10.34
Model Inference	9.26

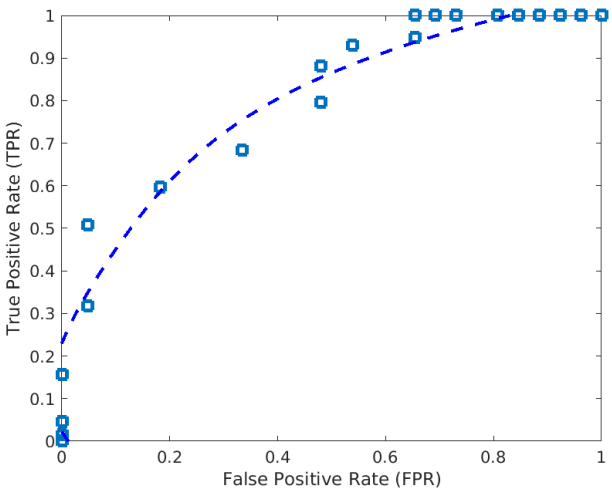

Figure 5: The corresponding ROC curve of the classification results.

5.2.4 Other Aspects Apart from the test set that the system was evaluated on, the system performed poorly on images of faces with glasses. The majority of these images were not identified correctly.

6 Conclusions

We implemented a facial recognition system based on a deep learning architecture GoogLeNet on a Raspberry Pi model 3B+. The neural network maps each input to a Euclidean hypersphere where the distance between each mapping correlates to a measure of face similarity. A library developed in the C describes the inference of the GoogLeNet architecture.

We evaluate the system on a test set of 103 images. The pictures were collected using the Raspberry Pi camera module. The algorithm had an accuracy and precision of 77.38% and 81.25%, respectively, on a group of 15 people registered in the database.

A drawback of our system is that the persons are required not to use glasses because our results suggest that the individuals with glasses had a higher probability of not being identified correctly.

The inference time of the GoogLeNet based model on the Raspberry Pi was 9.26 s.

For future studies, the number of images in the training set can be treated as a hyperparameter. Likewise, the effect of the size of the input image on the model would be interesting to see. Furthermore, optimization of the C library is a task that affects the performance of the system.

References

[1] H. H. Lwin, A. Khaing, and H. Tun, "Automatic door access system using face recognition," *International Journal of Scientific & Technology Research*, vol. 4, no. 6, pp. 294–299, 2015. http://www.ijstr.org/final-print/june2015/Automatic-Door-Access-System-Using-Face-Recognition.pdf 79

[2] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823. 79, 81, 82, 83

[3] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," University of Massachusetts, Amherst, Tech. Rep. 07-49, October 2007. https://hal.inria.fr/inria-00321923/file/Huang_long_eccv2008-lfw.pdf 79, 83

[4] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 79

[5] X. Zhang, Z. Fang, Y. Wen, Z. Li, and Y. Qiao, "Range loss for deep face recognition with long-tailed training data," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. https://arxiv.org/pdf/1611.08976.pdf 79

[6] J. Benito-Picazo, E. Domínguez, E. J. Palomo, E. López-Rubio, and J. M. Ortiz-de Lazcano-Lobato, "Deep learning-based anomalous object detection system powered by microcontroller for ptz cameras," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–7. https://doi.org/10.1109/IJCNN.2018.8489437 79

[7] S. Voghoei, N. Hashemi Tonekaboni, J. G. Wwallace, and H. R. Arabnia, "Deep learning at the edge," *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2018. https://doi.org/10.1109/csci46756.2018.00177 79

[8] V. A, D. Hebbar, V. S., K. Balasubramanya, Murthy, and N. Subramanyam, "Two novel detector-descriptor based approaches for face recognition using sift and surf," *Procedia Computer Science*, vol. 70, pp. 185–197, 12 2015. https://doi.org/10.1016/j.procs.2015.10.070 80

[9] Y. Kortli, M. Jridi, A. Falou, and M. Atri, "Face recognition systems: A survey," *Sensors*, vol. 20, p. 342, 01 2020. https://doi.org/10.3390/s20020342 80

[10] M. Annalakshmi, S. M. M. Roomi, and A. S. Naveedh, "A hybrid technique for gender classification with slbp and hog features," *Cluster Computing*, vol. 22, no. S1, pp. 11–20, 2018. https://doi.org/10.1007/s10586-017-1585-x 80

[11] M. XI, L. Chen, D. Polajnar, and W. Tong, "Local binary pattern network: A deep learning approach for face recognition," in *2016 IEEE international conference on Image processing (ICIP)*. IEEE, 2016, pp. 3224–3228. 80

[12] O. A. Arigbabu, S. M. Syed Ahamad, W. A. Wan Adnan, and S. Mahmood, "Soft biometrics: Gender recognition from unconstrained face images using local feature descriptor," *Journal of Information and Communication Technology*, 2015. 80

[13] R. Koshy and A. Mahmood, "Optimizing deep cnn architectures for face liveness detection," *Entropy*, vol. 21, no. 4, p. 423, 2019. 81

[14] L. Song, D. Gong, Z. Li, C. Liu, and W. Liu, "Occlusion robust face recognition based on mask learning with pairwise differential siamese network," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. https://doi.org/10.1109/iccv.2019.00086 81

[15] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," 2013. 81

[16] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. https://doi.org/10.1109/cvpr.2015.7298594 81, 83

[17] M. Schultz and T. Joachims, "Learning a distance metric from relative comparisons," in *Advances in Neural Information Processing Systems*, S. Thrun, L. Saul, and B. Schölkopf, Eds., vol. 16. MIT Press, 2004. https://proceedings.neurips.cc/paper/2003/file/d3b1fb02964aa64e257f9f26a31f72cf-Paper.pdf 82

[18] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *J. Mach. Learn. Res.*, vol. 10, pp. 207–244, Jun. 2009. https://doi.org/10.5555/1577069.1577078 82

[19] L. Wolf, T. Hassner, and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," *CVPR 2011*, 2011. https://doi.org/10.1109/cvpr.2011.5995566 83

[20] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," 2015. 83, 84

[21] P. Viola and M. Jones, "Robust real-time face detection," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2, 2001, pp. 747–747. https://doi.org/10.1109/iccv.2001.937709 83, 86

[22] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000. 83

[23] Coursera. Neural networks and deep learning. https://www.coursera.org/learn/neural-networks-deep-learning 83

[24] GitHub. onednn: Understanding memory formats. https://oneapi-src.hub.io/oneDNN/dev_guide_understanding_memory_formats.html 84

[25] C. M. Orostegui. Facenet_embeddedsystem. https://github.com/cristianMiranda-Oro/FaceNet_EmbeddedSystem 86

[26] J. A. Hanley and B. J. McNeil, "A method of comparing the areas under receiver operating characteristic curves derived from the same cases." *Radiology*, vol. 148, no. 3, pp. 839–843, 1983. https://doi.org/10.1148/radiology.148.3.6878708 90

[27] B. Hanczar, J. Hua, C. Sima, J. Weinstein, M. Bittner, and E. R. Dougherty, "Small-sample precision of roc-related estimates," *Bioinformatics*, vol. 26, no. 6, pp. 822–830, 2010. https://doi.org/10.1093/bioinformatics/btq037 90