Ministry of Education and Science of Ukraine

National University of “Kyiv-Mohyla Academy”

Informatics Department of the Faculty of Informatics

Implementation of the face recognition system using deep learning methods

Text part of coursework

in specialty “Computer science” 122

Coursework supervisor
Yushchenko Y.O.

____

(signature)

«__» __________ 2023 р.

Made by student
Shpir M. V.

«__» __________ 2023 р.

Kyiv 2023

Ministry of Education and Science of Ukraine
National University of “Kyiv-Mohyla Academy”

Informatics Department of the Faculty of Informatics

APPROVED
Lecturer at the Informatics Department, Docent

Yushchenko Y. O.

____
(signature)
«__» __________ 2023 р.

INDIVIDUAL TASK
for coursework

Of the 3rd year student of the Faculty of Informatics, Shpir Mariia Vadymivna

THEME: Implementation of the face recognition system

using deep learning methods

Content of the coursework:

Issue date «__» __________ 2023. Supervisor Yushchenko Y.O. _______________
(signature)
Task received _______________
(signature)

Theme: Implementation of the face recognition system using deep learning methods.

Calendar plan:

№	Name	Date
1.	Receiving a coursework assignment	01.10.2022
2.	Literature review	11.11.2022
3.	Familiarity with LFW and VGGFace2 datasets	03.12.2022
4.	Familiarity with OpenCV library	19.12.2022
	Data preprocessing and analysis	25.01.2023
5.	Implementation of the siamese neural network and FaceNet model	03.03.2023
6.	Familiarity with Tkinter library	11.04.2023
7.	Implementation of the face recognition system	15.04.2023
8.	Writing a text part	01.05.2023
9.	Providing the text part for review	06.05.2023
10.	Adjustment of the work	07.05.2023
11.	Finalizing the work, preparing slides	09.05.2023
12.	Submission of work for plagiarism check	15.05.2023
13.	Presentation of the work	23.05.2023

Shpir M.V. _______________

Yushchenko Y.O. _______________

«__» __________

Contents

Annotation

The study is devoted to an important direction of AI - the application of deep learning of neural networks for face recognition. The text focuses on the effectiveness of siamese neural networks in face recognition. Attention is paid to the improvement of one of the most popular varieties of Siamese models - FaceNet model. The technology for creating fine-tuning of neural network models for training on given data sets has been applied. To solve the problem of recognizing people from their images, the training of the Siamese model was carried out using the proposed settings and their effectiveness was evaluated. A face recognition system was implemented using a fine-tuned model to demonstrate the effectiveness of the proposed method.

Introduction

Face recognition is a fast-growing field of computer vision and artificial intelligence with numerous practical applications such as security, marketing, and social media. Deep learning has allowed researchers to make outstanding advancements in developing face recognition systems with great accuracy and reliability.

The history of deep learning dates to the early work of researchers like Ukrainian scientist Anatoliy Ivakhnenko, who is often referred to as the "father of deep learning." Ivakhnenko's pioneering work in the 1960s on the Group Method of Data Handling (GMDH), an inductive modeling method using a hierarchical structure and polynomial functions, laid the foundation for modern deep learning models [1]. Over the years, deep learning has evolved significantly, with advancements in computational power, algorithm development, and the availability of large datasets fueling the growth of this field. Today, deep learning models have permeated various industries, revolutionizing areas such as computer vision, natural language processing, and speech recognition.

In recent years, the usage of siamese neural networks has been one of the most promising approaches in the field of face recognition. Siamese networks are a type of neural network structure which consists of two identical sub-networks that share the same set of weights. This architecture has proven to be effective in one-shot image recognition tasks, in which the model is trained to detect an object or person based on only one sample.

One of the most successful face recognition models based on a siamese neural network architecture is FaceNet. FaceNet is a deep learning model developed by Google researchers that learns a high-dimensional embedding of face images, where images of the same person are clustered closely together in the embedding space, while images of different people are separated by a large distance. This approach has been shown to be highly effective in recognizing faces in real-world environments.

FaceNet has a number of advantages over other facial recognition models. First, it can recognize faces with high accuracy even under difficult conditions such as changes in lighting, position, and facial expressions. Second, it is designed to work with a database of recognized faces, making it perfect for security and surveillance applications. Finally, because of a siamese neural network architecture, the model is able to perform well even with minimal training data, which is a common problem in face recognition research.

This paper focuses on the development of a face recognition system that employs deep learning techniques, especially siamese neural networks, to achieve high accuracy and reliability in face recognition. One of the objectives of this research is to investigate the efficiency of the FaceNet model, which is based on a siamese neural network architecture, in detecting faces in real-world environments. The method is to fine-tune the pre-trained FaceNet model and evaluate its performance on the Labeled Faces in the Wild (LFW) dataset.

The structure of this text consists of an introduction, overview of the deep learning models for face recognition, methodology for applying models and datasets, results analysis and implementation, and conclusion. First section covers the siamese neural network architecture, FaceNet model, theirs advantages and usage. Second section describes relevant datasets used in face recognition research, the implementation of the FaceNet model and the fine-tuning process on the LFW dataset. Third section presents the evaluation of the FaceNet model on the LFW dataset, provides insight into the performance of the model, and describes the implementation of the FaceNet model in face recognition system. Finally, the conclusion summarizes the findings and outlines potential future directions for research in this area.

Section 1. Overview of the deep learning models for face recognition

1.1. Description of Siamese model architecture

With advances in deep learning and its numerous applications, it is critical to investigate and improve the performance of face recognition systems. In this regard, the use of siamese neural networks has demonstrated promising results in one-shot learning scenarios with minimal training data. It is possible to extract discriminative features and make reliable comparisons between input photos by exploiting the common weights and feature representations in siamese networks. This approach opens new opportunities for enhancing the accuracy and effectiveness of face recognition systems by tackling the obstacles given by lighting, occlusion, and position fluctuations.

"Siamese Neural Networks for One-shot Image Recognition" is a research paper published in 2015 by Koch et al [2]. The paper proposes an architecture for one-shot image recognition using siamese neural networks. The siamese network architecture has proven effective in tasks like signature verification and facial recognition, where the network is trained to distinguish between two input images based on their similarity.

The architecture of the siamese network is built of two similar sub-networks that share weights and learn identical feature representations from the input images. Each sub-network generates a low-dimensional feature vector that represents the input image. The distance between these feature vectors is calculated, and a threshold is applied to determine whether the two input images are similar. The siamese network learns to minimize the distance between feature vectors for input images of the same class while maximizing the distance for input images of different classes during training. Figure 1 describes the architecture in more detail.

One of the main advantages of siamese neural networks is their ability to perform well in one-shot learning scenarios, where only one or a few examples of a class are available for training. This is achieved by learning a low-dimensional representation of the input images that captures the essential features necessary for discrimination. In addition, siamese networks can be used for various applications, such as face recognition, object recognition, and signature verification.

Siamese neural networks have been successfully applied in various face recognition applications. For instance, Schroff et al. introduced the FaceNet model, which uses a siamese network to learn a similarity metric between pairs of face images [3]. On various benchmark datasets, FaceNet achieved state-of-the-art accuracy in face recognition.

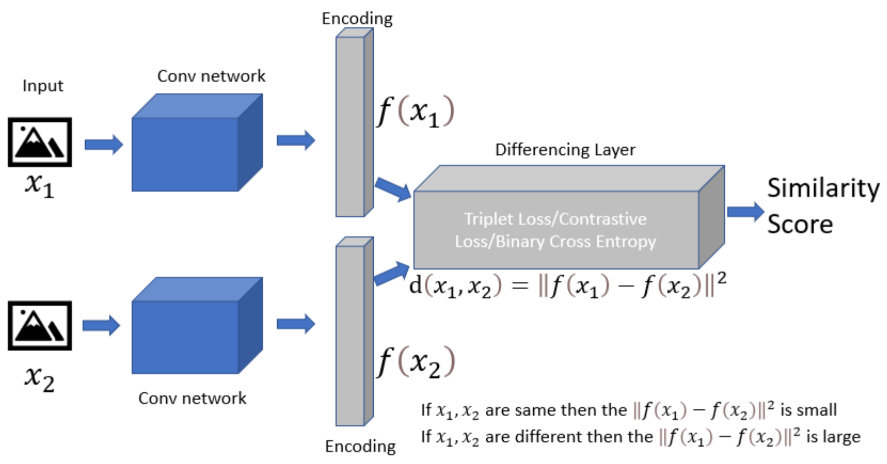

Encoding
Input
Conv network
$f(x_1)$
Differencing Layer
Similarity Score
x_1
Triplet Loss/Contrastive Loss/Binary Cross Entropy
$d(x_1, x_2) = \|f(x_1) - f(x_2)\|^2$
x_2
$f(x_2)$
Conv network
If x_1, x_2 are same then the $\|f(x_1) - f(x_2)\|^2$ is small
If x_1, x_2 are different then the $\|f(x_1) - f(x_2)\|^2$ is large
Encoding

Figure 1. Siamese model architecture [7]

1.2. FaceNet: an improved variation of the Siamese model

The FaceNet model uses a siamese neural network architecture with two main components: a deep convolutional neural network (CNN) for feature extraction and a triplet loss function for training as described in Figure 2. The CNN is based on the Inception architecture, which was designed to balance computational efficiency and model accuracy.

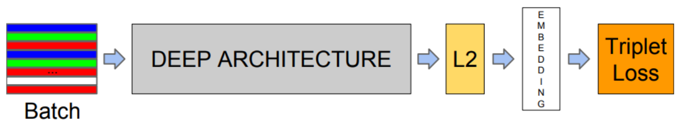


Figure 2. FaceNet model structure [3]

The triplet loss function used in FaceNet is a variation of the contrastive loss function, which is commonly used in siamese networks for image similarity tasks. The loss is calculated as $L = \sum_i^N [\|f(x_i^a - x_i^p)\|_2^2 - \|f(x_i^a - x_i^n)\|_2^2 + \alpha]$, where the function minimizes the distance between the embeddings of two images of the same person (anchor x_i^a and positive x_i^p) and maximizes the distance between the embedding of an anchor image and a different person's image (negative x_i^n) to ensure that an image of a specific person is closer to all other images of the same person than it is to any image of any other person. The goal is to learn a feature representation that maps images of the same individual close together in the embedding space while keeping images of different individuals far apart as shown in Figure 3.

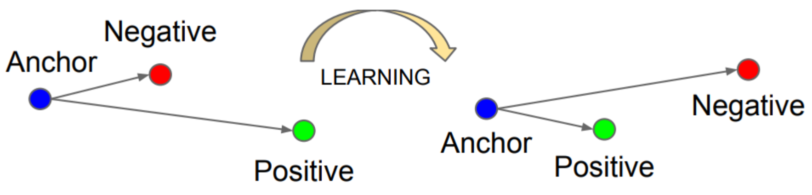
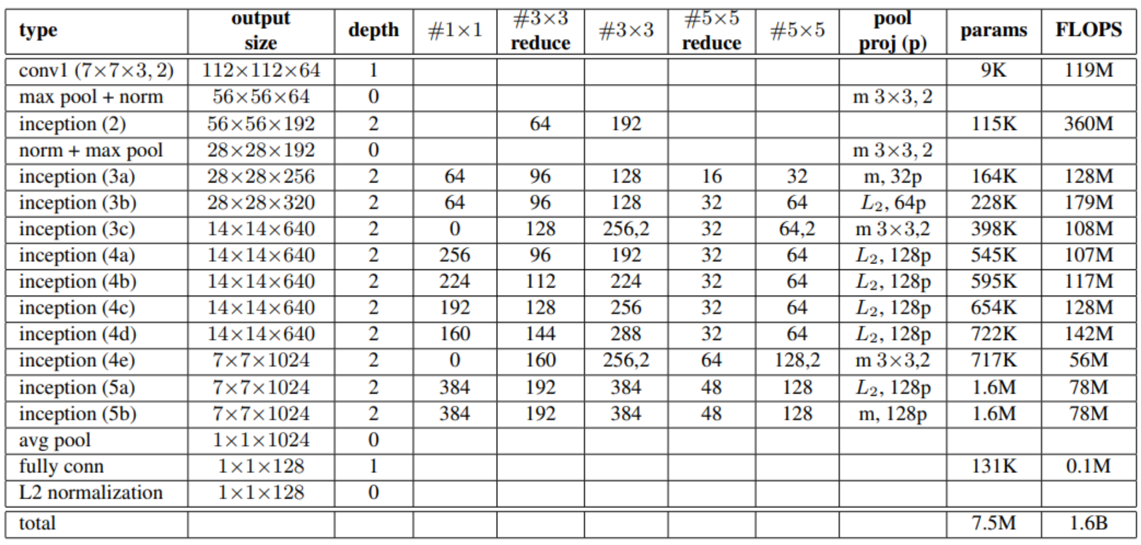


Figure 3. The Triplet Loss function [3]

The Inception architecture used in FaceNet consists of a sequence of convolutional and pooling layers followed by multiple inception modules. An inception module consists of multiple convolutional filters of different sizes followed by max-pooling and concatenation of the resulting feature maps. This design enables the network to capture features at different scales and resolutions. Schroff et al. compared two types of architectures in their FaceNet model, and their analysis showed that the NN2 architecture had better trade-off results. The architecture of this model is described in more detail in Table 1.



type	**output size**	**depth**	#1×1	#3×3 **reduce**	#3×3	#5×5 **reduce**	#5×5	**pool proj (p)**	**params**	**FLOPS**
conv1 (7×7×3, 2)	112×112×64	1							9K	119M
max pool + norm	56×56×64	0						m 3×3, 2		
inception (2)	56×56×192	2		64	192				115K	360M
norm + max pool	28×28×192	0						m 3×3, 2		
inception (3a)	28×28×256	2	64	96	128	16	32	m, 32p	164K	128M
inception (3b)	28×28×320	2	64	96	128	32	64	L_2, 64p	228K	179M
inception (3c)	14×14×640	2	0	128	256,2	32	64,2	m 3×3,2	398K	108M
inception (4a)	14×14×640	2	256	96	192	32	64	L_2, 128p	545K	107M
inception (4b)	14×14×640	2	224	112	224	32	64	L_2, 128p	595K	117M
inception (4c)	14×14×640	2	192	128	256	32	64	L_2, 128p	654K	128M
inception (4d)	14×14×640	2	160	144	288	32	64	L_2, 128p	722K	142M
inception (4e)	7×7×1024	2	0	160	256,2	64	128,2	m 3×3,2	717K	56M
inception (5a)	7×7×1024	2	384	192	384	48	128	L_2, 128p	1.6M	78M
inception (5b)	7×7×1024	2	384	192	384	48	128	m, 128p	1.6M	78M
avg pool	1×1×1024	0								
fully conn	1×1×128	1							131K	0.1M
L2 normalization	1×1×128	0								
total									7.5M	1.6B

Table 1. NN2 architecture used in FaceNet model [3]

To evaluate the performance of the FaceNet model, Schroff et al. used several benchmark face recognition datasets, including Labeled Faces in the Wild (LFW). The evaluation metrics used were accuracy, validation rate (VAL) and false accept rate (FAR). All faces pairs (i,j) of the same identity are denoted with P_{same}, whereas all pairs of different identities are denoted with P_{diff}. Schroff et al. defined the set of all true accepts and false accept as $TA(d) = \{(i,j) \in P_{same}, with\ D(x_i, y_i) \leq\ d\}$, $FA(d) = \left\{(i,j) \in P_{diff}, with\ D(x_i, y_i) \leq\ d\right\}$.

These are the face pairs (i,j) that were correctly and incorrectly classified at threshold d. The validation rate VAL(d) and the false accept rate FAR(d) for a given distance d are then defined as $VAL(d) = \frac{|TA(d)|}{|P_{same}|}, FAR(d) = \frac{|FA(d)|}{|P_{same}|}$.

In conclusion, Siamese neural networks have demonstrated promising results in a variety of applications, including face recognition, object recognition, and signature verification, because of their propensity to perform well in one-shot learning scenarios. The FaceNet model, an improved version of the Siamese model, uses a siamese neural network architecture for feature extraction and a triplet loss function for training. FaceNet achieved state-of-the-art accuracy in face recognition on a variety of benchmark datasets.

Section 2. Methodology for applying models and datasets

2.1. Description of the LFW dataset

To evaluate the performance of face recognition models, researchers often rely on benchmark datasets that accurately represent real-world scenarios. One such dataset is the Labeled Faces in the Wild (LFW) dataset, which has emerged as a widely adopted resource in the field of face recognition research.

The Labeled Faces in the Wild (LFW) dataset is a benchmark dataset for face recognition research. It contains more than 13,000 face images of 5,749 individuals collected from the web. Figure 4 provides an example of its contents. The dataset includes variations in pose, lighting, expression, and occlusion, making it challenging for face recognition algorithms. The LFW dataset has been widely used to evaluate the performance of face recognition models, including traditional and deep learning-based approaches.

The LFW dataset was first introduced in 2007 by Huang et al. in their paper "Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments." [6] Since then, it has been used in numerous face recognition research studies and competitions, including the Face Recognition Vendor Test (FRVT) and the Face Recognition Grand Challenge (FRGC).

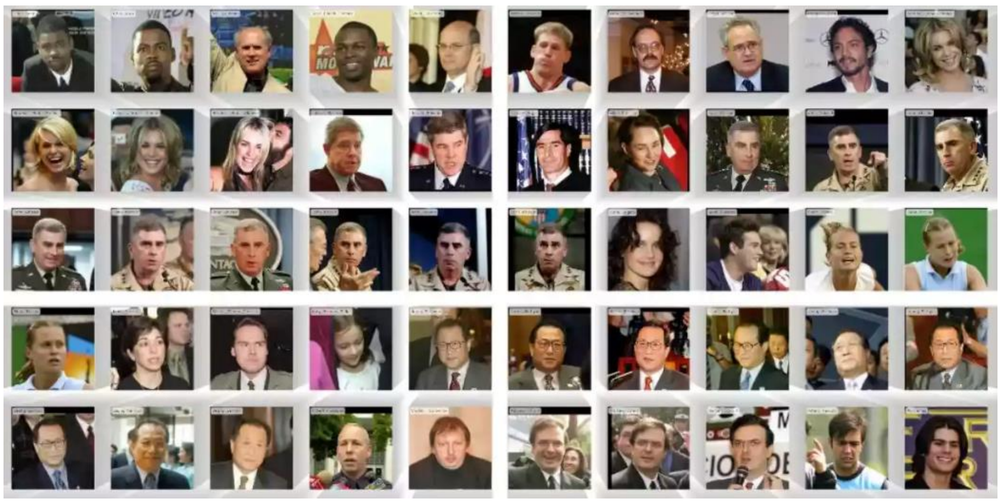


Figure 4. Visualization of the LFW dataset [4]

The LFW dataset provides a realistic evaluation of the performance of face recognition algorithms in unconstrained environments, where variations in pose, lighting, expression, and occlusion are common. Moreover, the dataset has a large number of individuals and images, making it suitable for training and testing deep learning-based models, which require a significant amount of data.

2.2. Description of the VGGFace2 dataset

In this study, the pretrained FaceNet model used for fine-tuning was initially trained on the VGGFace2 dataset. The VGGFace2 dataset contains around 3.3 million photos of 9,131 people, making it a large-scale face recognition dataset. Figure 5 provides an example of its contents. The dataset was created by the University of Oxford's Visual Geometry Group (VGG) for use in research on face recognition using deep learning algorithms. The collection contains images collected in a range of scenarios, including various positions, expressions, and lighting conditions, making it challenging for face recognition algorithms.

The VGGFace2 dataset was first introduced in 2018 by Qiong Cao et al. in their paper "VGGFace2: A dataset for recognising faces across pose and age."[8] The dataset is an upgraded version of the original VGGFace dataset, which was released in 2015 and contained around 2.6 million images of 2,622 individuals. The VGGFace2 dataset was collected from a wide range of sources, including the internet, movies, and TV shows, and is annotated with identity labels, gender, and age.

The VGGFace2 dataset has been widely used to evaluate the performance of deep learning-based face recognition models, including the FaceNet model. The dataset's large size and variability make it suitable for training and testing deep learning-based models, which require a significant amount of data to learn accurate representations of facial features. The dataset has been used to evaluate the performance of face recognition models across different poses, ages, and ethnicities, and has contributed to the development of more accurate and reliable face recognition systems.

Figure 5. Visualization of the VGGFace2 dataset [8]

2.3. Used method

The original FaceNet model was initialized randomly and trained on a CPU cluster for 1,000 to 2,000 hours. It contains 7.5 million parameters and requires 1.6 billion FLOPS for training. However, due to the computational cost and time required, it was decided to use the fine-tuning method to speed up the learning process and reduce the required computational resources.

Fine-tuning is a deep learning technique for adapting a pre-trained model to a new task or dataset. The technique is taking a pre-trained model that has been developed on a large dataset and using it as a starting point for training on a new dataset. The goal of fine-tuning the model is to increase its performance on its new task while minimizing overfitting.

One of the earliest works on fine-tuning in deep learning was by Yosinski et al. in their 2014 paper "How transferable are features in deep neural networks?"[9] The authors investigated the transferability of features learned in pre-trained deep neural networks across different tasks and datasets. They found that fine-tuning the pre-trained models on new datasets improved their performance compared to training from scratch.

Compared to learning from scratch, fine-tuning a pre-trained model can improve the generalization of the model, making it more robust to variations in the new dataset. For this work was chosen a pre-trained FaceNet model trained on the VGGFace2 dataset. Through the process of fine-tuning the pre-trained model on the Labeled Faces in the Wild dataset, the aim was to enhance the accuracy of the model.

The dataset was pre-processed using cv2.CascadeClassifier and haarcascade_frontalface_default.xml from OpenCV library. The cv2.CascadeClassifier is a tool used for object detection in images or videos. It is commonly used for detecting faces in images and videos. The haarcascade_frontalface_default.xml is a pre-trained classifier that is specifically designed to detect frontal faces in images. The pre-trained classifier is widely used in various face detection applications due to its high accuracy and efficiency. These tools were used to crop detected faces for improving models' accuracy and speeding up the learning process.

A total of 234,406 pairs of anchor, positive, and negative images were generated from the dataset. Figure 6 provides an example of these pairs. The model was validated every 10 batches on a separate 1000 images using the mini-batch learning. The threshold for the performance metrics was determined experimentally.

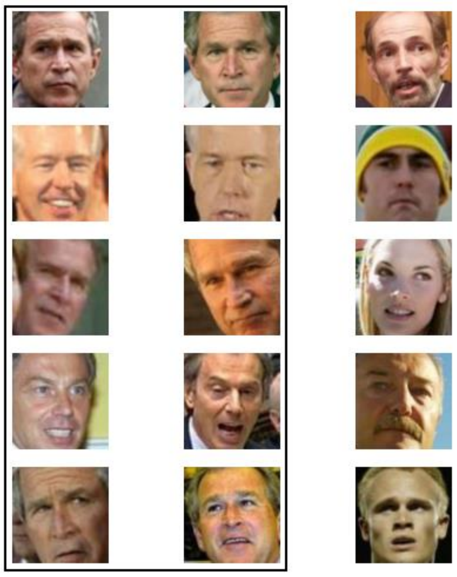


Figure 6. Visualization of the generated pairs generated from the LFW dataset [6]

The FaceNet model architecture and pre-trained weights on the VGGFace2 dataset developed by Tim Esler et al. [10] were used in this study. The optimization algorithm Adam was selected with a learning rate of 0.0001 and a batch size of 100. The triplet loss function was used with a margin of 0.2 and Euclidean distance. Described implementation was done in PyTorch.

Section 3. Results analysis and implementation

3.1. Training results

FaceNet model was fine-tuned using the methods described in Section 2.3. Figure 7 shows the average distance between embeddings calculated after each batch for the training and validation sets. The pre-trained model is able to easily distinguish the faces of different people since the negative distance did not increase during training.

The fluctuation amplitude of the validation set may be explained by the small size of the validation set and the dropout value embedded in the model, which can add some erratic behavior. Despite this, it is observed that the distance between images of the same person decreases during training, which can increase the accuracy of the model and enable setting a lower threshold in the future to avoid unwanted misclassification.

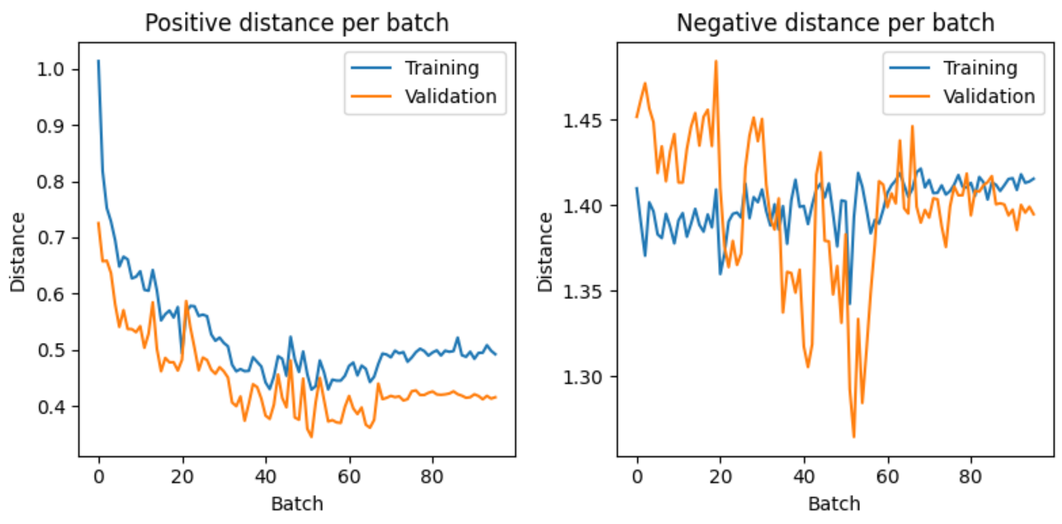


Figure 7. Averaged distances per batch

Figure 8 displays the metrics described in Section 1.2. The distance between identical faces is consistently below the threshold 1, indicating a 98% probability of correct classification. However, the plot of FAR (False Acceptance Rate) values shows that the model is misclassifying at a rate of 18%, which suggests that a lower threshold is required for further analysis.

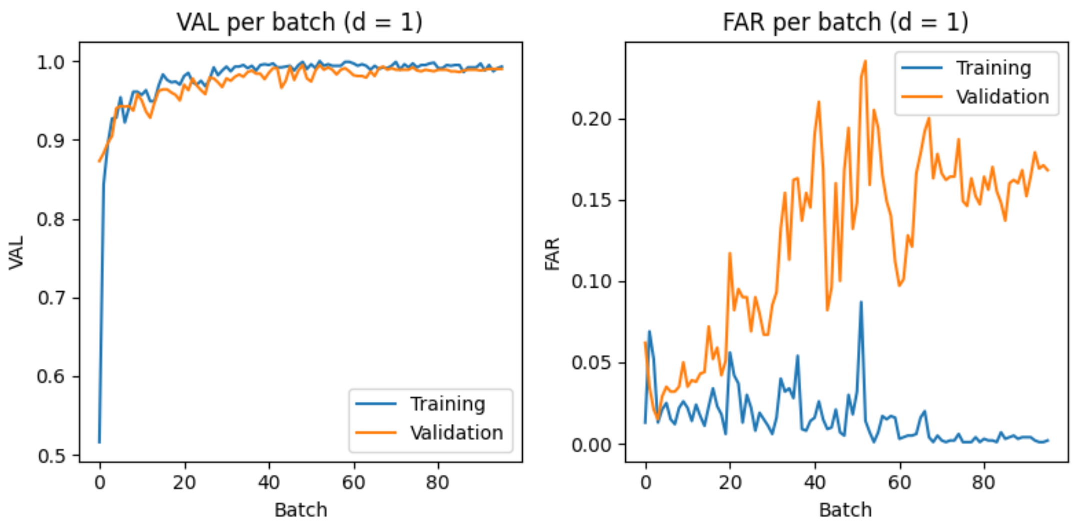


Figure 8. Averaged VAL and FAR metrics calculated per batch with threshold = 1

Figure 9 shows the density graphs of the positive and negative distances. These graphs were plotted to visualize the distribution of distances for positive and negative pairs. Upon examination, it was observed that the density graphs intersected at a value of 0.74. This specific threshold value was chosen to minimize the chances of misclassifying pairs. By selecting this threshold, the aim is to find a balance between correctly categorizing positive pairs (below the threshold) and correctly categorizing negative pairs (above the threshold).

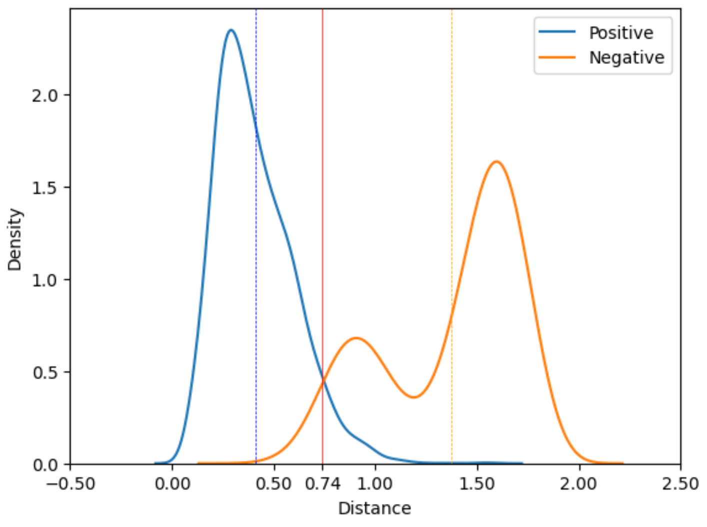


Figure 9. Distance distribution calculated between embeddings of fine-tuned model

The calculated metrics for the original model and the fine-tuned model are presented in Table 2, displaying the results. This table provides an overview of the performance measures obtained for each model.

	Original model	Fine-tuned model
Loss	0.02	0.01
Averaged positive distance	0.73	**0.41**
Averaged negative distance	1.26	1.38
VAL	0.89	0.94
FAR	0.05	0.04

Table 2. Calculated evaluation metrics of the original and fine-tuned models

Figure 10 illustrates the variation in distance between the same initial photo captured under normal lighting conditions and poor lighting conditions implemented in the face recognition system in Section 3.3. It is evident that the base model is more susceptible to noise, resulting in a distance increase of 0.61 compared to the new model, which exhibits a smaller distance of 0.27. The results correspond to the calculation of averaged positive distances, showing an improvement of 0.32 in the new model compared to the base model presented in Table 2. This indicates that the new model performs better in terms of capturing and measuring similarities between faces under varying lighting conditions.

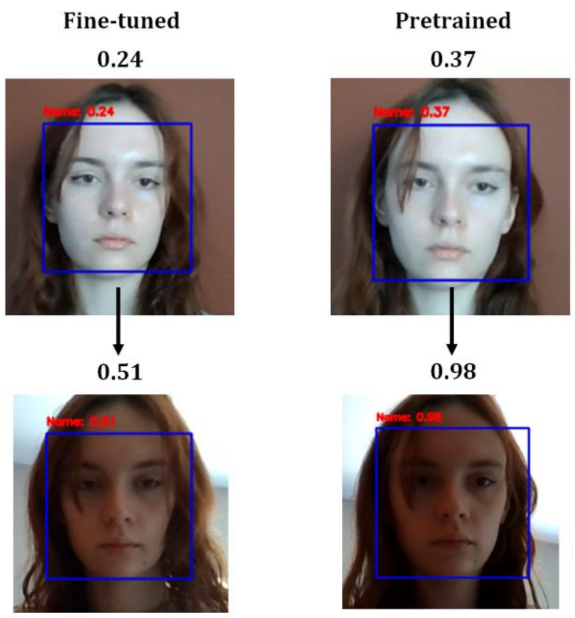


Figure 10. Distances under normal and poor lighting of pretrained and fine-tuned models

3.2. Face recognition pipeline

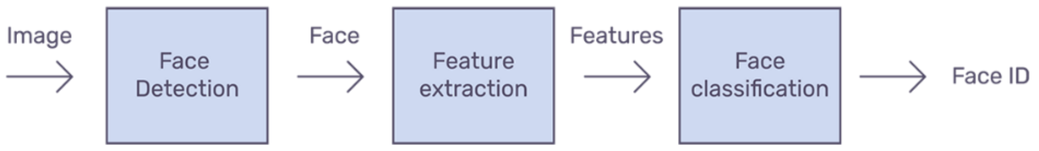


Figure 11. Face recognition pipeline [5]

Figure 11 shows the steps involved in the typical face recognition system:

Face detection: In this step, the system uses an algorithm to locate and extract faces from an image or video stream. One common technique for face detection is to use a convolutional neural network (CNN) that has been trained to recognize faces. Once the CNN has been trained, it can be used to identify the presence of faces in new images or video frames.

The face detection step is done using OpenCV's cv2.CascadeClassifier and the haarcascade_frontalface_default.xml file. This file contains a pre-trained Haar classifier model that is used to detect frontal faces in an image or video stream as shown in Figure 12. These preprocessing tools were used to prepare the LFW dataset for fine-tuning the FaceNet model.

Feature extraction: After the faces have been detected, the system extracts a set of features from each face. The network is trained to map each face image to a point in a high-dimensional space such that faces from the same person are mapped to similar points, while faces from different people are mapped to dissimilar points.

Once a face is detected, the implemented system extracts the detected face region and resizes it to 224x224 pixels. The resized face image is then passed to the FaceNet model to extract a 128-dimensional embedding vector for the face.

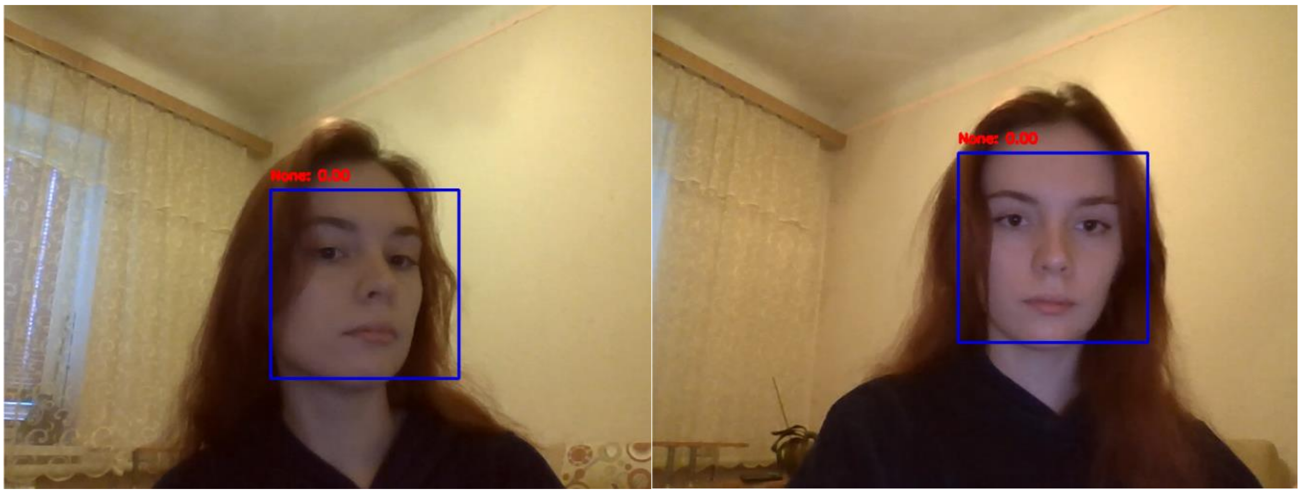


Figure 12. Face detection using OpenCV's cv2.CascadeClassifier

Face matching (classification): Once the embedding vectors have been extracted, the system compares them to a database of known faces to identify the person in the image. The system computes the distance between the embedding vector of the query face and the embedding vectors of the faces in the database using a distance metric such as Euclidean distance or cosine similarity. If the distance between the two vectors is below a certain threshold, the system considers the faces to be a match.

The get_identity() function is used to compare the embedding vector of the detected face with those in the system's known faces. The function computes the distance between the two vectors using the nn.PairwiseDistance() function, and returns the name of the matched face if the distance is below a threshold.

Face database management: In order to recognize faces, the system needs to maintain a database of known faces. This involves storing the embedding vectors of each face along with metadata such as the person's name, ID, and any other relevant information. The system also needs to be able to add new faces to the database.

The add_unknown_faces() function is called when the "Scan" button is clicked. It loops over all the unknown faces that were detected in the camera feed and creates a FaceFrame instance for each one. The FaceFrame class is a separate Tkinter window that displays the image of the unknown face and prompts the user to enter a name for the face. Once the user enters a name, the face image is saved, and the embedding vector is added to the system.

3.3. System description

Face recognition system implemented in Python using the Tkinter library for the graphical user interface (GUI). The system is composed of two main components: the camera interface and the user list interface. The camera interface displays a live video feed from the user's camera and uses the OpenCV library to detect faces in the video stream.

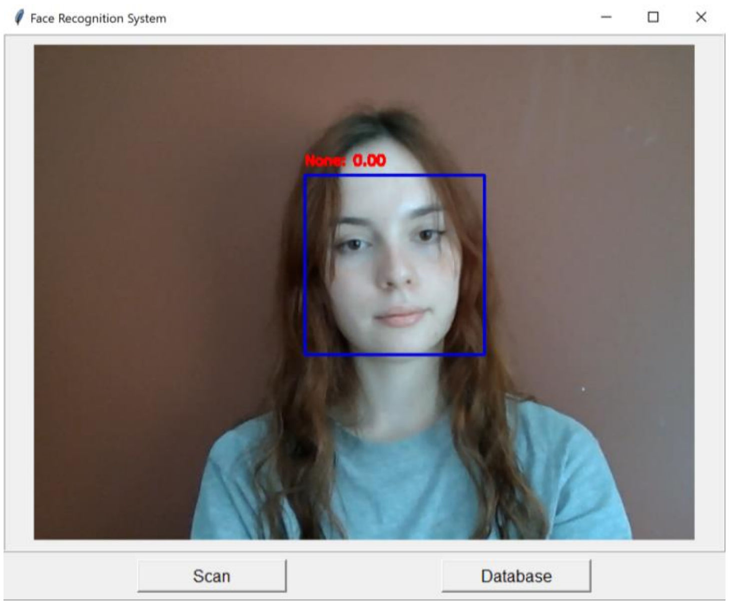


Figure 13. The camera interface

The user can add new faces to the system by clicking the "Scan" button. If the face is not already saved and recognized, the user will be prompted to enter a name for the new face. The new face is then saved to the JSON file with the entered name and a feature vector computed using the fine-tuned FaceNet model. After saving the system automatically detects face and displays both the name and the distance.

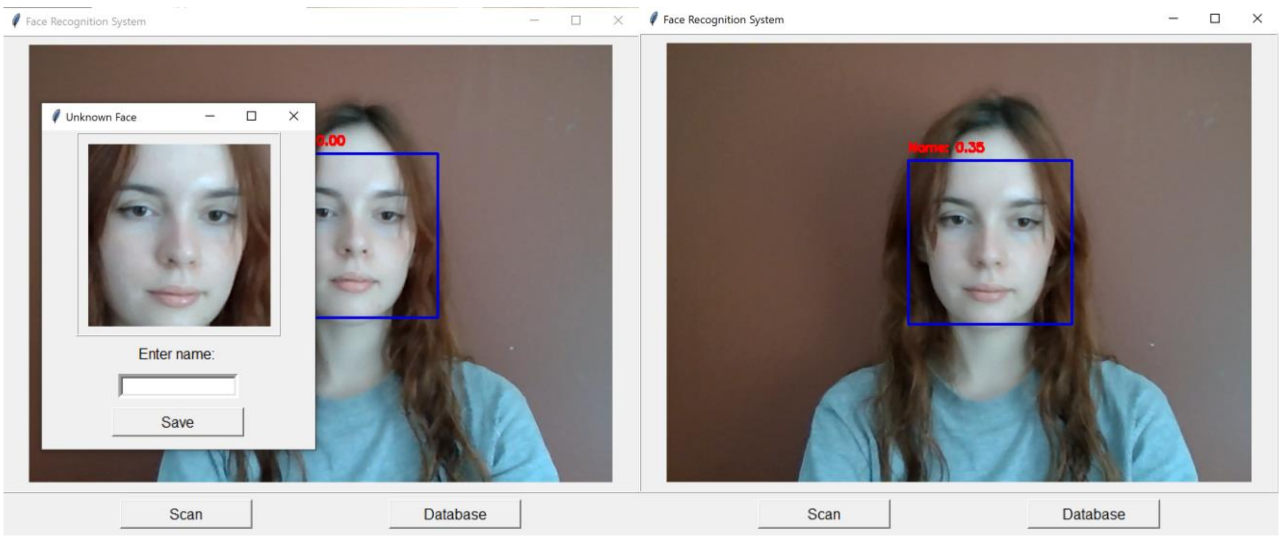


Figure 14. The window for saving detected unknown face

The user list interface allows the user to view the faces that have been saved in the system. The user can view a list of names of the saved faces and click on a name to view the corresponding face image. Additionally, the user can delete chosen faces from the system or clear all the saved information.

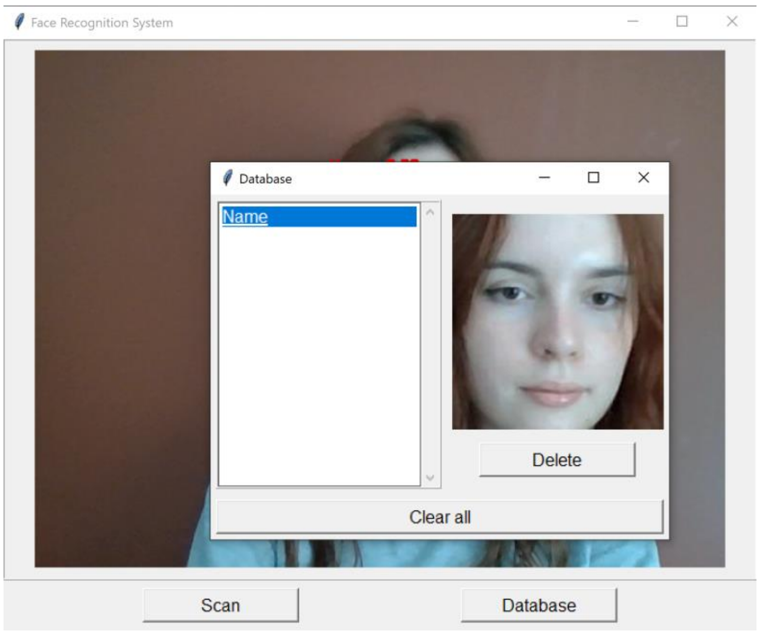


Figure 15. The user list interface

When the system recognizes multiple faces and identifies an unknown one, it will prompt the user to enter a name and save their face in the system. After this, the system will be able to distinguish the saved faces and display the corresponding distances with names, as shown in the Figure 16.

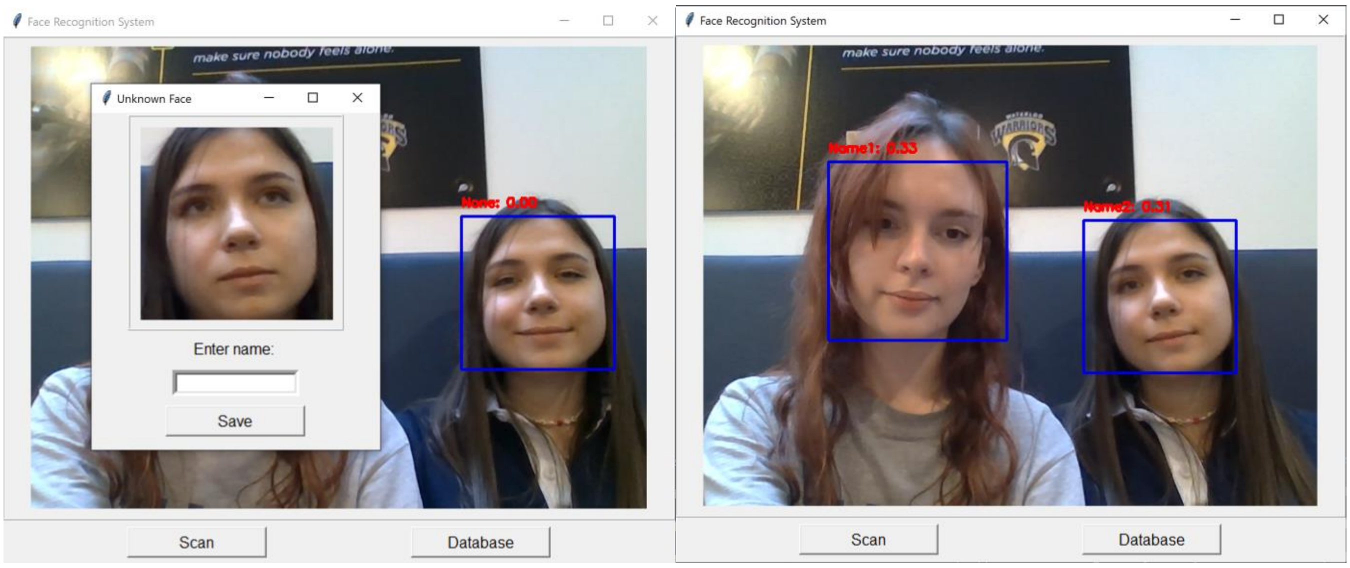


Figure 16. The system detects unknown face

The user can exit the program by clicking the close button on the window. The user data is automatically saved to a JSON file when the program is closed.

Conclusion

Face recognition systems are becoming increasingly significant in today's world due to its multiple uses in security, identity, and authentication systems. They have the ability to replace traditional authentication techniques like passwords and PIN codes with a more dependable and secure alternative. Furthermore, facial recognition technology has the potential to greatly improve public safety surveillance and monitoring systems.

Face recognition systems can be used in a variety of ways, including mobile applications, cameras, and kiosks. Some companies, such as Apple's Face ID and Facebook's photo tagging tool, are already using face recognition technology in their goods and services.

It's essential to increase the reliability of face recognition systems to lessen the possibility of false positives and negatives, which can have negative consequences. A false positive is when the system wrongly detects a person, whereas a false negative is when the system fails to accurately identify a person. False positives can lead to unjust arrests or detentions, while false negatives can result in security breaches and other problems.

The main objective of this work was to improve the accuracy of a pre-trained face recognition model on a new dataset. Fine-tuning the pre-trained model offered improvements with less used computational resources. The efforts to improve the accuracy of the face recognition system resulted in a reduction in the distance between similar faces, leading to more precise recognition. Moreover, there was a slight increase in the distance between faces of different individuals, enhancing the system's ability to differentiate between faces.

The implemented model showcased its capability to recognize faces even in challenging conditions such as poor lighting, surpassing the newly established threshold for accuracy. However, for further advancements, it is recommended to incorporate additional features like face alignment and noise reduction.

By incorporating face alignment techniques, the system can ensure consistent and standardized face positioning, minimizing variations caused by different poses or angles. Additionally, noise reduction techniques can help eliminate unwanted artifacts or disturbances, leading to clearer and more precise facial representations.

With enhanced accuracy and robustness, the system will excel in real-world applications such as surveillance, access control, and identity verification, making it more reliable and effective in ensuring security and privacy. As technology advances, further research and development will likely introduce more innovative techniques and algorithms. This ongoing progress will continue to shape the future of face recognition systems, ultimately providing even more precise, efficient, and versatile solutions for a wide range of applications.

Literature references

1. Ivakhnenko, A. G., & Lapa, V. G. (1965). Cybernetic Predicting Devices. CCM Information Corporation.
2. Koch, G., Zemel, R., and Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. Proceedings of the 32nd International Conference on Machine Learning, 1302-1310.
3. Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 815-823.
4. LFW Dataset (2023) Machine Learning Datasets. Available at: https://datasets.activeloop.ai/docs/ml/datasets/lfw-dataset/
5. Face recognition with FaceNet and MTCNN – Ars Futura. Available at: https://arsfutura.com/magazine/face-recognition-with-facenet-and-mtcnn/
6. Huang, G. B., Mattar, M., Berg, T., & Learned-Miller, E. (2008). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report, University of Massachusetts, Amherst.
7. Khandelwal, R. (2021) One-shot learning with Siamese network, Medium. The Startup. Available at: https://medium.com/swlh/one-shot-learning-with-siamese-network-1c7404c35fda
8. Cao, Q., Shen, L., Xie, W., Parkhi, O. M., & Zisserman, A. (2018). VGGFace2: A dataset for recognising faces across pose and age. In 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018) (pp. 67-74). IEEE.
9. Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In Advances in neural information processing systems (pp. 3320-3328).
10. Esler, T. et al. (2019) Timesler/facenet-pytorch: Pretrained Pytorch Face Detection (MTCNN) and facial recognition (InceptionResnet) models, GitHub. Available at: https://github.com/timesler/facenet-pytorch