Enhancing Facial Recognition Using Advanced Deep Learning Techniques

Abstract

Facial recognition systems have gained considerable traction in various applications, including security, authentication, and human-computer interaction. This paper meticulously outlines the design and implementation of a facial recognition system that seamlessly integrates face detection and recognition. The system employs Haar cascades for precise face detection, followed by Eigenfaces for dimensionality reduction and a Support Vector Machine (SVM) for robust classification. Extensive preprocessing techniques ensure quality inputs, and a comprehensive evaluation strategy guarantees reliable performance. The design's strength lies in its flexibility, scalability, and attention to detail, promising a significant contribution to the evolving field of facial recognition. Extensive testing on the system has been performed, a variety of dataset compositions, the inclusion or absence of face detection, and whether SMOTE resampling provides an advantage in the system have been considered and tested. The system shall be able to verify a person’s face against complete strangers to the system, or, put simply, “You” vs “Not You.”

**Keywords**: Facial Recognition, Face Detection, Haar Cascades, Eigenfaces, Principal Component Analysis (PCA), Support Vector Machine (SVM), Image Preprocessing, Dimensionality Reduction, Machine Learning, Classification, Synthetic Minority Oversampling Technique (SMOTE) Labeled Faces in The Wild (LFW)

# Introduction

Facial recognition has become a cornerstone technology in modern society, with applications ranging from security and surveillance to personalized user experiences. The intersection of machine learning, computer vision, and advanced algorithms allows for the development of systems that can recognize and interpret human faces with remarkable accuracy.

## Background

As facial recognition technology continues to advance, understanding its roots and evolutionary trajectory becomes vital. This section delves into the foundation of facial recognition, exploring both its technological underpinnings and the context in which it has emerged and thrived.

### Historical Context

Facial recognition, the computational and technological process of identifying or verifying an individual's identity using their face, has its roots in the mid-20th century, with the advent of computer science. Initially focusing on simple geometric measurements and patterns, the field has since advanced to utilize sophisticated machine learning and deep learning algorithms.

### Modern Applications

In today's rapidly advancing technological landscape, facial recognition has found extensive applications across various domains:

* **Security and Law Enforcement**: It aids authorities in criminal investigations, airport security checks, and border control.
* **User Authentication**: It provides a secure way for users to unlock devices or access restricted areas.
* **Personalized Marketing**: Businesses use this technology to offer customized experiences to customers by recognizing their preferences.
* **Healthcare**: Medical professionals are exploring facial recognition for patient identification and even potential diagnostic applications.

### Evolution of The Field

The growth of facial recognition has been heavily influenced by two key factors:

1. **Data Availability**: The proliferation of large, publicly available datasets has enabled researchers to train more complex models. In the case of this research, we have the Labeled Faces in the Wild dataset. A publicly available and easily obtainable dataset of labeled images of faces.
2. **Algorithm Advancements**: The shift from traditional image processing techniques to deep learning has revolutionized the field. Neural networks have been instrumental in automating feature extraction, contributing to the significant improvement in accuracy and efficiency [3].

## Objective and Scope

### Research Objectives

The main goal of this research is to design, implement, and evaluate a cutting-edge facial recognition system. The focus will be on integrating both traditional methods and advanced deep learning techniques to achieve a comprehensive understanding of facial features.

### Specific Goals:

1. **System Design**: To conceptualize a robust system that combines various algorithms for feature extraction and recognition.
2. **Algorithm Selection**: To explore different methodologies, including Eigenfaces, Convolutional Neural Networks (CNNs), and Support Vector Machines (SVMs), and select the most suitable combination.
3. **Algorithm Optimization**: Show comparison of results between different methods for preprocessing data, providing the algorithm features, and hyperparameter tuning, and make conclusions about the best configuration.
4. **Data Handling**: To gather, prepare, and augment a diverse dataset that represents various human faces with different expressions, angles, lighting conditions, and ethnic backgrounds.
5. **Training and Evaluation**: To implement an effective training strategy and conduct a thorough performance evaluation, encompassing aspects like accuracy, scalability, and real-time responsiveness.

### In Scope

The research will concentrate on creating a system that excels in face verification, where the objective is to confirm or deny the identity claim of an individual. This focus aligns with the vision articulated by Taigman et al. in DeepFace, striving to bridge the gap between machine and human-level performance in this task [1]. The research seeks to understand the limitations of the algorithm and how best to optimize the algorithm in the following ways:

1. **Impact of Data Composition:** In optimization testing, the composition of the data that will subsequently be used for training and testing will be controllable to some extent. The system has been tested with a variety of composition weights of the “You” vs “Not You.”
2. **Face Detection Usage:** This research tests whether inclusion of face detection during preprocessing makes a meaningful difference in performance. Face detection will be optionally toggled on and off in the algorithm. Whether the raw image or the image of the detected faces go through the system may have an impact on the PCA portion of the algorithm, thus impacting performance of the overall system.
3. **SMOTE Resampling:** Applying SMOTE Resampling to the system, data augmentation for the purpose of synthesizing more examples of the minority class in the data set, may have an impact on performance. This research aims to determine that impact.

### Out of Scope

The following items are out of scope for this research.

1. **Comparison of Algorithms:** The aforementioned “Algorithm Selection” will be based upon research into previous work in the field, this paper will not make assertions about which model performs better or are easier to work with in a practical sense.
2. I**mplementations Beyond Testing:** The trained and tested model will not be used, with regard to what is shown in this research, for implementations beyond testing the system. For example, face verification using a phone camera. This research is targeted at testing outcomes in performance under

## Summary

The ever-growing field of facial recognition offers immense potential and poses unique challenges. This research aims to contribute meaningfully to the field by marrying traditional methods with modern deep learning techniques. In doing so, it hopes to provide insights into creating a system that is not only technically advanced but also cognizant of the broader societal and ethical implications. By thoroughly understanding the past work and meticulously designing the experiment, the research sets the stage for a promising exploration into the frontiers of facial recognition technology.

# Previous Work

## Traditional Approaches

Earlier approaches laid the groundwork for understanding facial features and their geometric and statistical relationships:

1. **Eigenfaces (Turk and Pentland)**: The Eigenfaces method introduced a way to represent faces using principal component analysis (PCA). By capturing the variance between facial images and representing them with a set of principal components or eigenfaces, this method allowed for efficient recognition. However, it struggled with variations in lighting and pose [4].
2. **Fisherfaces (Belhumeur et al.)**: Building on the Eigenfaces method, Fisherfaces utilized linear discriminant analysis (LDA) to maximize between-class variations while minimizing within-class variations. This enhanced discrimination between different faces, improving recognition accuracy [13].
3. **3D Models (Bowyer et al.)**: By employing 3D models, researchers attempted to address the limitations of 2D recognition, such as pose variations. This research led to more robust systems that could understand facial structures in three dimensions [14].

## Deep Learning Revolution

The introduction of deep learning methods has significantly advanced the field, leading to innovative solutions:

1. **DeepFace (Taigman et al.)**: DeepFace marked a breakthrough by using 3D face alignment and a nine-layer deep neural network. The 3D alignment corrected for pose, illumination, and expression variations, while the deep network learned a compact representation of faces. This approach dramatically reduced error rates in face verification [1].
2. **FaceNet (Schroff et al.)**: FaceNet extended facial recognition by directly learning a mapping from face images to a compact Euclidean space. Using a triplet loss function, FaceNet ensured that similar faces were closer in the embedded space, achieving impressive results on various benchmarks [2].
3. **Deep Face Alignment (Yang et al.)**: This study empirically evaluated several face alignment methods, showing that proper alignment can dramatically improve recognition accuracy. The authors demonstrated the effectiveness, in the form of an alignment sensitivity analysis, of both two-stage and cascaded methods for face alignment [6].
4. **Imagenet Classification (Krizhevsky et al.)**: The source introduces several innovations and techniques to improve the performance and efficiency of the CNN, such as using rectified linear units (ReLU) as activation functions, using dropout as a regularization method, using overlapping max-pooling layers, and using data augmentation to increase the size and diversity of the training set [8].
5. **Joint Identification-Verification (Sun et al.)**: This source is relevant for our project because it shows how deep learning can be used to learn effective face representations that are invariant to age, pose, expression, and illumination. The paper proposes a deep learning approach for face representation that uses both identification and verification signals as supervision. The paper introduces a novel network architecture that consists of two sub-networks: one for identification and one for verification. The two sub-networks share the same convolutional layers but have different fully connected layers [12].

## Other Relevant Works

1. **Gradient-based learning (LeCun et al.)**: The source provides a comprehensive overview of the gradient-based learning methods applied to document recognition, such as optical character recognition (OCR), handwritten digit recognition, and text categorization. This source also provides us with some useful references to other related works and datasets that we can explore further [7].

## Summary

The rich body of previous work in facial recognition spans a wide array of methods and considerations. From foundational geometric and statistical approaches to cutting-edge deep learning models, the evolution of the field reflects an ongoing dialogue between theory, technology, and ethics. This diverse landscape informs the current research, underscoring both the remarkable progress made and the opportunities for future innovation and refinement.

# Experiment Design

## Introduction

The field of facial recognition has witnessed tremendous advancements, yet building a system that is robust, efficient, and adaptable requires a careful orchestration of various components. This research is tailored to design a facial recognition system that intricately integrates machine learning models and image processing techniques. The system's architecture is founded on a deep analysis of current practices, challenges, and emerging trends in the field. The design is aimed at a balance between computational efficiency and the ability to handle real-world variations. The section that follows offers a detailed roadmap of the research design, exploring each element with precision and contextual relevance.

In the process of developing the scope of the research, CNNs, SVMs, as well as Eigenfaces and PCA were all considered. After researching extensively, it was decided that the more modern SVM should be combined with PCA. The features outputted by the PCA are then fed into the SVM.

## Face Detection Module

The Face Detection Module is a pivotal component of the facial recognition system, responsible for identifying and locating human faces within images. This section explores the various techniques and algorithms used in face detection, with an emphasis on the application of Haar cascades.

### Algorithm and Haar Cascades

Haar cascades are a critical element within the face detection module. This sub-section delves into the algorithms used for face detection and the specific application of Haar cascades, providing an understanding of how they function within the system.

#### Applying OpenCV's Haar Cascades:

Haar cascades are machine learning models trained to detect objects for which they have been trained, using simple features. Originating from Viola-Jones detection algorithm, they've become popular for their efficiency. OpenCV's pre-trained Haar cascades, such as haarcascade\_frontalface\_default.xml, have been applied to detect faces within images. It works by sliding a window across the image and applying a series of binary feature classifiers to assess the presence of a face.

#### Tuning and Parameters:

Parameters like the scaling factor, minimum size, and neighbors can be adjusted for optimal detection. They define how the detection window scales and how many neighboring candidate rectangles should be retained. This system was found to detect faces reasonably well, in the data it was used on, with the following parameters: scaleFactor=1.1, minNeighbors=5, minSize=(400, 400).

## Face Data Sets Module

Face datasets form the bedrock of any facial recognition system. By having a diverse and appropriately processed dataset, the system ensures that predictions made on new, unseen data are accurate and reliable. The provided code offers a clear insight into how images are sourced, processed, and later used in a facial recognition system.

### Preprocessing

Given the variability in images, preprocessing is crucial to ensuring that the data fed into the system is consistent and free of noise. The preprocessing pipeline involves several steps:

#### Resizing:

* To ensure uniform processing across all images, they are resized to a consistent dimension, specifically to 47×62 pixels. Uniformity in size ensures that the model receives inputs of the same shape during both training and prediction.
* The **preprocess\_image** function in the code takes care of resizing. Whether the input is a direct image path or a numpy array, the function ensures that the image is resized to the target dimensions.

#### Color Conversion:

* While color images contain a wealth of information, for facial recognition, grayscale often suffices. Grayscale images not only reduce computational complexity but also ensure that the focus remains on structural features rather than color variations.
* Within the **preprocess\_image** function, the image is converted to grayscale using the **convert('L')** method from the PIL library.

#### Noise Reduction:

* It's crucial to minimize any noise in the image to highlight the primary facial features. Techniques like Gaussian blurring can be applied to reduce random noise.
* While Gaussian blurring isn't explicitly applied in the system, the face detection mechanism might implicitly handle some noise reduction. This is because the Haar or LBP cascades used in OpenCV's **CascadeClassifier** typically work better with clean and distinct features.

#### Normalization:

* Normalizing pixel values ensures that they lie in a specific range, usually [0, 1]. This step aids in the stability of neural networks and other algorithms sensitive to input scales.
* In the **preprocess\_image** function, after converting the image to an array, pixel values are normalized to the [0, 1] range.

#### Face Detection:

* The code leverages the **FaceDetector** class to identify faces within the images. This step ensures that only the relevant portions of an image (the face) are processed, eliminating any background or unrelated objects. If a FaceDetector is not provided to the class, the system will simply skip the face detection
* The detection mechanism also rotates the image through various angles (0, 90, 180, and 270 degrees) to ensure faces are detected regardless of their orientation in the image.
* Post detection, the face is further processed (resized, grayscaled, and normalized) before being added to the dataset.

### Dataset Structure and Extensions:

* The **FaceDataset** class fetches the LFW (Labeled Faces in the Wild) dataset. This dataset's images undergo the preprocessing pipeline to ensure they are suited for facial recognition tasks.
* The **ExtendedFaceDataset** extends the **FaceDataset** by introducing the capability to inject custom images (referred to as the "true" images). This allows for a balanced dataset where custom images can be added to achieve a desired percentage in the dataset.

In conclusion, the preprocessing and dataset structuring in the provided code aim to ensure that the facial recognition system works reliably across varied input images. Proper preprocessing ensures that only the most relevant and cleaned features are fed into the system, thus enhancing prediction accuracy.

## Face Recognition Module

The Face Recognition Module is where the previously detected faces are identified. This complex task requires a carefully orchestrated series of algorithms and techniques, discussed in detail below.

### Eigenfaces and Dimensionality Reduction

#### Introduction to Eigenfaces:

Eigenfaces represent a facial recognition method based on Principal Component Analysis (PCA). They provide a way to reduce the dimensionality of the image space, enabling effective computation and classification. This technique treats face recognition as a two-dimensional problem, where images are converted into a set of basis faces. The mathematics of PCA, its relevance to facial recognition, and its comparison with other dimensionality reduction techniques are explored here.

In the implementation of this system, eigenfaces are not directly handled by the programmer, rather the PCA implementation utilized by the programmer handles this behind the scenes.

#### PCA Implementation:

Implementing PCA involves creating a covariance matrix from the face vectors, followed by extracting eigenvectors and eigenvalues. These form the Eigenfaces when applied to original faces.

**Steps involved:**

* 1. **Standardize the data**: Before applying PCA, it is crucial to scale the features such that each one has a mean of zero and a standard deviation of one.
  2. **Calculate the Covariance Matrix**: Given **n** standardized samples of dimension **d**, form a **d x d** covariance matrix **C**. Each element Cij*Cij*​ of **C** measures the covariance between the **i-th** and **j-th** features.
  3. **Calculate Eigenvectors and Eigenvalues**: Obtain eigenvectors and corresponding eigenvalues of the covariance matrix. Eigenvectors represent the directions of maximum variance in the data, and the eigenvalues give the magnitude of this variance.

Again, much of this is abstracted away due to the conveniences of modern libraries.

#### Eigenfaces with SciKit Learn:

SciKit Learn offers an effective toolkit for PCA implementation. The library's **PCA** module streamlines Eigenface extraction, eliminating the manual computation of covariance matrices and eigenvalues.

* **Face Projection**: SciKit Learn allows for the transformation of new faces into the Eigenface subspace, representing them as a linear combination of the selected Eigenfaces.
* **Recognition Mechanism**: The projected representation can be juxtaposed with known faces using distance metrics. A minimized distance suggests successful facial recognition.

#### PCA in SciKit Learn:

The PCA class in the sklearn.decomposition module is a dimensionality reduction tool that uses singular value decomposition of the data and can project it to a lower-dimensional space.

The PCA object in SciKit Learn provides a user-friendly interface to perform Principal Component Analysis, making it easier for data scientists and researchers to apply this powerful technique for dimensionality reduction and feature extraction. However, a clear understanding of its workings and inherent assumptions is crucial to appropriately apply it to real-world datasets [15].

### Support Vector Machine (SVM)

#### SVM Overview:

Support Vector Machine (SVM) is a powerful classification method that has been implemented efficiently in the Scikit-learn module. This section provides an in-depth overview of how SVM operates, detailing the mathematical formulations of the hyperplanes and margin optimization. Particularly, it explains how Scikit-learn's SVM implementation (`sklearn.svm.SVC`) fits within the facial recognition paradigm [15].

#### Kernel Selection:

One of SVM's standout features is its flexibility, primarily through the use of kernel functions. The Scikit-learn module offers built-in support for a variety of these functions. This sub-section delves into different kernel functions available in Scikit-learn, such as linear (`'linear'`), polynomial (`'poly'`), and Radial Basis Function (RBF, `'rbf'`), and discusses their relevance to face recognition. Additionally, it examines the decision-making process for kernel selection in Scikit-learn's SVM and how to apply these kernels within the system effectively [15]. For this project, it was found that the linear kernel was the best choice over RBF. The polynomial option, admittedly, was never tried due to the strong performance of the linear Kernel.

#### Hyperparameter Tuning:

SVM's performance, especially when implemented via Scikit-learn, is highly sensitive to specific hyperparameters. Key among these are the regularization parameter (C) and the kernel coefficient (gamma). This section sheds light on the methodologies available in Scikit-learn, like `GridSearchCV` and `RandomizedSearchCV`, to fine-tune these hyperparameters. It further expounds on the implications of selecting different values, emphasizing potential issues such as overfitting and the trade-offs concerning model complexity [15].

Incorporating Scikit-learn's utilities and tools can streamline the SVM modeling process, making tasks like kernel selection and hyperparameter tuning more systematic and efficient.

### Training

#### Dataset Generation:

Creating a diverse and representative dataset is critical. This section discusses the principles and practices of dataset construction, including the selection of sources, diversity in facial expressions, lighting, angles, and potential biases. SciKit Learn provides direct access to the fetch\_lfw\_people function, which grants access to the Labeled Faces in the Wild (LFW) Dataset. This data set is used to generate the false cases, and the true cases are presently jpeg images, brought in from the internet, of Tom Cruise. Effectively, the system is trying to learn to authenticate on images of Tom Cruise.

#### Data Splitting:

The data is partitioned into training (75%) and testing (25%) sets, ensuring an unbiased evaluation. The partitioning of data into training and testing sets is a delicate process, impacting model evaluation. This is done using SciKit Learn’s train\_test\_split functionality [15].

#### Validation Strategy:

Cross-validation, e.g., 5-fold, is employed to minimize overfitting and provide a robust estimate of model performance. Cross-validation, such as k-fold validation, is a robust methodology to ensure that the model's performance is generalizable.

The data contains True (1) and False (0) cases, the experiment aims to understand the effect of weighting True to False in different ratios. The system aims to produce data sets that contain 5%, 10%, 15%, 20%, 25%, 30%, 50%, and 70% True images and then run them through the algorithm. A cross validation report will be produced for each case.

## Experiment Procedure

### Data Collection

#### Source Selection:

Data is collected from varied sources including public datasets with controlled and uncontrolled environments.

### Face Detection:

When using the LFW (Labeled Faces in the Wild) dataset, the faces have already undergone preprocessing. This means that the facial features are well-aligned and centered within the frame. Such preprocessing is crucial for consistent training in facial recognition systems. However, the challenge arises when the images being recognized, particularly the target person or the “you” case, do not possess the same alignment or scale characteristics as the LFW dataset.

To mitigate this discrepancy, the system employs a face detection mechanism. This mechanism ensures that the faces in the target images are identified and aligned consistently with the training data.

**Implementation Details:**

The system uses OpenCV, a widely-used computer vision library, for the face detection task. The main class responsible for this is **FaceDetector**. Here's a deeper dive into its workings:

1. **Initialization**: The **FaceDetector** is initialized using a given Haar or LBP cascade file. This cascade file contains features that the detector uses to identify faces. The class can also be set up to generate an HTML report showcasing detected faces, controlled by the **generate\_report** flag.
2. **Face Detection**: The **detect\_faces** method is the core of this class. It attempts to detect faces by rotating the image through various angles - 0, 90, 180, and 270 degrees. This rotation ensures that faces are detected regardless of their orientation in the image. Once the image is rotated, it's converted to grayscale, which simplifies the detection process. The detected faces are then filtered based on their aspect ratios to ensure that only valid faces are considered.
3. **Image Saving**: Detected faces and source images can be saved to predefined directories for further analysis. This feature is particularly useful when generating reports.
4. **Visualization**: The system offers two utilities, **matplotlib\_face** and **matplotlib\_faces**, for visualizing the detected faces using the **matplotlib** library. This is beneficial for quick verification and quality checks.
5. **Report Generation**: If the **generate\_report** flag is set during initialization, the **finalize\_report** method can generate an HTML report that showcases all the source images and their detected faces. This report provides a visual summary of the face detection process and its outcomes.

In summary, the face detection system ensures that the target images are processed and aligned consistently with the training data, thereby enhancing the accuracy and reliability of the facial recognition process. This step is crucial, especially when the input images have varied orientations and scales.

Face detection is an option when building the training and testing data. It can be turned on and off at the will of the programmer, and experiments can be run on whether or not its usage contributes to better outcomes.

### Preprocessing

The both the data from LFW and the detected faces from the FaceDetector are passed through a common preprocessor. The preprocessed by converting the image to a grayscale image via Python’s Pillow (PIL) library.

### Feature Extraction:

Algorithms such as Local Binary Patterns (LBP) or Histogram of Oriented Gradients (HOG) are employed to extract geometric and texture features.

### Training the Recognizer:

Training includes applying PCA for dimensionality reduction and then feeding these features to the SVM for classification.

### Evaluation

#### Metrics:

Metrics such as accuracy, precision, recall, and F1-score provide a quantitative assessment.

#### Robustness Testing:

The system is tested against variations in expressions, lighting, occlusion, etc.

#### Comparative Analysis:

Performance is compared with other recognition methods to highlight the strengths and weaknesses.

## Conclusion of Design

The experiment's design encapsulates a holistic approach to facial recognition, from detection to classification. The meticulous planning, selection of algorithms, validation strategies, and robust evaluation showcase a well-rounded and rigorous design. The next phases will involve implementation, tuning, and extensive testing to ensure that the design's promise translates into an effective real-world application.

This expanded content adds depth to each section, detailing the steps, choices, and considerations in the design of the facial recognition system. Feel free to modify or ask for further details on specific areas.

# Results:

A blue and white graph

Description automatically generated with medium confidence

A screenshot of a graph

Description automatically generated

**Bibliography**

[1] Y. Taigman, M. Yang, M. Ranzato and L. Wolf, “DeepFace: Closing the gap to human-level performance in face verification,” 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 2014, pp. 1701-1708, doi: 10.1109/CVPR.2014.220.

[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, pp. 815-823, 2015.

[3] R. Chellappa, C. L. Wilson and S. Sirohey, "Human and machine recognition of faces: a survey," in *\_Proceedings of the IEEE\_*, vol. 83, no. 5, pp. 705-741, May 1995, doi: 10.1109/5.381842.

[4] M. Turk and A. Pentland, “Eigenfaces for recognition,” Journal of Cognitive Neuroscience, vol. 3, no. 1, pp. 71-86, Jan. 1991, doi: 10.1162/jocn.1991.3.1.71.

[5] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus and Y. LeCun, “Overfeat: Integrated recognition, localization and detection using convolutional networks,” arXiv preprint arXiv:1312.6229, 2013.

[6]Yang, H., Jia, X., Loy, C. C., & Robinson, P. (2015). An empirical study of recent face alignment methods. arXiv preprint arXiv:1511.05049.

[7] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, “Gradient-based learning applied to document recognition,” in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.

[8]A. Krizhevsky, I. Sutskever and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in Neural Information Processing Systems, pp. 1097-1105, 2012.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov et al., “Going deeper with convolutions,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1-9, 2015.

[10] Parkhi, O. M., Vedaldi, A., & Zisserman, A. (2015, September). Deep face recognition. In bmvc (Vol. 1, No. 3, p. 6).

[11] Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Le, Q. V., ... & Ng, A. Y. (2012). Large scale distributed deep networks. In Advances in neural information processing systems (pp. 1223-1231).

[12] Y. Sun, Y. Chen, X. Wang and X. Tang, “Deep learning face representation by joint identification-verification,” 2014 28th International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 2014, pp. 1988-1996.

[13] Y. Sun, Y. Chen, X. Wang and X. Tang, “Deep learning face representation by joint identification-verification,” 2014 28th International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 2014, pp. 1988-1996.

[14] K. W. Bowyer, K. Chang, and P. Flynn, "A survey of approaches and challenges in 3D and multi-modal 3D + 2D face recognition," Computer Vision and Image Understanding, vol. 101, no. 1, pp. 1–15, 2006.

[15] Scikit-learn developers. Scikit-learn: Machine Learning in Python. Documentation, version 1.3.0. Available: https://scikit-learn.org/stable/documentation.html. [Accessed: August 15, 2023].