Table of Contents

Machine Learning D/HD Project
Face Recognition Attendance System with Liveness Detection

by Zun Yang Chin, 102781103

A. INTRODUCTION

The development of robust and accurate face recognition systems has become increasingly important in various applications, including security, surveillance, and automated attendance systems. This project report presents a comprehensive exploration and comparison of two distinct approaches to building a face recognition application: face verification (self-supervised learning) and face classification (supervised learning).

Face verification involves determining whether two face images are of the same person, effectively solving an open-set problem where the model may encounter new face identities. This approach utilizes techniques such as metric learning to model the similarity between images directly. In contrast, face classification assigns a face image to a predefined identity class, addressing a closed-set problem where the model recognizes only the identities it has been trained on. This method employs supervised learning techniques to classify faces based on known identities [1].

This project documents the process of implementing both face verification and face classification models, evaluating their performance, and selecting the most effective approach for integration into a face attendance system. The chosen dataset for this project is sourced from Kaggle [2], providing a robust collection of face images for training and testing the models.

In the implementation phase, convolutional neural networks (CNNs) were used to generate face embeddings, which are compact, low-dimensional feature vectors that retain critical discriminative information. These embeddings are then utilized for both face verification and classification tasks. The performance of these models is assessed using appropriate evaluation metrics, including the Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC), which are standard in measuring the accuracy of face recognition systems.

Following the model training and evaluation, the best-performing model is integrated into a face attendance system. This system includes an anti-spoofing module [3] designed to detect and prevent spoofing attempts using liveness detection techniques, thereby enhancing the security and reliability of the face recognition process.

This report not only highlights the methodologies employed in developing the face recognition models but also provides a detailed analysis of their performance. The findings from this project contribute valuable insights into the comparative effectiveness of face verification and classification approaches in real-world applications, particularly in automated attendance systems.

B. REQUIREMENTS

The face recognition attendance system project involves several critical requirements and steps to ensure successful implementation and performance. The first key aspect is image preprocessing. The dataset images are 64x64 pixels, which is relatively low for face classification tasks that require capturing fine facial features. Therefore, preprocessing steps need to consider this limitation. Additionally, with 3000 classes and each class containing 30-50 images, it is essential to ensure efficient data loading to avoid excessive training times. Trimming down the dataset may be necessary due to computational power constraints.

To expedite model training, leveraging Google Colab Pro's A500 GPUs is recommended. These GPUs offer significantly faster training times, saving valuable time during the model development phase. Efficient utilization of computational resources is crucial for handling the extensive data and complex models involved in this project.

The dataset preparation for the classification model is straightforward as the dataset is already split into training, validation, and test sets, eliminating the need for further partitioning. However, for the Siamese model, it is necessary to create anchor, positive, and negative data directories. Positive pairs consist of anchor and positive images, while negative pairs consist of anchor and negative images. This preparation is essential for training the Siamese model effectively.

Defining the model architecture for extracting face embeddings is another critical step. This involves specifying the convolutional neural network (CNN) structure that will generate compact, low-dimensional feature vectors retaining critical discriminative information. Additionally, appropriate metrics for metric learning tasks, specifically for the verification task, need to be suggested to ensure accurate performance evaluation.

Once the models are defined, the next step is model training. The verification model is trained to take two input images and provide a similarity score, while the classification model is trained to take a single input image and output the corresponding class label. Training these models requires careful consideration of the dataset, computational resources, and chosen architectures.

After training, the models are evaluated using performance metrics such as the ROC (Receiver Operating Characteristic) curve and AUC (Area Under the Curve). These metrics help compare the performance of the verification and classification models, providing insights into their effectiveness in face recognition tasks.

The final phase involves integrating the best-performing model into a face attendance system. The system includes a basic user interface with 'login', 'logout', and 'register' buttons. It integrates face detection to capture images and store them in a directory for the model to perform face verification. The system displays output indicating whether the person is logged in (if their image is in the dataset) or identified as an 'unknown' user if the person cannot be identified. An anti-spoofing module is also implemented to detect real versus spoofed images (e.g., phone images, printed pictures), displaying a "fake face detected" message for spoof attempts. Additionally, the system includes registration functionality, allowing users to capture their face, input their name, and add new face data to the system. All login, logout, and registration operations are logged in a .csv file for record-keeping.

C. METHODOLOGY

I. Classification Based on Identity Classes (Supervised learning)

First and foremost, the dataset is uploaded as a zip file to Google Drive and unzipped. The dataset is already classified into train, test, and validation sets, so no additional partitioning is needed. However, the original dataset contains 3000 classes, which is too extensive for effective training and could lead to overfitting. Therefore, the dataset is trimmed down to 500 classes for each partition.

To select 500 classes out of 3000, a practical approach is employed using the SSD face detector model for face detection [4]. The `select_classes_with_faces` function loops through all 3000 classes in the dataset, running face detection on each class. Only classes with a minimum of 30 images with detected faces are selected. From these selected classes, 500 are randomly chosen to be kept. This method not only reduces the dataset size but also ensures that the retained data is of high quality, with visible faces detected by the SSD face detector, a lightweight and efficient model for face detection.

To ensure consistency and ease of operation, the selected 500 classes are renamed. The VGG model, recognized for its high accuracy in face embedding extraction, is used to define the face embedding layer [5]. Transfer learning is applied by loading pre-trained VGG face weights to save training time. The embeddings for the dataset are then generated and saved to Google Drive, ensuring they are reusable and not lost if the Google Colab runtime disconnects. This allows for different classifiers to be tested by simply loading the saved embeddings using the `load_embeddings` function.

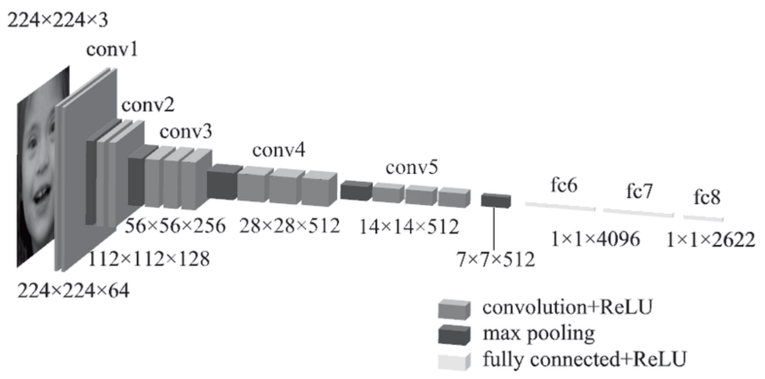

Figure 1 VGGFace network architecture [6]

Next, the saved embeddings are loaded for training and evaluation. This includes loading the embeddings and labels for train, validation, and test sets, and the label map. The labels are encoded into integer format using `LabelEncoder`. Label encoding is essential for converting categorical labels into numerical form, which is a prerequisite for training machine learning models. The feature values (embeddings) are standardized to have a mean of 0 and a standard deviation of 1 using `StandardScaler`. This ensures that each feature contributes equally to the model's performance, avoiding issues where features with larger scales dominate those with smaller scales. Dimensionality reduction is performed using Principal Component Analysis (PCA), reducing the data to 128 components while retaining most of the variance in the dataset. PCA is selected for its effectiveness in reducing the computational complexity of the model by transforming the high-dimensional data into a lower-dimensional form, capturing the most significant features while minimizing information loss. This step is particularly important given the high dimensionality of face embeddings.

An SVM classifier is then trained using the standardized and reduced-dimension embeddings. SVM (Support Vector Machine) is chosen for its robust performance in high-dimensional spaces and its effectiveness in classification tasks, particularly when the number of features exceeds the number of samples. SVM is also capable of handling multi-class classification efficiently with the use of a one-vs-one or one-vs-rest approach [7].

The classifier's performance is evaluated by making predictions on the train, validation, and test sets. Accuracy metrics are calculated for each set, indicating the model's performance. The trained classifier and other components (PCA, scaler, label encoder) are saved using `joblib` for future use. For evaluation, the saved components are loaded, and the test embeddings are standardized and reduced in dimensions using the loaded scaler and PCA. The labels are encoded using the loaded label encoder. Predictions are made using the SVM classifier, and the probabilities are used to compute the micro-average ROC curve and its area under the curve (AUC). The ROC curve is plotted to visualize the model's performance in distinguishing between different classes. The micro-average ROC curve is selected as it aggregates the contributions of all classes, providing a comprehensive view of the model's overall performance.

II. *Metric Learning (Self-supervised learning)*

The approach for face verification using metric learning involves developing a Siamese neural network, which is designed to determine whether two face images belong to the same person. Unlike traditional classification tasks, this method addresses an open-set problem, making it suitable for real-world applications where the model may encounter new face identities.

To build a robust verification model, data collection was done using a webcam instead of using the provided dataset. This approach was chosen because the original dataset was too large, leading to long processing times. Using `cv2`, 300 anchor samples (images of my face), 300 positive samples (images of my face) and 300 negative samples (images not of my face, selected from the train dataset) were captured. This method ensures a balanced dataset with clear distinctions between positive and negative pairs. The collected images were resized to 100x100 pixels to ensure uniformity and compatibility with the model architecture. The pixel values were normalized by dividing by 255, converting them to a range of [0, 1], which is crucial for speeding up the convergence of the model during training. Data partitioning was performed to create training and testing datasets. The dataset was divided such that 70% of the data was used for training and 30% for testing. The data was then batched and prefetched to improve the efficiency of the training process.

The Siamese network architecture follows the guidelines from the official Siamese network paper [8]. The architecture consists of two identical subnetworks that share the same weights. Each subnetwork extracts embeddings from the input images, creating a compact, low-dimensional representation of the faces. The embedding layer of the model is defined using a Convolutional Neural Network (CNN) to extract meaningful features from the images. The CNN is designed to output a feature vector for each input image, capturing the critical aspects needed for distinguishing between different faces.

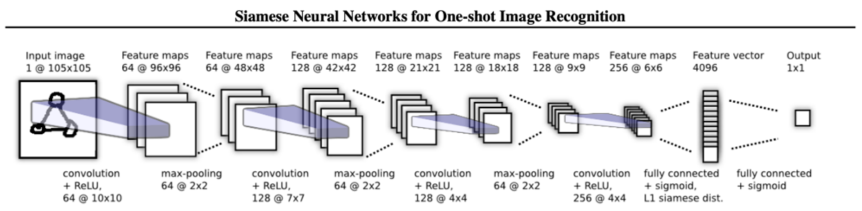

Figure 2 Siamese Network Architecture

The L1Dist layer is defined to calculate the absolute differences between the feature vectors (embeddings) of the two input images. This layer is crucial for measuring the similarity between the two embeddings, with the L1 distance (Manhattan distance) being chosen as the metric. The L1 distance is effective in capturing the differences between feature vectors, making it suitable for this verification task.

The training process involves defining the training step and training loop. The training step includes forward propagation through the Siamese network, calculating the loss using a contrastive loss function, and performing backpropagation to update the model weights. The contrastive loss function is designed to minimize the distance between embeddings of similar pairs (positive pairs) and maximize the distance between embeddings of dissimilar pairs (negative pairs). The training loop iterates over the dataset for a specified number of epochs, continuously updating the model weights to improve its performance. The loop also includes validation steps to monitor the model's performance on a separate validation set, ensuring that the model does not overfit the training data. After training, the Siamese network model is saved for future use. This allows for the model to be reloaded and used for face verification tasks without retraining, saving computational resources and time.

For verification, the saved model is loaded, and the embeddings of new input image pairs are generated. The L1Dist layer calculates the distance between the embeddings, and a threshold is applied to determine whether the images belong to the same person. The threshold is chosen based on the validation set to balance precision and recall. The model's performance is evaluated using metrics such as accuracy, precision, recall, and the ROC curve. These metrics provide a comprehensive view of the model's ability to distinguish between positive and negative pairs. The ROC curve is particularly useful for visualizing the trade-off between true positive and false positive rates at different threshold settings.

III. *Anti-Spoofing Module*

Building an anti-spoofing module from scratch requires significant additional work and expertise. Through online research, I discovered the Silent-Face-Anti-Spoofing model developed by MiniVision AI [3]. This model demonstrated impressive performance and seemed well-suited for integration into my face attendance system application. After a thorough review, I decided to incorporate this model into my application due to its robustness and effectiveness in detecting spoofing attempts.

The Silent-Face-Anti-Spoofing model uses liveness detection techniques to differentiate between real and spoofed faces. The primary approach employed by this model involves the use of Fourier spectrum analysis to capture the frequency domain differences between real and fake faces. This method is effective in distinguishing various spoofing methods, such as printed photos, display screens, silicone masks, and 3D portraits.

The model architecture comprises a classification branch and a Fourier spectrum auxiliary supervision branch. This dual-branch approach enhances the model's capability to detect spoofing attempts by leveraging the unique characteristics of real and fake faces in the frequency domain.

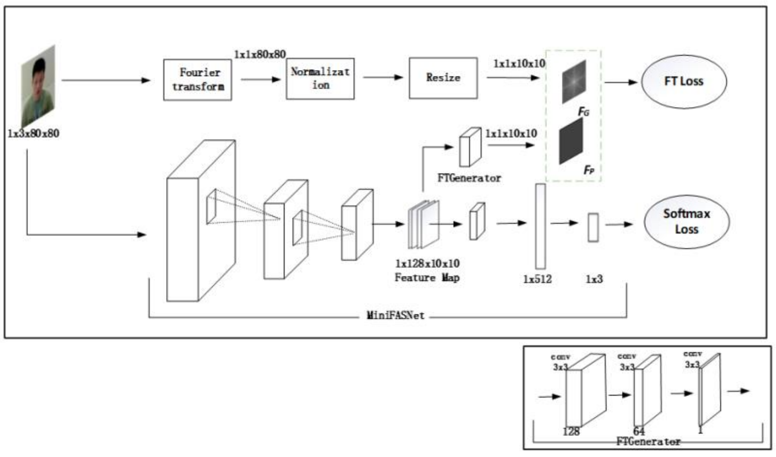

Figure 3 Anti-Spoofing Model Architecture

The classification branch is responsible for the primary classification task, which involves determining whether an input face is real or fake. It uses convolutional layers to extract features from the input images and applies fully connected layers to make the final classification. The Fourier spectrum auxiliary supervision branch assists the classification branch by providing additional supervision based on the Fourier spectrum of the input images. By analyzing the frequency components, this branch helps the model capture subtle differences between real and spoofed faces, improving overall detection accuracy.

In terms of implementation, the input images are first converted to grayscale and resized to a fixed size. The grayscale conversion simplifies the data, reducing computational complexity while retaining essential information. The Fourier transform is then applied to the preprocessed images to convert them from the spatial domain to the frequency domain. This step highlights the frequency components that distinguish real faces from spoofed ones. Both the spatial and frequency domain features are extracted using convolutional neural networks. These features are then combined and fed into fully connected layers for final classification.

By incorporating the Silent-Face-Anti-Spoofing model into my face attendance system, I ensured that the application could reliably detect and prevent spoofing attempts, thereby enhancing its security and reliability. This integration highlights the importance of leveraging state-of-the-art techniques and pre-existing models to build robust and effective machine learning applications.

IV. *Application Building*

The final step of this project involves building a face attendance application using the trained Siamese model for face verification. This application aims to provide a reliable and efficient system for tracking attendance by verifying the identity of individuals based on their facial features. The application also integrates an anti-spoofing module to enhance security by detecting and preventing spoofing attempts. To begin, the trained Siamese model and the L1Dist layer are loaded. These components are essential for performing face verification tasks. The Siamese model generates embeddings for the input images, while the L1Dist layer calculates the distance between these embeddings to determine the similarity between faces.

The images captured for verification need to be preprocessed to ensure they are compatible with the model. The preprocessing steps include reading the image file, decoding the JPEG format, resizing the image to 100x100 pixels, and normalizing the pixel values to a range of [0, 1].

The verification function utilizes the Siamese model to compare the input image with images stored in the verification database. The function iterates through each person's folder in the verification images directory, preprocessing each image and using the model to predict the similarity score between the input image and the verification images. A detection threshold is applied to determine whether the input image matches any stored images. If the verification score exceeds the threshold, the function returns the matched person's name; otherwise, it returns 'None', indicating no match.

An event logging function is implemented to record attendance events, such as login, logout, and registration, into a CSV file. This provides a record of all interactions with the system.

The application includes a face registration feature, allowing new users to register their faces in the system. During registration, the webcam captures multiple images of the user's face, which are then saved in a designated folder. This process involves using

OpenCV to capture images, detect faces, resize them to 100x100 pixels, and save them to the appropriate directory.

The user interface is built using `Tkinter`, providing a user-friendly and interactive experience. The interface includes buttons for logging in, logging out, and registering new users. Each button is associated with a specific function to handle the corresponding tasks. The login function captures an image using the webcam, performs an anti-spoofing check, preprocesses the image, and verifies it against the stored images using the Siamese model. If the verification is successful, the user is logged in, and the event is logged. The logout function operates similarly to the login function, capturing an image, performing an anti-spoofing check, and verifying it. If successful, the user is logged out, and the event is logged. The registration function captures multiple images of the new user's face, saves them to the appropriate directory, and logs the registration event.

The application integrates real-time webcam functionality using OpenCV. The webcam captures live video feeds, and face detection is performed on each frame. The detected faces are highlighted with rectangles, and the most recent capture is used for verification or registration. An anti-spoofing check is performed, using the Silent-Face-Anti-Spoofing before verification or logout to ensure the face being captured is not a spoof (e.g., a photo or video of a face).

D. RESULTS and DISCUSSION

The results of this project are presented by comparing the performance of the two models: the VGG classification model (supervised) and the Siamese verification model (self-supervised). The evaluation metrics used are the Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC). For the classification model, the micro-average ROC curve is used, which aggregates the contributions of all classes to provide a comprehensive view of the model's performance. The AUC for the VGG classification model is 0.99, while the AUC for the Siamese verification model is 1.00.

The use of ROC and AUC as evaluation metrics is justified by their effectiveness in providing a detailed performance analysis of the models, especially in scenarios where class distribution is imbalanced. Unlike pure accuracy, which only measures the proportion of correctly classified instances, AUC evaluates the model's ability to distinguish between classes by considering the trade-off between true positive and false positive rates at various threshold settings. This makes AUC a more reliable metric for assessing the performance of face recognition systems, as it accounts for the model's discrimination power across all decision thresholds [9].

The micro-average ROC is used for the VGG classification model to handle the multi-class classification task. This approach aggregates the performance of each class into a single ROC curve, providing a balanced view of the model's overall ability to correctly classify faces among multiple classes. This method ensures that the evaluation reflects the model's performance across all classes, rather than being biased by the performance on any single class.

The high AUC values for both models indicate their strong performance in face recognition tasks. For the VGG classification model, the high quality of the dataset and the rigorous preprocessing steps significantly contributed to its effectiveness. By selecting 500 high-quality classes out of 3000 and ensuring that all selected classes contain well-detected faces by the SSD model, the dataset was refined to enhance the model's training process and performance.

For the Siamese verification model, the perfect AUC can be attributed to the controlled data collection process. By capturing 300 anchor images and 300 positive images of my own face using a webcam in a consistent environment, the model was trained on highly similar positive pairs. The negative class was formed by combining images from the entire training dataset, creating a stark contrast between positive and negative pairs. This significant difference between positive and negative samples simplified the training process, enabling the Siamese model to perform exceptionally well during evaluation. However, this controlled data collection process may also lead to overfitting in real-world applications. Since the model was trained on images captured in a consistent environment with similar lighting and angles, it might struggle to generalize to variations in real-world conditions. When applied to the face attendance application, the Siamese model sometimes underperforms because it cannot differentiate subtle differences between faces captured in varying conditions. This highlights the importance of training models on diverse datasets to ensure robust performance in real-world scenarios.

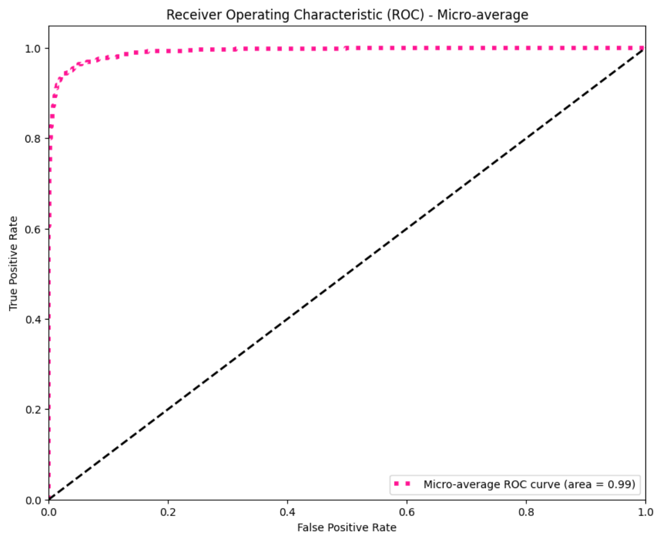

Figure 4 VGG Classification Model ROC Curve

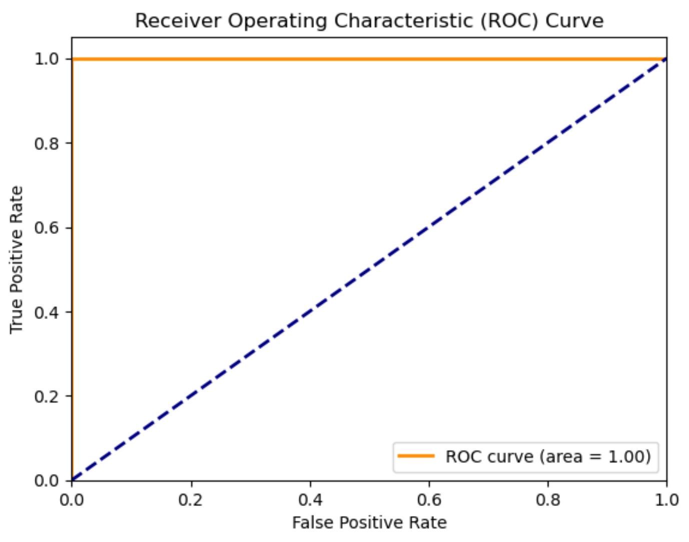

Figure 5 Siamese Verification Model ROC

E. CONCLUSION AND FUTURE WORK

The development of the face recognition attendance system with liveness detection demonstrates the potential of leveraging advanced machine learning techniques to create secure and efficient applications. This project compared two distinct approaches—supervised learning with the VGG classification model and self-supervised learning with the Siamese verification model. Both models achieved high AUC values, indicating strong performance in face recognition tasks.

However, the controlled data collection for the Siamese model, while ensuring high performance in evaluation, led to overfitting issues when applied in real-world scenarios. This underlines the necessity of training models on diverse datasets to enhance their robustness and generalization capabilities. Future work should focus on collecting a more varied and high-quality dataset that includes different backgrounds, lighting conditions, and angles. This would help in developing a more resilient face verification model capable of handling real-world variations. Additionally, future improvements could include integrating more advanced image preprocessing steps to enhance the quality of the input data. The user interface could also be refined to provide a more seamless user experience, incorporating features like user feedback mechanisms and automated system updates.

The findings from this project contribute valuable insights into the comparative effectiveness of face verification and classification models and highlight the importance of comprehensive data collection and preprocessing in developing robust face recognition systems. By addressing the current limitations and incorporating these improvements, the face recognition attendance system can be made more reliable and effective for broader applications.

F. REFERENCES

[1] A. Sahni, "Face Classification and Verification," 2019. [Online]. Available: https://github.com/anjandeepsahni/face_classification/blob/master/README.md.

[2] Abrsh, Akshat Gupta, Antioch John Sanders, Baccano!, Bharat Gaind, BionicKitty, BiteMe, CMU11785, Et tu, Beam Search?, Jacob Lee, Jiachen Lian, Jingwei Zhang, Jinhyung Park, Mansi Anand, Reshmi Ghosh, Rhiksha Baj, Sean Pereira, Viking, "Face Verification Using Convolutional Neural Networks," Kaggle, 2020. [Online]. Available: https://www.kaggle.com/c/11-785-fall-20-homework-2-part-2/overview/evaluation.

[3] zhuyingSeu, LZCai, nbadalls, "Silent-Face-Anti-Spoofing," GitHub, 2020. [Online]. Available: https://github.com/minivision-ai/Silent-Face-Anti-Spoofing/tree/master.

[4] K. Rathod, "face-detection," GitHub, 2018. [Online]. Available: https://github.com/keyurr2/face-detection/tree/master.

[5] S. I. Serengil, "deepface," GitHub, 2024. [Online]. Available: https://github.com/serengil/deepface.

[6] Vyacheslav Kumov, Andrey Samorodov, "Recognition of Genetic Diseases Based on Combined Feature Extraction From 2D Face Images," in *ResearchGate*, 2020.

[7] J. Brownlee, "Machine Learning Mastery," 2021. [Online]. Available: https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/.

[8] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, "Siamese Neural Networks for One-shot Image Recognition," Department of Computer Science, University of Toronto. Toronto, Ontario, Canada.

[9] T. Fawcett, "Pattern Recognition Letters," *An introduction to ROC analysis,* 2006.

G. APPENDIX

1. YouTube Video: Siamese Model Tutorial
2. YouTube Video: Face Attendance App UI & Anti Spoofing
3. Jupyter Notebook & Google Colab Pro
4. ChatGPT for debugging
5. GitHub Repository for reference
6. Week 8 Learning Materials for understanding requirements