FPT UNIVERSITY

SMART MEDITATION

Capstone Project Report

Nguyen Trung Thanh
Nguyen Sy Duc Dai
Huynh Tien Dung

Supervisor: **MSc. Do Thai Giang**

Bachelor of Artifial Intelligence
FPT University - Hoa Lac Campus
28th Aug, 2024

Acknowledgement

Our deepest gratitude extends to MSc Do Thai Giang, our esteemed thesis advisor, for his invaluable guidance, insightful feedback, and unwavering support throughout our graduation thesis journey. His profound knowledge, constructive criticism, and steadfast encouragement have been crucial in shaping the outcome of this thesis as well as our overall academic development. It has been both a privilege and an honor to work under his dedicated supervision.

We wish to express our sincere appreciation to FPT University for providing the essential prerequisites, state-of-the-art infrastructure, and the appropriate channels that were crucial for the successful completion of our graduation thesis. The university's unwavering commitment to advancing education and research has significantly fueled our aspirations. As proud recipients of this prestigious academic institution's support, we take great pride in contributing to its esteemed scholarly legacy.

Moreover, we formally acknowledge the indispensable role played by our beloved families and steadfast friends. Their unwavering care, support, and motivation have been a cornerstone of our success. Their faith in our abilities and relentless encouragement bolstered our resilience and determination, empowering us to pursue our goals with unwavering commitment. It is with deep gratitude that we recognize their pivotal role in our achievements, affirming that their steadfast support was essential to our success. Our accomplishments would not have been possible without their continuous and steadfast backing.

Lastly, we extend our heartfelt thanks to the participants of this study for their time, openness, and willingness to share valuable insights. Their contributions provided a solid foundation for this research and enriched our understanding of the subject matter.

This thesis stands as a testament to the collective efforts of all those mentioned above, and we are profoundly grateful for the profound impact each of you has had on this academic endeavor.

Abstract

Today, health issues are a top concern with the emergence of numerous applications related to yoga, running, and gym workouts. These applications help individuals improve their health, reduce stress, and increase energy, paving the way for the development of smart meditation applications. In this thesis, we develop a system aimed at integrating user authentication and real-time posture verification. The application combines Siamese Neural Network (SNN) and ML Kit to detect, track, and verify faces, ensuring continuous awareness of the user's presence throughout the meditation session; MediaPipe, a powerful technology for posture detection and tracking, ensures that users maintain the correct meditation pose. This project demonstrates how merging deep learning technology with other technologies can lead to innovative and improved wellness solutions in the field of mobile applications.

Keywords: Smart Meditation, Facial Detection, Facial Recognition, Posture Verification, Posture Monitoring.

Table Of Contents

Contents

List of Figures

1 Introduction

1.1 Overview

Traditional meditation is an ancient practice that helps individuals attain inner peace and enhance mindfulness through breathing exercises and concentration techniques. In today's modern context, technology has been integrated into meditation to elevate the user experience. Meditation apps not only offer a variety of exercises but also allow users to track their progress, set reminders, and even tailor sessions to match their psychological state. Nowadays, there are numerous advanced meditation apps available that make it easier for users to maintain their practice and fully enjoy the benefits of meditation.

Meditation apps today boast numerous outstanding features that enhance the user experience. They offer a wide range of meditation exercises, from basic to advanced, allowing users to choose methods that suit their personal needs. Additionally, these apps often include features like time reminders, progress tracking, and even nature sounds to create an ideal meditation environment. However, while these features significantly improve the user experience, they still fall short in providing the necessary intuitiveness for meditation practice. This is where the integration of machine learning models to monitor and adjust meditation sessions in real-time represents a groundbreaking innovation, paving the way for unprecedented opportunities in optimizing meditation effectiveness.

The camera application that directly tracks the user's posture while meditating is an unprecedented breakthrough in current meditation applications, which gave us the opportunity to develop this project. With the goal of improving the effectiveness and safety of meditation practice, we aim to create an application that will use advanced image recognition technology to track and analyze the user's posture in real time. By using the camera and machine learning models to accurately detect any deviations, the application ensures that the user maintains correct posture, thereby optimizing the effectiveness of their practice, minimizing the risk of injury and improving overall comfort. Additionally, the technology provides instant feedback, allowing the practitioner to immediately adjust their posture to respond to any deviations, even without professional supervision. With the ability to track and adjust posture in real time, the app not only enhances the meditation experience but also pioneers a new approach by seamlessly integrating technology with traditional methods. This groundbreaking project meets the growing need for mental and physical health in the digital age.

1.2 Related works

1.2.1 Meditation applications

Popular meditation apps on the market today such as Calm, Mindbliss, Balance, Insight Timer, Mindfully, Meditopia, and Headspace all offer useful and rich features to help users easily access and practice meditation. These apps often have diverse content such as guided voice meditation sessions, personalized experiences, and progress tracking, thereby helping users monitor their progress over time.

Calm, Mindbliss, Balance are popular meditation apps with diverse content, including guided meditation sessions with voice instructions, accompanied by visual illustrations, enhancing the user experience making it easy for users to follow and practice. They stands out with its high level of personalization, allowing users to track their meditation progress. Insight Timer stands out with its highly diverse content and a wide range of instructors, offering many options for users at different levels. This app also supports multiple languages, making it more accessible to users worldwide. Notably, Insight Timer is completely free, which is a significant advantage. Meditopia stands out with its diverse content, including music and video, enhancing visual appeal and the meditation experience. This app features high personalization and progress tracking, making Meditopia an attractive choice for many users. However, the average subscription price might be a barrier for some people who want to access this app.

Even while the apps available today for meditation give users access to pictures and educational videos, they frequently lack instant feedback and real-time engagement. This restriction becomes especially important when users have trouble altering their postures since they are unable to get precise, real-time guidance from the program to determine whether they are striking the poses correctly. Lack of immediate feedback can cause novices to practice improperly, decreasing the exercise's effectiveness and possibly injuring themselves. Furthermore, the entire user experience is lowered by this lack of engagement, which makes it harder for users to stay motivated and focused when practicing.

1.2.2 Meditation pose

In meditation, different postures play an important role in creating the ideal environment for meditation, each offering its benefits. The Full Lotus posture is considered the gold standard in many meditation traditions because of its symmetry and the absolute stability it provides. However, this posture requires a high level of flexibility and can be difficult for many beginners. The Half Lotus posture is a gentler variation that maintains the same stability but simultaneously creates more comfort for the practitioner. The Quarter Lotus posture stands out for its perfect balance between the two. One foot is placed on the opposite calf while the other remains on the floor, creating a posture that is easy to maintain for long periods without causing pain. This pose is suitable for all skill levels, from beginners to experienced practitioners, and for this reason, it has become a popular choice in modern meditation practices. Finally, the Burmese Position is the simplest and easiest option, especially for beginners or those seeking maximum comfort in meditation without excessive muscle strain. Each pose, with its characteristics, contributes to the success of the meditation process by helping the practitioner maintain stability, comfort, and focus.

Not only do meditation postures include sitting, but they also have a significant impact on the best possible meditation experience for practitioners. The Full Lotus and Half Lotus are renowned for their capacity to produce complete stability, which enables the body to hold a single posture for extended periods of time without requiring adjustment. This is significant because mental stability reduces internal and external distractions when there is physical stability. During meditation, the mind may effortlessly concentrate on the breath, thoughts, or more profound emotions while the body is still. Furthermore, people who want to be comfortable and relaxed during their meditation sessions might benefit greatly from poses like the Quarter Lotus and Burmese Position.These poses ease tense muscles, lessen the chance of pain from prolonged sitting, and make it easier for the practitioner to attain a profound level of meditation without being disturbed by pain. In addition, these positions' comfort promotes prolonged meditation, which helps the practitioner progressively attain better meditation outcomes. The body's energy circulation is further aided by meditation poses. This is particularly true in symmetrical, balanced poses like the the Half Lotus, Full Lotus and the Quarter Lotus, where energy is thought to flow smoothly and enhance both mental and physical well-being.

1.2.3 Meditation detection

Meditation detection, particularly for ensuring correct posture and improving the effectiveness of meditation sessions, has increasingly gained attention and interest in recent years. The integration of advanced technologies such as computer vision and machine learning has enabled precise and real-time monitoring of meditation practices, which is crucial for improving both safety and outcomes in meditation.

Recent advancements in deep learning have enabled the development of more sophisticated meditation detection systems. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have been employed to analyze video feeds of users during meditation, allowing for more accurate detection and classification of meditation postures. These models can be trained on large datasets to recognize subtle variations in posture and provide personalized feedback to users. For instance,Welekar et al.(2023)[1] developed a meditation accuracy detection system using deep learning. They combined deep learning models such as CNN, RNN, LSTM and achieved high accuracy in classifying and detecting correct and incorrect meditation postures. However, the model faced some difficulties in variations in body, clothing, environment and camera angle for the model to generalize well.

Additionally, Chen et al. (2021) [2] developed a system for recognizing facial expressions during meditation using Deep CNNs, aiming to assess the relaxation and focus levels of users. Some research has focused on using 3D sensors to capture the full-body posture during meditation. Li et al. (2020) [3] developed a system using 3D sensors to monitor the entire body, allowing for detailed feedback on posture to ensure safety and effectiveness in meditation practices. This approach provides a comprehensive view of the user's posture, helping to identify and correct even minor deviations.

Another significant line of research focuses on the application of wearable sensors to monitor physiological signals during meditation. These sensors can track metrics such as heart rate variability, breathing patterns, and muscle tension, offering insights into the user's relaxation level and overall meditation quality. By combining these physiological data with posture detection, researchers have developed comprehensive systems that ensure users maintain correct posture and achieve optimal meditation states. For example, Yang et al. (2019) [4] explored the use of wearable motion sensors to monitor and correct meditation postures. Their machine learning model detected posture deviations and provided alerts for immediate adjustments, focusing on enhancing the safety of meditation by minimizing the risk of injury.

Despite significant progress, challenges remain in making meditation detection systems more accessible and user-friendly. Current research is focusing on reducing the complexity of these systems, improving accuracy across diverse body types and environments, and seamlessly integrating them into consumer-grade meditation apps. As this field evolves, there is a growing emphasis on developing systems that not only detect but also adapt to individual user needs, offering tailored guidance to enhance the overall meditation experience. The combination of pose estimation, wearable sensors, and deep learning techniques represents a promising direction for future research in meditation detection, paving the way for more effective and personalized meditation practices.

1.2.4 Facial detection

Face detection and analysis have been extensively studied in the field of computer vision, with various approaches and algorithms proposed over the years. Traditional methods, such as Viola-Jones [5] and Histogram of Oriented Gradients (HOG) [6], have demonstrated effectiveness in detecting faces in images. However, these methods often face challenges in handling variations in pose, illumination, and occlusions.

The advent of deep learning has revolutionized face detection and analysis, with convolutional neural networks (CNNs) achieving state-of-the-art performance [7]. Frameworks like OpenCV [8] and Dlib [9] provide robust face detection capabilities, but their integration into mobile applications can be complex and computationally demanding.

ML Kit's Face Detection API addresses these challenges by offering a lightweight, on-device solution optimized for mobile platforms. It leverages pre-trained deep learning models to provide accurate and efficient face detection, even in resource-constrained environments [10]. Moreover, ML Kit's modular architecture enables developers to select specific features and customize the performance based on their application requirements [10].

1.2.5 Facial verification

Research on one-shot learning algorithms remains relatively underdeveloped and has not garnered significant attention from the machine learning community. Nonetheless, several important studies have laid the groundwork for this field.

The foundational work on one-shot learning began in the early 2000s with Li Fei-Fei and her team, who proposed a variational Bayesian framework for single-shot image categorization. This framework leverages previously learned classes to predict future classes, even with only a few examples from a specific class ([11], [12]). In more recent development, Lake explored one-shot learning from a cognitive science perspective, focusing on character recognition through Hierarchical Bayesian Program Learning (HBPL) [13]. Their research involved modeling the generative process of drawing characters by breaking down the image into smaller components ([14], [15]). The aim of HBPL is to offer a structural explanation for the observed pixel patterns, but the vast joint parameter space makes inference under HBPL challenging, leading to an integration problem that is intractable.

Researchers have also branched out, exploring diverse data modalities and leveraging transfer learning strategies. Lake and colleagues recently employed a generative Hierarchical Hidden Markov model for speech primitives, combined with Bayesian inference, to recognize new words from unknown speakers [16]. Maas and Kemp have conducted some of the limited studies using Bayesian networks to predict attributes for Ellis Island passenger data [17]. Wu and Dennis investigated one-shot learning in the context of path planning algorithms for robotic actuation [18]. Lim's research focused on "borrowing" examples from other classes in the training set by adjusting the weight of each category in the loss function [18]. This approach is beneficial for datasets with limited examples for certain classes, offering a flexible and continuous method to integrate inter-class information into the model

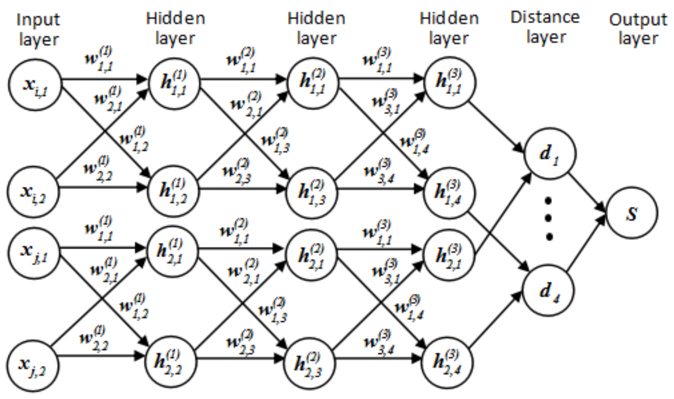


Figure 1: A straightforward Siamese network with three hidden layers designed for binary classification.[19]

1.3 Motivation

This project was initiated with the vision of merging traditional meditation practices with cutting-edge technology, addressing the need for a more accessible and effective meditation experience for people at all levels of practice. As meditation becomes increasingly popular, the diversity among practitioners grows, ranging from beginners to experienced meditators. However, many lack access to qualified instruction, and even seasoned practitioners may struggle to maintain proper posture and focus during their sessions, potentially diminishing the benefits of their practice.

By harnessing the power of camera technology and advanced algorithms, we saw the opportunity to create a system that not only monitors but also adjusts the user's posture in real-time. In addition to posture detection, the project incorporates facial detection and recognition to further enhance personalization and user experience. The facial detection feature allows the system to identify subtle changes in the user's expressions, providing insights into their emotional state, level of concentration, and relaxation during meditation.

Moreover, the facial recognition component enables the system to recognize individual users, tailoring feedback and adjustments to each person's specific meditation style and progress. This personalized approach ensures that the guidance provided is not only relevant but also continuously adapted to the user's evolving needs. For instance, the system can track improvements over time and adjust its feedback to challenge the user in ways that promote deeper focus and relaxation.

The objective of the project is to offer a sophisticated, user-friendly, and tailored solution that elevates the meditation experience for each individual. This will enable users to achieve greater relaxation, focus, and balance in their daily lives, thereby improving their overall quality of life. We believe that by integrating technology into meditation, including camera-based posture monitoring and facial recognition, we can make the practice more accessible, safer, and ultimately more beneficial for people in today's fast-paced world.

1.4 Contribution

In our thesis, we present a comprehensive Smart Meditation application designed to enhance the user experience through real-time posture correction and personalized feedback. A key contribution of our work is the integration of camera-based posture detection and facial recognition algorithms, which together provide accompanying meditation guidance and improved personalization.

Our research opens a new direction by combining advanced AI technologies with meditation methods, establishing a new direction in the field of AI-Mindfulness. This project combines learning with practical application, using AI to enhance meditation and explore its broader potential in developing meditation apps as well as promoting mindfulness and happiness.

2 Background

2.1 Meditation Pose Estimation Approach

In this paper, we use Mediapipe Pose, an open source framework developed by Google to attain estimates of 2D human joint coordinates in each image frame. This task to identify key body locations, analyze posture, and categorize movements. This task uses machine learning (ML) models that work with images , video or in real-time. The pose landmarker model

[20] track 33 landmark sites on the body, which roughly correspond to the locations of body parts (Figure 2). It is extensively utilized in motion analysis and fitness, sports, and health applications. Among the estimated landmarks, we use 12 points to detect meditation posture, which are positions 11, 12, 13, 14, 15, 16, 23, 24, 25, 26, 27 and 28 as shown in Figure 2.

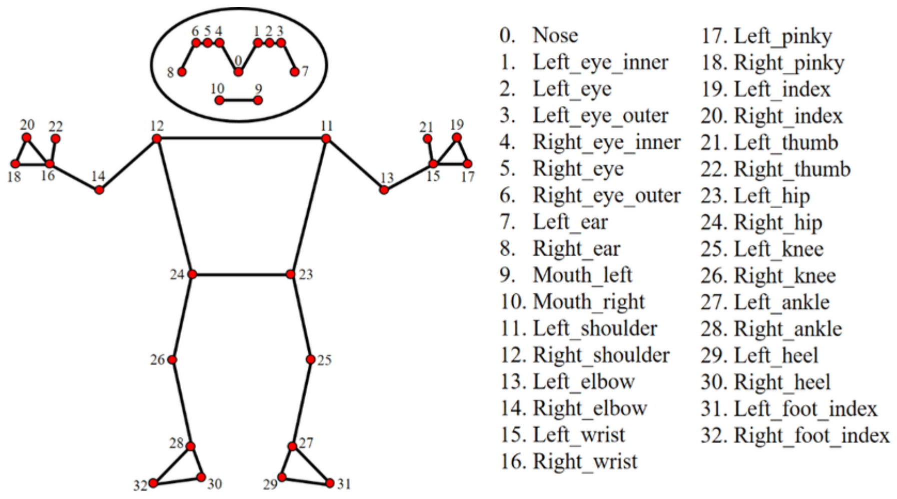


Figure 2: 33 pose landmarks.[21]

Our meditation posture estimation process calculates angles at the following positions: hips, shoulders, elbows, knees.Calculate the angle between three points using the arctan function as follows:

$$\text{angle} = \arctan 2(c_y - b_y, c_x - b_x) - \arctan 2(a_y - b_y, a_x - b_x)$$

Where (a_x, a_y), (b_x, b_y), and (c_x, c_y) respectively represent the coordinates of three points.

For example, you want to calculate the angle at the shoulder:

- (a_x, a_y) represents the coordinates of point **a**, which is the right elbow,
- (b_x, b_y) represents the coordinates of point **b**, which is the right shoulder, and
- (c_x, c_y) represents the coordinates of point **c**, which is the right hip.

Aiming for a comfortable meditation position for users, we use a basic meditation posture as a standard: **The Quarter Lotus** (Figure 3).

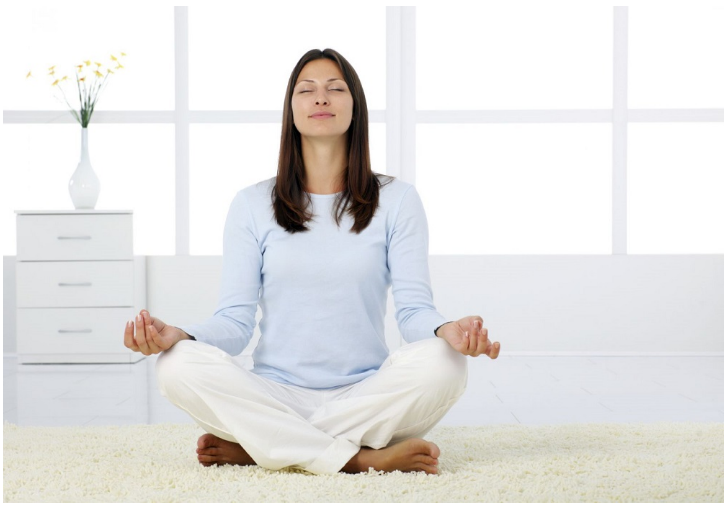
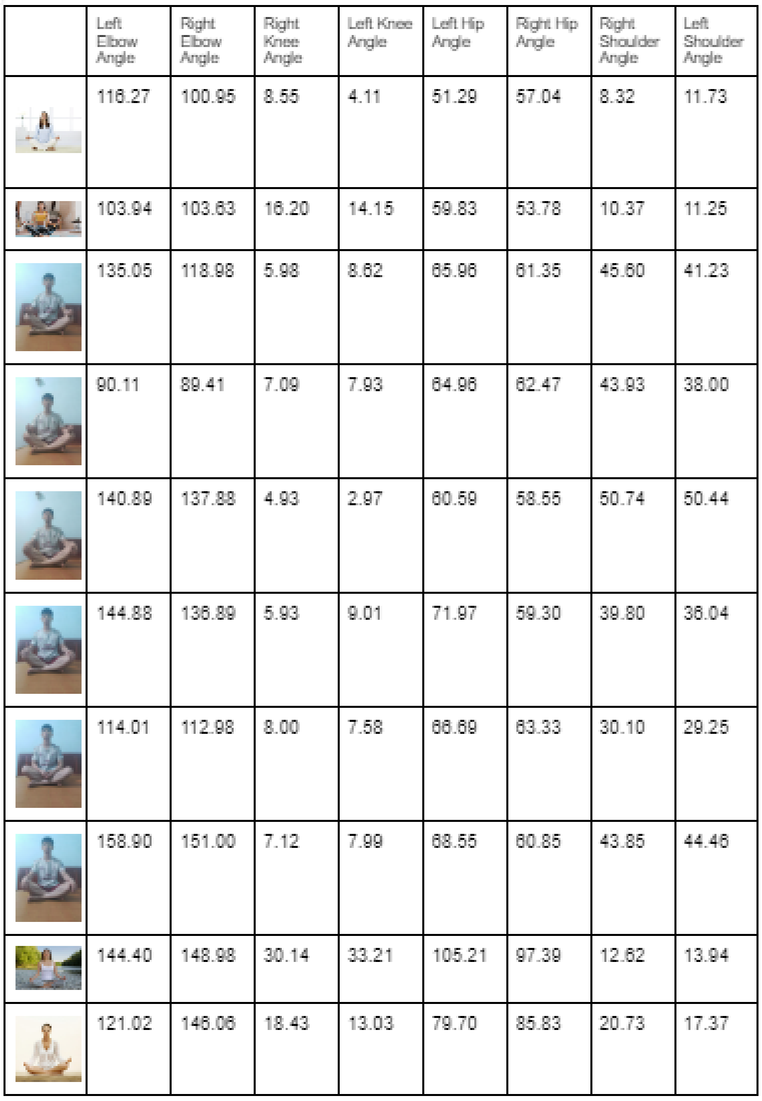


Figure 3: The Quarter Lotus.[22]

After calculating through a number of images, the angles are converted to degrees and extracted in Table 1.



	Left Elbow Angle	Right Elbow Angle	Right Knee Angle	Left Knee Angle	Left Hip Angle	Right Hip Angle	Right Shoulder Angle	Left Shoulder Angle
	116.27	100.95	8.55	4.11	51.29	57.04	8.32	11.73
	103.94	103.63	16.20	14.15	59.83	53.78	10.37	11.25
	135.05	118.98	5.98	8.62	65.96	61.35	45.60	41.23
	90.11	89.41	7.09	7.93	64.96	62.47	43.93	38.00
	140.89	137.88	4.93	2.97	60.59	58.55	50.74	50.44
	144.88	136.89	5.93	9.01	71.97	59.30	39.80	36.04
	114.01	112.98	8.00	7.58	66.69	63.33	30.10	29.25
	158.90	151.00	7.12	7.99	68.55	60.85	43.85	44.46
	144.40	148.98	30.14	33.21	105.21	97.39	12.62	13.94
	121.02	146.06	18.43	13.03	79.70	85.83	20.73	17.37

Table 1: Angles measured for different body parts

To ensure accurate recognition of the meditation stance during real-time operation, some adjustments were made to pose tracking. These adjustments involved finding the appropriate limits for each angle based on real-time experiments. After conducting these experiments, we established the following predicted limits for each angle to enable effective meditation stance recognition:

- **Elbow angle:** 60° - 160°
- **Shoulder angle:** 10° - 60°
- **Hip angle:** 50° - 110°
- **Knee angle:** 0° - 40°

2.2 Face Detection by ML kit

The proliferation of mobile devices equipped with powerful cameras has paved the way for a surge in applications leveraging computer vision technologies [8]. Facial recognition, in particular, has garnered significant attention due to its potential to revolutionize various sectors, ranging from security and authentication to healthcare and entertainment [5].

Traditional approaches to face detection and analysis, often reliant on complex algorithms like Viola-Jones [23] and extensive computational resources, posed challenges for real-time implementation on mobile platforms. However, advancements in machine learning, coupled with the increasing computational capabilities of modern smartphones, have enabled the development of efficient and accurate on-device face detection solutions [24].

ML Kit, Google's comprehensive machine learning SDK, provides a suite of ready-to-use APIs for Android and iOS developers [10]. The Face Detection API within ML Kit stands out as a versatile tool for incorporating facial recognition features into mobile applications [10]. It employs sophisticated deep learning models, trained on extensive datasets, to deliver robust face detection and attribute extraction capabilities.

This surge in accessible face detection technology has spurred a wave of innovative applications. In the realm of social media and photography, real-time face detection enables features like automatic tagging, augmented reality filters, and personalized avatars [25]te. In healthcare, it aids in monitoring patient emotions and detecting potential health issues through facial cues [26]. Security systems utilize face detection for access control and surveillance [27]. Moreover, the gaming industry has embraced face detection to create immersive experiences that respond to players' expressions [28].

This research seeks to tap into the potential of ML Kit's Face Detection API by developing an Android application capable of real-time facial analysis. The study investigates the practical implementation of face detection, landmark identification, emotion classification, and head pose estimation, demonstrating the efficacy of ML Kit for on-device, real-time facial analysis on Android platforms.

2.3 Deep Siamese Networks for Image Verification

The first Siamese networks were proposed in the beginning of the 1990 by Bromley and LeCun with an aim of solving the problem of signature verification posing it as an image matching task [29]. Siamese neural network is made up of two networks where the networks take two different inputs, and are linked with an energy function at the final layer. This function calculates a distance between the highest level features between the sides of the network (Figure 1). The parameters of twin networks are same and hence two nearly similar images cannot be placed at far off positions in the feature space because both the twin networks compute the same function. Additionally, the output of the network will be symmetric; that means if two different images are fed to the twin networks, then the top joining layer should compute similar metric as that when the images are interchanged between the networks.

LeCun et al. used a contrastive energy function with two terms in it to make the energy of the similar pairs low and the dissimilar pairs high [30]. However, in this work, we use the weighted L1 difference between the twin feature vectors h1 and h2, having applied a sigmoid transformation function to the results, which will map the output into the interval [0, 1]. Therefore, in order to optimize weights of the network a cross-entropy loss function is suitable. In contrast to that, LeCun et al. , which learned the similarity metric implicitly defined by the energy loss in, we only fix it as described above following the DeepFace paper from Facebook [31].

In the best learned models we have multiple instances of convolution layers prior to the completely interconnected layers and the highest order power functions. Deep Convolutional neural networks (CNNs) showed significant performance in a large number of large-scale problems in computer vision and especially in the image recognition tasks ([32]; [33]; [34]; [35]).

Several facts make convolutional networks especially attractive. This suggests that local connectivity inherently decreases the number of parameters in the model, acting as a form of regularization. However, convolutional layers are generally more computationally demanding compared to some standard nonlinearities, such as ReLU. Further, in these networks, convolution operation's main role is to filter inputs in which each feature map filters inputs against patterns as pixel groupings. Hence, the outputs of each convolutional layer can be interpreted as important spatial features within the original input space, while they are invariant to simple transformations. Furthermore, the fast CUDA libraries make it possible to train the large convolutional network whether or not the training time will be extremely long ([36]; [33]; [34]).

In the subsequent sections, we provide more information about the architectures of the Siamese networks used in our study and the characteristics of the learning process that was employed by us.

2.3.1 Model

Our primary model is a Siamese convolutional neural network consisting of L layers, where each layer comprises N_l units. In this setup, $h_{1,l}$ indicates the hidden vector at layer l for the first twin, whereas $h_{2,l}$ signifies the hidden vector for the second twin. We apply Rectified Linear Units (ReLU) exclusively in the first $L-2$ layers, while the last layers utilize sigmoidal units.

The model is composed of a series of convolutional layers, each operating on a single channel with filters of different sizes and a consistent stride of 1. The number of convolutional filters is chosen as a multiple of 16 to enhance performance. The network employs a ReLU activation function on the resulting feature maps, which can optionally be followed by max-pooling using a filter size and stride of 2. Consequently, the k-th filter map in each layer is expressed as follows:

$$a_{1,m}^{(k)} = \text{max-pool}\left(\max\left(0, W_{l-1,l}^{(k)} * h_{1,(l-1)} + b_l\right), 2\right) \tag{1}$$

$$a_{2,m}^{(k)} = \text{max-pool}\left(\max\left(0, W_{l-1,l}^{(k)} * h_{2,(l-1)} + b_l\right), 2\right) \tag{2}$$

Here, $W_{l-1,l}$ represents the 3-dimensional tensor corresponding to the feature maps of layer l, with $*$ denoting the valid convolution operation that returns only the valid part of the convolution. The units in the final convolutional layer are flattened into a single vector. This is followed by a fully connected layer and then an additional layer that calculates the induced distance metric between the Siamese twins, which is then passed to a single sigmoidal output unit. Specifically, the prediction vector is defined as $p = \sigma\left(\sum_j \alpha_j \left\|h_{1,L-1}^{(j)} - h_{2,L-1}^{(j)}\right\|\right)$, where σ is the sigmoidal activation function. This final layer creates a metric in the learned feature space of the $(L-1)$-th hidden layer, assessing the similarity between the two feature vectors. The α_j parameters, which are learned during training, weigh the importance of each component-wise distance. This defines the final L-th fully connected layer, which merges the two Siamese twins.

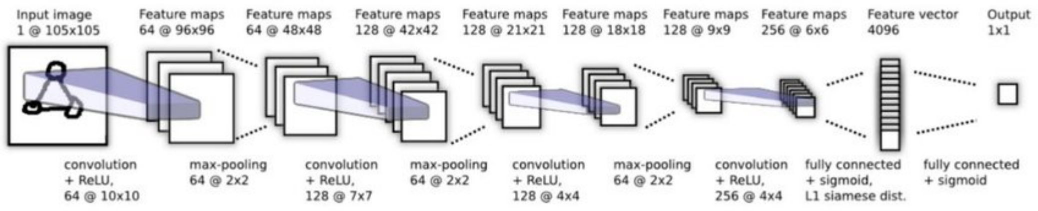


Figure 4: Best convolutional architecture selected for verification task. Siamese twin is not depicted, but joins immediately after the 4096 unit fully-connected layer where the L1 component-wise distance between vectors is computed. [37]

An example is illustrated above (Figure 2), displaying the largest version of our model that was tested. This network also delivered the best performance among all networks tested on the verification task.

2.3.2 Loss function

Let M denote the size of the minibatch, with i indexing the i-th minibatch. The vector $y(x_1^{(i)}, x_2^{(i)})$ of length M contains the labels for the minibatch, where $y(x_1^{(i)}, x_2^{(i)}) = 1$ whenever x_1 and x_2 are from the same character class and $y(x_1^{(i)}, x_2^{(i)}) = 0$ otherwise. We impose a regularized cross-entropy objective on our binary classifier of the following form:

$$\begin{aligned} L(x_1^{(i)}, x_2^{(i)}) = {} & y(x_1^{(i)}, x_2^{(i)}) \log p(x_1^{(i)}, x_2^{(i)}) \\ & + (1 - y(x_1^{(i)}, x_2^{(i)})) \log(1 - p(x_1^{(i)}, x_2^{(i)})) \\ & + \lambda^T |w|^2 \end{aligned} \tag{3}$$

2.3.3 Optimization

This goal is incorporated into the conventional backpropagation algorithm, which allows the gradient to accumulate across twin networks due to the shared weights. We use a fixed mini-batch size of 128 and use the Adam optimizer with a learning rate η_j set to 1×10^{-4}. To prevent overfitting, we also incorporate L2 regularization, where the regularization weights λ_j are assigned on a per-layer basis. The rule for updating the weights at epoch T follows:

$$w_{kj}^{(T)}\left(x_1^{(i)}, x_2^{(i)}\right) = w_{kj}^{(T)} + \Delta w_{kj}^{(T)}\left(x_1^{(i)}, x_2^{(i)}\right) + 2\lambda_j |w_{kj}|$$

Here, the weight change $\Delta w_{kj}^{(T)}\left(x_1^{(i)}, x_2^{(i)}\right)$ is computed based on the gradient of the loss function and uses momentum to accelerate convergence:

$$\Delta w_{kj}^{(T)}\left(x_1^{(i)}, x_2^{(i)}\right) = -\eta_j \nabla w_{kj}^{(T)} + \mu_j \Delta w_{kj}^{(T-1)}$$

Where ∇w_{kj} is the partial derivative of the loss function in reference to the weight connecting the jth neuron of one layer to the kth neuron in the subsequent layer.

2.3.4 Weight Setup

To achieve high performance, the convolutional layer weights in this model were initialized using a normal distribution with a mean of 0 and a standard deviation of 10^{-2}. Similarly, the biases in these layers were initialized from a normal distribution, with a mean of 0.5 and the same standard deviation. For the fully connected layers, the biases were initialized just like in the convolutional layers. However, the weights for the fully connected layers were drawn from a broader normal distribution, characterized by a mean of 0 and a standard deviation of 2×10^{-1}.

2.3.5 Learning Schedule

We allowed each layer to have a different learning rate. However, these rates were uniformly reduced across the network by 1 percent per epoch, resulting in $\eta_j^{(T)} = 0.99\eta_j^{(T-1)}$. By annealing the learning rate, the network could more easily converge to local minima without getting stuck in the error surface. Momentum was initially set to 0.5 in every layer, increasing linearly each epoch until reaching the individual momentum term μ_j for the (j)-th layer.

Each network was trained for up to 50 epochs. Training was stopped if the error on the validation set failed to improve after 10 epochs, and the parameters of the model from the epoch with the best performance were used. If the validation error decreased during the learning schedule, the model's final state was preserved.

2.4 Smart meditation app

The "Smart Meditation" app is a tool designed to help users practice meditation accurately and effectively. With the use of cutting-edge image recognition and analysis technologies, the app is designed to assist users in maintaining the proper posture for meditation and attaining maximum focus. The app's two primary features are as follows:

Meditation posture detection: The software keeps track of and evaluates the user's posture during meditation by using the camera. The technology assesses the user's posture and recognizes bodily landmarks automatically. The software helps the user improve their meditation skill by alerting them and pausing the timer if their posture is improper. If the posture is correct, the app confirms and begins timing the meditation session. The user's experience is optimized by this real-time input, which enables quick adjustments and a more customized meditation routine.

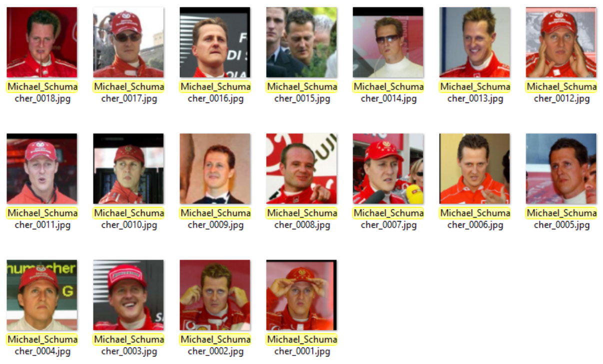
Face Detection and Verification: The software makes use of facial recognition technology to recognize and validate the user's identification during the meditation session. The technology verifies that the correct user is in a meditative state by analyzing facial features and applying a deep learning model (Siamese Neural Network). This guarantees that each meditation session is unique to each user while simultaneously improving personalization and privacy.

3 Dataset

3.1 Labeled Faces in the Wild

The LFW dataset is a collection of labeled face images, primarily intended for research in the field of face recognition. This dataset contains over 13,000 images of different individuals, collected from publicly available sources on the internet.

The reason for selecting the LFW dataset is that it provides a rich and diverse source of facial data under various conditions, enabling your AI


Michael_Schumacher_0018.jpg
Michael_Schumacher_0017.jpg
Michael_Schumacher_0016.jpg
Michael_Schumacher_0015.jpg
Michael_Schumacher_0014.jpg
Michael_Schumacher_0013.jpg
Michael_Schumacher_0012.jpg
Michael_Schumacher_0011.jpg
Michael_Schumacher_0010.jpg
Michael_Schumacher_0009.jpg
Michael_Schumacher_0008.jpg
Michael_Schumacher_0007.jpg
Michael_Schumacher_0006.jpg
Michael_Schumacher_0005.jpg
Michael_Schumacher_0004.jpg
Michael_Schumacher_0003.jpg
Michael_Schumacher_0002.jpg
Michael_Schumacher_0001.jpg

h

Figure 5: 18 images for Michael Schumacher in LFW dataset

model to learn to recognize faces in a wide range of real-world situations.

This dataset is suitable for research related to face recognition, emotion classification, or other applications involving facial image processing.
The LFW dataset consists of directories corresponding to individual persons. Each directory contains multiple images of the same person taken from different angles and with varying facial expressions. For example, the directory `George_HW_Bush` contains 13 images of George H.W. Bush, while the directory `Michael_Schumacher` contains 18 images of Michael Schumacher.

Each image in the dataset is labeled with the name of the person in the image. The main attribute of the data is the facial image, while the label is the name of the individual. The goal is to use facial features to train an AI model to recognize or classify these faces.

The LFW dataset is not balanced in terms of the number of images per individual. Some people have many images (e.g., 18 images for Michael Schumacher), while others have only a few (e.g., 1 image for Curtis Strange). This distribution can be visualized through a chart to give readers a clearer understanding of the diversity and balance of the data.

3.2 MyFace300

In addition to the LFW dataset, personal images were also utilized in the project. You created two separate data files, each containing 300 images of your face, with a size of 100x100 pixels.

Besides using the LFW dataset to enhance the diversity and generalization of the model, your personal images were used to customize and evaluate the model under specific conditions. This could involve testing the model's accuracy with data it has not frequently encountered during training.

Using personal images adds an additional layer of practical testing for the model, helping to assess its performance and accuracy in specific applications such as face recognition or authentication.

In addition to the LFW data, you have two separate data files, each containing 300 images of your face, totaling 600 images, all with a size of 100x100 pixels. This ensures that the model can effectively process both public dataset images and personal images.

Personal images not only provide additional data for the model but also help test the model's accuracy in recognizing specific face.

3.3 Data Preprocessing

3.3.1 Labeled Faces in the Wild

After extracting the file, place the Labeled Faces in the Wild dataset into the data folder and randomly select 300 images to save into the negative folder. The images are read from the file and decoded from the JPEG format. Then, the images are resized to 100x100 pixels to ensure consistent input size for the model. Next, the pixel values of the images are normalized to the range [0, 1] by dividing by 255, which helps improve the model's performance. Finally, the processed images are returned and are ready to be fed into the machine learning model.

3.3.2 MyFace300

The images are captured continuously, with 300 face images saved into the positive folder and 300 images saved into the anchor folder. The images are read from the file and decoded from the JPEG format. Then, the images are resized to 100x100 pixels to ensure consistent input size for the model. Next, the pixel values of the images are normalized to the range

Figure 6: LFW data

[0, 1] by dividing by 255, which helps improve the model's performance. Finally, the processed images are returned and are ready to be fed into the machine learning model.

4 Methodology

4.1 Mediapipe

4.1.1 Input Processing

- Process: By utilizing the CameraX API, the application uses the front camera of the device to record live video input. The MediaPipe processing pipeline receives a continuous stream of video frames for analysis.
- Real-time Processing: By capturing frames at a predetermined rate, the MediaPipe model can process them instantly and without any discernible lag. This is essential to give the user instant feedback on how they are doing in their meditation stance.

4.1.2 Pose Detection

- Graph Setup: To process the incoming video stream and identify and track human body landmarks, a MediaPipe graph was set up with a combination of TensorFlow Lite models and custom nodes (Figure 7). The "pose_tracking_gpu.binarypb" graph was used to leverage GPU acceleration for faster processing.

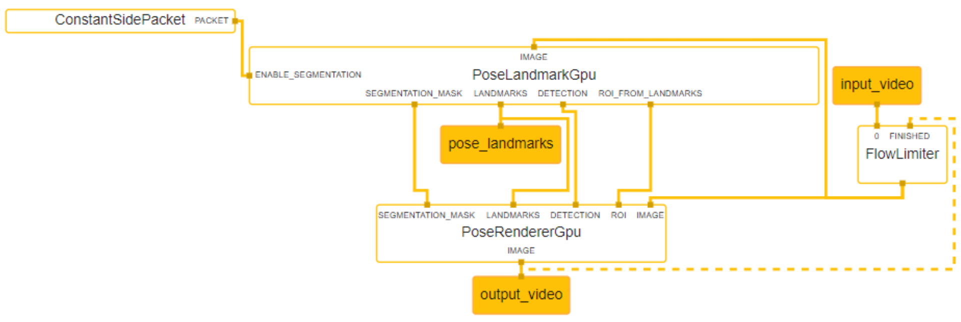


Figure 7: Mediapipe graph-view

- Landmark DetectionThe graph outputs landmark data for key points on the human body (e.g., shoulders, hips, knees). This data is used to calculate angles and assess whether the user is in the correct meditation pose.

4.1.3 Pose Evaluation

- Angle Calculation: The angles between particular joints, such as the knee-hip-shoulder, are calculated by processing the landmark data. The user's compliance with the proper meditation position is then assessed by comparing these angles to predetermined thresholds.
- Feedback Mechanism: The application shows the user real-time feedback by showing messages such as "Meditation Pose" or "Wrong Meditation Pose" right on the screen, based on the angle calculations.

4.1.4 User Interface (UI)

- Design: The UI was made to be simple and easy to use, with an emphasis on giving the user fast, unambiguous feedback. A simple overlay is used to display pose status messages, while the video feed is shown in the background.
- Interaction: Using a timer, the UI shows how long the ideal meditation stance will last and lets users start and stop the pose identification process. The UI dynamically updates based on the user's posture.

4.2 ML Kit Integration

4.2.1 Project setup and ML Kit integration

The initial phase involves the creation of a new Android project within the Android Studio development environment. Subsequently, the requisite ML Kit dependencies are incorporated into the project's build.gradle file, ensuring the availability of the necessary libraries and functionalities.

Following the integration of ML Kit, the Face Detection API is instantiated, configured with specific options to suit the project's requirements. These options might encompass the detection of facial landmarks, classification of facial expressions, and other relevant parameters [10].

4.2.2 Camera integration

To facilitate real-time face analysis, the device's camera is seamlessly integrated into the application through the implementation of CameraX [38]. This library provides a streamlined interface for accessing the camera and acquiring a live video stream.

The preview use case is configured to render the camera feed directly onto the device's screen, enabling users to visualize the real-time input. Concurrently, the image analysis use case is set up to process each individual frame captured by the camera, paving the way for subsequent face detection and analysis [38].

4.2.3 Face detection and analysis

Within the image analysis use case, each camera frame is passed to ML Kit's Face Detector for meticulous analysis. The detector, equipped with pre-trained deep learning models, adeptly identifies faces within the frame, providing precise bounding box coordinates for each detected face [10].

These bounding boxes are then used to crop the original frame, isolating the region containing the detected face. This cropped image can then be used for further processing or analysis, such as facial landmark detection, emotion classification, or head pose orientation determination, leveraging the capabilities of ML Kit's API[10].

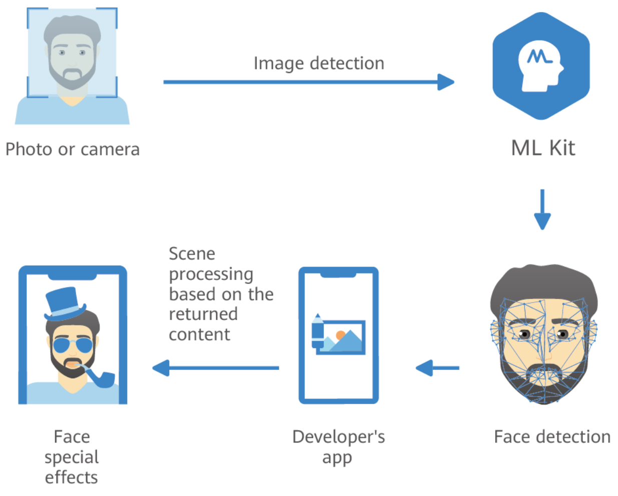


Figure 8: Workflow of Real-Time Face Detection Using ML Kit in Mobile Applications. [39]

4.2.4 Result presentation

To enhance the user experience and visualize the detected faces, bounding boxes and facial landmarks are superimposed onto the live camera preview. Additionally, the extracted facial attributes are presented in a clear and comprehensible manner, offering users real-time insights into the analyzed facial data.

4.3 One-shot learning

This methodology utilizes a supervised learning approach with Siamese neural networks to tackle the problem of one-shot learning. These networks are trained to differentiate between pairs of images, focusing mainly on character recognition. The model employs convolutional layers to learn generic image features, which are then applied to one-shot learning tasks. The architecture consists of two identical networks, combined at the output by a similarity measurement function, and trained using a cross-entropy loss function. This method emphasizes the generalization of learned features to unseen classes and has demonstrated strong performance in one-shot classification tasks.

After completing the training process, the model is converted to the TensorFlow Lite (TFLite) format for optimized performance on mobile devices. The model is then integrated into Android Studio and deployed using Kotlin. In the Android application, the TFLite model is used to handle the image processing logic, from input data acquisition to image classification based on the learned features. These tasks are executed with high efficiency and low resource requirements, suitable for mobile devices.

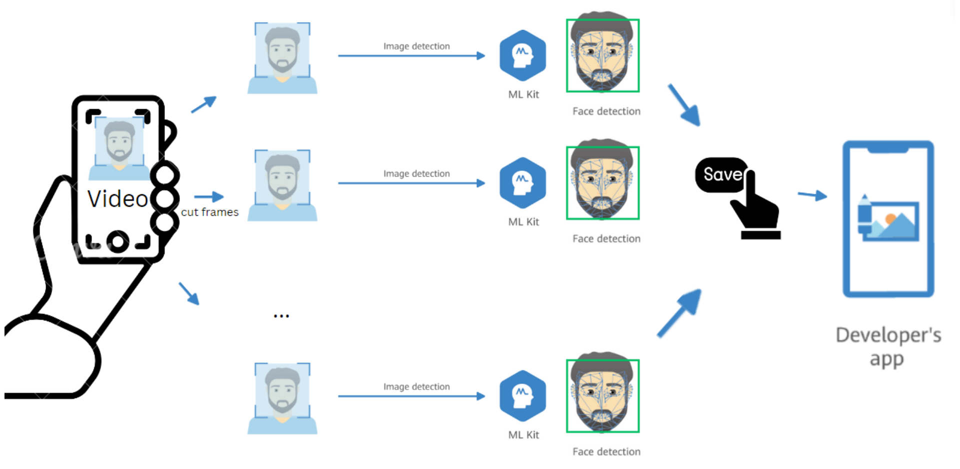


Figure 9: Crop the frame to store comparison data

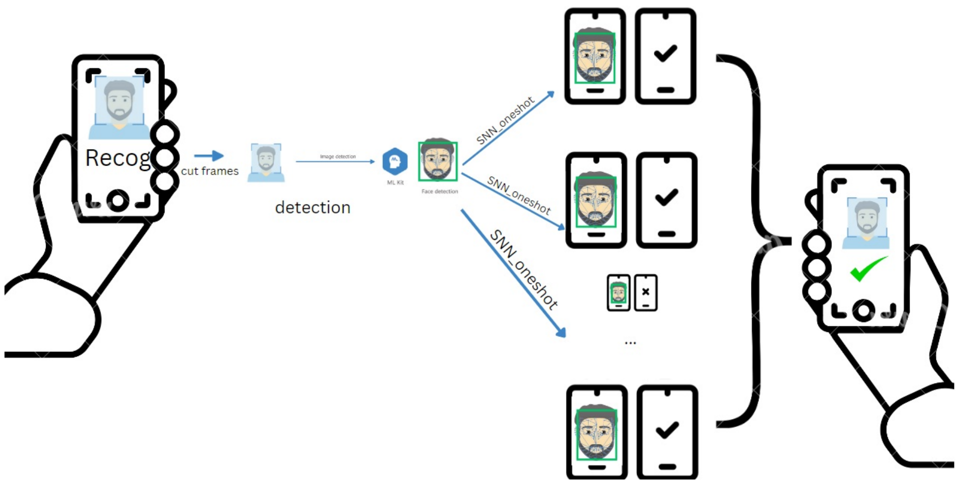


Figure 10: Comparison through the SNN model

5 Experiments and Result

5.1 Hardware Configuration

These are the device parameters we used to train the SSN one-shot learning model.

Component	Specification
CPU	Intel(R) Core(TM) i7-10700 CPU
GPU	NVIDIA GeForce GTX 1650
RAM	32 GB 64-bit

Table 2: Configuration of training device

In addition to training models on computers, we also utilize Android devices to conduct real-world testing. By using Android Studio, we can create and run virtual machines directly on the Android platform, allowing us to test and optimize the application right on the phone. This not only ensures compatibility but also enhances performance and user experience before the official deployment.

5.2 Experiments

In this experiment, the data was meticulously prepared before being fed into the model for training. First, the data was loaded and underwent preprocessing through the preprocess_twin function, where each pair of input images was normalized to ensure consistency during training. After preprocessing, the data was shuffled randomly with a buffer size of 1024, helping to reduce dependency on the order of samples in the training set, and was cached to optimize data retrieval speed in training loops.

The data was then split into two parts: 70% was used for training and the remaining portion for testing the model's performance. To ensure efficiency during training, the training data was divided into small batches of 16 samples each. Additionally, the data was prefetched with a depth of 8, allowing for data to be loaded into memory beforehand, ensuring that the training process was not interrupted by data shortages.

The model was trained over 50 epochs. In each epoch, the model underwent a loop over the batches of data, where the train_step function was called to update the model's weights based on the calculated loss. The model's performance metrics, including Recall and Precision, were continuously monitored and updated after each batch to assess the model's ability to accurately classify image pairs.

Throughout the training process, progress was displayed via a progress bar (Progbar), providing users with visual information about the current state of the process. To ensure continuity in training, model checkpoints were saved every 10 epochs. This allowed the model to be restored and continue training in the event of an interruption, or simply to preserve the model's state at key stages.

At the end of the training process, metrics like Recall and Precision were expected to gradually improve with each epoch, reflecting an enhancement in the model's ability to accurately differentiate between image pairs. The checkpointing process also ensured that the model could continue to evolve or be evaluated for stability across different phases of training.

5.3 Result

Condition	**ML Kit Face Detection**	**One-shot Learning SNN**	**Meditation Pose Detection**
Accuracy (Normal Lighting)	98%	85%	92%
Accuracy (Low Lighting)	95%	78%	88%
Accuracy (Varied Angles/Distances)	95%	82%	72%
Accuracy (Multi-Person)	92%	N/A	87%
Training Time	N/A	About 4-5 hours	N/A
Environment	On-Device	Pycharm	On-Device

Table 3: Comparison of Different AI Models and Applications

5.3.1 Face detection Accuracy

The ML Kit face detection system displayed exceptional accuracy across various conditions, proving its reliability within the Smart Meditation app. The dataset used in this study is our own, collected through approximately 100 experimental trials. In normal lighting conditions, the face detection model achieved an accuracy of 98%, demonstrating its capability to detect and identify faces with high precision. This level of accuracy is crucial for the app's functionality, as it ensures that the meditation session only begins when the user's face is correctly detected, preventing any false starts. Even in less-than-ideal low-light conditions, the model maintained a strong performance with a 95% accuracy rate, highlighting its robustness in handling different lighting scenarios.

When tested with varied angles and distances, the face detection system continued to perform reliably, maintaining an accuracy of 95%. This indicates that the system can accurately detect faces even when users are not directly facing the camera, which is essential for real-world use where perfect positioning is not always possible. In multi-person scenarios, the accuracy slightly decreased to 92%, but it still provided reliable performance, ensuring that the app can function effectively in group settings. The strong performance across these varied conditions underscores the reliability and versatility of ML Kit face detection in the Smart Meditation app.

5.3.2 SNN Verification Accuracy

Based on the Data (the accuracy of the model will depend on the data and lighting conditions of the input image as well as the verification image), the Siamese Neural Network (SNN) model used once for user verification has achieved commendable results, especially under controlled conditions. With 85% accuracy under normal lighting conditions, the SNN model has proven its effectiveness in verifying users based on facial features. This level of accuracy is sufficient to meet the needs of the application, ensuring that only authorized users can initiate a meditation session. However, the accuracy drops to 78% under low-light conditions, indicating that the model is more sensitive to lighting variations, which can affect the reliability of user verification in darker environments.

The SNN model also encountered challenges with different angles and distances, where the accuracy was recorded at 82%. This shows that the model's ability to recognize and verify users decreases as the face angle or distance from the camera deviates from the optimal settings. Since the SNN is not intended for multi-user scenarios, this aspect was not tested, as the primary goal of the SNN is to verify individual users rather than manage multiple users at once. These findings suggest that while the SNN model performs well under ideal conditions, further optimization may be needed to improve its robustness in more challenging environments.

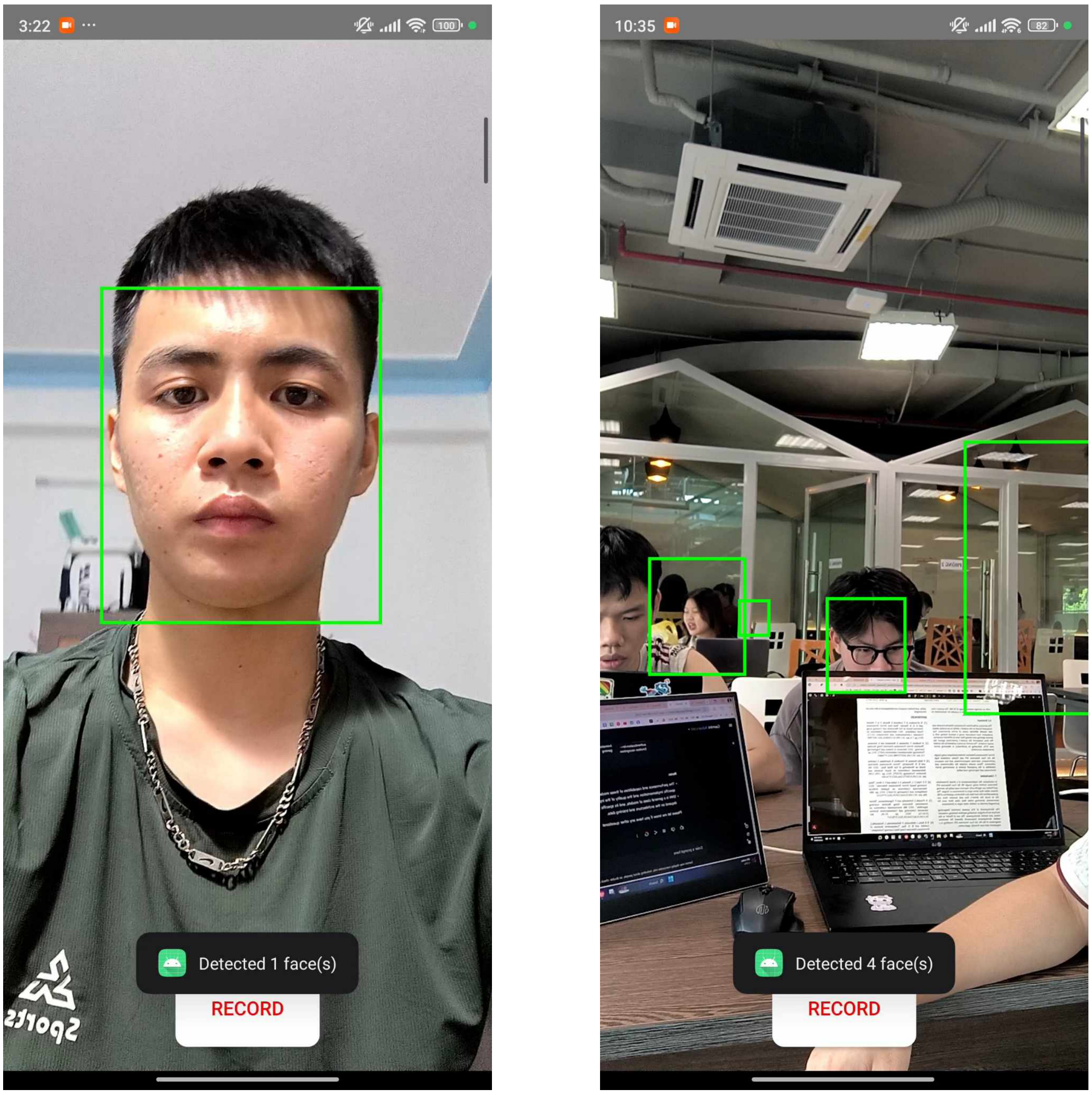


Figure 11: Real-time Face Detection System in One-Person and Multi-Person Scenarios

5.3.3 Posture Verification Precision

The MediaPipe Pose Detection API was integrated into the Smart Meditation app to ensure that users maintain the correct meditation posture, and it demonstrated high precision in this task. Under normal lighting conditions, the API achieved a precision rate of 92%, effectively identifying correct postures and contributing to the app's overall functionality. This high precision is vital as it ensures that users are guided to maintain proper posture throughout their meditation session, which is a key aspect of the app's goal to promote effective meditation practices. However, the precision rate slightly declined to 88% in low-light conditions, suggesting that while the API is reliable, it may struggle slightly in environments with poor lighting.

The precision of the posture detection system was notably affected by varied angles and distances, with a precision rate of 72%. This indicates that the system finds it challenging to accurately detect and verify postures when users are not positioned directly in front of the camera or are at varying distances. Despite this, the system performed well in multi-person scenarios, maintaining an accuracy of 87%, demonstrating its ability to handle group meditation sessions effectively. These results highlight the importance of ensuring proper user positioning during meditation sessions to maximize the accuracy of posture verification.

5.3.4 Training Time and Environment

The SNN model training process took approximately 4-5 hours, conducted using Python in the PyCharm IDE. The training was executed in a controlled environment with specific hardware, including Realme 5 Pro, Xiaomi 12T, and Android Virtual Devices (AVDs) on Android Studio. The training time reflects the complexity of the one-shot learning model, as it requires careful adjustment of hyperparameters to achieve the desired level of accuracy. The on-device environments for both ML Kit face detection and MediaPipe Pose Detection ensured real-time processing, contributing to the app's efficiency and responsiveness.

The choice of environment for each component was made to optimize performance and ensure seamless integration within the Smart Meditation app. ML Kit face detection and MediaPipe Pose Detection were run entirely on-device, eliminating the need for continuous internet connectivity and enhancing the app's privacy and security features. The combined use of these environments and tools underscores the app's capability to operate effectively across different devices, providing a consistent user experience regardless of the hardware used. This combination of training and runtime environments played a critical role in achieving the app's high accuracy and efficiency.

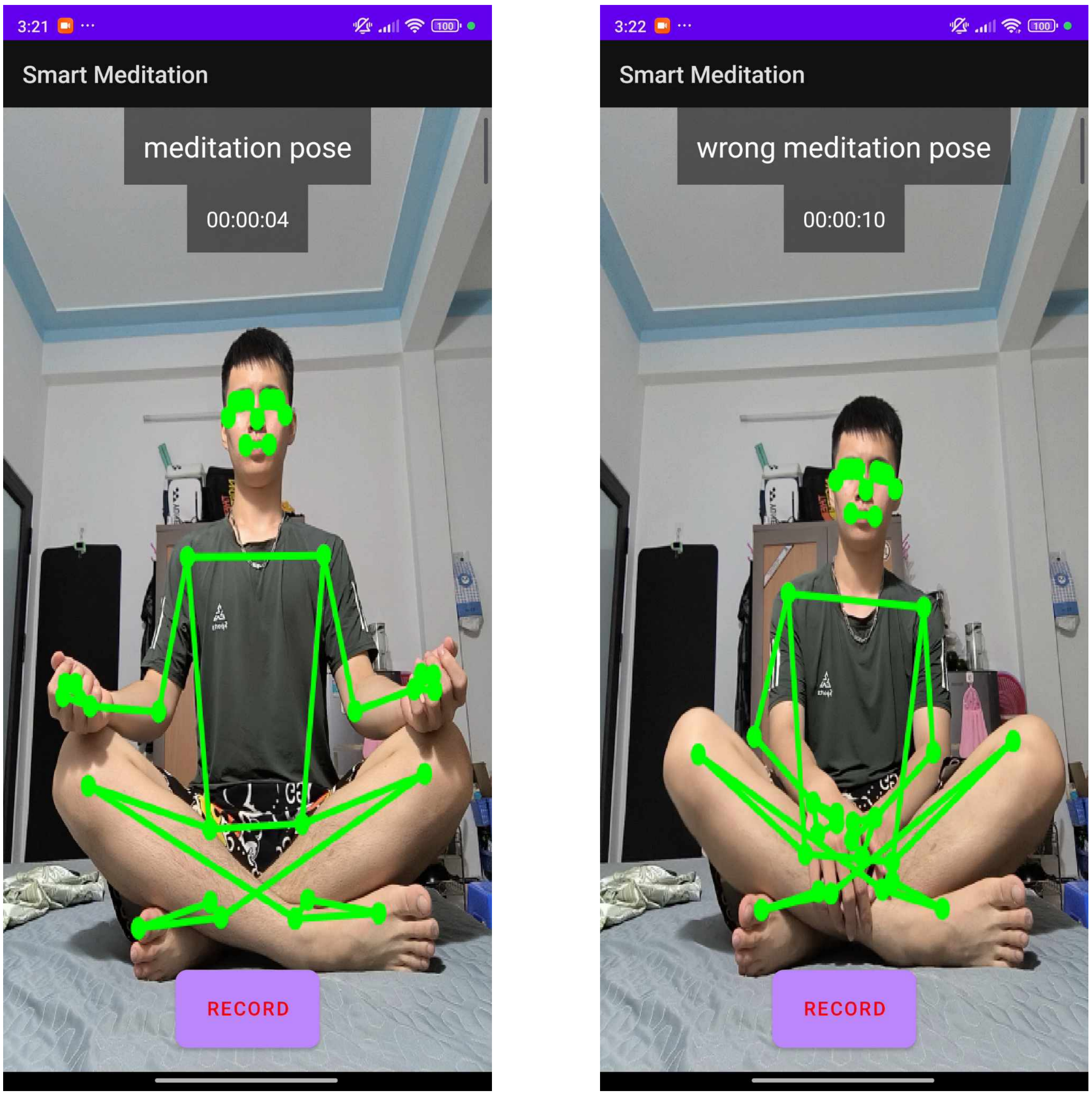


Figure 12: Real-time Meditation Pose Detection System using Mediapipe.

6 Conclusion and Future Works

6.1 Conclusion

This thesis has explored the integration of advanced machine learning models to enhance the accuracy and effectiveness of meditation posture detection. The models employed include ML Kit Face Detection, SNN Face Verification, and MediaPipe Pose Detection, each contributing uniquely to the overall system.

ML Kit Face Detection was utilized primarily for its robust real-time face detection capabilities. This model enabled precise identification and cropping of facial regions, ensuring that the system accurately recognized the user's face during meditation sessions. SNN Face Verification played a crucial role in verifying the identity of the user, adding an extra layer of personalization and security to the meditation experience. These two models worked in tandem to ensure that the system only activated when the correct user was identified in the correct meditation posture.

The focus of this research, however, has been on the application of MediaPipe Pose Detection for detecting and analyzing meditation postures. MediaPipe's powerful pose estimation framework allowed for real-time tracking of key body landmarks, enabling accurate calculation of joint angles and assessment of meditation postures. By focusing on essential joints such as the hips, shoulders, elbows, and knees, the system effectively identified deviations from the correct posture, providing immediate feedback to users.

Through extensive testing, it was found that the combination of these models offered a robust solution for monitoring and correcting meditation postures. However, challenges were encountered in maintaining accuracy across different body types, clothing, and environmental conditions. Additionally, while the current system provides a solid foundation, there is room for further enhancement.

6.2 Future Works

In the future, the primary objective is to integrate the three models—ML Kit Face Detection, SNN Face Verification, and MediaPipe Pose Detection—into a unified application named the Smart Meditation App. This app will be designed to verify the user's identity and monitor their meditation pose, ensuring that both conditions are met simultaneously before starting a meditation session timer.

6.2.1 Model Integration

The integration process will begin with ensuring effective communication between the three models within the application. The ML Kit Face Detection model will be responsible for detecting a face in real-time. Once a face is detected, the SNN Face Verification model will verify whether the detected face matches a pre-registered user. This verification ensures that only authorized users can start a meditation session.

6.2.2 Meditation Pose Detection

Following successful user verification, the MediaPipe Pose Detection model will analyze the user's body posture. This model will track key landmarks on the user's body and calculate the angles between joints to determine if the user is maintaining the correct meditation pose. The accuracy of pose detection is critical, as the timer will only start if the user's posture is correct.

6.2.3 Timer Functionality

The core functionality of the app revolves around its timer mechanism. The timer will only run when both conditions—user verification and correct meditation pose—are satisfied. If either condition fails at any point during the session, the timer will stop immediately. This feature ensures that users maintain the correct posture throughout their meditation practice, enhancing the effectiveness and safety of the session.

6.2.4 Challenges and Future Enhancements

Challenges anticipated in this integration include ensuring the robustness of the models in varying lighting conditions, different user appearances, and diverse environments. Addressing these challenges will require iterative testing and optimization.

Future enhancements could involve incorporating additional sensors or feedback mechanisms to further ensure that users maintain correct posture. Additionally, expanding the app's functionality to include more advanced meditation guidance features could be explored.

In conclusion, the integration of these three models into the Smart Meditation App represents a significant step towards creating a reliable tool that not only verifies user identity but also ensures correct meditation posture, enhancing the overall meditation experience.

References

[1] R. Welekar, A. Dubey, and S. Hablani, “Meditation accuracy detection system using deep learning”, *Multimedia Tools and Applications* **82** (2023), pp. 43625–43633, DOI: `10.1007/s11042-023-15273-5`.

[2] L. Chen, H. Xu, and J. Wang, “Facial expression recognition in meditation using deep convolutional neural networks”, *Pattern Recognition Letters* **140** (2021), pp. 245–252.

[3] Y. Li, S. Wang, and Z. Hu, “Full-body posture recognition for meditation using 3D sensors”, *Sensors* **20**.(18) (2020), p. 5287.

[4] X. Yang, Q. Li, and P. Zhang, “Posture monitoring and correction system for meditation using wearable motion sensors”, *IEEE Transactions on Human-Machine Systems* **49**.(3) (2019), pp. 276–286.

[5] Wenyi Zhao et al., “Face recognition: A literature survey”, *ACM Computing Surveys (CSUR)* **35**.(4) (2003), pp. 399–458.

[6] Navneet Dalal and Bill Triggs, “Histograms of oriented gradients for human detection”, *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1, IEEE, 2005, pp. 886–893.

[7] Kaipeng Zhang et al., “Joint face detection and alignment using multitask cascaded convolutional networks”, *IEEE Signal Processing Letters* **23**.(10) (2016), pp. 1499–1503.

[8] Gary Bradski, “The OpenCV Library”, *Dr. Dobb’s Journal of Software Tools* (2000).

[9] Davis E King, “Dlib-ml: A machine learning toolkit”, *Journal of Machine Learning Research* **10** (2009), pp. 1755–1758.

[10] Google, *ML Kit — Google for Developers*, `https://developers.google.com/ml-kit`, 2023.

[11] Li Fei-Fei, Robert Fergus, and Pietro Perona, “A bayesian approach to unsupervised one-shot learning of object categories”, *Proceedings Ninth IEEE International Conference on Computer Vision*, IEEE, 2003, pp. 1134–1141.

[12] Li Fei-Fei, Robert Fergus, and Pietro Perona, “One-shot learning of object categories”, *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **28**.(4) (2006), pp. 594–611.

[13] Brenden M. Lake, Ruslan R. Salakhutdinov, and Joshua B. Tenenbaum, “One-shot learning by inverting a compositional causal process”, *Advances in Neural Information Processing Systems*, 2013, pp. 2526–2534.

[14] Brenden M. Lake et al., “One shot learning of simple visual concepts”, *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*, vol. 172, 2011.

[15] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum, “Concept learning as motor program induction: A large-scale empirical study”, *Proceedings of the 34th Annual Conference of the Cognitive Science Society*, 2012, pp. 659–664.

[16] Brenden M Lake et al., “One-shot learning of generative speech concepts”, *Cognitive Science Society*, 2014.

[17] Andrew Maas and Charles Kemp, “One-shot learning with bayesian networks”, *Cognitive Science Society*, 2009.

[18] Joseph Jaewhan Lim, “Transfer learning by borrowing examples for multiclass object detection”, Master’s thesis, Massachusetts Institute of Technology, 2012.

[19] Linchang Zhao et al., “Siamese Dense Neural Network for Software Defect Prediction With Small Data”, *IEEE Access* **PP**.(99) (Dec. 2018), pp. 1–1, DOI: 10.1109/ACCESS.2018.2889061, **available**: https://doi.org/10.1109/ACCESS.2018.2889061.

[20] Google AI, *MediaPipe Pose: Real-time Pose Detection*, https://github.com/google-ai-edge/mediapipe/blob/master/docs/solutions/pose.md, Accessed: 2024-08-19, 2024.

[21] Chen Kuan-Yu et al., “Fitness Movement Types and Completeness Detection Using a Transfer-Learning-Based Deep Neural Network”, *Sensors* **22**.(15) (July 2022), DOI: 10.3390/s22155700, **available**: https://doi.org/10.3390/s22155700.

[22] California Fitness & Yoga, *12 tu the thien dung cach, thien tinh tam*, https://cali.vn/blog/ngoi-thien-dung-cach, Accessed: 2024-09-04, 2024.

[23] Paul Viola and Michael Jones, “Rapid object detection using a boosted cascade of simple features”, *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, IEEE, 2001, pp. I–I.

[24] Abhishek Kumar, Anamika Kaur, and M Kumar, “Face detection techniques: a review”, *Artificial Intelligence Review* **52** (2019), pp. 927–948.

[25] Jiankang Deng et al., “Arcface: Additive angular margin loss for deep face recognition”, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, 2019, pp. 4690–4699.

[26] S L Happy and Aurobinda Routray, “Automatic facial expression recognition using features of salient facial patches”, *IEEE transactions on Affective Computing* **6**.(1) (2015), pp. 1–12.

[27] Kresimir Delac, Mislav Grgic, and Panos Liatsis, “Appearance-based statistical methods for face recognition”, *Proceedings of the 47th International Symposium ELMAR, 2005*, IEEE, 2005, pp. 151–154.

[28] Stylianos Asteriadis, Kostas Karpouzis, and Stefanos Kollias, “Visual emotion recognition using facial expression analysis”, *International Workshop on Image Analysis for Multimedia Interactive Services, 2005*, IEEE, 2005, pp. 11–14.

[29] Jane Bromley et al., “Signature Verification Using a Siamese Time Delay Neural Network”, *International Journal of Pattern Recognition and Artificial Intelligence* **7**.(04) (1993), pp. 669–688.

[30] Sumit Chopra, Raia Hadsell, and Yann LeCun, “Learning a Similarity Metric Discriminatively, with Application to Face Verification”, *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, 2005, pp. 539–546.

[31] Yaniv Taigman et al., “DeepFace: Closing the Gap to Human-Level Performance in Face Verification”, *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 2014, pp. 1701–1708.

[32] Yoshua Bengio, “Learning Deep Architectures for AI”, *Foundations and Trends in Machine Learning* **2**.(1) (2009), pp. 1–127.

[33] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks”, *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[34] Karen Simonyan and Andrew Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, *arXiv preprint arXiv:1409.1556* (2014).

[35] Nitish Srivastava, “Improving Neural Networks with Dropout”, MA thesis, University of Toronto, 2013.

[36] Volodymyr Mnih, *Cudamat: A CUDA-Based Matrix Class for Python*, 2009.

[37] Ruslan Salakhutdinov Gregory Koch Richard Zemel, “Siamese Neural Networks for One-shot Image Recognition”, *Department of Computer Science, University of Toronto* (2015), GKOCH@CS.TORONTO.EDU, ZEMEL@CS.TORONTO.EDU, RSALAKHU@CS.TORONTO.EDU.

[38] Android Developers, *CameraX Overview — Android Developers*, `https://developer.android.com/training/camerax`, 2023.

[39] Huawei Developer, *Face Detection Guide*, Accessed: 2024-08-22, Huawei Technologies Co., Ltd, 2023, **available**: `https://developer.huawei.com/consumer/en/doc/hiai-Guides/face-detection-0000001050038170`.