Al'adzkiya International of Computer Science and Information Technology (AIoCSIT) Journal
Vol. 5, No. 1, May 2024, pp. 17~32
ISSN: 2722-0001

Convolutional Neural Network Based Human Posture Correction Implementation for Yoga Health Motion Classification

Elza Ahmad Raihan
Department of InformationTechnology, Universitas Muhammadiyah Sumatera Utara, Indonesia

ABSTRACT

Post-pandemic lifestyle patterns have undergone many changes with the implementation of digital transformation, one of which is the meditation pattern such as yoga practice that can be done independently at home without direct interaction with the instructor. This study also aims to develop a yoga movement classification system using Convolutional Neural Network (CNN) based on human posture correction. Using the Movenet model, this system can recognise and classify different yoga poses to provide accurate feedback on correct posture. Training data was collected from yoga photographs and processed into pose images that were analysed using CNN. The results of this study indicate that the developed system is able to achieve a high level of accuracy in identifying yoga poses, which has the potential to help users improve their posture and reduce the risk of injury. This system is also implemented in a mobile application, making it easier for users to access posture correction in real time. As such, this research makes a significant contribution to the fields of health and technology by providing innovative solutions for safer and more effective yoga practice.
Keyword : Convolutiona Neural Network , Yoga , Mobile App

Corresponding Author:
Elza Ahmad Raihan,
Department of Information Technology,
Universitas Muhammadiyah Sumatera Utara,
Jalan Kapten Muktar Basri No 3 Medan 20238, Indonesia.
Email: elzaahmad@gmail.com

1. INTRODUCTION

The advancement of technology, particularly in the field of Artificial Intelligence (AI), has significantly influenced various aspects of human life. One critical branch, Deep Learning, mimics the workings of the human brain by learning from data such as images, videos, and audio to perform complex tasks. In this domain, Computer Vision has seen rapid development over the years, with researchers continually striving to create more accurate systems. One such advancement is the Convolutional Neural Network (CNN), widely used for image classification, often supported by models like MoveNet for movement and posture recognition (Karnadi et al., n.d.).

Cardiovascular diseases remain a leading cause of death in Indonesia, with unhealthy dietary patterns—such as the frequent consumption of fast food high in fats, salt, and low in fiber—being a significant contributing factor (Ayuningtyas, 2023). In contrast, yoga has been recognized for its benefits in promoting both physical and mental health. Studies have demonstrated yoga's effectiveness in reducing risk factors for cardiovascular diseases, including hypertension and cholesterol, and even managing these conditions without surgical intervention (Mooventhan & Nivethitha, 2020). Moreover, yoga has been shown to improve emotional regulation and mental well-being, especially among students, with benefits lasting for months after regular practice (Elstad et al., 2020).

Despite these benefits, many beginners hesitate to start yoga due to perceived barriers such as the need for group sessions or a professional instructor, which can be costly. Practicing yoga alone at home often leads to uncertainties about posture correctness, increasing the risk of injury or ineffective results (Sharma et al., 2022). Many beginners attempt to learn yoga through smartphone tutorials without feedback, making it difficult to evaluate their own posture.

To address these challenges, recent innovations have explored the use of Machine Learning and Computer Vision to develop AI-powered yoga instructor applications. These systems can analyze body movements and provide real-time feedback on the accuracy of yoga poses, thus allowing individuals to safely practice yoga without live supervision.

This research proposes the development of a CNN-based yoga pose classification system, supported by MoveNet, to recognize and classify five yoga poses—Downward Dog, Warrior, Tree, Cobra, and Chair—using image and video data. The model is designed for implementation on Android devices to ensure accessibility, ease of use, and real-time posture evaluation for beginners.

Through this work, the study aims to promote independent yoga practice, improve physical and mental well-being, and reduce the risk of injury. Additionally, it contributes to the field of health technology by integrating machine learning and yoga, potentially paving the way for future AI-driven wellness applications.

2. RESEARCH METHOD

A. Research and Product Development Method

The research method is a form of scientific validation used to obtain information in order to achieve desired goals and benefits. Typically, research requires efforts based on scientific characteristics, such as rationality, being based on experience or empirical evidence, and being structured. The research conducted must be acceptable to human reason or be rational, meaning that any steps taken during the research process can be logically understood and accepted.

The validation of this research effort can be seen and understood through the use of certain procedures, where research must be structured and systematic, thus having a logical or reasonable nature. In general, there are several methods commonly used in research as approaches, including:

a) Research and Development (R&D)-Based Research
   This type of research is a study aimed at finding patterns and sequences of growth as a function of time. The research object is the change or progress achieved by individuals, such as the author in this study. The purpose of this research is to explore the development of machine learning models and the applications currently being developed.

b) Qualitative Research
   Qualitative research is used to deeply address problems within the context of developing situations. This research is typically conducted naturally and objectively according to real field conditions. At this stage, the author has conducted interviews with several individuals who need a yoga meditation trainer application. In developing the application or evaluating the machine learning model, the process differs from standard research; the Agile method will be used. This allows researchers to quickly and flexibly respond to changing needs, building applications and models iteratively to improve quality.

Among the various commonly used research approaches, this study applies an R&D (Research and Development) approach, often referred to as development research. This method is used to produce a specific product and test the effectiveness of that product. In its development, the researcher will adopt the Agile concept.

B. Research Flow

a) Literature Review

In the initial stage of the research process, as a researcher, I collected, reviewed, and analyzed relevant literature related to the research topic—this includes gathering information, reading and taking notes, and processing the information into research material. Additionally, as the author, I used the literature review to understand the existing knowledge on the subject, identify knowledge gaps, and establish a strong foundation for the upcoming research.

Through the literature review, I studied theories related to yoga and deep learning, using TensorFlow as the testing platform. It is expected that the model can later be developed into a mobile application product. The literature reviewed includes books, journals, and previous studies.

b) Dataset Collection

This data acquisition stage is an integral part of this research. The dataset used to train and evaluate the model consists of various yoga poses sourced from Kaggle. The data used in this study are classified into five categories: Downward Dog, Tree, Warrior, Chair, and Cobra.

This research aims to evaluate the performance of the MoveNet deep learning model in detecting and classifying types of yoga poses. Like other deep learning models, MoveNet requires a dataset of labeled images to be used in the training process—or in this case, for transfer learning.

Below is a sample of images representing several yoga poses that will be used in the dataset for developing the yoga classification model.

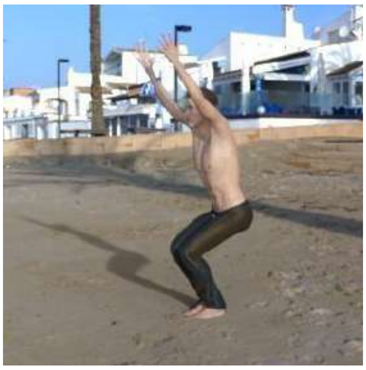

Figure 1. Chair Pose

Figure 2. Cobra Pose

Figure 3. Downdog Pose

Figure 4. Tree Pose

Figure 5. Warrior Pose

When collecting this dataset, we need to make sure we have enough images for different poses. In this study, the researchers will use training data from 942 images with different poses, and then test data from 675 images.

C. MoveNet Model Training

Since the MoveNet model is a pre-trained model, this training process is a transfer learning process that only trains the final layer to adjust the model to detect classes in the given dataset.

Before this training process is carried out, a little configuration needs to be done on the MoveNet model to be used, such as the dataset directory, input image size and image depth, then set the training configuration such as the number of epochs, batch size, and callback for the target accuracy number. After the configuration is complete and in accordance with the model, the training process can take place. The model will perform the training process on the dataset as many as the number of epochs given or until the accuracy number given in the callback. After the training process is complete, the model is automatically exported in tflite (tflt) format and can be used for movement prediction.

a) Model Testing and Evaluation

After the model has been successfully trained and produces satisfactory training accuracy, the next step is to evaluate the model's performance on yoga pose movements. In the evaluation process, predictions are made on groups of movements that have labels to find out the original results and this group of images must never be used in the training process to ensure the integrity of the evaluation results. The evaluation process is carried out by making predictions on the directory containing the evaluation dataset group, and producing evaluation numbers that will be analyzed in the next process.

b) Model Implementation and Model Performance Testing in Applications

The next step is to integrate the model into the application that will be used for practical yoga movement classification. This involves developing a user-friendly user interface, which allows users to record or obtain yoga movement videos and get real-time movement classifications from the MoveNet model and the last stage is testing the performance of the model implemented in the developed application. This involves testing the application with real users or with relevant data to evaluate how well the MoveNet model performs in classifying yoga movements in practical situations

c) Hardware & Software Requirements

Hardware Requirements

In implementing this research, some hardware is required that uses the x64 architecture with certain minimum specifications. These specifications are critical requirements for running the model and provide a basis for evaluating the speed performance of the model in the context of this research.

Table 1. Hardware Requirements

No	Hardware	Specification
1	*Processor*	Intel Core i7-7700HQ
2	*RAM*	8 GB
3	*GPU*	Geforce GTX 1050 Ram 16Gb
4	Storage	1 *Terabyte*
5	Operation System	*Windows* 10 / *Windows* 11

In the context of conducting research, it is possible to choose to use a system with specifications as previously explained, or to use cloud computing through platforms such as Google Collab or other cloud services to run the model.

Software Requirements

In designing this system, several software will be needed to support the research. There are two software that will be used, namely Android Studio and Google Collaboratory. In this design, the researcher will use Android Studio as the IDE software to design the mobile app that will be developed, and the researcher will use Google Collaboratory as a place to design the machine learning logic that the researcher will use to create the yoga movement classification system.

3. RESULTS AND DISCUSSION

A. Android Display Design Scheme

In this initial stage before building the dataset, we will focus on building the initial look and feel of the Android-based application, where the use of figma will be used to provide an overview of how the Android application will look in the implementation of yoga movement classification using a convolutional neural network. Later, after successfully creating the UX design display, we will implement the image on Android using Kotlin programming to make it easier to determine the layout in Kotlin-based Android programming.

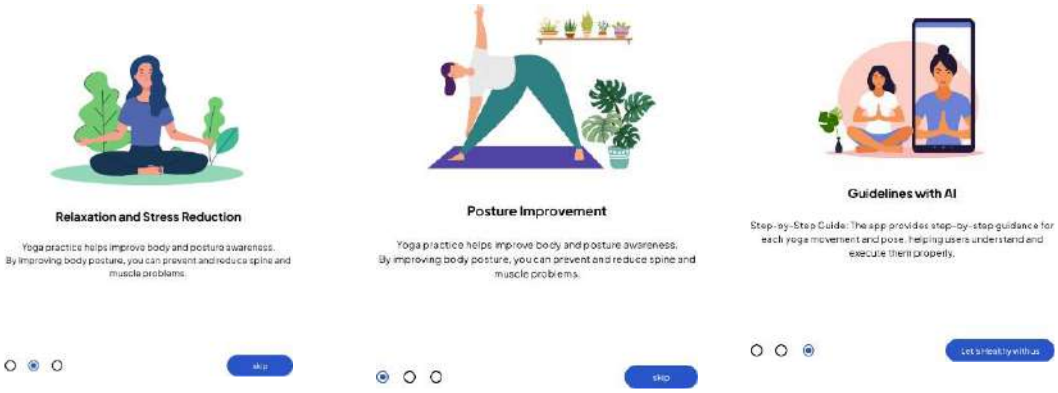

Figure 6. Onboarding Initial View

As illustrated above, the onboarding image informs users of the purpose of the application they are using. Image (a) provides an explanation of how yoga can be beneficial in terms of relaxation and stress relief, image (b) details the use of applications that can improvise movements independently, and image (c) explains how machine learning can facilitate the meditation guidance process. Following this, the login and registration account display will be designed so that users can access the application using an account. The display will be as follows:

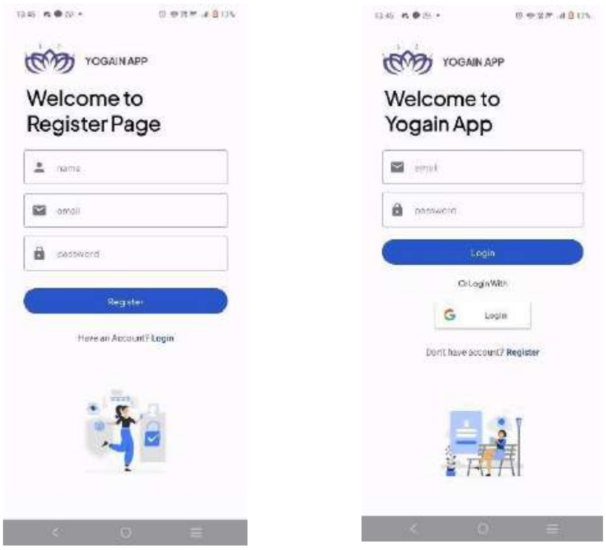

Figure 7. Login and Register View

The register display is where users can register a new account by entering their name, email address and password. This account is required to access the application.

The second display is the login display, which requires an account to access the application. This is based on the registration data previously stored in the register menu.

In the subsequent stage, it is imperative to construct a display that presents a range of yoga movement classification options. These options will be utilised in the identification of movement choices, the correction of postures, and the real-time camera detection. Prior to this, a display is to be implemented that provides static images and explanations pertaining to the selected poses.

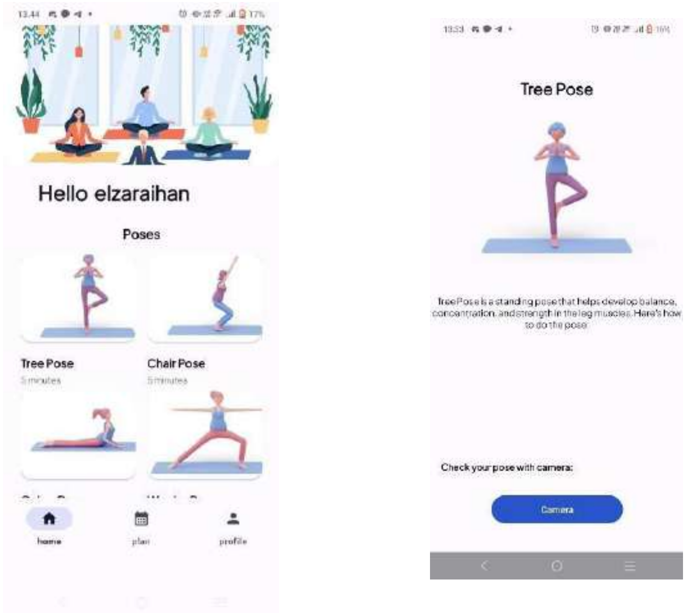

Figure 8. Yoga Selection Page View and Guide Details

As illustrated in image (a), the user has the capacity to select the yoga pose they wish to perform on a daily basis. Prior to initiation of the yoga pose movement, a pictorial guide and elucidation of the movement are presented. The camera button is then engaged to initiate the camera in yoga movement detection.

B. Kotlin Implementation of Android View

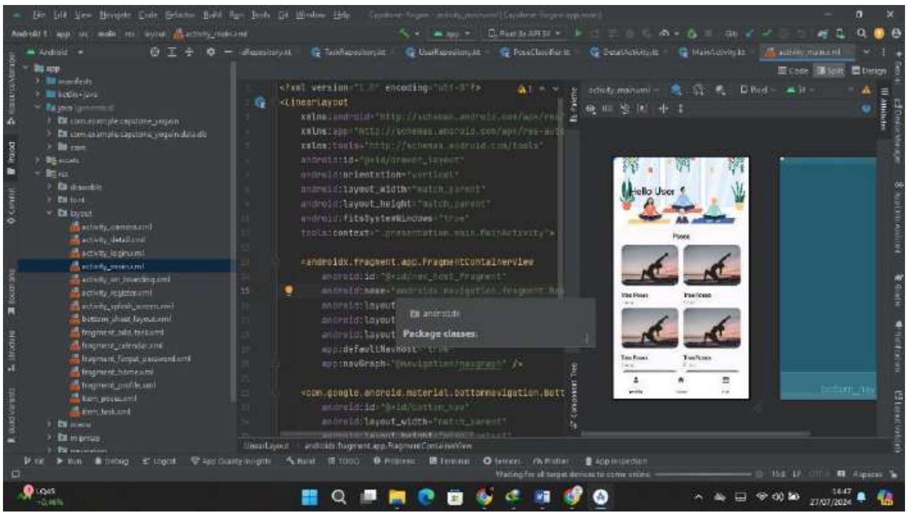

Figure 9. Layout Implementation in Kotlin

In the subsequent phase, the display design will be implemented in the form of an Android application. The Kotlin programming language will be utilised for the construction of the layout, with Android Studio being employed for the development of the layout. Subsequently, widgets will be employed to select UI components that align with our requirements for the interaction of the display interface.

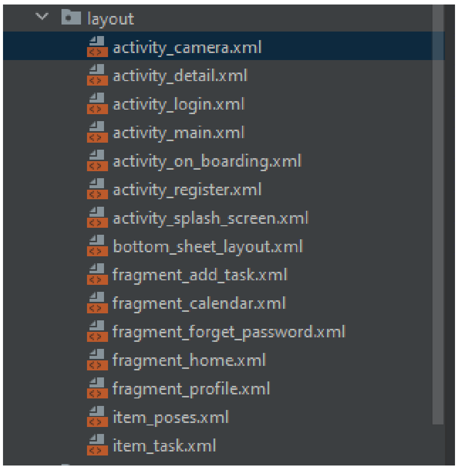

Figure 10. Layout Activity Display

In the course of developing the Android layout, researchers divided several layouts, each of which has a different function. These layouts will subsequently be combined and unified in the form of an APK in the Android implementation.

C. Model Development and Training and Testing Data

At this juncture, the results of the dataset collection are collated in a folder containing folder labels with differing image contents according to the yoga labels to which they correspond. It is imperative to ensure that all datasets and model folders are built in Google Drive, as this is necessary for their utilisation in Google Colab during the training and data development processes.

Figure 11. Google Drive Source Google Collab

In this particular drive, the directories have been subdivided into threefold. The 'app' folder contains the mobile application that has previously been built, whilst the 'datasets' folder is further subdivided into the 'testing' and 'training' folders. The 'training' folder contains the training data, and the 'model' folder contains the output that will be used from the results of the training data that has been built on Google Collaborator.

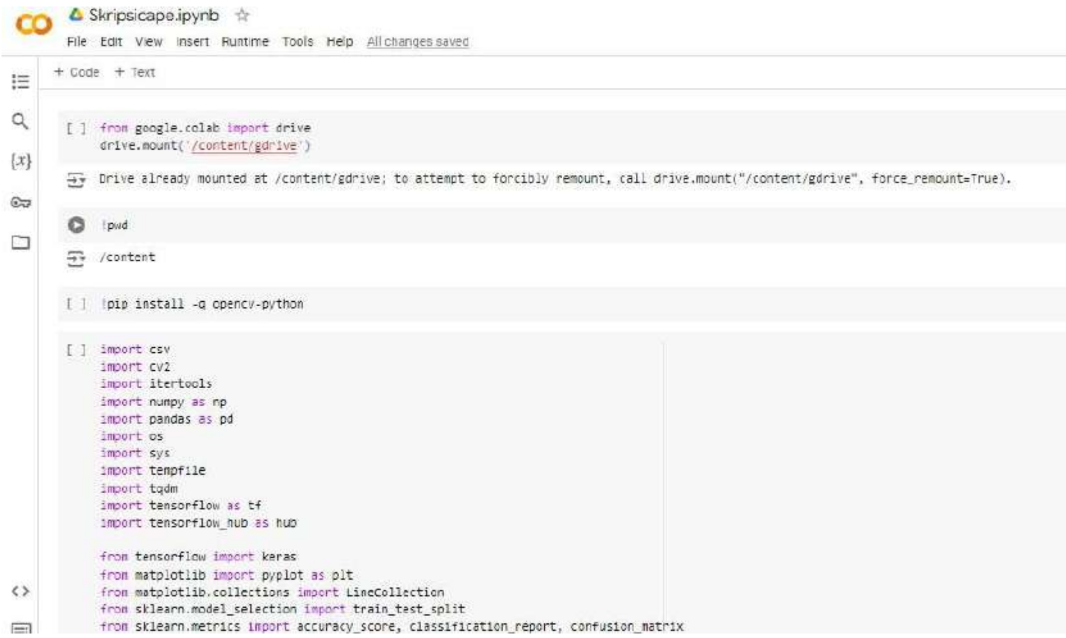

Figure 12. Initial Import Installation Command

In the subsequent phase, Google Collaborator will be integrated with Google Drive via the Google Collaborator command '.google.colab import drive'. This will result in the automatic import of Google Drive as a data collection tool. In the subsequent line, a library module is utilised to prepare data processing and to prepare for the subsequent utilisation of OpenCV to carry out direct testing on Google Collaborator.

Figure 13. Import Movenet Model in Pose Prediction

Following the establishment of a connection between Google Drive and Google Collab, Movenet will be utilised as the primary model for the detection of poses on the human body in the form of images. Subsequently, the human image will be assigned a keypoint for the determination of the pose of the human body. Furthermore, Movenet will be imported into a personal Google Drive, where the results of the model are stored as Movenet TFLite files, which can be utilised at a later stage.

Subsequently, a prolonged process of training the Movenet model as a pose model detector will be undertaken. The training of the Movenet model is based on the images that have been previously saved on Google Drive. The model is then used to detect images based on the poses determined by the labels that have been created previously. The image input process is then initiated to ascertain whether the Movenet model has succeeded in detecting a pose in the image using keypoints.

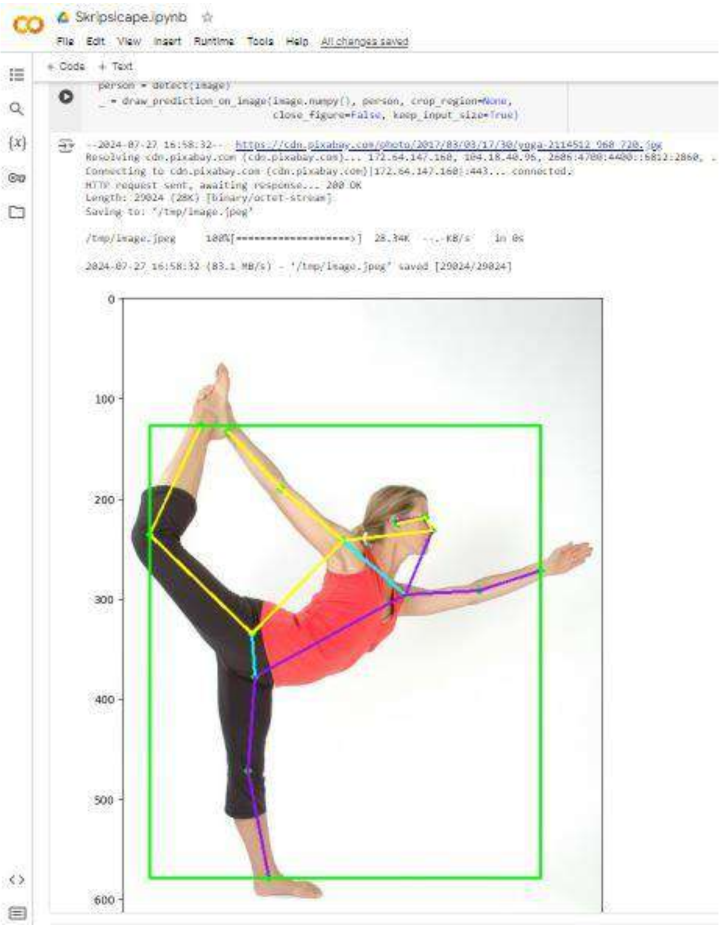

Figure 14. Movenet Model Results in Keypoint Detection

The experimental findings pertaining to the Movenet model have demonstrated that the model is capable of detecting movement poses on the human body. Consequently, the utilisation of the keypoint model is deemed feasible. Subsequent to the training of the movement model as a foundation for utilising the pose detection model, it is imperative to undertake the training of new data pertaining to the recently acquired images stored on Google Drive. These images are accompanied by labels that serve as descriptors of their content. The training process is anticipated to yield image detection data that is derived from the movement characteristics inherent within the images.

Figure 15. Yoga Label Image Data Training

At this stage, there is preprocessing of several poses that have been determined, where this process takes time. Subsequently, the training data can be used in the model testing process to predict output without adjusting the weights and to ascertain the accuracy and performance calculations of the model.

```
images_in_test_folder = "/content/gdrive/MyDrive/YOGA/datasets/TEST/"
images_out_test_folder = 'poses_images_out_test'
csvs_out_test_path = 'test_data.csv'

preprocessor = MoveNetPreprocessor(
    images_in_folder=images_in_test_folder,
    images_out_folder=images_out_test_folder,
    csvs_out_path=csvs_out_test_path,
)

preprocessor.process(per_pose_class_limit=None)
```

```
Preprocessing chair
100%|██████████| 168/168 [09:30<00:00,  3.40s/it]
Preprocessing cobra
100%|██████████| 232/232 [13:08<00:00,  3.40s/it]
Preprocessing downdog
100%|██████████| 97/97 [05:49<00:00,  3.60s/it]
Preprocessing tree
 86%|████████▌ | 59/69 [03:41<00:35,  3.56s/it]/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:520: RuntimeWarning: Mean of empty slice.
  avg = a.mean(axis, **keepdims_kw)
/usr/local/lib/python3.10/dist-packages/numpy/core/_methods.py:129: RuntimeWarning: Invalid value encountered in scalar divide
  ret = ret.dtype.type(ret / rcount)
100%|██████████| 69/69 [04:15<00:00,  3.70s/it]
Preprocessing warrior2
100%|██████████| 109/109 [05:57<00:00,  3.28s/it]Skipped /content/gdrive/MyDrive/YOGA/datasets/TEST/cobra/guy3_cobra036_flipped.jpg. No pose was confidentally detected.
```

Figure 16. Testing Data of Pose

This process prepares the test image data by converting it to a format suitable for the MoveNet model. The objective is to ascertain that the model is capable of accurately evaluating images and identifying poses in the test data. Warnings and errors serve to indicate areas that may require further attention in order to ensure accuracy. The process subsequently yields a model architecture, which can be utilised subsequently. The ensuing results are as follows:

```
Skripsicape.ipynb
File  Edit  View  Insert  Runtime  Tools  Help  All changes saved
+ Code  + Text

                                                                    ']
 tf.compat.v1.gather_17 (TF  (17, 2)            0        ['tf.math.subtract_5[0][0]']
 OpLambda)

 tf.compat.v1.norm_2 (TFOpL  ()                 0        ['tf.math.subtract_4[0][0]']
 ambda)

 tf.compat.v1.norm_3 (TFOpL  (2,)               0        ['tf.compat.v1.gather_17[0][0]
 ambda)                                                  ']

 tf.math.multiply_17 (TFOpL  ()                 0        ['tf.compat.v1.norm_2[0][0]']
 ambda)

 tf.math.reduce_max_1 (TFOp  ()                 0        ['tf.compat.v1.norm_3[0][0]']
 Lambda)

 tf.math.maximum_1 (TFOpLam  ()                 0        ['tf.math.multiply_17[0][0]',
 bda)                                                     'tf.math.reduce_max_1[0][0]']

 tf.math.truediv_1 (TFOpLam  (None, 17, 2)      0        ['tf.math.subtract_3[0][0]',
 bda)                                                     'tf.math.maximum_1[0][0]']

 flatten_1 (Flatten)         (None, 34)         0        ['tf.math.truediv_1[0][0]']

 dense_3 (Dense)             (None, 128)        4480     ['flatten_1[0][0]']

 dropout_2 (Dropout)         (None, 128)        0        ['dense_3[0][0]']

 dense_4 (Dense)             (None, 64)         8256     ['dropout_2[0][0]']

 dropout_3 (Dropout)         (None, 64)         0        ['dense_4[0][0]']

 dense_5 (Dense)             (None, 5)          325      ['dropout_3[0][0]']

==================================================================================================
Total params: 13061 (51.02 KB)
Trainable params: 13061 (51.02 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 17. The Results of Model Architecture

The model results obtained demonstrate the following:

1. Total Params: The total number of parameters in the model is 13,061. This encompasses all weights and biases within the network.
2. Trainable Parameters: It is evident that all parameters (13,061) are trainable. Consequently, each parameter can be updated during the training process in order to minimise loss and enhance model accuracy.

3. The following parameters are not trainable: It is evident that there are no non-trainable parameters (0). This finding suggests that all parameters within the model have been optimised during the training process.

The total parameter size is 51.02 KB, indicating that this model is relatively light and compact for computational purposes.

The subsequent process entails the validation of the accuracy that has been built where there is code implementation. Employing techniques such as checkpointing and early stopping, the model is optimised for performance and resistance to overfitting. The employment of this strategy is pivotal in the development of a robust machine learning model, as evidenced by the findings that the model exhibits satisfactory classification capabilities. However, there is still scope for enhancement, whether through refining the model architecture, utilising additional data, or employing supplementary regularization techniques. The employment of strategies such as early stopping and checkpointing is imperative in optimising the performance and stability of models.

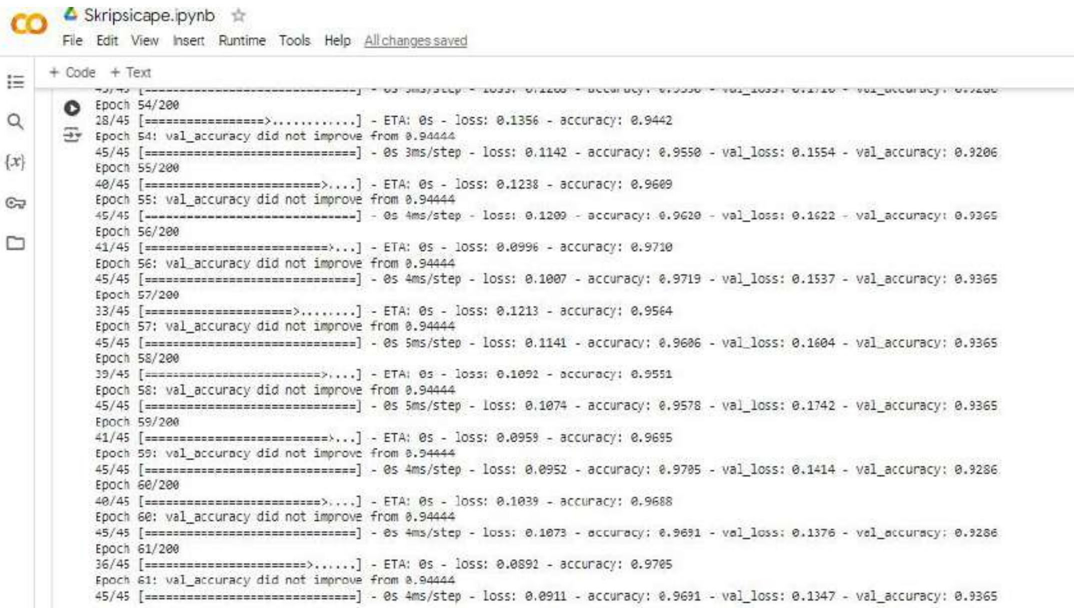

Figure 18. Accuracy Validation Results

The findings demonstrate that the model attains a high level of training accuracy, exceeding 97%, thereby signifying its capacity to discern patterns from the training data. Conversely, the validation accuracy is typically lower than the training accuracy, a phenomenon that is prevalent and indicative of the model's generalisation capability. The model demonstrates a high level of accuracy, with a mean value of 93%-94%, suggesting that it functions effectively. However, there is scope for enhancement. Despite the negligible disparity between training and validation accuracy, the early stopping mechanism is instrumental in averting overfitting, thereby ensuring the model maintains its generalisability.

Then the model accuracy results show the following results:

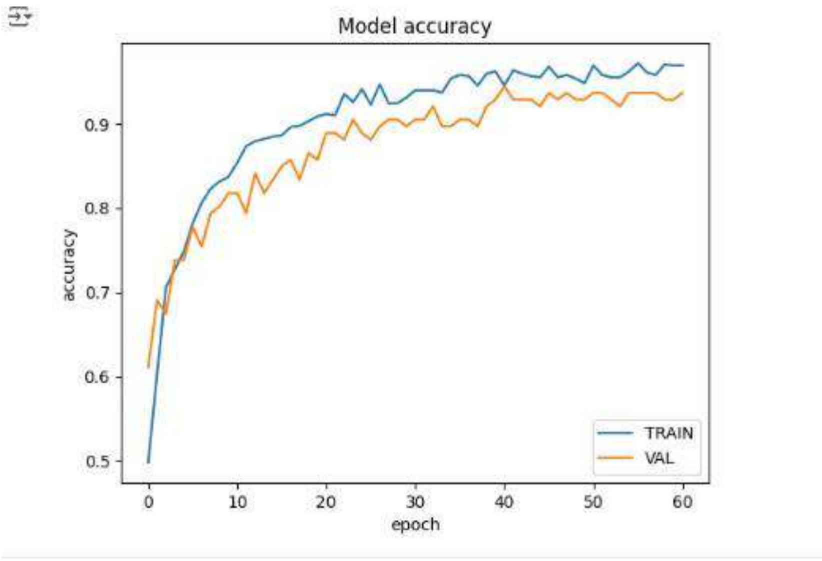

Figure 19. Model Accuracy

The graph demonstrates that the model has been adequately trained and exhibits strong performance in classification; however, there is potential for enhancement with regard to generalization. It is recommended that strategies such as regularization, data augmentation, or fine-tuning hyperparameters be considered in order to further improve validation accuracy. Despite the slight discrepancy between the training and validation accuracies, the model's overall performance remains at a satisfactory level, ensuring its reliability for practical applications.

Classification Report:

	precision	recall	f1-score	support
chair	0.88	0.48	0.62	168
cobra	0.99	1.00	0.99	194
downdog	1.00	0.98	0.99	87
tree	0.37	0.81	0.51	64
warrior2	0.99	1.00	1.00	103
accuracy			0.83	616
macro avg	0.85	0.85	0.82	616
weighted avg	0.90	0.83	0.84	616

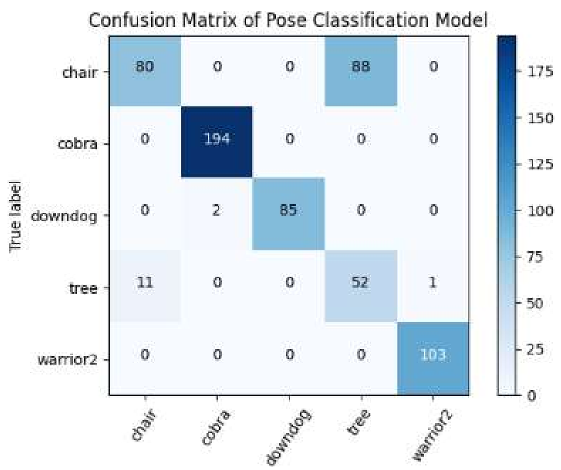

Figure 20. Classification Report Results with Confusion Matrix

The classification report provides an overview of the model's performance in predicting five different pose classes. The primary metrics employed in this study are precision, recall, and F1-score, each of which offers valuable insights into the efficacy of the model's predictions. The overall model demonstrates an accuracy of 92%, underscoring the effectiveness of the proposed methodology. However, a notable vulnerability is exhibited by the Chair class, which is frequently erroneously categorised as Tree. The precision and recall for this class are lower than the other classes, indicating the necessity for further treatment, such as the addition of more training data or the improvement of the distinguishing features for this class.

It is evident that the Cobra, Downdog and Warrior classes demonstrate commendable performance, exhibiting near-flawless precision, recall, and F1-score metrics. This finding indicates that the model exhibits a high degree of capacity in differentiating between these distinct classes. The model demonstrates considerable potential; however, there is scope for enhancement, particularly with regard to addressing imbalance and the misclassification of specific categories. Subsequently, the results of direct use will be evaluated by means of a webcam, with this evaluation being conducted on Google Collaborator.

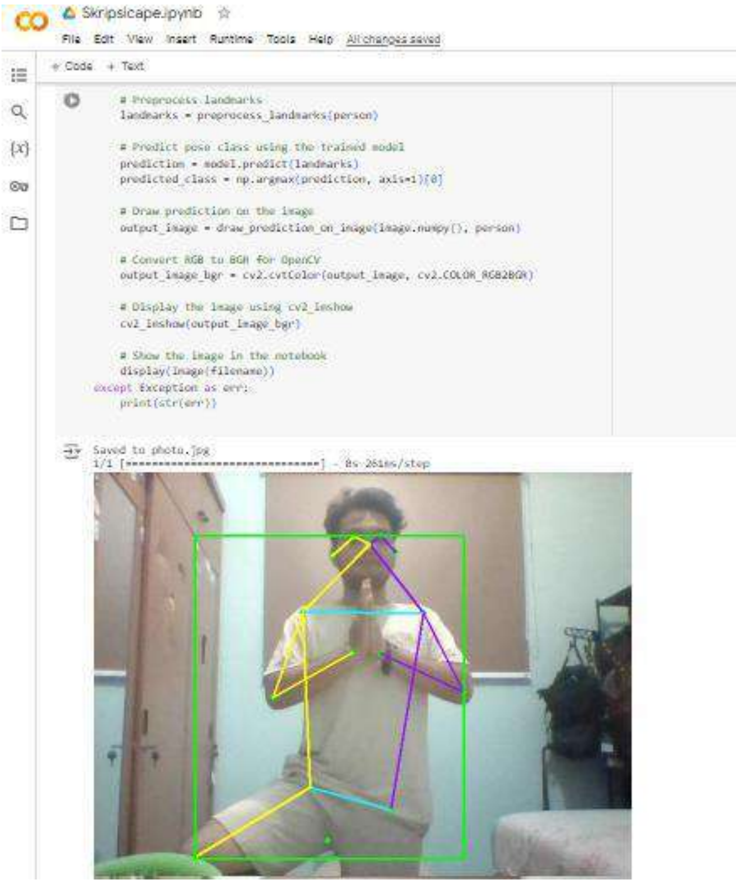

Figure 21. Box Pose Webcam Detection Results

As demonstrated by the webcam, the image is capable of utilising the motion model to detect the pose being performed.

D. Implementation of Machine Learning in Android

Subsequent to the validation of accuracy and the execution of training and testing data via Google Collaboratory, the concluding process is conducted by TFLite. This process generates an API, which is the framework derived from the constructed model. The model can be utilised to operate on a device with constrained resources, specifically the mobile application that has been developed in Kotlin. This application has been demonstrated to produce precise results, as illustrated in the subsequent image.

Figure 22. Tree Results

Figure 23. Warrior Results

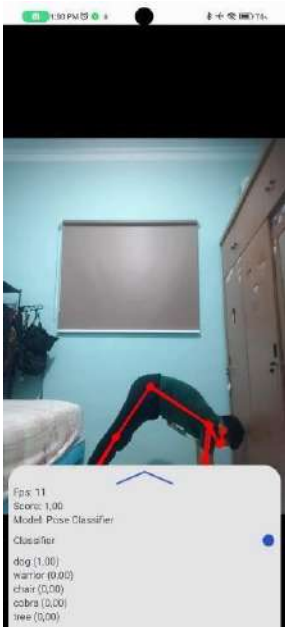

Figure 24. Downdog Results

Figure 25. Cobra Results

Figure 26. Chair Results

The image results demonstrate the precision of the movement, with the detection accuracy corresponding to the movement's precision. The accuracy values vary across different pose results, as illustrated by the Pose Tree Results, which exhibit an accuracy figure of approximately 0.99. The Warrior pose results demonstrate an accuracy figure of up to 1.00, the Down Dog pose shows a figure of up to 1.00, the Cobra pose exhibits a figure of 1.00, and the Chair pose displays an accuracy figure of around 0.81. It is evident that the ensuing outcomes are adequate for utilisation in a mobile application. The preceding model has demonstrated commendable accuracy and seamless integration capabilities, rendering it well-suited for implementation in a mobile context. The integration of machine learning within the mobile application is executed with notable efficacy, ensuring the model operates with precision and minimal overhead.

4. CONCLUSION

The training process and the construction of models utilising Movenet are conducted through Google Collaboratory. The model demonstrates a high level of training accuracy, with a percentage of approximately 97%, which is indicative of its capacity to discern patterns from the training data. Subsequently, it is evident that the validation accuracy is generally lower than the training accuracy. This phenomenon is a common occurrence and is indicative of the model's generalisation capability. The model demonstrates commendable performance, exhibiting an accuracy of approximately 93% - 94%, though there is scope for enhancement. Despite the negligible disparity between training and validation accuracy, the early stopping mechanism is instrumental in averting overfitting, thereby ensuring the model maintains its generalisability. Consequently, when subsequently converted via tflite for importation into a mobile application, the model remains applicable. The findings of the model incorporated into the mobile application demonstrate that real-time cellphone movements yield relatively precise feedback in terms of movement classification. In instances where the movement is deemed inappropriate, the classification is not rendered. Consequently, the movement must be rectified in accordance with the preceding image guide, utilising keypoints to identify the correct movement points during yoga exercises.

REFERENCES

[1] Abed, A. A., Al-Ibadi, A., & Abed, I. A. (2023). Real-time multiple face mask and fever detection using YOLOv3 and TensorFlow lite platforms. Bulletin of Electrical Engineering and Informatics, 12(2), 922–929. https://doi.org/10.11591/eei.v12i2.4227

[2] Ayuningtyas, E. Y. (2023). Penerapan Senam Yoga Terhadap Tekanan Darah Pada Penderita Hipertensi Di Kelurahan Jebres Surakarta. 1(4), 131–145. https://doi.org/10.59680/anestesi.v1i4.529

[3] Dufan J. P. Manajang, Sherwin R.U.A. Sompie, & Agustinus Jacobus. (2020). Implementasi Framework Tensorflow Object Detection dalam mengklasifikasi jenis kendaraan bermotor. Jurnal Teknik Informatika, 15(Jurnal Teknik Informatika), 171–178.

[4] Elstad, T., Ulleberg, P., Klonteig, S., Hisdal, J., Dyrdal, G. M., & Bjorndal, A. (2020). The effects of yoga on student mental health: a randomised controlled trial. Health Psychology and Behavioral Medicine, 8(1), 573–585. https://doi.org/10.1080/21642850.2020.1843466

[5] Gohel, M., Phatak, A., Kharod, U., Pandya, B., Prajapati, B., & Shah, U. (2021). Effect of long-term regular Yoga on physical health of Yoga practitioners. Indian Journal of Community Medicine, 46(3), 508–510. https://doi.org/10.4103/ijcm.IJCM_554_20

[6] Hindarto, H., Sumarno, S., & Rosid, M. A. (2023). Buku Ajar Kecerdasan Buatan/Artificial Intelegent (AI). In Buku Ajar Kecerdasan Buatan/Artificial Intelegent (AI). Umsida Press. https://doi.org/10.21070/2022/978-623-464-034-2

[7] Jo, B. J., & Kim, S. K. (2022). Comparative Analysis of OpenPose, PoseNet, and MoveNet Models for Pose Estimation in Mobile Devices. Traitement Du Signal, 39(1), 119–124. https://doi.org/10.18280/ts.390111

[8] Karnadi, B., Lubis, C., Agus,), & Dharmawan, B. (n.d.). Jurnal Ilmu Komputer dan Sistem Informasi Integrasi Metode Convolutional Neural Networks dengan Arsitektur Model PoseNet untuk Pengembangan Sistem Klasifikasi Gerakan serta Monitoring Repetisi pada Olahraga Bulu Tangkis.

[9] Mehindra Prasmatio, R., Rahmat, B., & Yuniar, I. (2020). ALGORITMA CONVOLUTIONAL NEURAL NETWORK. In Jurnal Informatika dan Sistem Informasi (JIFoSI) (Vol. 1, Issue 2).

[10] Mooventhan, A., & Nivethitha, L. (2020). Role of yoga in the prevention and management of various cardiovascular diseases and their risk factors: A comprehensive scientific evidence-based review. In Explore (Vol. 16, Issue 4, pp. 257–263). Elsevier Inc. https://doi.org/10.1016/j.explore.2020.02.007

[11] Purna Irawan, Y., Susilawati, I., & Kunci, K. (n.d.). Klasifikasi Jenis Aglaonema Berdasarkan Citra Daun Menggunakan Convolutional Neural Network (CNN).

[12] Putro Eko Cahyono, & Rolly Maulana Awangga. (2020). Tutorial Gender Classification Using The You Look Only Once (YOLO (Vol. 1). books.google.com.

[13] Rere, L. M. R., Usna, S., & Soegijanto, D. (2019). Studi Pengenalan Ekspresi Wajah Berbasis Convolutional Neural Network. Seminar Nasional Teknologi Informasi Dan Komunikasi STI&K (SeNTIK), 3.

[14] Santoso, A., & Ariyanto, G. (n.d.). IMPLEMENTASI DEEP LEARNING BERBASIS KERAS

[15] UNTUK PENGENALAN WAJAH. Jurnal Teknik Elektro, 18(01). https://www.mathworks.com/discovery/convol

[16] Sharma, A., Shah, Y., Agrawal, Y., & Jain, P. (2022). REAL-TIME RECOGNITION OF YOGA POSES USING COMPUTER VISION FOR SMART HEALTH CARE A PREPRINT.

[17] Singh PK. (2017, July 1). International Day of Yoga 2017-World Health Organization. WHO. Yudistira, N. (2021). Peran Big Data dan Deep Learning untuk Menyelesaikan Permasalahan

[18] Secara Komprehensif. EXPERT: Jurnal Manajemen Sistem Informasi Dan Teknologi, 11(2), 78. https://doi.org/10.36448/expert.v11i2.2063

[19] Zuo, X., Yang, X., Dou, Z., & Wen, J. R. (2019). RUCIR at TREC 2019: Conversational Assistance Track. 28th Text REtrieval Conference, TREC 2019 - Proceedings. https://doi.org/10.1145/1122445.1122456

[20] Motamed, S., & Askari, E. (2022). Recognition of Attention Deficit/Hyperactivity Disorder (ADHD) Based on Electroencephalographic Signals Using Convolutional Neural Networks (CNNs). Journal of Information Systems and Telecommunication, 10(39), 222–228. https://doi.org/10.52547/jist.16399.10.39.222

[21] Moayyed, H., Mohammadpourfard, M., Konstantinou, C., Moradzadeh, A., Mohammadi-Ivatloo, B., & Aguiar, A. P. (2022). Image Processing Based Approach for False Data Injection Attacks Detection in Power Systems. IEEE Access, 10, 12412–12420. https://doi.org/10.1109/ACCESS.2021.3131506

[22] He, X., & Dong, F. (2023). A deep learning-based mathematical modeling strategy for classifying musical genres in musical industry. Nonlinear Engineering, 12(1). https://doi.org/10.1515/nleng-2022-0302