Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

June 2026. Vol. 5. No. 3; e-ISSN: 2808-4519

Design of an Android-Based Sitting Posture Detection Application Using Deep Learning

Jhonshen[1] *, Octara Pribadi [2], Andy [3]

[1,2,3]*STMIK-TIME, Indonesia*
gunawanjhonshen@gmail.com[1*], octarapribadi@gmail.com[2], andy@mjsolusindo.com[3]

Abstract

Prolonged poor sitting posture is a major cause of musculoskeletal disorders including lower back pain and spinal abnormalities. This study designs and implements PosturApp, a deep learning-based Android application for real-time sitting posture detection using Kotlin. A Multi-Layer Perceptron (MLP) model was trained on 3,526 keypoint datasets sourced from the Kaggle public dataset (Posture Recognition) and direct image capture using an Android front camera, extracting 66 coordinate values from 33 body landmarks via MediaPipe BlazePose. The model was converted to TensorFlow Lite (TFLite) format at approximately 78 KB for on-device inference without internet connectivity. Evaluation results show an accuracy of 97.81% with precision 0.99, recall 0.99, and F1-Score 0.98. The application provides real-time visual feedback through interface color changes and corrective notifications, along with a gallery-based classification feature. Functional testing across eight posture scenarios yielded entirely correct results with confidence values ranging from 59% to 99%.

Keywords: *Sitting Posture, Deep Learning, Multi-Layer Perceptron, MediaPipe BlazePose, TensorFlow Lite, Android, Pose Estimation*.

1. Introduction

Prolonged sitting with poor body posture is one of the primary causes of musculoskeletal disorders, including lower back pain and spinal abnormalities. This problem is increasingly prevalent as modern lifestyles demand hours of digital activity in work and educational environments [1]. The body gradually adapts to incorrect sitting positions, increasing the risk of musculoskeletal disorders over time without the user realizing it

Advances in computer vision and deep learning offer innovative solutions for analyzing human body movement through pose estimation. MediaPipe BlazePose, developed by Google, is capable of detecting up to 33 body keypoints in real-time with high efficiency, even on resource-constrained devices such as smartphones [2]. Research has demonstrated that combining keypoint extraction with neural network classifiers such as Multi-Layer Perceptron (MLP) is effective for accurately recognizing body posture from coordinate data [3].

This study develops PosturApp, an Android application that integrates MediaPipe BlazePose-equivalent keypoint extraction via ML Kit Pose Detection with a TensorFlow Lite MLP model. The system provides immediate visual feedback to users regarding their sitting posture quality without requiring an internet connection. To the best of our knowledge, this approach combining on-device MLP inference with ML Kit landmark extraction for sitting posture detection on Android has not been previously reported in the literature [4][5].

2. Methodology

2.1. System architecture

The PosturApp system operates as a standalone Android application without server dependency. The architecture consists of three main layers: (1) input layer using the device front camera captured via CameraX API, (2) processing layer using ML Kit Pose Detection to extract 33 body landmarks producing 66 coordinate values (x, y per landmark), and (3) classification layer using a TFLite MLP model that classifies posture as good or poor based on the extracted keypoints.

2.2. Dataset

The dataset used in this study consists of 3,526 body keypoint data samples collected from two sources. First, the public Kaggle dataset titled "Posture Recognition" by Sahasraditya Thyadi (https://www.kaggle.com/datasets/sahasradityathyadi/posture-recognition/data), which provides diverse sitting posture images. Second, direct image collection using the front camera of an Android device involving the researcher and several volunteers as subjects, ensuring data relevance to actual application usage conditions. The dataset is balanced with 1,763 samples of good (ergonomic) sitting posture and 1,763 samples of poor sitting posture.

Each data sample consists of 66 feature values representing the x and y coordinates of 33 body landmarks extracted using MediaPipe BlazePose. The dataset was split into 80% training data (2,821 samples) and 20% test data (705 samples) using stratified splitting to maintain class proportion in both subsets.

2.3. Model Training

The classification model uses a Multi-Layer Perceptron (MLP) architecture built with TensorFlow/Keras in Python. The model consists of four layers: three hidden layers with ReLU activation (128→64→32 neurons) and one output layer with sigmoid activation. Dropout regularization (rate=0.3) is applied to the first two hidden layers to prevent overfitting. Training uses the Adam optimizer with binary cross-entropy loss function, a batch size of 32, a maximum of 100 epochs, and an early stopping mechanism with patience=15 based on validation loss monitoring. The complete training parameters are shown in Table 1.

Table 1: Model Training Parameters

Parameter	Value / Description
Model Type	Multi-Layer Perceptron (MLP)
Architecture	128 → 64 → 32 → 1 neurons
Activation Function	ReLU (hidden), Sigmoid (output)
Dropout	0.3 (layers 1 and 2)
Optimizer	Adam
Loss Function	Binary Crossentropy
Batch Size	32
Max Epochs	100
Early Stopping (patience)	15 epochs
Training Data	2,821 samples (80%)
Test Data	705 samples (20%)
Input Features	66 (x,y × 33 keypoints)
Output	1 sigmoid neuron (0=Good, 1=Poor)

After training, the model is converted to TFLite format using TFLiteConverter and stored in the Android application's assets folder with a file size of approximately 78 KB. The conversion preserves all model weights while optimizing for mobile inference

2.4. Android Implementation

The Android application is developed using Kotlin in Android Studio. ML Kit Pose Detection (Google) is used as the on-device keypoint extraction library, producing 33 normalized landmarks compatible with the MLP model input format. CameraX processes front-camera frames in real-time, converting each frame to a bitmap for pose detection. The 66 coordinate values extracted from each frame are passed directly to the TFLite interpreter for inference without requiring internet connectivity. The system classifies posture as Good, Poor, or Uncertain, and immediately updates the UI with color-coded feedback (green header for good posture, red header for poor posture) along with a confidence percentage progress bar.

3. System implementation

3.1. Application Interface

PosturApp provides five main interface screens that guide users through posture detection. Each screen has a distinct visual indicator to communicate the system status clearly and in real-time.

Fig. 1: Main screen in standy mode

Fig. 1 shows the main screen when the application is first launched. The camera preview area is inactive (black), with a LIVE badge in the top-right corner confirming the camera is ready. A gallery access button (blue camera icon) appears in the top-left. Instruction text at the bottom guides the user to tap the screen or press the play button to begin detection.

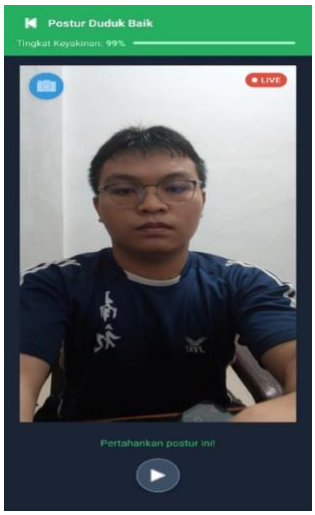

Fig. 2 *:* Good d detected with 99% confidence

Fig. 2 displays the detection result when the system identifies an ergonomic sitting position. The header turns green with the label "Postur Duduk Baik" (Good Sitting Posture) and shows a fully-filled green progress bar at 99% confidence. The notification "Pertahankan postur ini!" (Maintain this posture!) is shown at the bottom, providing positive reinforcement to the user.

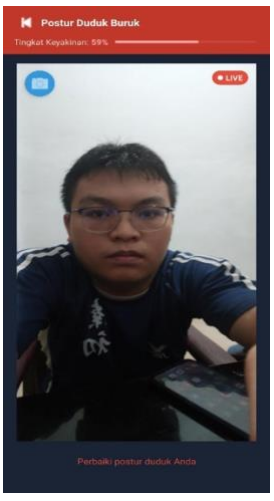

Fig. 3*:* Poor posture detected with 59% confidence

Fig. 3 shows the system response when poor posture is detected. The user is visibly leaning forward toward the camera, causing the header to turn red with the label "Postur Duduk Buruk" (Poor Sitting Posture) and a partially-filled red progress bar at 59% confidence. The message "Perbaiki postur duduk Anda" (Correct your sitting posture) is displayed at the bottom as a corrective instruction.

Fig. 4: Gallery feature for photo-based classification

Fig. 4 illustrates the gallery feature, which provides an alternative to real-time camera detection. When the user taps the gallery button, the device photo picker appears as a bottom sheet overlay showing available photos organized by Foto (Photo) and Koleksi (Collection) tabs. Users can select any existing image from their device for posture classification.

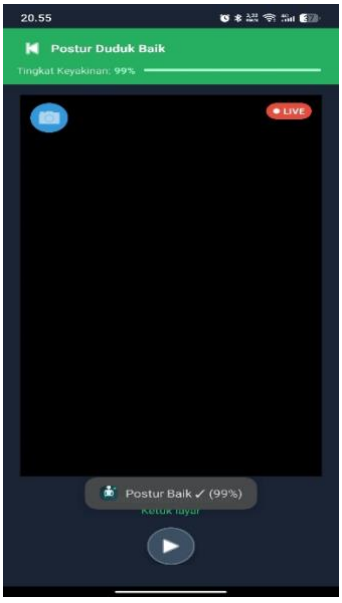

Fig. 5: Real-time detection with on-screen posture status notification

Fig. 5 demonstrates the real-time notification overlay during active detection. A status badge at the bottom of the camera view displays "Postur Baik (99%)" with a checkmark icon, providing an at-a-glance confirmation of the current posture quality without disrupting the camera view. This compact notification allows users to quickly verify their posture status while the full camera preview remains visible.

3.2. Detection Mechanism

The detection pipeline processes approximately 30 frames per second from the front camera. Each frame undergoes: (1) format conversion from YUV to NV21 to JPEG to Bitmap, (2) ML Kit Pose Detection landmark extraction producing 66 coordinate values from 33 body landmarks, and (3) TFLite MLP inference generating a sigmoid output value. The system classifies posture as Good (output below 0.40), Poor (output above 0.60), or Uncertain (0.40 to 0.60). The UI color, header text, confidence bar, and bottom notification all update immediately upon each classification result.

4. Results and Discussion

4.1. Model Evaluation

The MLP model training ran for 50 epochs before being stopped by the early stopping mechanism, with the best model selected at epoch 35 based on minimum validation loss. Evaluation on the test dataset produced an accuracy of 97.81%. The detailed classification report is presented in Table 2.

Table 2: Classification Report

Class	Precision	Recall	F1-Score	Support
Good	0.99	0.97	0.98	352
Poor	0.97	0.99	0.98	353
Average	**0.98**	**0.98**	**0.98**	**705**

The model demonstrates excellent performance on both classes with a balanced F1-Score of 0.98. The high precision (0.99) for the Good class indicates that 99% of predictions classified as good posture are truly good posture, while the high recall (0.99) for the Poor class shows the model can detect 99% of actual poor posture cases in the test data.

4.2. Confusion Matrix

The confusion matrix analysis provides detailed insight into the model's prediction distribution, as shown in Table 3.

Table 3: Confusion Matrix

Actual \ Predicted	Predicted Good	Predicted Poor
Actual Good	341 (TN)	11 (FP)
Actual Poor	5 (FN)	348 (TP)

Of 352 good posture test samples, 341 were correctly predicted (TN) and 11 were misclassified as poor (FP). Of 353 poor posture samples, 348 were correctly predicted (TP) and only 5 were misclassified as good (FN). Total misclassifications are 16 out of 705 test samples, yielding an overall accuracy of 97.73%.

4.3. Functional Testing

Functional testing was conducted by evaluating eight distinct posture scenarios to assess the system's real-world detection capability. Results are presented in Table 4.

Table 4: Functional Testing Results

No	Posture Condition	Detection Result	Confidence	Status
1	Upright sitting, head aligned with shoulders	Good Posture	99%	✓
2	Leaning forward toward camera	Poor Posture	59%	✓
3	Shoulder tilted to one side	Poor Posture	65%	✓
4	Head bowed downward	Poor Posture	72%	✓
5	Body leaning sideways	Poor Posture	61%	✓
6	Good posture with hand movement	Good Posture	95%	✓
7	Good posture image from gallery	Good Posture	88%	✓
8	Poor posture image from gallery	Poor Posture	76%	✓

All eight tested posture scenarios were correctly detected by the system. Confidence values ranged from 59% to 99%, with clear and distinct postures yielding higher confidence values and borderline postures yielding lower confidence. This variation is expected and reflects the model's calibrated uncertainty in ambiguous cases.

4.4. System Performance

The PosturApp system performance evaluation is summarized in Table 5.

Table 5: System Performance

Performance Parameter	Result
Platform	Android (Kotlin)
TFLite Model Size	±78 KB
Real-Time Detection	Yes (no internet required)
Model Accuracy	97.81%
Minimum Android Version	Android 7.0 (API 24)
Feedback Type	Real-time visual notification

4.5. Analysis

The MLP model achieves 97.81% accuracy due to three key factors: balanced dataset distribution preventing class bias, Dropout regularization preventing overfitting, and Early Stopping selecting the best model weights. The coordinate keypoint approach proved effective, as 66 geometric coordinates from 33 body landmarks are sufficient to capture the geometric patterns distinguishing good and poor posture, consistent with the findings of Swain et al. [3].

The use of ML Kit Pose Detection as a MediaPipe BlazePose equivalent on Android produced compatible landmark coordinates with the MLP model input, enabling accurate on-device inference. The 78 KB TFLite model size demonstrates successful model optimization for the mobile environment, allowing deployment on mid-range Android devices without sacrificing detection accuracy.

5. Conclusion

This study successfully designed and implemented PosturApp, a deep learning-based Android application for real-time sitting posture detection. The MLP model trained on 3,526 keypoint samples achieved 97.81% accuracy with precision 0.99 and recall 0.99 for both classes. Integration with TensorFlow Lite produced a compact 78 KB model enabling on-device inference without internet connectivity. Visual feedback through color-coded interface changes and corrective notifications was successfully implemented, with functional testing across eight posture scenarios yielding entirely correct detection results with confidence values between 59% and 99%.

Future work should focus on expanding posture categories beyond the current binary classification, incorporating audio feedback and posture history tracking, extending to iOS platforms using Core ML, and collecting more diverse datasets with varied camera angles and lighting conditions to improve model generalization.

References

[1] Ienaga, N., Takahata, S., Terayama, K., & others. (2022). Development and verification of postural control assessment using deep-learning-based pose estimators. Occupational Therapy International, 2022, 1–11. https://doi.org/10.1155/2022/6952999

[2] Wu, J., Li, W., Chen, Y., & others. (2024). A human pose estimation method based on the BlazePose model. Journal of Modern Educational Research, 2024, 1–10. https://doi.org/10.53964/jmer.2024021

[3] Swain, D., Satapathy, S. K., Acharya, B., & others. (2022). Deep learning models for yoga pose monitoring. Algorithms, 15(11), 403. https://doi.org/10.3390/a15110403

[4] Krishna, R. (2023). Real time human body posture analysis using deep learning. International Journal for Science Technology and Engineering, 4(3), 1–6. https://doi.org/10.22214/ijraset.2023.52099

[5] Ranjan, R., Ahmedt-Aristizabal, D., Armin, M. A., & others. (2024). Developing normative gait cycle parameters for clinical analysis using human pose estimation. arXiv preprint, 2411.13716. https://doi.org/10.48550/arxiv.2411.13716

[6] Chung, J. L., Ong, L. Y., & Leow, M. C. (2022). Comparative analysis of skeleton-based human pose estimation. Future Internet, 14(12), 380. https://doi.org/10.3390/fi14120380

[7] Torun, C., & Karaci, A. (2024). Comparison of 2D human pose estimation models based on deep learning. Journal of Sustainable Engineering Applications and Technological Developments, 4(2), 1–10.

[8] Haghighat, M. (2023). Introduction to computer vision with convolutional neural networks. Edge AI and Vision Alliance.

[9] Yauri, R., & Espino, R. (2023). Edge device for movement pattern classification using neural network algorithms. Indonesian Journal of Electrical Engineering and Computer Science, 30(1), 229–236.

[10] Romindo, R., Barus, O. P., Pangaribuan, J. J., & others. (2022). Implementation of support vector machine algorithm for ballet pose classification. Building of Informatics, Technology and Science, 4(3), 1–7.

[11] Object-Based Hybrid Deep Learning Technique for Recognition of Sequential Actions. (2023). IEEE Access, 1–12. https://doi.org/10.1109/access.2023.3291395

[12] Chiddarwar, G. G., Ranjane, A., Chindhe, M., & others. (2020). AI-based yoga pose estimation for Android application. International Journal of Innovative Science and Research Technology, 5(10), 30–36.