

A User Fatigue Prediction System Based on Heart Rate and SpO₂ Data from Smartwatch on Android Platform

Ridho Nur Fahrendy^{1*}, Fara Triadi², Agus Triyono³

^{1,2,3}Computer Engineering Technology Study Program, Departemen of Information Technology, state Polytechnic of Samarinda
ridhonf18@gmail.com^{1*}, fara_triadi@polnes.ac.id², triyono@polnes.ac.id³

Abstract

The high demands of modern activities frequently lead to fatigue, a condition that diminishes physical and cognitive capacity while elevating the risk of human error. Smartwatches equipped with physiological sensors offer continuous, non-invasive health monitoring, presenting an opportunity for early fatigue detection. This study developed an Android application that predicts user fatigue status by analyzing heart rate (BPM) and blood oxygen saturation (SpO₂) data recorded by a smartwatch. A machine learning model based on Long Short-Term Memory (LSTM) was trained on a public dataset to recognize fatigue patterns, then converted to TensorFlow Lite format for on-device integration. The application retrieves historical BPM and SpO₂ data through Health Connect API and classifies the user's condition as either Normal or Fatigued. Offline evaluation yielded 100% accuracy on test data; however, this performance was identified as an artifact of overfitting caused by deterministic synthetic labeling rules. On-device inference encountered compatibility issues with TensorFlow Lite runtime, preventing successful execution of the LSTM model on Android. Despite this limitation, the study demonstrates the feasibility of bridging heterogeneous wearable ecosystems and highlights critical technical challenges in deploying recurrent neural networks on mobile devices.

Keyword: Android; Fatigue prediction; heart rate; LSTM; smartwatch; SpO₂

1. Introduction

Modern life is characterized by intense productivity demands across both professional and personal domains. These conditions substantially increase the risk of fatigue—a state of diminished physical and cognitive capacity that goes far beyond ordinary drowsiness [1]. Unmanaged fatigue is a major contributor to human error, with consequences ranging from impaired workplace performance to fatal traffic accidents. Systematic reviews have linked driver fatigue to approximately 10–20% of all road traffic incidents [2]. Such findings underscore the urgent need for proactive fatigue monitoring systems that can safeguard health, safety, and sustained performance [3].

The proliferation of wearable technology, particularly smartwatches, has opened a new paradigm in personal health monitoring. These devices are equipped with sophisticated sensors capable of continuously recording physiological data in a non-invasive manner [4]. The global wearable medical device market continues to grow exponentially, reflecting strong public acceptance and substantial potential for transforming healthcare delivery [5]. Integration of smartwatches with dominant mobile platforms such as Android provides an ideal ecosystem for developing accessible health applications for broad populations [6].

Among the measurable physiological parameters, heart rate (beats per minute, BPM) and blood oxygen saturation (SpO₂) are two key indicators highly relevant to fatigue detection. Research has shown that mental fatigue disrupts heart rate variability (HRV), reflecting dysregulation of the autonomic nervous system [7]. Decreased SpO₂ levels can indicate hypoxia—a reduction in oxygen supply to body tissues including the brain—which is one of the primary causes of fatigue onset and cognitive decline [8]. Monitoring both parameters simultaneously therefore offers a comprehensive approach to objectively assessing fatigue status [9].

The Android platform, with its open architecture, provides a flexible development environment supported by a mature ecosystem. Google Fit serves as a centralized health data aggregation hub, enabling developers to access data from various hardware brands through standardized Application Programming Interfaces (APIs). Nevertheless, practical challenges arise from limited cross-platform interoperability, which remains a significant barrier to holistic health data utilization [10]. For instance, the Huawei Health ecosystem

operates within a closed environment and does not provide direct data synchronization to Google Fit, necessitating intermediary solutions for data integration.

Physiological data becomes more meaningful when employed not merely for monitoring but also for prediction. Systems that only display historical data are inherently reactive, whereas predictive systems enable proactive intervention. Davidson et al. [11] demonstrated that smartwatch data can predict perceived exertion levels—a proxy for fatigue—with high accuracy using machine learning. Similar deep learning approaches have proven effective in detecting driver fatigue and distraction from wearable data [12]. These findings establish the feasibility of transforming raw BPM and SpO₂ data into actionable predictive insights such as Fatigued or Normal status.

Based on the urgency of fatigue monitoring, the potential of smartwatch technology and the Android platform, the relevance of BPM and SpO₂ parameters, data integration challenges, and predictive analytics opportunities, this study was proposed. The research aims to design and build a User Fatigue Prediction System based on heart rate and SpO₂ data from a smartwatch on the Android platform. The system is expected to bridge the gap between the availability of physiological data from commercial devices and its utilization for meaningful health interventions.

The research objectives are threefold: (1) to build a functional Android application that integrates and visualizes BPM and SpO₂ data from a Huawei smartwatch through Health Connect synchronization; (2) to develop a fatigue prediction model using the LSTM algorithm and evaluate its accuracy in offline testing; and (3) to evaluate the technical feasibility of deploying a deep learning model on-device on the Android platform using TensorFlow Lite.

2. Related work

Several prior studies have explored related domains, including fatigue detection using physiological data from wearable devices, health monitoring system architectures on the Android platform, and algorithm implementation for physiological data analysis. The following discussion reviews key contributions and identifies gaps that the present study addresses.

Marce-Santurio et al. [10] investigated the interoperability challenges of mobile health applications, highlighting the fragmentation of health data across proprietary ecosystems. Their work emphasized the need for unified data hubs—a role that Health Connect now fulfills on Android. Similarly, Madavarapu [6] presented an IoT-based wearable health monitoring system using smartwatch sensors, demonstrating the viability of continuous physiological data collection but without incorporating predictive analytics.

Alruwaili et al. [9] developed a health monitoring system using an IoT-based Android mobile application that collected heart rate and SpO₂ data. While their system achieved reliable data acquisition and visualization, it relied on threshold-based alerts rather than machine learning prediction. Reza and Suryana [13] built a health monitoring application utilizing a smartwatch on Android, focusing on data presentation and user interface design. Their work provided a foundational architecture that the present study extends by incorporating a deep learning prediction module.

Castaldo et al. [7] conducted a systematic review with meta-analysis on acute mental stress assessment via short-term HRV analysis. Their findings confirmed that HRV parameters are sensitive indicators of autonomic nervous system changes induced by stress and fatigue, providing physiological justification for the features selected in this study. Mizuno et al. [8] demonstrated that prolonged cognitive load triggers sympathetic hyperactivity and parasympathetic suppression, establishing the autonomic nervous system mechanisms through which mental fatigue can be detected via heart rate monitoring.

In the domain of driver fatigue detection, Díaz-Santos and Caballero-Gil [1] employed smartwatch data with deep learning techniques to detect drowsiness in drivers. Their work validated the use of wearable sensor data for real-time fatigue classification. Lu et al. [14] conducted a systematic review on detecting driver fatigue using heart rate variability, reporting detection accuracies ranging from 44% to 100% across various studies. These results demonstrate both the promise and variability of HRV-based fatigue detection approaches.

Karim et al. [15] proposed LSTM fully convolutional networks for time series classification, demonstrating the effectiveness of LSTM architectures in capturing temporal dependencies in sequential data. This work informed the model architecture choice in the present study. Bent et al. [16] investigated sources of inaccuracy in wearable optical heart rate sensors, noting that factors such as motion artifacts, skin pigmentation, and device pressure can affect measurement accuracy—considerations relevant to the data quality limitations acknowledged in this study.

The present study differs from prior work by integrating data from a closed wearable ecosystem (Huawei) into the Android Health Connect platform, implementing an LSTM-based prediction model for fatigue classification, and evaluating the challenges of on-device deep learning inference using TensorFlow Lite. Table 1 summarizes the comparison with related studies.

Table 1: Comparison of related studies (State of the Art)

Reference (Author, Year)	Proposed System / Method	Physiological Parameters	Key Results / Accuracy	Limitations / Research Gap
Davidson et al. (2020)	Fatigue (RPE) prediction using ML algorithms (CNN, kNN, SVM) on a commercial smartwatch.	Heart Rate, Speed (GPS), Cadence	CNN achieved the highest accuracy of 84.8%.	Small sample size; only predicts two fatigue classes.

Díaz-Santos & Caballero-Gil (2022)	Driver drowsiness detection app using Samsung Galaxy Watch 4.	HRV (Heart Rate Variability), Motion	Successfully detects drowsiness and triggers vibration alerts.	No accuracy metrics provided; heavily dependent on Samsung's closed ecosystem.
Reza & Suryana (2022)	Health monitoring app integrating smartwatch data via Google Fit and Firebase.	Heart Rate, SpO ₂	Functional architecture for data synchronization and display.	Lacks a predictive feature; evaluation was limited to functional testing.
Alruwaili et al. (2023)	IoT-based health monitoring system using ESP32 and mobile app.	Heart Rate, SpO ₂ , Temperature, ECG	Highly accurate real-time monitoring comparable to medical devices.	No predictive analytics; relies entirely on continuous internet connectivity.
Madavarapu (2024)	IoT wearable (HOT Watch) using the Pan-Tompkins algorithm.	Temperature, ECG, SpO ₂	Achieved 99.4% accuracy in heart rate calculation.	Dependent on continuous Bluetooth connection; potential user constraints. Focuses on cross-ecosystem integration.
Proposed Study (2026)	Fatigue prediction using LSTM model integrated via Health Connect.	Heart Rate (BPM), SpO ₂	Trained LSTM model deployed locally (on-device) on Android.	(Huawei to Android); highlights on-device deployment challenges.

3. Methodology

3.1. Research design

This study adopted a Research and Development (R&D) methodology, which systematically produces and validates a specific product [17]. Unlike pure research aimed at discovering or testing theories, the R&D approach is oriented toward creating a functional, effective product that addresses practical problems. The methodology followed a structured cycle comprising requirements analysis, system design, implementation, testing, and deployment phases.

3.2. System architecture

The system architecture comprises four principal layers: the data source layer, the synchronization layer, the application layer, and the presentation layer.

At the data source layer, the Huawei Watch Fit 3 smartwatch employs a photoplethysmography (PPG) sensor to continuously collect raw BPM and SpO₂ data. The PPG sensor operates by emitting LED light (green for BPM; red and infrared for SpO₂) into wrist tissue. The light reflected back to a photodetector varies with blood volume fluctuations corresponding to each heartbeat, and these variations are processed by internal algorithms to compute BPM and SpO₂ values [9].

The synchronization layer addresses the interoperability challenge posed by Huawei's closed ecosystem. Data flows from the smartwatch to the Huawei Health companion application, then through a third-party intermediary application (Health Sync) that transfers data to Google Fit. Google Fit subsequently writes the data to Health Connect, which functions as the centralized, on-device health data hub on Android [17].

Health Connect, introduced by Google, operates as a unified health and fitness data platform that stores all data locally on the device, ensuring that sensitive physiological information remains under user control [17]. The platform supports standardized data schemas for over 40 common health data types, including HeartRateRecord and OxygenSaturationRecord, ensuring format uniformity regardless of the originating device brand.

The application layer follows the Model-View-ViewModel (MVVM) architecture pattern. The Repository component (HealthRepository) serves as the single source of truth, responsible for communicating with the Health Connect API to read BPM and SpO₂ time-series data and for executing the TensorFlow Lite prediction model. The ViewModel component (HealthViewModel) bridges the Repository and the user interface, managing data state and exposing it to the UI through observable state objects. The View component (MainActivity) renders the user interface using Jetpack Compose and delegates user actions back to the ViewModel.

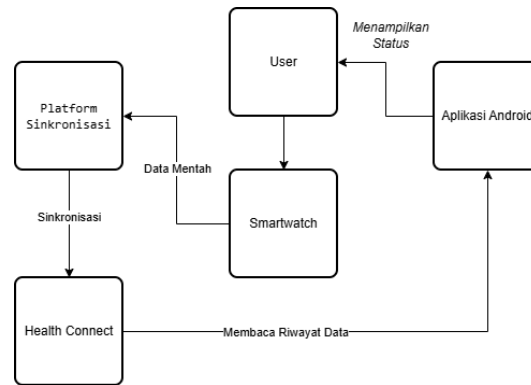


Fig. 1: System block diagram showing data flow from smartwatch through synchronization chain to Android application

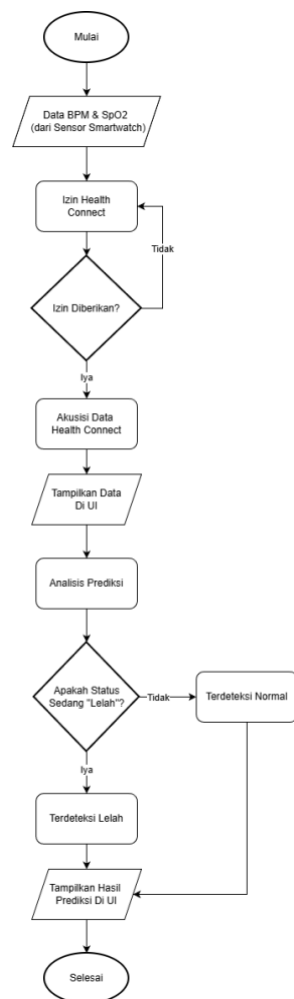


Fig. 2: System flowchart illustrating the operational logic from data acquisition to fatigue prediction

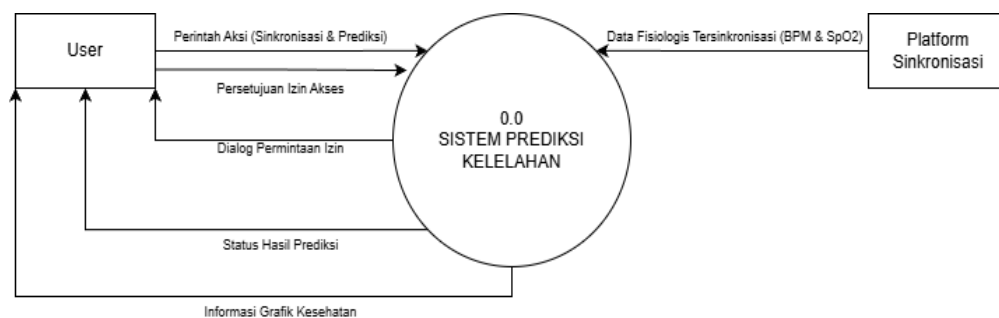


Fig. 3: Data Flow Diagram (Level 0) showing system boundary and external entity interactions

3.3 LSTM model development

The fatigue prediction model employs Long Short-Term Memory (LSTM), a recurrent neural network (RNN) architecture specifically designed to learn long-term dependencies in sequential data [15]. Standard RNNs struggle with vanishing gradients when processing extended sequences, but LSTM overcomes this through an internal cell state regulated by three gating mechanisms: the forget gate (determining which information to discard), the input gate (determining which new information to store), and the output gate (determining which information to produce as output). This selective memory mechanism enables LSTM to retain important patterns from distant past data while discarding irrelevant information.

The model was trained on a publicly available dataset titled "Sports Injury Prediction" sourced from Kaggle. This dataset contains time-series physiological data including heart rate (Heart_Rate_BPM) and blood oxygen level (Blood_Oxygen_Level_Percent). The use of an external dataset was necessary to obtain sufficient volume and variation in physiological data for training a robust model, which would be difficult to collect from a single individual in a short time period.

Data preprocessing involved several steps. First, the dataset was cleaned of noise and missing values. Input features were then normalized using Min-Max Scaling to transform values to the [0, 1] range, which stabilizes neural network training convergence. The normalization formula is:

$$x^1 = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (1)$$

where x is the original value, $\min(x)$ and $\max(x)$ are the minimum and maximum values of each feature in the dataset. Fatigue labels were assigned synthetically using a thresholding logic rule: a sample was labeled as Fatigued when Heart_Rate_BPM exceeded 100 and Blood_Oxygen_Level_Percent fell below 95; otherwise, it was labeled as Normal. The dataset was split into 80% training data and 20% testing data.

The LSTM model architecture consisted of two stacked LSTM layers with 60 units each, followed by Dropout layers to prevent overfitting, and a final Dense layer with a single sigmoid activation output. The sigmoid function converts the output to a probability score between 0 and 1:

$$y = \sigma(z) = \frac{1}{1 + e^{-z}} \quad (2)$$

A classification threshold of 0.5 was applied: scores above 0.5 were classified as Fatigued, and scores at or below 0.5 as Normal. The model was trained using binary cross-entropy loss with the Adam optimizer. Input sequences were constructed using a sliding window of 60 time steps, producing input dimensions of [samples, 60, 2].

After training, the model was converted to TensorFlow Lite (.tflite) format with Select TF Ops for LSTM operation compatibility on Android devices.

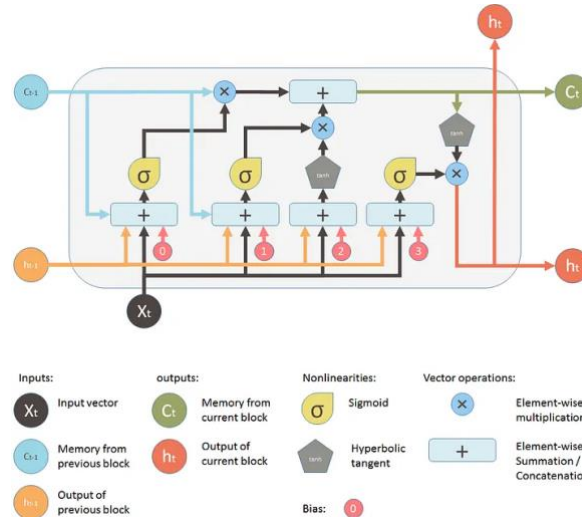


Fig. 4: LSTM model architecture diagram

3.4 Application implementation

The Android application was developed using Android Studio with Kotlin and the Jetpack Compose UI toolkit. The permission management for Health Connect follows a check-first-then-request pattern recommended by Android Developers [17]. When the user presses the data retrieval button, the application first verifies Health Connect SDK availability, then checks whether the required permissions (reading HeartRateRecord and OxygenSaturationRecord) have already been granted. If permissions are missing, the system launches the official Health Connect permission dialog. Once permissions are granted, the HealthRepository reads the last 24 hours of BPM and SpO₂ history data.

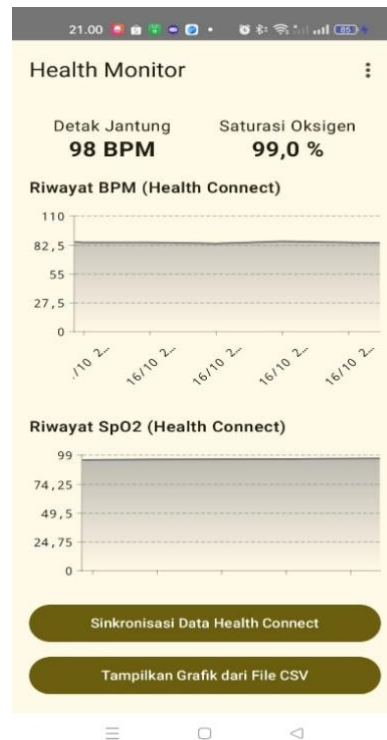


Fig. 5: Main application interface showing BPM and SpO₂ data visualization

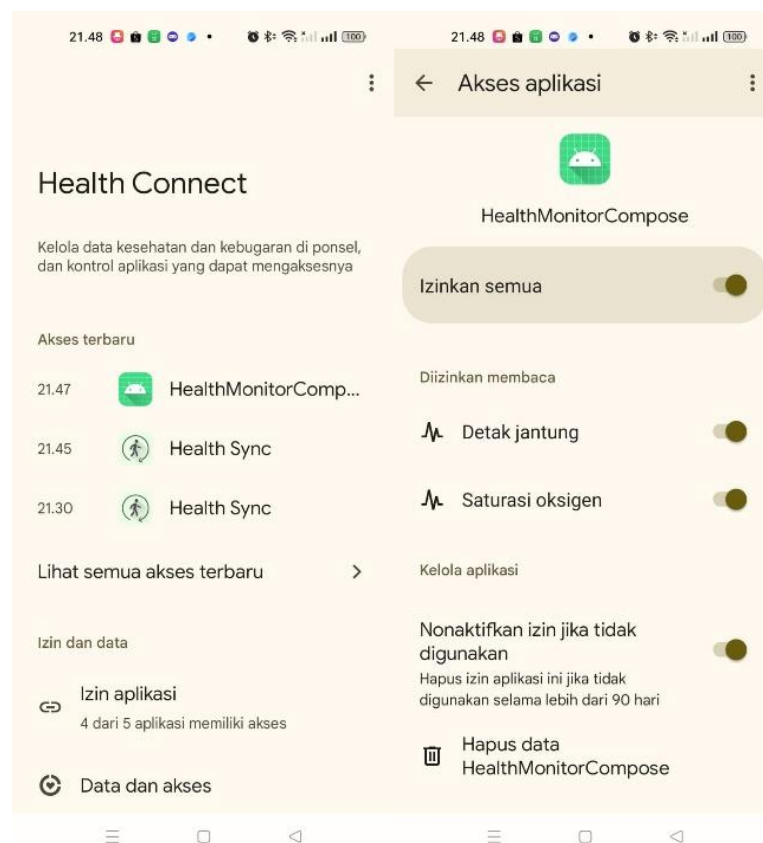


Fig. 6: Health Connect permission request dialog

3.5 Testing approach

Testing was conducted in three phases. First, sensor validity testing verified the PPG sensor's ability to distinguish human skin signals from inanimate objects by testing on various surfaces including human skin, water, table surface, and tissue paper. Second, functional

testing using black-box methodology verified all application features on a Realme 10 smartphone with Health Connect installed. Third, model performance evaluation measured classification accuracy, precision, and recall on the held-out test dataset.

4. Results and discussion

4.1 Sensor validity test results

Sensor validity testing confirmed the reliability of the Huawei Watch Fit 3 PPG sensor's wear-detection mechanism. On water and table surfaces, the sensor did not respond, as it failed to detect light absorption profiles consistent with hemoglobin. On tissue paper, the sensor briefly displayed false positive heart rate readings due to the high reflectivity of white tissue resembling skin reflection. However, the sensor's internal validation algorithm successfully identified the absence of dynamic pulse wave patterns and automatically disabled readings within seconds. This demonstrates effective error-correction mechanisms that prevent invalid data from entering the system.

Table 2: Sensor validity test results across different surfaces

No	Test Object (Surface)	Surface Characteristics	Sensor Response	Status
1	Human Wrist	Biological, presence of blood flow	Displayed stable BPM readings	Valid
2	Container with Water	Liquid, transparent, refracts light	Did not display BPM readings	Valid
3	Table Surface	Solid, hard, opaque	Sensor light turned off automatically	Valid
4	Tissue Paper Roll	White, fibrous, high light reflectance	Briefly displayed false BPM (80-90) for ± 5 seconds, then turned off	Valid (Auto-corrected)



Fig. 7: Documentation of sensor validity testing on various objects

4.2. Functional test results

Functional testing confirmed that all core application features operated correctly. The Health Connect SDK availability check, permission management flow, data retrieval from Health Connect, BPM and SpO₂ history visualization, and CSV data export all functioned as designed. However, the fatigue prediction feature failed during execution. When the user triggered the prediction, the application successfully acquired sensor data from Health Connect but experienced a force close when that data was fed into the TensorFlow Lite interpreter.

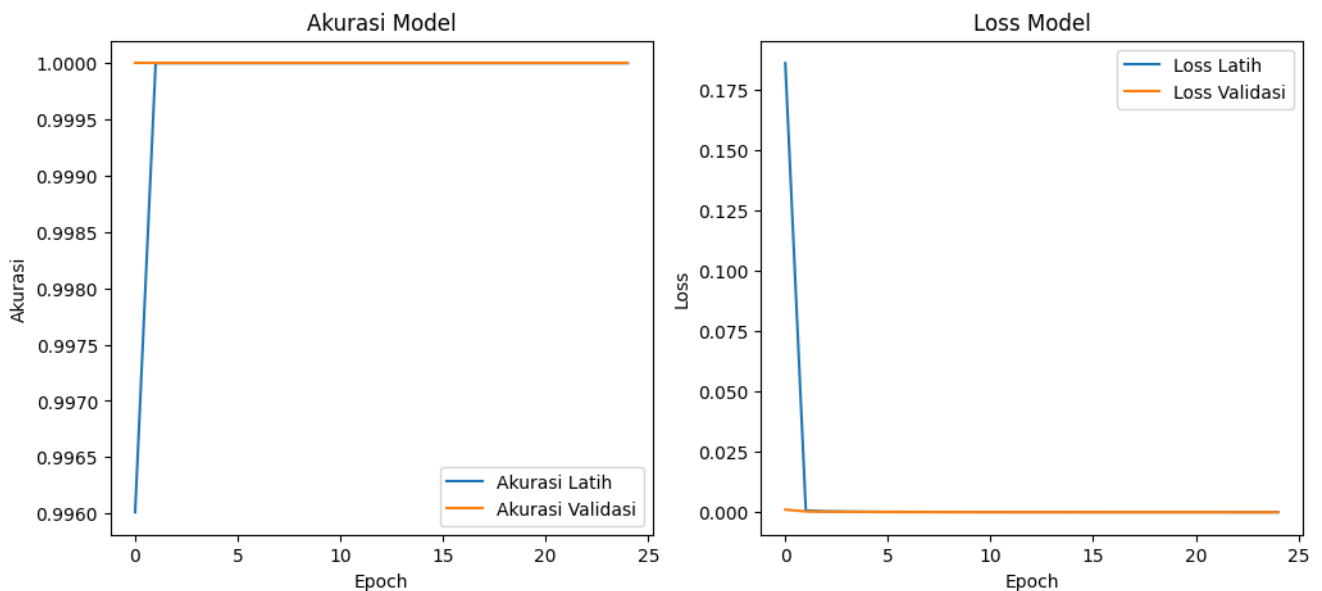
Table 3: Functional test results with pass/fail status for each feature

No	Tested Feature	Test Scenario	Expected Result	Actual Result	Status
1	Permission Acquisition	User clicks the "Fetch Data" button when permissions are not yet granted.	The application displays the official Health Connect permission request dialog.	The application successfully displayed the correct Health Connect permission dialog.	Pass
2	Data Retrieval	After permissions are granted, the user clicks the "Fetch Data" button.	The latest BPM and SpO ₂ data are read from Health Connect and displayed on the user interface (UI).	The latest BPM and SpO ₂ data were successfully read from Health Connect and displayed correctly on the UI.	Pass
3	Prediction Function	User clicks the "Fatigue Prediction Analysis" button after data is successfully retrieved.	The application executes the model and displays the prediction result ("Normal" or "Fatigued") on the UI.	The application failed to execute the model. Encountered a technical issue (app crash) during on-device inference.	Fail
4	Permission Management Menu	User opens the top menu (three dots icon) and selects "Manage Permissions".	The application opens the permission settings page for the app within the Health Connect system.	The link successfully redirected the user to the corresponding permission settings page within Health Connect.	Pass

Analysis of system error logs revealed unsupported operation compatibility issues. The LSTM architecture employed has specific computational complexities that the standard TensorFlow Lite library on the tested Android device did not fully support. This finding confirms that although the model was trained with high accuracy in the offline environment, on-device implementation of recurrent neural network architectures such as LSTM requires additional optimization or the use of hardware-specific delegates (GPU/NNAPI) for stable execution.

4.3. Model performance evaluation

Offline model evaluation was conducted in the development environment after training completion. The model achieved 100% accuracy on the held-out test dataset from the public dataset. This result indicates that the LSTM architecture possessed sufficient memory capacity to replicate the synthetic labeling rule logic ($HR > 100$ combined with $SpO_2 < 95$) perfectly within the controlled environment.

**Fig. 8:** Model training history showing accuracy and loss curves over epochs

However, this perfect performance warrants critical interpretation. The 100% accuracy constitutes a strong indication of overfitting, where the model memorized the deterministic rules of the training data rather than learning complex, natural fatigue patterns. The synthetic labeling approach—defining fatigue through a simple threshold combination—produced data with an inherently deterministic decision boundary. The LSTM model, with its high parameter capacity, easily learned to replicate this boundary without developing generalizable representations of fatigue.

This overfitting concern is further supported by the practical deployment results. The model's failure to execute on-device, while primarily caused by runtime compatibility issues, also illustrates the gap between controlled offline evaluation and real-world deployment. Even if

the runtime issue were resolved, the model's reliance on synthetic labels means it would likely perform poorly on real-world physiological data, which is inherently noisier and more variable.

4.4 Discussion

The results of this study present a significant paradox between theoretical promise and practical implementation. The research succeeded in fulfilling part of its objectives—the integration architecture design proved technically viable and effective in bridging the closed Huawei ecosystem into the Android Health Connect platform. The MVVM architecture demonstrated clean separation of concerns, and the check-first-then-request permission flow for Health Connect created a robust and user-friendly workflow, consistent with findings from Reza and Suryana [13] on functional health monitoring applications.

The most significant finding, however, is the dual failure of the prediction component. At the model level, the 100% offline accuracy was revealed as an artifact of overfitting rather than genuine predictive capability. This finding echoes concerns raised in the broader machine learning literature about the risks of using deterministic synthetic labels for training models intended for deployment on real-world data [15]. The labeling rule (BPM > 100 AND SpO₂ < 95) was too rigid and simplistic, causing the LSTM to memorize the rule rather than learn the complex, multifactorial nature of fatigue.

At the deployment level, the on-device inference failure highlights a fundamental technical challenge. The gap between training environments (Python-based frameworks) and mobile deployment runtimes (TensorFlow Lite) introduces compatibility constraints that are not always apparent during the development phase. This is particularly acute for recurrent architectures like LSTM, which involve sequential computation patterns that may not be fully optimized in standard TFLite runtimes.

Taken together, these findings reframe the study's contribution from "creating a fatigue prediction system" to "a case study identifying technical challenges in on-device deep learning deployment for health monitoring." This reframing, while acknowledging the system's limitations, still constitutes a valuable contribution to the field, as it provides concrete, experience-based guidance for future researchers attempting similar integrations.

5. Conclusion

This study developed an Android application for predicting user fatigue based on heart rate and SpO₂ data from a smartwatch. Three principal conclusions emerge from the research.

First, a functional Android application architecture was successfully designed to integrate physiological data from a heterogeneous wearable ecosystem. The system effectively utilizes the data synchronization chain through Huawei Health, Health Sync, Google Fit, and Health Connect, with Health Connect API serving as the application's single data source. The MVVM architecture ensured clean separation between data logic and presentation, while the Health Connect permission management flow provided a secure and user-controlled data access mechanism.

Second, the LSTM-based fatigue prediction model was successfully developed and trained in an offline environment. Evaluation on the public dataset test subset yielded 100% accuracy. However, this perfect performance was identified as an artifact of overfitting caused by deterministic synthetic labeling rules. The model learned to memorize the threshold logic rather than developing generalizable fatigue pattern recognition, indicating insufficient complexity in the training data labels.

Third, the on-device prediction framework was successfully constructed, encompassing TensorFlow Lite model loading and data preprocessing pipelines within the application. Nevertheless, significant technical challenges were encountered during model inference execution on the Android device, preventing the fatigue prediction feature from functioning as intended in the final prototype.

Based on these findings, several recommendations are offered for future research. Resolving the TensorFlow Lite runtime compatibility issue should be the primary priority, potentially through exploring alternative TFLite library versions, adjusting the model architecture, or employing hardware acceleration delegates such as NNAPI or GPU. The development of a dataset with subjective labels—collecting BPM and SpO₂ data from participants paired with self-reported fatigue ratings rather than synthetic thresholds—would substantially improve real-world prediction relevance. Once stable prediction is achieved, the system could be extended with notification features and personalized recommendations. Finally, background synchronization using WorkManager would ensure that prediction data remains current without requiring manual user intervention.

References

- [1] S. Díaz-Santos and P. Caballero-Gil, "Drowsiness detection in drivers with a smartwatch," *arXiv preprint arXiv:2210.04066*, 2022.
- [2] K. Lu, A. Sjörs Dahlman, J. Karlsson, and S. Candefjord, "Detecting driver fatigue using heart rate variability: A systematic review," *Accident Analysis & Prevention*, vol. 178, p. 106830, 2022.
- [3] S. Jiba and M. H. Mazlan, "Wearable health monitoring device based on IoT application for forklift driver," *Evolution in Electrical and Electronic Engineering*, vol. 3, no. 1, pp. 191–198, 2022.
- [4] M. F. Utomo and Sulhatun, "Eksplorasi peran smartwatch Android berbasis IoT dalam bidang kesehatan," *Jurnal Sains Dan Teknologi 4.0*, vol. 1, no. 2, pp. 15–23, 2024.
- [5] L. Lu, J. Zhang, Y. Xie, F. Gao, S. Xu, X. Wu, and Z. Ye, "Wearable health devices in health care: Narrative systematic review," *JMIR mHealth and uHealth*, vol. 8, no. 11, p. e18907, 2020.
- [6] J. B. Madavarapu, "HOT Watch: IoT-based wearable health monitoring system," *IEEE Sensors Journal*, vol. 24, no. 20, pp. 33252–33259, 2024.

- [7] R. Castaldo, P. Melillo, U. Bracale, M. Caserta, M. Triassi, and L. Pecchia, "Acute mental stress assessment via short term HRV analysis in healthy adults: A systematic review with meta-analysis," *Biomedical Signal Processing and Control*, vol. 18, pp. 370–377, 2015.
- [8] K. Mizuno, M. Tanaka, K. Yamaguti, O. Kajimoto, H. Kuratsune, and Y. Watanabe, "Mental fatigue caused by prolonged cognitive load associated with sympathetic hyperactivity," *Behavioral and Brain Functions*, vol. 7, p. 17, 2011.
- [9] M. Alruwaili, M. H. Siddiqi, K. Farid, M. Azad, S. Alanazi, A. Khan, and A. Khan, "A health monitoring system using IoT-based Android mobile application," *Computer Systems Science and Engineering*, vol. 47, no. 2, pp. 2293–2311, 2023.
- [10] V. Marce-Santurio, J. L. Moyano-Cuevas, and F. Lobillo, "Interoperability of mobile health applications," *Journal of Medical Systems*, vol. 43, no. 7, p. 178, 2019.
- [11] P. Davidson, P. Düking, C. Zinner, B. Sperlich, and A. Hotho, "Smartwatch-derived data and machine learning algorithms estimate classes of ratings of perceived exertion in runners: A pilot study," *Sensors*, vol. 20, no. 9, p. 2637, 2020.
- [12] J. Liang, X. Chen, and Y. Zhang, "Deep learning-based driver fatigue detection using wearable data," 2019.
- [13] Z. F. Reza and T. Suryana, "Aplikasi monitoring kesehatan dengan memanfaatkan smartwatch berbasis Android," *JUPITER: Jurnal Penelitian Mahasiswa Teknik Dan Ilmu Komputer*, vol. 2, no. 2, pp. 85–92, 2022.
- [14] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "LSTM fully convolutional networks for time series classification," *IEEE Access*, vol. 6, pp. 1662–1669, 2018.
- [15] B. Bent, B. A. Goldstein, W. A. Kibbe, and J. P. Dunn, "Investigating sources of inaccuracy in wearable optical heart rate sensors," *npj Digital Medicine*, vol. 3, no. 1, p. 18, 2020.
- [16] Android Developers, "Health Connect API," 2024. [Online]. Available: <https://developer.android.com/health-and-fitness/guides/health-connect>
- [17] R. Castaldo, "Mental fatigue impairs sport performance: A systematic review," *Journal of Sports Medicine and Physical Fitness*, vol. 59, no. 12, pp. 2030–2039, 2019.
- [18] Chaudhuri and P. O. Behan, "Fatigue in neurological disorders," *The Lancet*, vol. 363, no. 9413, pp. 978–988, 2004.
- [19] Google Developers, "Google Fit APIs," 2024. [Online]. Available: <https://developers.google.com/fit/overview>
- [20] E. Grandjean, *Fitting the Task to the Man: A Textbook of Occupational Ergonomics*. Taylor & Francis, 1988.