A GPS-Integrated Fitness Tracking System with Exercise Classification and Personalized Health Recommendations

Purushottam Kafle[1,*], Rakesh Mahar[2], Ramesh Tamang[3]
[1,2,3]Himalaya College Of Engineering, Tribhuvan University (TU), Lalitpur, Nepal
*Corresponding author: pk@okafal.com

Abstract
A GPS-Integrated Fitness Tracking System with Exercise Classification and Personalized Health Recommendations is a mobile app for increasing fitness through real-time activity tracking, exercise classification, and personalized health recommendations. Using GPS, it monitors walking and running, calculates distance and calories burned. Machine learning model, leveraging the device's camera, classify exercises (e.g., push-ups, squats) in real-time, also it counts the exercises with the algorithm analyzing calories burns providing immediate feedback. Also, with the data synchronization, with notifications delivering a daily health reminder. Addressing limitations of existing apps such as limited real-time classification and generic guidance this system integrates tracking, machine learning, and dynamic recommendations to promote sustained engagement and healthier lifestyles.

Keywords: GPS tracking, fitness app, exercise classification, machine learning, real-time tracking

1. Introduction

A healthy lifestyle has become a top priority in today's world, and smartphones play a major role in transforming how physical activity is tracked. While mobile applications can leverage sensors such as GPS and cameras to monitor activities, most still lack real-time analysis and personalized guidance [1]. To address these gaps, this project develops a GPS-Integrated Fitness Tracking System, a mobile application that combines activity tracking, exercise classification, and personalized health recommendations.

The system uses GPS to track outdoor activities, calculating distance, speed, and calories burned. Real-time exercise recognition is performed via CNN, classifying exercises such as push-ups and squats through the device camera and providing immediate feedback to enhance engagement and efficiency. Calories from exercise-specific MET values are combined with GPS-derived movement to yield a more accurate total energy expenditure [2], [3].

Unlike existing applications such as Apple Watch, Strava and Fitbit, which focuse on metrics like steps or heart rate without offering real-time exercise analysis or accurate calorie counts during workouts [4]–[6], the proposed system integrates multiple functionalities into a single user-friendly platform.

1.1 Problem Statement

Many existing fitness applications focus on providing exercises in animated forms and track only basic metrics such as steps, distance, or calories burned [7]. However, they often lack real-time exercise analysis, repetition counting, and personalized feedback [8]. In addition, most applications fail to integrate multiple features such as GPS tracking for outdoor activities, exercise classification and repetition counting, and personalized health suggestions into a single integrated system, which leads to fragmented user experiences [7], [9]. Such fragmentation across different functionalities makes it difficult for users to maintain consistent progress and effectively achieve their health goals.

1.2 Research Objectives

- To examine the effectiveness of a fine-tuned CNN model in real-time exercise classification.
- To validate if GPS can correctly track distance, speed, and calorie use with MET values.
- To investigate whether integrating exercise classification with GPS-based movement data improves the accuracy of calorie estimation compared to using either method alone.
- To examine how combining real-time exercise data, GPS tracking, and calorie estimation enables personalized health feedback and recommendations that enhance user motivation and progress.

1.3 Scope and Applications

- Real-Time Exercise Monitoring: Accurately identify and track multiple exercise for immediate feedback and guidance.
- GPS-Based Tracking: Measure distance, speed, and movement calories.
- Calorie Estimation Integration: Combine exercise and GPS movement data to improve total energy expenditure accuracy.
- Personalized Health Feedback: Generate tailored suggestions and motivational guidance to support user engagement and fitness goals.
- Medical Application: Can be used to track patients' locations in medical institutions.

2. Literature Review

2.1 Research on Fitness Tracking Applications

Fitness tracking applications have gained significant popularity as tools for monitoring physical activities and supporting

Article History:
Submission: 2025-05-29, Revision: 2025-10-30, Acceptance: 2025-11-11

healthy lifestyles. Platforms such as Google's Fit and Strava enable users to track activities including walking, running, and cycling, while also recording basic metrics such as step count, distance covered, and heart rate when integrated with wearable devices [1], [5].

However, despite their widespread adoption, these applications primarily rely on predefined activity metrics and offer limited functionality for detailed, real-time exercise analysis. They generally cannot accurately recognize or classify complex body movements, such as squats, push-ups, or lunges, which restricts their usefulness for comprehensive workout monitoring. Additionally, the functional focus of different applications is often segregated: some (e.g., Strava) prioritize GPS-based activity tracking, while others (e.g., Google Fit) emphasize general health monitoring. This lack of integration results in fragmented user experiences, requiring individuals to utilize multiple applications to track movements, analyze exercises, and receive personalized feedback.

These limitations highlight a clear gap in the current fitness technology landscape [10].

2.2 Pose Estimation and Exercise Classification

Pose estimation is essential for exercise classification in fitness applications, as it detects body keypoints and analyzes movements. Tools such as OpenPose [11] use Part Affinity Fields to estimate multi-person 2D poses in real time, accurately recognizing exercises like squats and push-ups, but require high computational resources and perform poorly under occlusions or low light [12]. MediaPipe Pose [13] is optimized for mobile devices, detecting 33 keypoints for exercises such as push-ups and lunges with real-time efficiency, though its accuracy decreases for fast or angled movements and it lacks built-in activity tracking or calorie estimation [14]. Classical computer vision approaches using OpenCV, including edge detection and Histogram of Oriented Gradients (HOG) descriptors [15], [16], can be adapted for pose estimation but typically require extensive customization, are less accurate for complex exercises, and are not ideal for mobile real-time tracking [2].

Overall, these pose estimation tools provide foundational capabilities for exercise recognition, but most lack integrated features such as repetition counting or calorie tracking. This limits their applicability in comprehensive fitness monitoring applications, motivating the development of integrated systems that combine real-time pose estimation, exercise classification, and personalized recommendations [15].

2.3 Integration of GPS in Fitness Applications

GPS-based fitness applications, such as Strava and Nike Run Club, use GPS to track users' outdoor movements, including running, walking, and cycling. By recording consecutive coordinates, these apps calculate metrics such as distance traveled, average speed, and route maps [4], [17].

Strava, in particular, is widely used by runners and cyclists due to its robust route-mapping features [4]. However, these apps predominantly target outdoor activities, leaving a gap for users who prefer indoor exercises or holistic fitness tracking.

2.4 Personalized Health Suggestions

Applications like MyFitnessPal provide static health recommendations based on predefined algorithms that calculate calorie intake and macronutrients. These suggestions lack dynamic adaptability to real-time user activities like walking, running, or exercise performance [18].

2.5 Local Storage for Data Management

Fitness applications manage user data using both local storage and cloud services. For example, Fitbit stores structured data such as steps, exercise duration, and calories locally using SQLite, and simple key-value data like user preferences using Shared Preferences [5], [19], [20]. Additionally, Fitbit supports cloud synchronization, allowing users to access their records across multiple devices [5].

Also, popular fitness apps such as Strava, Fitbit and MyFitnessPal typically separate activity tracking, exercise classification, and personalized recommendations into different modules [10], [18]. They do not provide a fully integrated, real-time system that combines local storage, GPS-tracked activity, exercise recognition, and personalized health guidance. In contrast, our system unifies these features into a single platform, ensuring comprehensive, timely, and user-centered fitness management.

2.6 Challenges in Existing Solutions

Despite significant advancements, existing fitness applications face several challenges:

- Limited Real-Time Feedback: Few apps provide immediate feedback during workouts, making it difficult for users to correct their form [21].
- Fragmented Features: Users often rely on multiple apps to manage different aspects of their fitness journey [1].

These challenges highlight the need for an integrated solution that combines real-time exercise detection, GPS tracking, and personalized health insights.

3. Related Theory

This section presents the theoretical foundations for the proposed system.

3.1 CNN Model

Convolutional Neural Networks (CNNs) are central to visual analysis tasks. The fundamental operation in a CNN is convolution, which slides a filter over the input to produce a feature map [22].

CNNs are a class of deep learning models designed to extract hierarchical features from input images or video frames. CNNs extract features via convolutional, pooling, and fully connected layers [23], [24].

3.2 MobileNetV2

MobileNetV2 is a lightweight CNN architecture tailored for mobile devices, designed to overcome the high resource demands of traditional CNNs. Researchers have used pretrained MobileNetV2 (e.g., with ImageNet weights) and fine-tuned it on exercise video datasets, achieving around 88% accuracy in detecting workouts like squats and planks, with low latency ideal for mobile inference [25]. Its architecture includes two key block types—stride=1 and stride=2—to extract features efficiently [2].

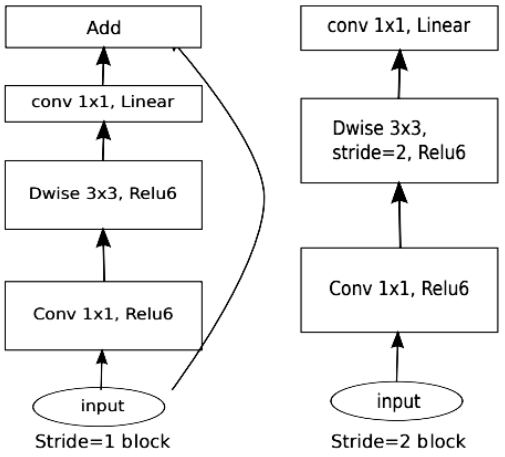

Figure 1. Convolutional Block used in the Mobile NetV2 Model

The network starts with the initial convolution layer.

- Initial Convolution Layer: Applies a 3x3 convolution with 32 filters, stride 2, and ReLU6 activation:

MobileNetV2 introduces depthwise separable convolutions combined with linear bottlenecks and ReLU6 activation, which significantly reduce computational cost while maintaining accuracy. [2].

4. Methodology

4.1 System Design

This section explains how the system components interact with each other to achieve the desired functionality.

Use Case Diagram:

The use case diagram outlines the interactions between the user and the application's features.

The primary actor in the system is the User, who interacts with the application to track fitness activities, receive health recommendations, and manage their profile.

*1) **Data Flow Diagram (DFD Level 0)***

Figure 3 illustrates the data flow between the user and the system, showing how inputs from the user are processed and how outputs are returned.

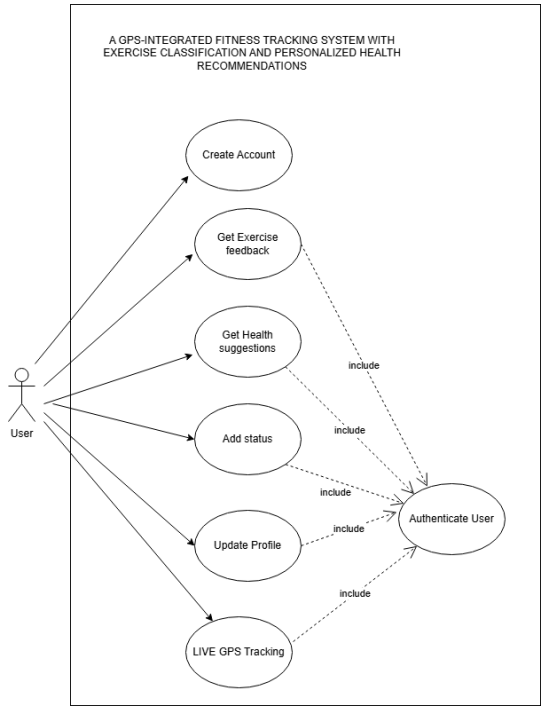

Figure 2. Use Case Diagram

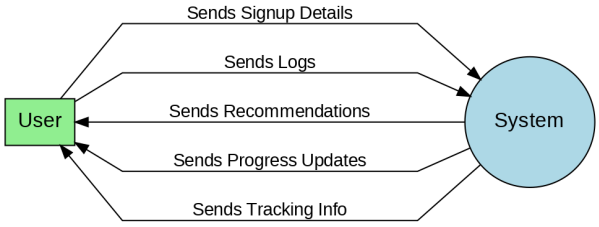

Figure 3. DFD Level 0 Diagram

*2) **Data Flow Diagram (DFD Level 1)***

The DFD Level 1 expands on Level 0 by breaking down the central "System" into its major internal processes, showing how data flows between these processes, external entities, and data stores. It provides a more detailed view of the system's operations while maintaining the same external entities as in Level 0.

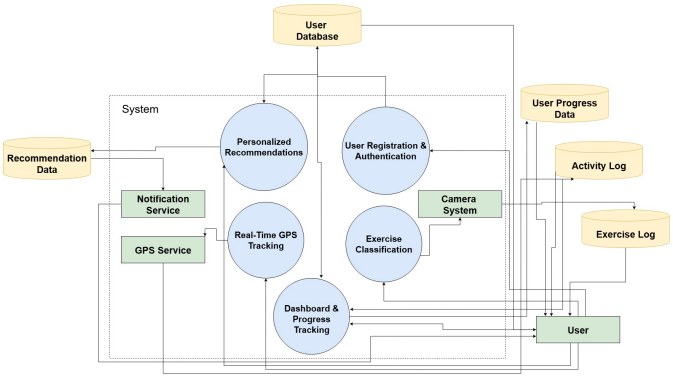

Figure 4. DFD Level 1 Diagram

*3) **Flowchart Diagram***

The Flowchart Diagram visually represents the step-by-step process flow within the application, helping to simplify the understanding of system operations.

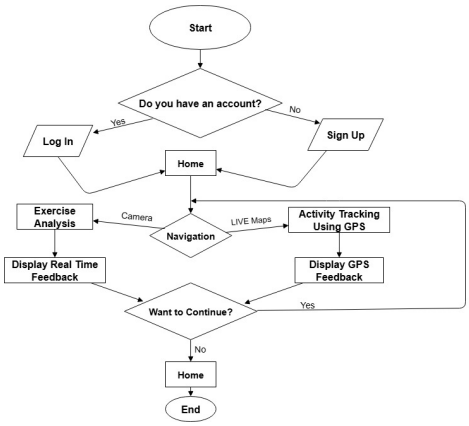

Figure 5. Flowchart Diagram

4.2 Basic System Architecture

Our system architecture is designed to integrate multiple components to deliver real-time fitness tracking, exercise detection, and personalized feedback.

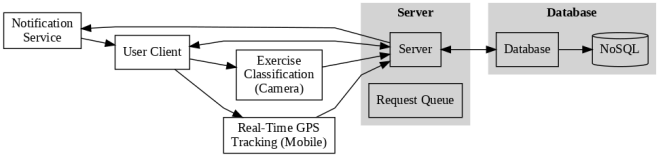

Figure 6. Basic System Architecture

The User Client enables users to interact with the application. The Server supports real-time sync and removes the need for separate backend infrastructure. The Request Queue prioritizes tasks to maintain responsiveness. The Server handles secure logins using Email/Password with support from custom session management. The NoSQL Database stores profiles and logs in a scalable JSON format ideal for real-time use. The Notification Service uses Cloud Messaging, push alerts, and Alarm Manager for timely updates. Exercise Classification (Camera) enables real-time workout detection, and Real-Time GPS Tracking (Mobile) provides live location, distance, and calorie tracking.

4.3 Exercise Classification

To classify exercises in real time, we utilized a Convolutional Neural Network (CNN) trained on a custom-built dataset. The images were labeled based on folder structure and encoded into Base64 format, then saved in a .csv file to streamline the data pipeline.

1) Data Collection

The initial dataset was sourced from Kaggle [26], consisting of 208 Base64-encoded images. To increase the diversity and robustness of the dataset, additional images were gathered from stock platforms, such as Shutterstock, expanding the dataset to 4,572 images, with 508 images per class across nine exercise categories: Bent Over Row, Glute Bridge, Lunges, Push-up, Chest Dips, Plank, Shoulder Press, Pull-up, and Squat.

The images were processed as follows:

- **Data Filtering**: First of all, Image Conversion technique was applied, the Base64-encoded images were decoded and saved as JPEG files. After decoding, the images were manually reviewed and cleaned, with erroneous images being removed from the dataset.
- **Labeling**: Images were organized into folders corresponding to each exercise class. We extracted only the relevant images by selecting the exercise name and then grouped them into the appropriate folder. In this way, each image was labeled according to its exercise category.
- **Additional Data Sourcing**: Images from stock image websites like Shutterstock were also used to expand the dataset, ensuring each exercise had 508 images. This led to a final dataset of **4572 images**.

The following preprocessing and augmentation techniques were implemented:

- **Brightness and Contrast Adjustment**: Adjustments to brightness and contrast were randomly applied to each image.
- **Flipping**: The images were flipped horizontally for diversity.
- **Data Generator**: Data augmentation techniques such as zooming, rotating, and adjusting brightness/contrast were applied to create more varied and robust data.

2) Data Preprocessing

Sampling: Data sampling was done by reading from the CSV file containing the image paths and labels. A random sampling approach was employed to ensure that all images from the dataset were used equally during training.

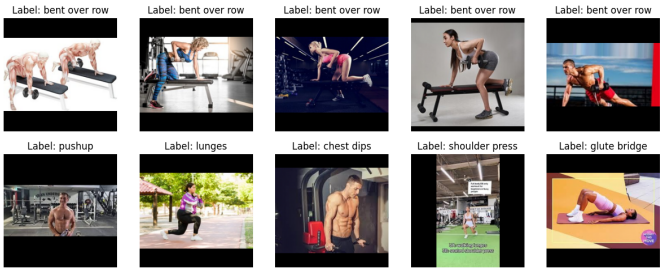

Figure 7. A Random Sample in Data Preprocessing

Image Preprocessing:

- **Base64 Decoding**: The Base64 images were decoded and resized by maintaining the aspect ratio also ensuring all images were converted to RGB format. This step is crucial for handling the encoded image representations efficiently and ensuring they are ready for further processing, as demonstrated in the data_generator function.

- **Normalization**: Images were normalized to a $[0, 1]$ scale using the formula: After converting the PIL image to a NumPy array, the pixel values originally in the range $[0, 255]$ for RGB channels are divided by 255.0 and this scaling is mathematically expressed as:

$$\text{Normalized Pixel Value} = \frac{\text{Original Pixel Value}}{255.0} \quad (1)$$

  For compatibility with the standard pretrained MobileNetV2 feature extractor, images were further normalized from the $[0, 1]$ range to the $[-1, +1]$ range, matching the input distribution used during model training and supporting stable and reliable feature extraction.
- **Padding**: For images that were resized to preserve their aspect ratio, black padding was added to ensure that all images were the target size of 224x224 pixels without distorting exercise poses.

The full generator implementation, including batch processing and label encoding.

3) Model Architecture

MobileNetV2, pretrained on ImageNet, was fine-tuned by unfreezing its last 30 layers and adding custom layers: Global Average Pooling, Batch Normalization, and Dense layers (1024, 512, 9 units) with ReLU activation, L2 regularization (0.005), and Dropout (0.5). The output used softmax for nine-class classification. The architecture was compiled with categorical cross entropy loss and Adam optimizer (learning rate: 0.00001).

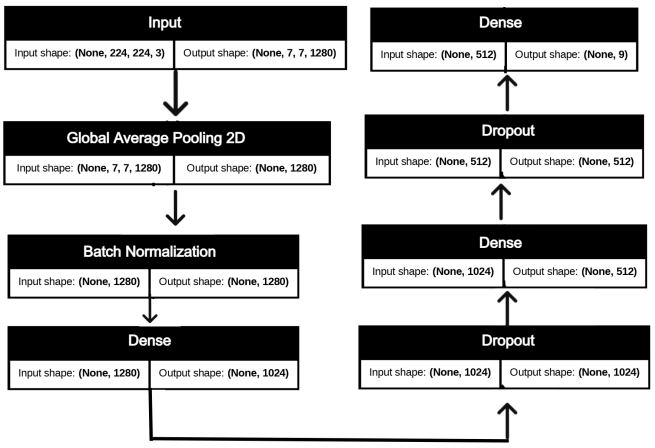

Figure 8. Fine-tuned MobileNetV2 architecture.

Figure 8 illustrates the fine-tuned model architecture used in this study.

The **ReLU (Rectified Linear Unit)** activation function was employed in the hidden Dense layers (1024 and 512 units) to introduce nonlinearity.

ReLU outputs the input directly if positive, otherwise zero, enabling the model to learn complex patterns efficiently while mitigating the vanishing gradient problem common in deep networks. [27].

The final layer used the **Softmax** activation function to generate class probabilities for the nine exercises. Softmax normalizes the output into a probability distribution, where each value sums to 1, allowing the model to assign confidence scores to each exercise class (e.g., Push-up: 0.9, Squat: 0.05) [28].

This facilitates multi-class classification and supports real-time decision-making, where the highest probability determines the predicted exercise.

The model was compiled using the Adam optimizer with a small learning rate, which adaptively balances past and current gradients. Categorical crossentropy was used as the loss function to penalize incorrect predictions, guiding the model to improve exercise classification over time.

4) Train and Test Split

In our approach, the dataset is not divided into fixed training and validation subsets by design. Two dynamic generators are employed: one applies light augmentations for training, while the other evaluates the same dataset with strong, unseen augmentations. Both generators reshuffle the data at the start of each epoch to produce randomized batches, ensuring exposure to diverse transformations. Importantly, augmented samples used for evaluation are never seen during training, allowing the design to assess both generalization and robustness under distributional shifts while maximizing data utilization.

5) Training

The model was fine-tuned by unfreezing the last 30 layers of the base model and adding custom dense layers on top. Training utilized separate generators for training, processing 4,572 images in batches of 32 over 50 epochs. Early stopping (patience=5) and model checkpointing ensured optimal performance.

Key training parameters included:

- **Learning Rate**: Set at 0.00001 to allow for gradual fine-tuning of the model.
- **Early Stopping**: To avoid overfitting, early stopping was used.
- **Dropout**: Dropout layers with a rate of 0.5 were added to regularize the model. A dropout rate of 0.5 was found to provide the optimal balance, reducing overfitting while maintaining high validation accuracy at Epoch 50. In our case, a 0.5 rate means that during each iteration, 50% of the neurons in the dropout layer were randomly set to zero, ensuring robust learning without significantly compromising the model's learning capacity.

6) Testing and Evaluation

After training, the model's performance was evaluated on the validation dataset. The model achieved good training accuracy and validation accuracy after 50 epochs. The performance graphs is included below and the training result are as follows:

- **Training Accuracy**: 87.36%
- **Validation Accuracy**: 96.08%

These results demonstrate that the model performs well on unseen data, indicating that the model is generalizing effectively.

7) Deployment

The model was deployed using the following approach:

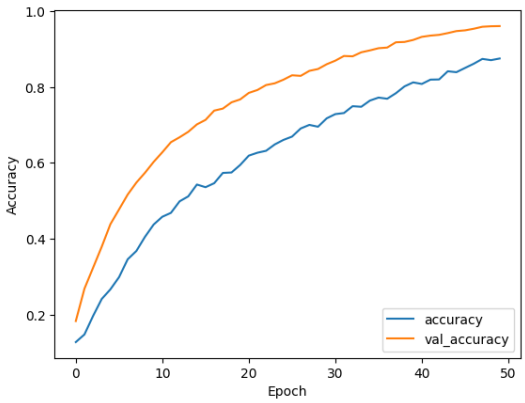

Figure 9. Accuracy vs epoch graph

- **Saved Model**: After training, the model was saved in the .keras format.
- **Real-Time Testing**: The trained model was tested using real-time input. A base64 image was passed to the model, and the prediction was made. In the Android app, the .tflite model was integrated with TensorFlow Lite and CameraX Core library to classify exercises from live camera input.
- **Prediction**: The model outputs the predicted exercise label using the label map provided.

The model was successfully able to predict the correct exercise from the base64 image input.

Now, to calculate calories burned during specific exercises, the system uses a custom Java method that combines exercise-specific MET values (e.g., Push-up: 7.0, Squat: 5.0) with user profile data (age, weight), logging results in the database.

Next, a delay of 500–800 ms is applied to ensure exercise posture consistency before confirming a prediction. Detected exercises (confidence greater or equals to 65%) are only considered valid if maintained across three consecutive frames. It reduces erroneous classification spikes (e.g., false Push-up or Squat detections).

Furthermore, exercises are classified in real time using a fine-tuned MobileNetV2 model deployed via TensorFlow Lite in the Android application. It captures video frames through Camera, preprocessed into 224×224 RGB images, and encodes them as Base64 for model input. The model processes these images, applying convolutional layers to detect spatial features (e.g., body postures), and outputs probabilities for nine exercise classes (e.g., Push-up, Squat) using Softmax, selecting the highest probability as the prediction.

We have used Base64 encoding for image input to streamline the data pipeline by converting images into a text-based format that can be easily stored, transmitted, and processed within the Android application. This approach ensures compatibility with the model input requirements, facilitates efficient handling of real-time camera frames, and simplifies integration

Finally, exercise type, repetitions, and duration are integrated with MET and personal attributes to compute calories, which are stored in user profiles to generate daily summaries. A rules-based scheduler provides personalized alerts for missed targets or inactivity, promoting engagement, healthier habits, and goal attainment.

4.4 GPS Tracking and Real-Time Maps

The GPS tracking component monitors the user's location in real time, calculates distances using the Haversine formula, and displays movement on an OpenStreetMap canvas.

1) Data Collection

Latitude and longitude coordinates are obtained from the mobile device's GPS sensor via Google Location Services.

2) Location Tracking

Coordinates are collected periodically to track movement and update the user's route in real time, supporting accurate activity metrics such as distance and speed. Distances between consecutive GPS coordinates were calculated using the standard Haversine formula, which determines the great-circle distance on a sphere based on latitude and longitude [3]. This approach allows accurate estimation of distance traveled for GPS-based tracking in real-world conditions.

The **Haversine formula** is used to calculate the distance between two geographic points based on their latitude and longitude.

3) Real-Time Mapping

- Map Integration: OpenStreetMap tiles are used as the base layer, with user positions plotted in real-time.
- Multi-User Display: A canvas layer draws the user's movement path using GPS coordinates. Other users within a defined radius (e.g., 5 km) are shown on the map, fetched from Realtime Database.

4) Data Storage

GPS coordinates, distance traveled, time spent, and calories burned are saved to the Database under the user's unique ID. And to store calorie values, we have used calorie calculation which is based on MET (Metabolic Equivalent of Task) values: Walking: 3.8 METs, Running: 8.0 METs.

$$\text{Calories} = MET * Weight(kg) * Time(hours) \quad (2)$$

To monitor user activity in real time, live GPS tracking is employed using Google Location Services. The application retrieves latitude and longitude coordinates at regular intervals and calculates distance changes using the Haversine formula. Based on calculated speed, movements are classified as walking (less than or equals to 6 km/h) or running (6–15 km/h). These activity metrics including distance, calories burned, and type of activity are stored daily in the database, allowing new tracking to commence every 24 hours. This approach ensures accurate monitoring of user movement and consistent logging for analysis.

Calorie estimation from movement is achieved by processing real-time GPS updates alongside user-specific information such as weight and age. Distance and speed are calculated to determine the activity type, and MET values are applied to estimate energy expenditure. The resulting calorie values are

formatted and stored with distance and activity type, providing a reliable foundation for personalized fitness recommendations.

To provide a view of user energy expenditure, the system integrates calories burned during exercises with those calculated from GPS-tracked movement. This combined approach produces total daily calorie data, capturing both structured exercises and routine physical activity. Such integration allows for a comprehensive representation of overall energy output, which is crucial for accurate feedback.

Based on the accumulated data, the system generates personalized health suggestions and notifications. Daily statistics including steps, calories, distance, and total activity points are used to create motivational messages and reminders. Notifications are scheduled to encourage engagement, such as prompting a user to take a short walk if step counts are low.

4.5 Personalized Suggestions

In the proposed system, the personalized suggestions module generates the best recommendations for users.

1) Data Collection

- User Profile: Collected during signup (name, age, weight) and stored in database.
- Activity Data: Real-time data from exercise classification and GPS tracking.

2) Points System

To quantify overall user activity, a points system is employed, combining distance, repetitions, exercise duration, and calories burned:

$$\begin{aligned} \text{Total Points} = & \frac{\text{Distance}}{80} \\ & + 2 \cdot \text{Exercise Reps} \\ & + 5 \cdot \text{Exercise Minutes} \\ & + 2 \cdot \frac{\text{Calories Burned}}{10} \end{aligned} \tag{3}$$

Components:

- Distance (meters): 1 point per 80 meters.
- Exercise Reps: 2 points per repetition.
- Exercise Minutes: 5 points per minute.
- Calories Burned: 2 points per 10 calories.

3) Time-Based Suggestions

To promote optimal health and adherence to daily activity, the system provides exercise prompts based on the user's time of day. This approach is grounded in circadian rhythm research and principles of longevity. For instance, higher-intensity exercises such as push-ups or squats are recommended in the morning when energy levels and metabolic responsiveness are typically higher, whereas lower-intensity or core-focused exercises like planks are suggested in the evening to reduce stress on the cardiovascular system.

These recommendations align with the findings of Sinclair and LaPlante [29], who highlight the importance of timing and regularity of physical activity in supporting metabolic health, mitochondrial function, and cellular repair mechanisms. By providing contextually appropriate prompts, the system encourages users to exercise at biologically favorable times, potentially enhancing both compliance and long-term health benefits.

4) Real-Time Notifications

The system delivers real-time feedback on user activity, including calorie expenditure and personalized motivational prompts. Drawing on longevity research, the system tailors recommendations according to cumulative activity points. Users with higher activity points receive messages encouraging the continuation of exercise intensity, whereas lower activity scores trigger prompts to increase activity to support metabolic function and overall health. This approach is informed by Sinclair's principles of aging, emphasizing the benefits of regular physical activity for cellular health and lifespan [29].

4.6 User Signup and Personalization

During the **user signup process**, the application collects basic details such as: Name, Age, Weight. This data is stored and used for personalized fitness recommendations:

The data collected during signup is used to recommend personalized goals, such as how many steps the user should take per day or the optimal workout plan.

5. Results & Discussion

This section presents the results obtained from the implementation of system and provides an in-depth analysis of its performance across its core components:

5.1 Exercise Classification

This section discusses the training outcomes and evaluation results based on the 50-epoch training process conducted. The selection of 50 training epochs was based on the model's convergence pattern: beyond roughly 40 epochs, improvements in validation accuracy became minimal, and model performance exhibited only minor fluctuations, indicating that further training would provide limited benefit.

*1) **Model Performance***

The training was performed using a batch size of 32 because it gives a good middle ground between faster training and stable updates to the model.

The training and validation results demonstrate that the MobileNetV2 model effectively learned exercise-specific features from the custom dataset. Training accuracy progressively increased from 11.96% to 87.36%, while training loss decreased from 12.1982 to 7.5508, indicating successful adaptation of pretrained weights to the new dataset and overcoming initial misalignment and input variance. Validation accuracy showed a sharper improvement, rising from 18.29% at Epoch 1 to 96.08% at Epoch 50, with validation loss decreasing from 11.2001 to 7.2678. Interestingly, validation accuracy consis-

tently outpaced training accuracy after Epoch 10 (e.g., 60.26% versus 42.91%), which can be attributed to regularization strategies such as dropout (0.5), L2 weight decay (0.005), and extensive data augmentation. These techniques mitigated the learning of noisy or distorted training patterns while enabling better generalization on unseen data.

We used L2 weight decay of 0.005 to avoid the model memorizing the training data, as it helps keep the weights smaller and more balanced.

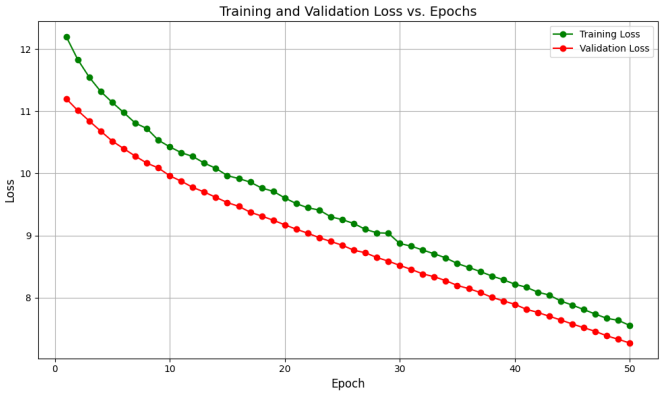

Figure 10. Training and validation loss vs. epochs

Figure 10 shows a steady decline in both metrics, with validation loss converging faster, suggesting the model effectively balanced learning and generalization.

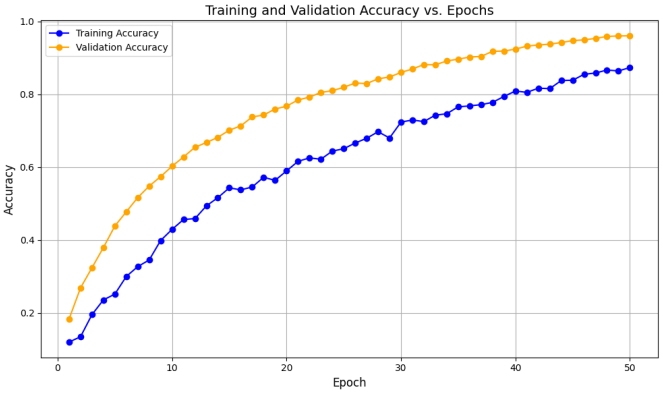

Figure 11. Training and validation accuracy vs. epochs

Figure 9 further illustrates that validation accuracy reached a plateau near 96%, while training accuracy increased more gradually. This gap is due to regularization and augmentation, which improved generalization rather than overfitting.

From a practical perspective, the strong generalization performance suggests that the model can accurately classify exercises across different users and contexts, even when environmental factors such as lighting or camera angle vary. The plateau in accuracy after around 40 epochs also indicates that further training beyond 50 epochs would yield diminishing returns, justifying the chosen training length as a balanced compromise between performance and efficiency.

2) Evaluation Outcomes

The trained model was evaluated on the validation dataset and deployed in the Android app for real-time testing:

- Quantitative Results: Final accuracies were 87.36% (training) and 96.08% (validation), demonstrating robust performance.
- Real-Time Performance: Integrated with TensorFlow Lite and CameraX, the model successfully classified exercises from live camera feeds. For example, a Base64-encoded Push-up image yielded a predicted label of "pushup" with high confidence (softmax output > 0.9), aligning with the label map.

The confusion matrix shown below demonstrates a clear assessment of the model's performance.

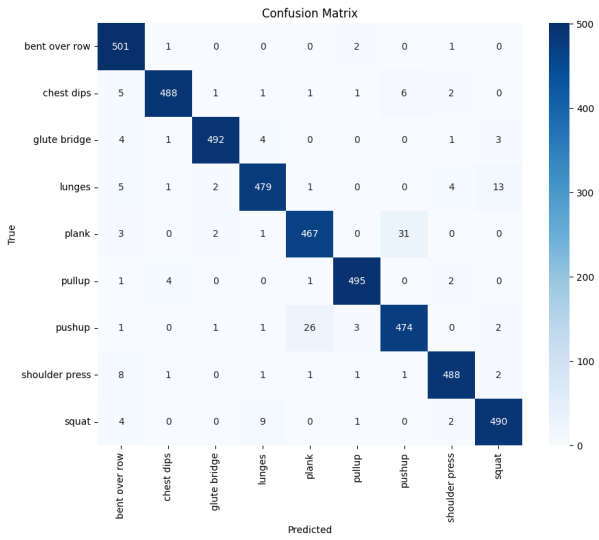

Figure 12. Confusion matrix

The confusion matrix reveals strong performance, with high diagonal values indicating accurate predictions for most classes. However, some classes, such as Push-up (474 correct, 26 misclassified as Plank) and Plank (467 correct, 31 misclassified as Push-up), show notable confusion, likely due to similar body postures. This suggests that the model struggles with distinguishing closely related exercises, particularly under varying lighting or angles, reflecting a limitation in its ability to generalize to subtle differences.

5.2 GPS Tracking and Real-Time Maps

This module tracks user movement using GPS, calculates distances via the Haversine algorithm, and visualizes activity on OpenStreetMap with a canvas overlay. Results are based on real-world testing over a 24-hour period.

1) Results

The application accurately displays user location, updating in real time as users walk or run, with a blue canvas path illustrating the route taken. Distances between consecutive GPS points are computed using the Haversine algorithm, and results including time spent and estimated calories burned are stored in the Realtime Database.

For a baseline case, a 25-year-old user weighing 70 kg completed a 1 km walk in 15 minutes (0.25 hours), resulting in 66.5 kcal burned (3.8 METs × 70 kg × 0.25 h), aligning with standard metabolic estimates.

To verify adaptability, a 24-year-old user weighing 48 kg walking the same distance under identical conditions burned 45.6 kcal, demonstrating that the system appropriately scales energy expenditure according to user-specific parameters.

Over 24-hour continuous usage, total movement was successfully aggregated, showing consistent values even after app restarts; for instance, one test day logged 5.2 km total distance with no loss of data integrity.

Additionally, markers for nearby users updated dynamically within a 5 km radius, with minimal device synchronization delays (less than equals to 5 seconds), confirming that the multi-user component supports responsive group-tracking scenarios.

Table 1. Calorie Expenditure by Subject

Subject	GPS(kcal)	Exercise(kcal)	Combined(kcal)
Subject 1	30	40	70
Subject 2	25	35	60
Subject 3	50	45	95
Subject 4	20	30	50

As shown in Table 1, combining GPS-based movement with exercise classification increases calorie estimates by 70–90% compared to using either method alone, providing a more accurate measure of total energy expenditure, especially for stationary activities.

5.3 Personalized Suggestions

The Personalized Suggestions Module generates tailored fitness recommendations based on a points system, time-based prompts, and real-time notifications, using data from exercise classification and GPS tracking.

*1) **Results***

Using the defined points system in Equation 3, the points were calculated as total. For a test case (1 km walked, 10 push-ups, 15 exercise minutes, 100 calories burned), the result was:

> Total Points = (1000/80) + (10×2) + (15×5) + ((100/10) ×2) = 12.5 + 20 + 75 + 20 ≈ 128
> A second test case was also performed (2 km walked, 20 push-ups, 10 exercise minutes, 200 calories burned), resulting in:
>
> Total Points = (2000/80) + (20×2) + (10×5) + ((200/10) ×2) = 25 + 40 + 50 + 40 ≈ 155
> Points and stats (steps, minutes, calories) were accurately saved to database.

These multiple measurements confirm that the points system consistently reflects user activity across different scenarios. These points serve as the basis for personalized health recommendations, guiding users with tailored suggestions to improve activity levels, calorie management, and overall fitness.

Aggregated results show high adherence to suggested exercises (mean = 9.6/10) and effective engagement with notifications (mean response = 4.6/5). Users responding consistently achieved higher points and burned more calories, demonstrating that real-time exercise classification, GPS tracking, and calorie estimation provide actionable feedback that motivates and supports measurable progress.

Table 2. Summary Statistics of Exercise Metrics

Metric	Mean	Std. Dev	Min	Max
Exercises Suggested	10.0	3.0	7	15
Exercises Completed	9.6	2.1	6	14
Total Reps	156	35	110	210
Calories Burned	344	80	250	470
Points Earned	128	22	100	160
Notifications Triggered	5.0	1.5	3	7
Notifications Responded	4.6	1.8	2	7

6. Conclusion

It advances personal health monitoring by integrating exercise classification (96.08% accuracy), GPS tracking, and personalized suggestions. MobileNetV2 and TensorFlow Lite enable efficient real-time analysis, while Firebase (NoSQL) ensures scalable data management. The system fulfills objectives of activity tracking, exercise classification, and tailored recommendations, supporting beginners and enthusiasts. We demonstrated that a fine-tuned CNN model can classify exercises in real time, and that GPS can accurately track distance, speed, and calories. The combination of exercise data and GPS improves calorie estimation, and integrating this information enables personalized feedback that facilitates user motivation and progress tracking. Furthermore, this study demonstrates the feasibility of implementing such a system on low-end Android devices using only core camera libraries, highlighting its practicality and accessibility.

Limitations include misclassifications under poor lighting, resets of the user trace path on the real-time map, simplistic suggestions, and the use of the same dataset for both training and validation with data augmentation, which was employed to maximize learning from limited data while providing a consistent measure of generalization without holding out a separate subset.

Future work includes fine-tuning additional layers to improve robustness against variations in posture, camera angles, and lighting, enhancing generalization across exercises. Additionally, background activity tracking could enable continuous monitoring, ensuring comprehensive data capture throughout the day. Also, incorporating a more diverse datasets in future iterations will further strengthen the model reliability.

Finally, personalization could be further enhanced by incorporating user-specific information, such as historical activity patterns, preferences, and physiological characteristics. Utilizing this data would allow the system to provide more accurate,

adaptive, and individualized health recommendations, thereby strengthening the usefulness of the system's recommendations. This project demonstrates the potential of integrated fitness tracking, with scalable applications in personal and medical contexts.

Acknowledgments

The author is grateful to Himalaya College of Engineering. We would also like to thank our supervisor, Ramesh Tamang, for reviewing the document and providing feedback to refine the report during the project. Also, the authors would like to thank their advisors and peers for their valuable insights and support during the development of this research. We also appreciate the institutional encouragement that made this work possible.

Conflict of Interest

The authors declare that there is no conflict of interest regarding the publication of this research.

References

[1] S. R. Mishra *et al.*, "Wearable technologies and health behaviors: A review," *Journal of Medical Internet Research*, vol. 20, no. 11, pp. 1–12, Nov 2018.

[2] M. Sandler *et al.*, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA, Jun 2018, pp. 4510–4520.

[3] R. W. Sinnott, "Virtues of the haversine," *Sky & Telescope*, vol. 68, no. 2, p. 159, 1984.

[4] Strava Inc., "Strava — run and cycling tracking," 2023, [Online]. Available: https://www.strava.com.

[5] Fitbit Inc., "Fitbit official site for activity trackers & more," 2023, [Online]. Available: https://www.fitbit.com.

[6] Apple Inc., "Apple watch: Health and fitness features," 2023, [Online]. Available: https://www.apple.com/watch/health-fitness/.

[7] D. King, F. Greaves, C. Exeter, and A. Darzi, "'gamification': Influencing health behaviours with games," *Journal of the Royal Society of Medicine*, vol. 106, no. 3, pp. 76–78, 2013.

[8] Z. Zhao, A. Arya, R. Orji, and G. Chan, "Effects of a personalized fitness recommender system using gamification and continuous player modeling: System design and long-term validation study," *JMIR Serious Games*, vol. 8, no. 4, p. e18808, 2020.

[9] A. Negreiros *et al.*, "Quality assessment of smartphone fitness apps used to increase physical activity," *Journal of Medical Internet Research*, vol. 24, no. 4, p. e38305, 2022.

[10] J. Kim and H. Lee, "Personalized health recommendations in mobile apps: A survey," *IEEE Transactions on Mobile Computing*, vol. 19, no. 5, pp. 1123–1135, May 2020.

[11] Z. Cao *et al.*, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 172–186, Jan 2021.

[12] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Columbus, OH, USA, Jun 2014, pp. 1653–1660.

[13] Google Inc., "Mediapipe pose documentation," 2023, [Online]. Available: https://developers.google.com/mediapipe/solutions/vision/pose_landmarker.

[14] R. W. Ouyang *et al.*, "Integrating gps and machine learning for fitness tracking," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7456–7465, Aug 2020.

[15] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 2008.

[16] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Kauai, HI, USA, Dec 2001, pp. I–511–I–518.

[17] Nike, Inc., "Nike run club: Running app for training plans and tracking runs," 2022, [Online]. Available: https://www.nike.com/nrc-app.

[18] Under Armour Inc., "Myfitnesspal - track your nutrition," 2023, [Online]. Available: https://www.myfitnesspal.com.

[19] A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database System Concepts*, 6th ed. McGraw-Hill, 2010.

[20] L. Miller and P. Brown, "Usability issues in fitness tracking applications," *ACM Transactions on Human-Computer Interaction*, vol. 27, no. 3, pp. 1–20, Sep 2020.

[21] J. Yang, Q. Liu, and K. Zhang, "Exercise recognition using depth camera and convolutional neural networks," in *Proc. Int. Conf. Machine Learning and Applications (ICMLA)*, Miami, FL, USA, Dec 2017, pp. 452–457.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[23] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[25] W. Chen, X. Li, and J. Wang, "Real-time exercise classification using mobilenetv2 on mobile devices," *IEEE Access*, vol. 8, pp. 123 456–123 465, Jul 2020.

[26] D. Contributor, "Exercise images with pose estimation," https://www.kaggle.com/datasets/asher213/exercise-images-with-pose-estimation, 2025, [Accessed: 20-Sep-2025].

[27] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010, pp. 807–814.

[28] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," *Neurocomputing: Algorithms, Architectures and Applications*, pp. 227–236, 1990.

[29] D. A. Sinclair and M. D. LaPlante, *Lifespan: Why We Age—and Why We Don't Have To*. New York, NY: Atria Books, 2019.