HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and Communication Technology

THESIS

Mobile-based Health Care Application Using Human Activity Recognition

NGUYEN DUC DUNG

Dung.nd160674@sis.hust.edu.vn

Supervisor:	Dr. Cao Tuan Dung	Signature
Department:	Department of Software Engineering	
Institute:	School of Information and Communication Technology	

Hanoi, May 2021

REQUIREMENTS OF THESIS

Student name: Nguyen Duc Dung

Phone number: +84347818811

Email: dung.nd160674@sis.hust.edu.vn

Class: ICT.02-K61

Program: Fulltime

Goal of the Thesis:

- Study and build a health care application applying machine learning in recognizing human activity.
- Build an application helping observe the health condition of users and their family.

Main Tasks:

- Week 1-2: Study and determine the topic of thesis.
- Week 3: Study and determine the main features of the application.
- Week 4-10: Develop and test the demo application.
- Week 11-15: Write report of thesis, user and implementation document and prepare final presentation.

Declaration:

I – Nguyen Duc Dung – warrants that the Work and Presentation in this thesis are performed by myself under the supervision of Dr. Cao Tuan Dung. All results presented in this thesis are truthful and are not copied from any other work. All references in this thesis – including images, tables, figures, and quotes – are clearly and fully documented in the bibliography. I will take full responsibility for even one copy that violates school regulations.

	Hanoi, June 2021 Author Nguyen Duc Dung

Attestation of the supervisor on the fulfillment of the requirements of the thesis

	Hanoi, June 2021 Supervisor Cao Tuan Dung

ACKNOWLEDGEMENT

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Cao Tuan Dung, for continuous support and guidance my graduation thesis for his patience. He not only gives me invaluable advices and comments to solve the challenges of the thesis but also teaches me a lot about the way to search needed information quickly and reliably. Without his suggestions and reminders, it is impossible for me to complete this thesis, so I would like to thank him again for great support and guidance.

Secondly, I would like to say thank you to all my lecturers who have taught me since the day I entered Hanoi University of Science and Technology five years ago. I am humble and grateful for the knowledge I learned from them and their willingness to support students.

For my friends during my 5-year span in Hanoi University of Science and Technology, I am really happy because of their appearance in my life and becoming my unforgettable memories. This university has given me more than I ever asked for and it shapes the person I am today.

Finally, I would like to thank my family, who have always stood by me, taken care for me, believed and encouraged my decision. Thank you for pushing me but at the same time giving me room to figure things out by myself. I am forever thankful for it.

ABSTRACT

In recent years, with the development of science and technology, Artificial Intelligence has become extremely powerful and achieved many marvelous milestones in performance and applicability. Applying Artificial Intelligence, especially Machine Learning, in building applications is considered the key point to develop features that can solve complex issues in real life.

Nowadays, although more and more mobile applications have been born with features that use Machine Learning model to improve their performance and accuracy, most of them are implemented with the architecture that mobile application uses model inference by calling APIs of server. It is also a good choice but it has an important weakness when building on the mobile platform is internet connection requirement. Store Machine Learning model directly in smartphones is the way that mobile applications can run features calling model inference even when the smartphone is disconnected from the internet. This way will be significantly effective in a specific domain like Healthcare because the application needs to be active every time and everywhere.

For those reasons, I chose to build a healthcare application on the Android platform that uses a Deep Learning, a branch of Machine Learning, model to recognize the user activity in real-time and then record it and remind the user if needed. The application also collects the data from sensors of the smartphone and calculates some metrics such as number of footstep and total burned calories but it can classify the footstep into many types like walking, jogging, downstairs and upstairs with recognition of human activity and improve the accuracy of metrics. However, the Deep Learning model I used just recognizes some basic human activities. I need to research and improve it to recognize more and more activities to help the application give user metrics with higher accuracy and helpful advice.

CONTENTS

LIST OF FIGURES

LIST OF EQUATIONS

LIST OF TABLES

LIST OF ACRONYM

API	Application Programming Interface
ANN	Artificial Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
HAR	Human Activity Recognition
SQL	Structured Query Language
NoSQL	Non Structured Query Language
JSON	JavaScript Object Notation
ML	Machine Learning
SDK	Software Development Kit
UI	User Interface
APK	Android Package Kit
XML	Extensible Markup Language

CHAPTER 1: INTRODUCTION

In the first chapter, I would like to introduce the overview topic of this thesis, the reasons that led to my decision to choose the problems to be solved in my graduation research. Moreover, this part contains the purpose and the solution of the topic, as well as the introduction of the report outline.

1.1 Motivation

Nowadays, smartphones are gradually becoming more popular and to be one of the most important belongings in human life. It makes many things in human life easier and more convenient for people to do such as take part in the courses, go shopping, carry out their payments and take care of their health themselves. Because of the utilities of smartphones, the time people spend using smartphones has increased tremendously, especially in the conditions like coronavirus pandemic, general lockdown and work from home implementations. Therefore, building and developing mobile applications plays an important role in the development strategies that make products be closer to users.

In the general growth of science and technology, Artificial Intelligence, Machine Learning, and Deep Learning have become more powerful and to be the hottest keywords in the fields that apply information technology. They can help programmers solve many difficult problems in real-life such as face detection, text recognition and human activity recognition. With applying Artificial Intelligence, applications will be smarter and can give users useful suggestions and advice. Hence, applying and implementing studies of Artificial Intelligence, Machine Learning, and Deep Learning in developing mobile applications is really important in making human life become more convenient and happier.

One of the difficult problems in real-life that needs to be applied to Machine Learning algorithms is Human Activity Recognition. It is not only the problem of Computer Vision when processing video input but also is the one when handling sensor-based data of mobile devices such as smartphones and smartwatches. Recognizing human activity from sensor-based data is one of key features for many important applications, especially about healthcare.

For these reasons, in this thesis, I have studied about the Human Activity Recognition problem, approaches to solve this problem and the way I can implement

Machine Learning techniques on mobile. From this theory, I build and train a simple deep learning model and develop an independent module that helps programmers use this model easily in recognizing human activity on mobile applications. With this module, I have built an application whose most of the main features use the result of the model. This application helps users track their activities all day and calculates the total burned calories with classifying the footsteps.

1.2 Objectives and scopes

For the motivation mentioned in the above section, the first objective of conducting this thesis is to study the Human Activity Recognition problem, approaching methods and its applications in real-life. With the objective that the application will be developed on mobile platform, I ought to consider the mobile-based approaches for this problem. Secondly, research popular machine learning models that have high performance in solving human activity recognition problem so that I can try to use to develop the application. Besides, I also need to learn the ways I can implement machine learning techniques in developing mobile. In detail, I need to find out the frameworks and libraries that help to implement a machine learning model on mobile and try to use them. Furthermore, I should research common models that are usually applied to implement a system with a mobile application using a machine learning model. Finally, with these studies, I try to implement a machine learning model on Android platform and build a health-care application that uses this model to recognize human activity in real-time.

With these objectives, in this thesis, I researched the Human Activity Recognition problem through published articles. Because of time limitations, I only tried to built and trained a basic deep learning model and applied it in developing the application. In addition, the application would be developed only on Android because Android is open environment and easier to develop. In order to implement Machine Learning techniques on mobile, I studied frameworks and libraries that support Android platform. And I also researched popular models for deploying systems using machine learning models. On the other hand, since the human activity recognition problem should be solved in mobile-based approaches, I needed to know what kinds of sensor are supported on Android, which of them are necessary for recognizing human activity and how to handle the data of them. Finally, for building the application, I researched the applications of human activity recognition, especially the result set of the studied deep learning model and developed main features using this result set.

1.3 Solution approach

For the objectives that are mentioned in the above section, I firstly researched machine learning models that are usually used to solve human activity recognition from data of smartphone sensors. I summarized the background theory of mentioned methods and chose the type of model that often has high performance. I also looked up dataset that is published on internet that can be used to solve this problem. Because of health-care application needs to use human activity recognition at everywhere in every time, in this thesis, I chose the model in which the machine learning model is stored directly on mobile. And of courses, the model would be trained on cloud since this approach can reduce the hardware requirements for smartphones. With this implementation model, the application can always call machine learning model inference and can recognize human activity even when the smartphone is disconnected from the internet. In order to implement this model, I needed to find the format that is not only the one machine learning model could be saved but also be supported to load and call inference on mobile. On the other hand, to collect sensor data of smartphones and process them to fit with the input of the model is an important requirement, so I needed to learn how to handle data of sensors on smartphones such as accelerometer and gyroscope on Android platform. In addition, I tried to build an independent module that implements storing and calling this model so that I could use this model easily and reuse it in developing new features or combining with other model to solve complex problems in the future. Finally, with the set of basic human activity that model can recognize, I developed the application that can recognize user activity real-time, report to users the chart of their activities in day and month and also warn users when they spend too much time to sit or stand. In addition, the application classifies footsteps of user into many activities so that it can give user more exact report about the total calories he or she burned in day and month.

1.4 Structure of thesis

The rest of this graduation thesis report is organized as follows:

Chapter 2 presents the background theory about Machine Learning – a subfield of Artificial Intelligence, Deep Learning and some algorithms and techniques of them. Secondly, it describes the Human Activity Recognition problem and approaches to solve it. The last content of this chapter is about implementing Machine Learning models on mobile and some popular models for deploying the system.

Chapter 3 presents the technologies used in this thesis. It contains techniques about programming on Android platform such as registering and processing data from the sensors of smartphone and handling local databases. It also introduces Google

Firebase and how to integrate its authentication and real-time database in mobile application. Finally, it lists some popular techniques that support developers implementing Machine Learning models on mobile applications.

Chapter 4 presents the content of application development and deployment including architecture design, package and class design, interface design, application construction, application testing and deployment.

Chapter 5 presents the content of the application's contributions and solutions in this system.

Chapter 6 presents a summary of the achieved results of the system and the orientation of future development solutions and makes experiences after completing building the system.

CHAPTER 2: BACKGROUND THEORY

Chapter 2 introduces the background theory of machine learning, artificial neural network, deep learning and deep learning models extended from artificial neural network: recurrent neural network (RNN), and long short-term Memory network (LSTM). Recurrent neural network and long short-term memory network are popular deep learning models that have high performance when used to handle input data in the form of sequence. In this chapter, I also present the definition of the Human Activity Recognition problem and some popular approaches for solving this problem.

2.1 Introduction to Machine Learning

2.1.1 Machine Learning

Machine learning is a branch of artificial intelligence, which is one of the hottest keywords of the field that applies information technology in recent years. Technically, machine learning is the study of computer algorithms that can improve automatically through analyzing data. Machine learning algorithms build a mathematical model based on sample data, known as a “training dataset”, in order to make predictions or decisions without being explicitly programmed to do so. Machine learning aims to provide increasing levels of automation in the knowledge engineering process, replacing much time-consuming human activity with automatic techniques that improve accuracy or efficiency by discovering and exploiting regularities in training data.

In mathematics, the machine learning method can be defined as: suppose that in a computer program, for a class of task T, which can be measured its performance by P, it requires experience E to improve, this program can be named as learning from experience E, for the task T, measured its performance by P. There are three main characteristics of the precise definition to be identified in machine learning: type of task T, experience E, and the specific criteria for the improvement of task P.

Machine learning has an essential position in the study of artificial intelligence. It is difficult to claim a system to be truly intelligent if it does not have the ability to learn, but intelligent systems in the past have generally lack the ability to learn. For example, they cannot be self-correcting an error, cannot improve their performance through experience, cannot automatically get and discovery the required knowledge. They are limited to deductive reasoning and lack of induction. Therefore, at most only able to prove the existing facts and theorems, but cannot discover new theorems, laws and rules. With the development of artificial intelligence, these limitations become more prominent. It is under such circumstance that machine learning gradually become the core of artificial intelligence research.

Machine learning algorithms are used in a wide variety of applications, where it is difficult or infeasible to develop conventional algorithms to perform the needed tasks. Some most trending real-world applications of machine learning are image recognition, speech recognition, automatic language translation, virtual personal assistant and self-driving cars. Recognizing human activity from video or sensor data is also a difficult problem that we need to apply machine learning algorithms to solve it.

2.1.2 Machine Learning approaches

Machine learning approaches are divided into four categories, depending on the nature of the “signal” or “feedback” available to the learning system: supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning.

Supervised learning is the method that is used to build the model for solving the problems based on the dataset containing both input data and their desired outputs. The training dataset of this kind of problem consists of pairs of input and output and corresponding labels. There exist two types of problems that are usually applied with supervised learning approach: classification and regression. The output set of the model of classification problem contains only discrete values while the one of regression problem is the continuous set of real numbers. Some popular methods used in supervised learning are K-nearest neighbors, linear regression, random forest and support vector machine.

On the other hand, unsupervised learning is applied for problems whose dataset contains only input data. Because no label is given to the learning algorithm, the method tries to find out the structure of inputs and bases on this structure to cluster data and discover the association of data. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning). The output of the problems that are applied with unsupervised learning can be a data cluster, hidden structure, or a trend. Some popular methods used in unsupervised learning are K-means, hierarchical clustering, mixture models and density-based spatial clustering of applications with noise (DBSCAN).

Because of wasting too much resources for labeling data or requirement of expert knowledge, the dataset of the problem may contain only a part of inputs that have corresponding labels. It cannot be completely applied with supervised learning or unsupervised learning. For this reason, semi-supervised learning will be used in this kind of problem. Semi-supervised learning is an approach to machine learning that combines a small amount of labeled data with a large amount of unlabeled data during training. This method may refer to either transductive learning or inductive learning. Some methods that are usually used in semi-supervised learning are generative models, low-density separation, graph-based methods and heuristic approaches.

Finally, reinforcement learning concerns how intelligent agents should take actions in an environment in order to maximize the notion of cumulative reward. And the feedback of the environment after performing these actions will support the model evaluate and update. Reinforcement learning differs from supervised learning in not needing labelled input/output pairs to be presented, and in not needing sub-optimal actions to be explicitly corrected. Some algorithms that are usually used in this method are associative reinforcement learning, deep reinforcement learning, inverse reinforcement learning and safe reinforcement learning.

In this thesis, human activity recognition is a classification problem and is approached with supervised learning.

2.2 Artificial neural network

2.2.1 Architecture

Artificial neural network (also called neural network) is a machine learning model designed to simulate the way the human brain analyzes and processes information. The human brain contains a large number of neurons that are connected together in order to spread nervous signals.

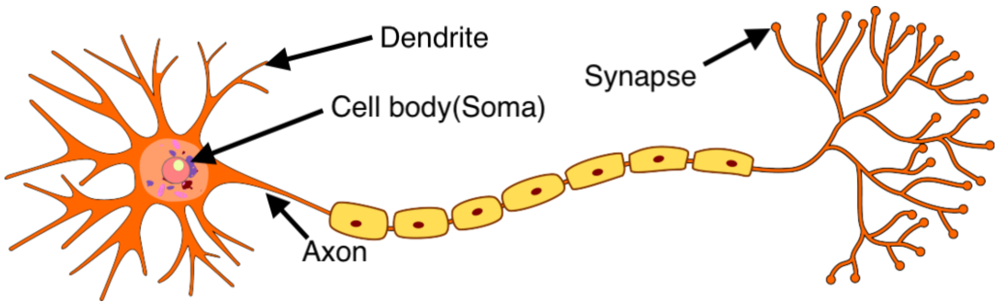


Figure 2.1 The architecture of neuron in human brain

Each neuron in the human brain has many dendrites for receiving signals, a cell body for processing and an axon for sending signals to other neurons. In order to simulate this architecture, each neuron in an artificial neural network called artificial neuron also has some input channels that will process input data and send them to the activation function and an output channel that can send signals to other neurons.

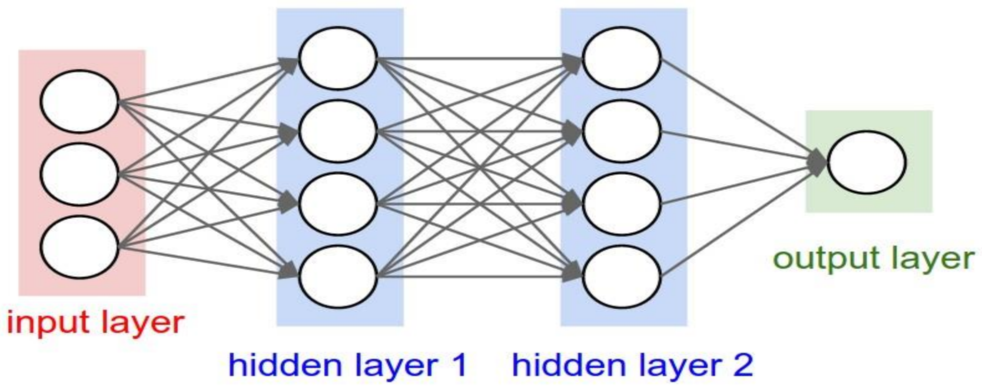


Figure 2.2 General architecture of Artificial Neural Network

The artificial neural network shown in Figure 2.2 is built from connected neuron layers, each of them can contain only one or many artificial neurons. An artificial neural network consists of an input layer, one or many hidden layers and an output layer. The amount of artificial neurons in each layer will depend on the problem that the model is solving.

$$z_j^k = \sum_i w_{i,j}^k \, . \, a_i^{k-1} + b^k$$

$$a_j^k = f(z_j^k)$$

Equation 2.1 Formula of value of a neuron

Suppose that n is the number of neurons in the input layer, X = {x_1, x_2, …, x_n} is the set of values of input data in the input layer. These values will be transferred to neurons in the next layer through connected edges. Let $w_{i,j}^k$ be the weight number of connected edge between ith neuron of layer k – 1 and jth neuron of layer k. In equation 1, a_j^k is the value of jth neuron of layer k, f is activation function and b is called bias. The value of each neuron in layer k is calculated from the value of neurons of the previous layer and the weight number of corresponding connected edges. With the input layer, the value of neuron a_j^1 normally equals to the corresponding input data x_j. The purpose of the activation function is to non-linearize input signals so that the model could be built with many hidden layers that have their own meaning. Some activation functions that are usually used are sigmoid, tanh, ReLU and Maxout. Training progress of a model is automatic regulation of weight numbers such that the output data are fitted with training dataset.

2.2.2 Parameter optimization

Artificial neural network is able to simulate a complex function through adjusting parameters w of the network. Training progress aims to create a simulated function that is more asymptotic with the objective function. It is a loop of two steps: first step is calculating the difference between the simulated function and the objective function and the second step is adjusting parameters w so that this difference could be decreased. This difference is calculated based on a loss function. The loss function can be chosen so that it would be suitable with the considered problem and used algorithms. Some popular loss functions are cross-entropy, triplet-loss and Euclidean distance.

The progress that the model adjusts its parameters w in order to reduce the difference between the simulated function and objective function is called minimization of loss function. The loss signal of each neuron can be calculated based on the value of loss function and transferred to neurons of the previous layer and the parameters w can be updated immediately. This progress is called backpropagation. The loop of feedforward and backpropagation will make the value of loss function be decreased and reach the minimum point as near as it can. There are many optimization algorithms that can be used such as Stochastic Gradient Descent, Nesterov accelerated gradient, Adagrad, RMSprop and Adam.

2.2.3 Overfitting issue and solutions

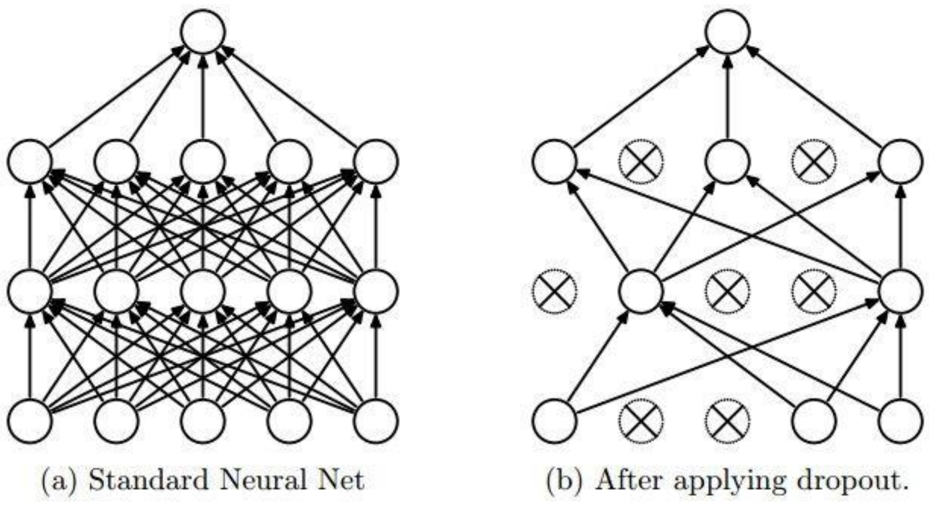


Figure 2.3 Neural network applying dropout

Artificial neural network models with architecture of many hidden layers are able to learn complex characteristics and relations between input and output data. However, if the training dataset is not big enough, the found complex relations can make noise for the model because these relations may exist in training data but not in practice data. For this reason, the accuracy of the model can be reduced. This problem is called overfitting.

A simple technique that can resolve the overfitting problem is Dropout. The Figure 2.3 describes the way the Dropout turns off some neurons of each layer randomly. At each step of training progress, when the signal is forwarded to each layer, each neuron of this layer has a probability of keeping on network p and a probability of removing from network 1 – p. Each removed neuron is turned off both its input and output channels. With this mechanism, Dropout creates a less narrow network than a standard neural network. This network can learn less complex relations from the dataset and reduce the impact of overfitting problems.

2.3 Introduction to Deep Learning

Deep learning (also known as deep structured learning) is a kind of artificial neural network model with large and deep architectures, yet different from traditional neural networks in specific computation in each layer which consists of many neurons and computational units. These neurons make a summation of data or information from previous neurons via an operation of a nonlinear function, simultaneously processing inputs and generating outputs sent to next neurons in the same layer. With a series of complicated computations in many layers in the middle of the neural network, the final layer will carry out a classification, regression or fitting.

In deep learning, the process of learning is an assignment of searching proper powers or weights making the neural network reach designed proposes. In order to learn with much more accuracy, deep learning is constructed based on plenty of neurons and layers as well as special connection fashions according to various practical problems.

Deep learning architectures such as deep neural networks, deep belief networks, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, machine vision, speech recognition, natural language processing, and machine translation. They have produced results comparable to and in some cases surpassing human expert performance. The power of deep learning has been proven with many successful researches and applications.

2.4 Long Short-Term Memory RNNs

The Long Short-Term Memory Recurrent Neural Networks (LSTM RNN) has proven highly efficient in utilizing memory cells in recurrent networks, under certain conditions. Before introducing LSTM, a brief introduction to recurrent neural networks follows.

2.4.1 Recurrent neural network – RNN

In some cases, in order to predict or resolve a problem, we need to follow the events occurring in the past. For example, when we watch a movie, we ought to follow from the beginning to the end so that we can understand the plot of the movie. For this reason, a traditional model with the input and output being independent is not able to resolve this kind of problem. This is the reason why we need recurrent neural networks.

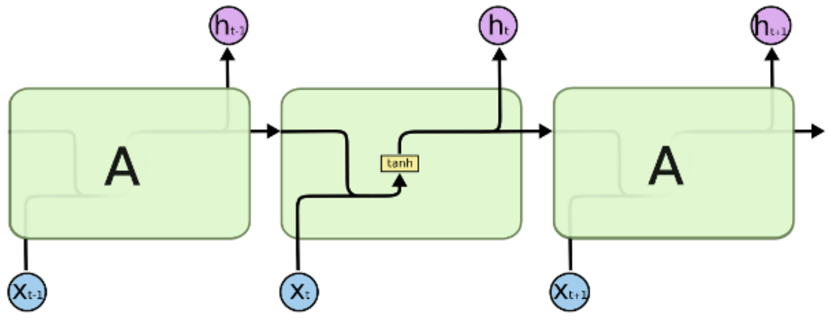


Figure 2.4 The architecture of Recurrent Neural Network

A recurrent neural network is an artificial neural network with connections in cycles, where data in a feed-forward neural network is only flowing in one direction over the layers. Information in an RNNs travels in loops from layer to layer so that influence is taken from previous steps. This group of networks adds to the ability to algorithmically preserve a temporal dynamic state or memory. It is the natural architecture of neural networks for sequential data.

In Figure 2.4, h_t is the hidden state at time t. It is calculated based on previous hidden states h_{t-1} and the input x_t of the current step and after that, it is going to be used to calculate the hidden state at the next time.

In theory, the combination of the state of the previous step and the input of the current step to calculate the value of the current state is able to memorize the information in time flow. However, in practice, the architecture of RNNs restricts its long-term memory capabilities, which are limited to only remembering a few sequences at a time. Consequently, the memory of RNNs is only useful for shorter sequences and short time-periods. This problem is called long-term dependencies. In order to resolve this problem, Long Short-Term Memory network was introduced.

2.4.2 Long Short-Term Memory network

Long short-term memory network is a recurrent neural network architecture that overcomes long-term dependencies. It learns to detect time intervals separated by thousands of steps without loss of short time lag capabilities. It operates over noisy and incompressible input sequences over an appropriate gradient-based learning algorithm. Results are achieved using a constant error flow through internal states of special units within the algorithm.

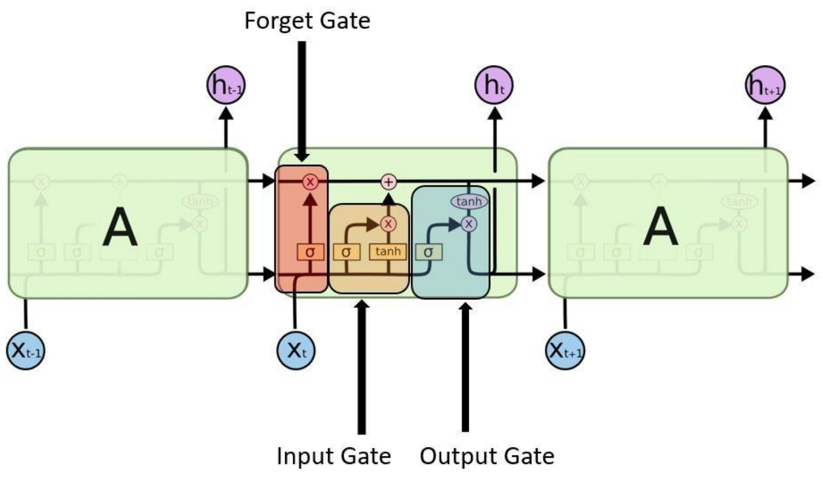


Figure 2.5 The architecture of Long Short-Term Memory network

Differ from recurrent neural networks that contain only an activation function *tanh* in cell core (Figure 2.4), Long Short-Term Memory networks are improved from RNNs with four layers that react together at each time step. The main point of LSTM is to retain the value of cell state through the loop from step to step. The value of cell state is only impacted by some linear transformers, so the information it stored seems to have no change. Hence, with the cell state, LSTMs can memorize the total information of time sequence.

Besides, LSTMs are able to add or remove information from cell state by using a gate mechanism. There are three gates in each cell core: input gate, forget gate and output gate. These gates can modify the amount of information going through by using a sigmoid layer and a vector product. The output of the sigmoid layer that is in [0, 1] is corresponding to the amount of retained information. The first gate that signal goes through is the forget gate. It is responsible for removing information that is no longer necessary for the completion of the task. This step is essential to optimizing the performance of the network. Secondly, input gate decides to add information from input of current step to cell state. Finally, the output gate is responsible for selecting and outputting necessary information based on the result of input gate and the value of cell state.

Distinguishing between the information that follows time flow and the one at each step makes LSTMs resolve the problem of long-term memory. Because a human activity is a sequence of sensor signals, in order to recognize it, we need to process the whole sequence of signals from the beginning to the end of this activity. Therefore, Long Short-Term Memory networks are suitable for applying to resolve problems in recognizing human activity.

2.5 Human Activity Recognition

Human Activity Recognition is a difficult problem that could not be resolved with high efficiency by using some normal algorithms. Machine learning and especially deep learning algorithms are proven that they have high performance when used to recognize human activity. So, this section gives an overview about the human activity recognition problem and some approaches to this problem.

2.5.1 Human Activity Recognition problem

Human Activity Recognition (HAR) is the problem of identifying a physical activity carried out by an individual dependent on a trace of movement within a certain environment. The task of HAR is to classify body gesture or motion, and then determine or predict state of action or behavior. The extensive applications of HAR, appearing in military health care, physical recovery from disability or injury and clinical deformity correction, are drawing more and more attention on the further development and exploitation from industry and academe. Especially, in public health care, with the pervasion of portable personal digital devices such as smartphones, intelligent watches and multimedia terminals, generating a great number of different types of chronic data, for instance, video recorders, photos streams and spatial-temporal logs, there will be the significant need for personal customization using human activity recognition.

2.5.2 Approaches

Human activity recognition tasks can be divided into two classes, including space-time approaches in computer vision and sequential approaches in time series analysis [1].

In space-time approaches, the essential for recognizing human activities is to measure the similarity between two volumes in images. For example, we can consider the changes of shapes on a series of images, which is corresponding to a moving human being or compare the patches of volumes. In recent years, with the development of neural network applied in processing images, deep convolutional neural network was applied in action recognition based on learning semantic trajectory-pooled data from raw video. Long-term recurrent convolutional networks have a significant advantage in solving visual recognition by its memory elements in each network layer.

In sequential approaches, traditional statistical techniques are initially proposed for handling human activity recognition tasks. With this approach, we can process one input as a signal, with sequential statistical features extractions, indicating all behaviors are the linear combination with different weighted statistical features. Linear time invariant model can also be used with its ability of classifying a new input with similar features such as slow walk and fast walk. The long short-term memory recurrent neural network was proven to be suitable to handle input data in series.

2.5.3 Human Activity Recognition with wearable sensors

In the scope of my thesis, I only concern the approaches that solve the human activity recognition problem through wearable sensors of some technical devices such as smartphones and intelligent watches. They are called sensor-based approaches while the approaches that use data of visual sensors are called vision-based ones.

For sensor-based human activity recognition, in order to record movement or change in movement, sensors such as tri-axial accelerometer and gyroscopes, capture data while the activity is being performed. A tri-axial accelerometer data detects acceleration or movement along the three axes and a gyroscope measures rotation along the three axes to determine direction. Data recorded is along three dimensions of the X, Y and Z axis at the specified frequency.

The challenge arises as there is no explicit approach to deduce human actions from sensor information in a general manner. The large volume of data produced from the sensors and use of these features to develop heuristics introduces the technical challenge. Storage, communication, computation, energy efficiency, and system flexibility are some of the aspects which need to be analyzed in detail to build a robust activity recognition system.

For sensor-based human activity recognition, information from wearable sensors are time series data. Traditional models were built by machine learning approaches from extracting features to classifying and prediction activity patterns, in which Hidden Markov Model (HMM) and Support Vector Machine (SVM) are always more popular previously. Recently, deep learning algorithms, like Recurrent Neural Networks (RNNs), play an essential role in constructing human activity recognition models because of its powerful learning ability would automatically have a comprehensive grasp of features from collected data, completely different from previous procedures of data-processing with handcrafted features. Another important advantage of modeling with RNNs is the ability of memorizing the information in time series. The performance for human activity recognition systems using RNNs is significantly successful with data of wearable sensors.

2.6 Implement Machine Learning model on mobile

After having overview about machine learning, artificial neural network, deep learning (a kind of artificial neural network model), Long Short-Term Memory recurrent neural network (a power model based on deep learning theory for processing input data in sequence) and the definition and some approaches of Human Activity Recognition problem, the last section of this chapter introduced about the way a machine learning model can be implemented on mobile application.

2.6.1 Implementing models

In order to apply Machine learning techniques into constructing new features in the mobile application, we not only need to design and develop user interface on the mobile app but also need to organize data, preprocess data, build the Machine learning model, and train it with our dataset. The problems we first face are where we will store our dataset, where we will train the model, where we will store this model, and how we can make our mobile application do prediction with this model. The following architectures are the most popular ones that are used in implementing features applying Machine learning techniques on mobile applications.

Firstly, we can organize and store our data on the server-side and we will build and train the Machine learning model here. After we finish training the model, we also store the model on the server and create some APIs for calling from the client. The mobile app will do predictions by calling these APIs and get the result from the server. This is the most popular way used in implementing Machine learning techniques in developing applications. The advantages of this architecture are that we can apply for any platform of the client (web, mobile, etc) and the client will easily use the APIs. However, the disadvantage of this way is that the client needs to connect to the internet in order to call APIs.

Secondly, we also store data on the server-side and train the model online. However, we will store our model that is trained with our dataset on our mobile app directly instead of storing it on the server-side. By using this architecture, the client can do predictions with the Machine learning model without connecting to the internet. With this model, the machine learning model must be saved in a suitable format that the application can easily load and use to predict.

Finally, we can store our dataset on cloud and do the model training directly on mobile here. And we will store the model on our mobile app of course and do predictions by using this model. The advantage of this way is we can update our dataset and re-train the model regularly. By being re-trained with a new dataset, the model may provide higher accuracy. However, storing data and training models directly on the mobile might make our app heavier and consume more resources on the mobile phone.

In this thesis, in order to make the healthcare application be able to use the machine learning model to recognize user’s activities every time at everywhere, the model must be stored directly on mobile. And with the purpose of making the training and building model easier, I decided to implement the model in which the model is built and trained at server-side and after that stored on a mobile phone.

CHAPTER 3: USED TECHNOLOGY

After illustrating the background theory of my thesis and introducing about human activity recognition problem in Chapter 2, this chapter will introduce main technologies used in my thesis. It contains techniques in Android programming about collecting sensor data and handling local storage of smartphones, introduction about Google Firebase, the service of Google that I used to implement quickly a backend for system and some frameworks and libraries that support developers implementing machine learning models on mobile.

3.1 Android platform framework

3.1.1 Collect sensor data

Most Android-powered devices have built-in sensors that measure motion, orientation, and various environmental conditions. These sensors are capable of providing raw data with high precision and accuracy, and are useful if we want to monitor three-dimensional device movement or positioning, or we want to monitor changes in the ambient environment near a device. For example, a game might track readings from a device's gravity sensor to infer complex user gestures and motions, such as tilt, shake, rotation, or swing. Likewise, a weather application might use a device's temperature sensor and humidity sensor to calculate and report the dew point, or a travel application might use the geomagnetic field sensor and accelerometer to report a compass bearing [2].

The Android platform supports three broad categories of sensors, including motion sensors, environmental sensors and position sensors. Motion sensors measure acceleration forces and rotational forces along three axes. This category contains accelerometers, gravity sensors, gyroscopes, and rotational vector sensors. Environmental sensors measure various environmental parameters, such as ambient air temperature and pressure, illumination, and humidity. They include barometers, photometers, and thermometers. Sensors of the last category – position sensors – measure the physical position of a device. This category consists of orientation sensors and magnetometers.

The Android sensor framework lets us access many types of sensors. Some of these sensors are hardware-based and some are software-based. Hardware-based sensors are physical components built into a handset or tablet device. They derive their data by directly measuring specific environmental properties, such as acceleration, geomagnetic field strength, or angular change. Software-based sensors are not physical devices, although they mimic hardware-based sensors. Software-based sensors derive their data from one or more of the hardware-based sensors and are sometimes called virtual sensors or synthetic sensors. The linear acceleration sensor and the gravity sensor are examples of software-based sensors.

However, few Android-powered devices have every type of sensor. For example, most handset devices and tablets have an accelerometer and a magnetometer, but fewer devices have barometers or thermometers. Also, a device can have more than one sensor of a given type. For example, a device can have two gravity sensors, each one having a different range.

3.1.2 RxJava

Now we can see that more and more developers are using RX related technologies to develop apps, Java backend and other fields. In open source communities and Internet companies, Rx, responsive programming, and functional programming are popular. Therefore, this subsection introduces Rxjava, a function library that helps developers handle asynchronous tasks in programming Android applications.

3.1.2.1. What is ReactiveX?

The definition given by Microsoft is that Rx is a function library, which allows developers to use observable sequences and LINQ style query operators to write asynchronous and event-based programs. With Rx, developers can use Observables to represent asynchronous data flows, LINQ operators to query asynchronous data flows, and schedulers to parameterize the concurrent processing of asynchronous data flows. RX can be defined as RX = observables + LINQ + Schedulers. The definition of ReactiveX.io is that Rx is a programming interface for asynchronous programming using observable data streams. ReactiveX combines the essence of observer mode, iterator mode and functional programming [3].

3.1.2.2. What exactly is RxJava

RxJava's self-introduction on GitHub homepage is "a library for compositing asynchronous and event based programs using observable sequences for the Java VM". This is RxJava, which is very precise. In fact, the essence of RxJava can be summed up as the concept of asynchrony. In essence, it is a library for asynchronous operations. The asynchronous implementation of RxJava is implemented through an extended observer pattern [3].

The main benefit of RxJava is also asynchronous. Why use it instead of the existing Thread, Thread Pool Executor, Android's AsyncTask, Handler? In fact, it is simple and easy to use. The key point of asynchronous operation is the simplicity of the program, because in the case of complex scheduling processes, asynchronous code is often difficult to write and understand. Just like the AsyncTask and Handler created by Android, they are all designed to make asynchronous code more concise. RxJava also has the advantage of simplicity, but the difference of its simplicity is that it can still keep simplicity as the program logic becomes more and more complex.

3.2 Google Firebase

In the era of rapid prototyping, we can get bright ideas, but sometimes they are not applicable if they take too much work. Often, the back-end is the limiting factor - many considerations never apply to server-side coding due to lack of knowledge or time. In this case, we can use Google Firebase instead.

Firebase is a Backend-as-a-Service (BaaS) which started as a YC11 startup. It grew up into a next-generation app-development platform on Google Cloud Platform. Firebase (a NoSQL JSON database) is a real-time database that allows storing a list of objects in the form of a tree. We can synchronize data between different devices.

The best thing about Firebase is that we don’t need to manage servers. We don’t need to write APIs. Firebase is your server, our API and our data store, all written so generically that we can modify it to suit most needs. Yeah, you’ll occasionally need to use other bits of the Google Cloud for your advanced applications. Firebase can’t be everything to everybody. But it gets pretty close.

Firebase lets us build more powerful, secure and scalable apps, using world-class infrastructure using:

- Cloud Firestore
- ML Kit
- Cloud Functions
- Authentication
- Hosting
- Cloud Storages
- Real-time Database

Firebase gives us insights into app performance and stability, so we can channel our resources effectively using:

- Crashlytics
- Performance Monitoring
- Test Labs

Firebase also helps us grow to millions of users, simplifying user engagement and retention using:

- In-App Messaging
- Google Analytics
- Predictions
- A/B Testing
- Cloud Messaging
- Remote Config
- Dynamic Links
- App Indexing

3.3 Implement Machine Learning model on mobile

In order to implement machine learning models on mobile applications, we need the support of a framework or a library. Some frameworks and libraries that help developers store and load machine learning models for calling inferences on mobile are introduced in this section.

3.3.1 Firebase ML Kit

Firebase ML Kit is a part of the Firebase suite that intends to give our apps the ability to support intelligent features with more ease. The SDK currently comes with a collection of predefined capabilities that are commonly required in applications - you will be able to implement these in your application regardless of whether you are familiar with machine learning or not.

Now, what Firebase ML Kit offers to us is already possible to implement ourselves using various machine - learning technologies. The thing with Firebase ML is that as well as offering these capabilities underneath a form of the wrapper, it also takes these technologies and offers their capabilities inside of a single SDK.

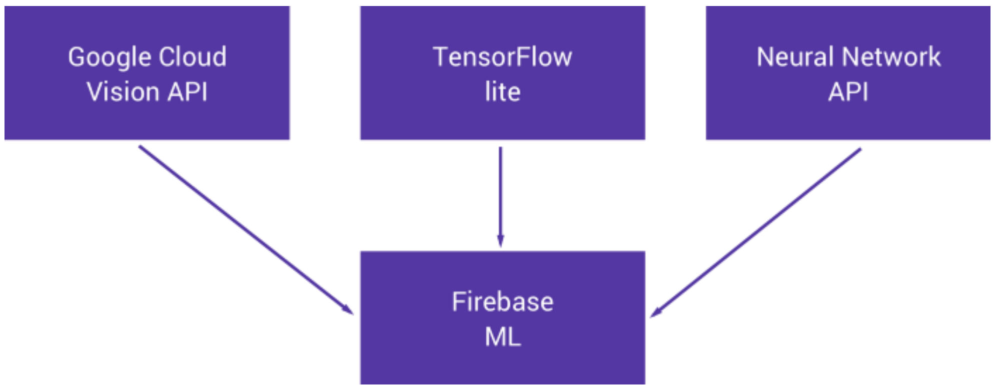


Figure 3.1 Features of Firebase ML Kit

As you can see in the Figure 3.1, the Firebase ML Kit SDK contains three main parts: Google Cloud Vision API, TensorFlow Lite, and Neural Network API. Therefore, almost the features ML Kit provides are about computer vision and provided by Google Cloud Vision API.

3.3.2 TensorFlowJs

TensorFlow.js is an open-source library you can use to define, train, and run machine learning models entirely in the browser, using JavaScript and a high-level layers API. If you are a JavaScript developer who is new to machine learning, TensorFlow.js is a great way to begin learning. Or if you are a machine learning developer who is new to JavaScript, read on to learn more about new opportunities for in-browser machine learning [4].

If you’re developing with TensorFlow.js, here are three workflows you can consider. Firstly, you can import an existing, pre-trained model for inference. If you have an existing TensorFlow or Keras model that you have previously trained offline, you can convert into TensorFlow.js format, and load it into the browser for calling inferences. After that, you can retrain an imported model. For instance, you can use transfer learning to augment an existing model trained offline using a small amount of collected data. This is one way to train an accurate model quickly, using only a small amount of data. Finally, you can also use TensorFlow.js to define, train and run a model entirely in the browser using Javascript and a high-level layers API. If you are familiar with Keras, the high-level layers API should feel familiar.

3.3.3 Freezing graph model

3.3.3.1. What is the Freezing graph model?

Freezing the model means producing a singular file containing information about the graph and checkpoint variables, but saving these hyper-parameters as constants within the graph structure. This eliminates additional information saved in the checkpoint files such as the gradients at each point, which are included so that the model can be reloaded and training continued from where you left off. As this is not needed when serving a model purely for inference they are discarded in freezing. A frozen model is a file of the photobuf file type (.pb).

3.3.3.2. Requirements for freezing

The requirements for freezing your model for inference are simple, however, you will probably need to install various other packages to actually perform inference depending on your application:

- Use Keras with a TensorFlow backend

- Although it should be working automatically, you will need to make sure that TensorBoard is working on your computer
- From the TensorFlow repository copy the freeze_graph.py python script to your working directory. Alternatively, you can use a custom designed freeze_graph function, which we will see later.

3.3.4 TensorFlow Lite

TensorFlow Lite is an open-source, product ready, cross-platform deep learning framework that converts a pre-trained model in TensorFlow to a special format that can be optimized for speed or storage. The special format model can be deployed on edge devices like mobiles using Android or iOS or Linux based embedded devices like Raspberry Pi or Microcontrollers to make the inference at the Edge.

TensorFlow Lite offers all the features required for making inferences at the Edge. Firstly, it saves the deep learning model in a light-weight format. Edge devices have limited resources in terms of storage and computation capacity while deep learning models are resource-intensive. So, the models we deploy on edge devices should be light-weight with smaller binary sizes. The second main feature of TensorFlow Lite is low latency. Deep learning models at the Edge should make faster inferences irrespective of network connectivity. As the inferences are made on the Edge device, a round trip from the device to the server will be eliminated, making inferences faster. Thirdly, the model is deployed in the Edge device, the inferences are made on the device, no data leaves the device or is shared across the network, so there is no concern for data privacy. The next feature of TensorFlow Lite is optimal power consumption. Network needs a lot of power, and Edge devices may not be connected to the network, and hence, the power consumption needed is low. Finally, models can be trained on-prem or cloud for different deep learning tasks like image classification, object detection and speech recognition and can be easily deployed to make inferences at the Edge.

In this thesis, I used TensorFlow Lite to implement the machine learning model on the application because it is simple and convenient for developers. In the case of recurrent neural network models and especially long short-term memory models, TensorFlow had detailed documentation for developers to save the model, load and call it on mobile.

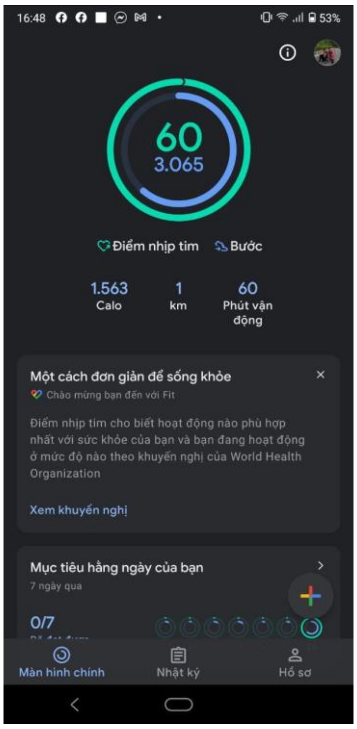
CHAPTER 4: DEVELOP HEALTHCARE APPLICATION

After explaining the background theory and the technologies and frameworks adopted in the application in the second and third chapter, this chapter will describe the process of requirements breakdown and application development. It is divided into five parts: Requirements breakdown, Application design, Application construction, Testing and Deployment.

4.1 Requirements breakdown

4.1.1 Context

Nowadays, when science and technology have been developed strongly, humans seem to spend more time and money on taking care of their health. In addition, smartphones gradually become more popular and to be one of the most important belongings. For these reasons, the ability to look after the health status with only a smartphone becomes extremely convenient and necessary for human life. For this reason, many healthcare applications were developed and published on the application store. Some popular applications about healthcare on Google Play Store are Google Fit and Home Workout.



Một cách đơn giản để sống khỏe

Chào mừng bạn đến với Fit

Điểm nhịp tim cho biết hoạt động nào phù hợp nhất với sức khỏe của bạn và bạn đang hoạt động ở mức độ nào theo khuyến nghị của World Health Organization

Mục tiêu hằng ngày của bạn

7 ngày qua

0/7

Figure 4.1 Home screen of Google Fit

Google Fit is the healthcare application developed by Google. It helps users monitor their activities in a day through some metrics such as total number of footsteps, amount of burned calories, total time and distance they performed. In addition, it also supports users connecting other wearable devices to collect other metrics such as heart rate and blood pressure. Google Fit can track the progress of many exercises and sports. After performing each exercise, the application will record it and save it to a diary so that users can easily monitor. Furthermore, users can also update their weight and blood pressure to their diary after they have had a measurement.

Another popular application in the healthcare field which has more than 100 million downloads is Home Workout. As not the same as Google Fit, it only calculates some workout metrics when users perform a particular exercise that the application supports. And of course, Home Workout has tutorials of many exercises that users can do at home. As similar as the diary of Google Fit, Home Workout provides users the workout report of each day. Besides, it also has its own alarm, so users can set up their plan for a workout.

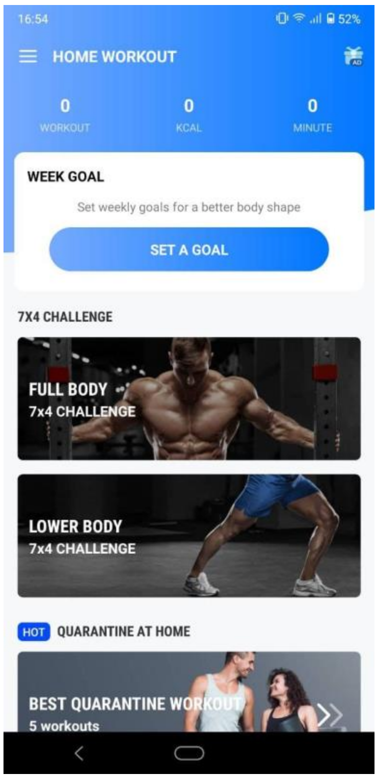


Figure 4.2 Home screen of Home Workout

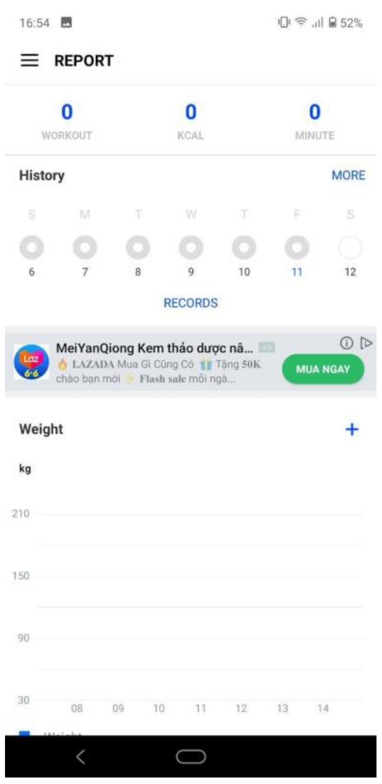


Figure 4.3 Report screen of Home Workout

Almost all popular healthcare applications on the market can help users understand their health status through some metrics such as the total footsteps they performed and the amount of calories they burned in a day. As Google Fit and Home Workout did, the feature that helps users monitor and review their activity and workout is important, so the healthcare application should provide workout reports of each day and month. With the implementation of a machine learning model in recognizing human activity, the healthcare application can also track the progress of user's activities. From that, users can realize some bad habits and have a suitable plan for changing them. This is the reason I tried to apply a machine learning model in solving human activity recognition for my healthcare application.

4.1.2 Feature overview

4.1.2.1. General use case diagram

In this system, there are two actors: User and Activity Detect Service. Users are all people who have their own smartphone and use this application to take care of their health by themselves. They need to register an account and be authenticated to use the functionalities of the healthcare application. They are allowed to monitor their activity, manage a group of users that can be considered as their family, manage their personal information and refer to some exercises that are suggested by the system. Activity Detect Service is a background service of the application. It is responsible for supervising user activity all day and reminding users if needed.

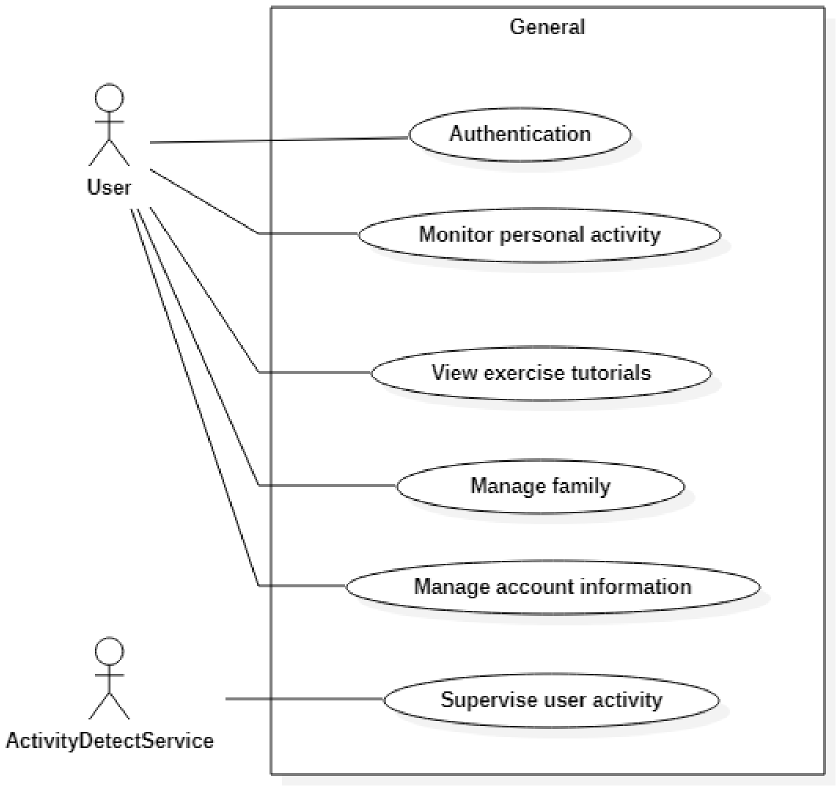


Figure 4.4 General use case diagram of application

4.1.2.2. Decomposition use case diagram

- Use case diagram for "Authentication"

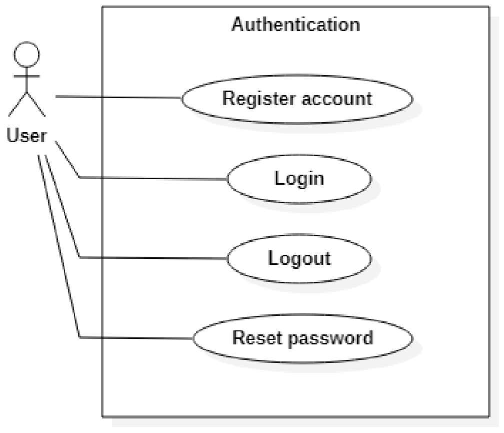


Figure 4.5 Decomposition use case diagram of "Authentication"

For authentication, users need to register their accounts with their email and some basic information. After having their own accounts, they can login to my system and use the functionalities of the healthcare application. They are also allowed to change the password of their accounts. The form for changing password will be sent to their email and they can access their email accounts and change it easily. If they do not want to use the system or another one is using their smartphone, they can logout from the system.

- Use case diagram for "Monitor personal activity"

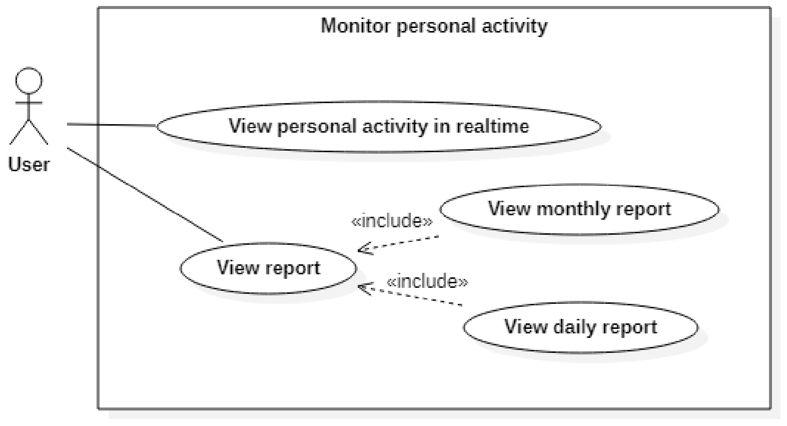


Figure 4.6 Decomposition use case diagram of "Monitor personal activity"

The use case "Monitor personal activity" is decomposed into two main use cases shown in Figure 4.6. Users can track their activities in the current day through their activity recognized in real-time, the number of footsteps they performed, the amount of calories burned and the line chart of their activities followed by time. In addition, they can get the report of their activity daily and monthly.

- Use case diagram for "Manage account information"

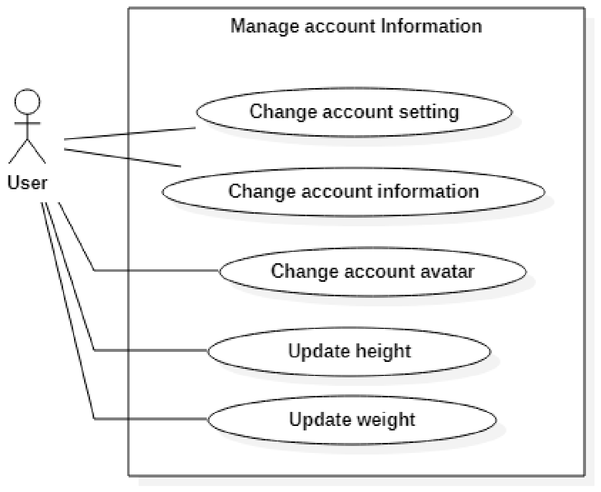


Figure 4.7 Decomposition use case diagram of "Manage account information"

The above use case diagram describes what users can do to manage their personal information. Firstly, users can modify the information of their account such as password, display name, age, and avatar. In addition, users can update their height and weight of the last time they have measured. These metrics will be stored so that users can see the change easily. Finally, users can also modify the configuration of the application that is individual for each account such as the time interval users spend on sleeping and the maximum time users can spend on sitting or standing.

- Use case diagram for "Manage family"

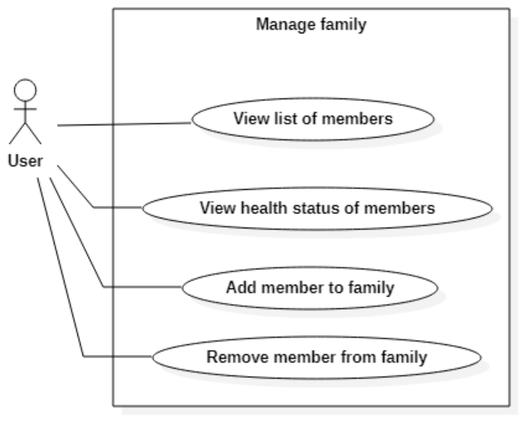


Figure 4.8 Decomposition use case diagram of "Manage family"

The use case "Manage family" is described in Figure 4.8. Users using this application have their own group that can be considered as their family. They can add other users that they want to look after into their family by using email. After that, they can follow the health status of all members in their family and receive notification when the status of anyone in their family seems to be worse. Finally, if a user does not want to take care of a family member anymore, he or she can remove it from his or her family.

- Use case diagram for "Supervise user activity"

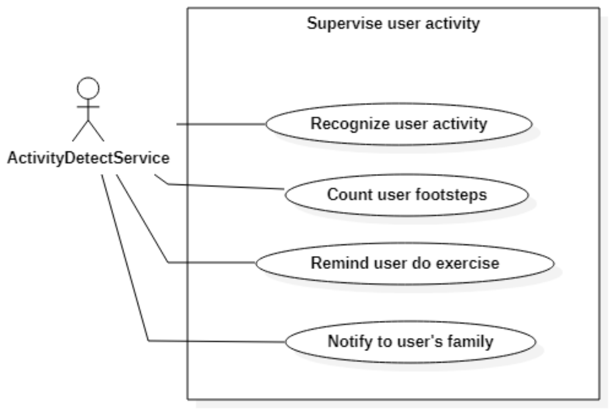


Figure 4.9 Decomposition use case diagram of "Supervise user activity"

The use case diagram shown in Figure 4.9 describes what the background service of the healthcare application can do. This service is responsible for recognizing the activity and counting the footsteps of the user in real-time, warning the user if he or she sits or stands for a long time and notifying all members in the user's family if the health status of the user seems to be worse.

4.1.3 Feature specification

The following table is the list of all use cases of the system.

General use case	**Use case code**	**Use case name**
Authentication	UC001	Register new account
	UC002	Reset account’s password
	UC003	Login
	UC004	Logout
Supervise user activity	UC005	Recognize user activity
	UC006	Count user footsteps
	UC007	Remind user do exercise
	UC008	Notify to user’s family
Monitor personal activity	UC009	View personal activity in real-time
	UC010	View daily report
	UC011	View monthly report
Manage account information	UC012	Change account setting
	UC013	Change account information
	UC014	Change account avatar
	UC015	Update height
	UC016	Update weight
Manage family	UC017	View list of members
	UC018	View health status of members
	UC019	Add member to family
	UC020	Remove member from family
	UC021	View exercise tutorials

Table 4.1 List of use cases

In the following part, there is specification of use cases of the system. Because of the limitation of the thesis, I would like to specify some main functions only.

4.1.3.3. Use case specification for "Count user footsteps"

Use case code	UC006	**Use case name**	Count user footsteps
Actor	Activity Detect Service (ADS)		
Pre-condition	User logged in to the application		
Main flow of events	**No.**	**Actor**	**Action**
	1.	User	Perform activity
	2.	ADS	Collect data from accelerometer of user's smartphone
	3.	ADS	Perform calculation and check if user performed a step
	4.	ADS	Check if user's state is walking, jogging, downstairs or upstairs
	5.	ADS	Update the number of footsteps
	6.	ADS	Calculate the distance and burned calories
	7.	ADS	Update the distance and burned calories
	8.	ADS	Update these metrics in Home screen
Alternative flow of events	**No.**	**Actor**	**Action**
	3a.	ADS	Skip if user did not perform a step
	4a.	ADS	Skip if user's state is sitting, standing, biking or unknown
Post-condition	None		

Table 4.2 Use case "Count user steps" specification

4.1.3.4. Use case specification for "Warn user if user sits or stands in long time"

Use case code	UC006	**Use case name**	Remind user do exercise
Actor	Activity Detect Service (ADS)		
Pre-condition	User logged in to the application		
Main flow of events	**No.**	**Actor**	**Action**
	1.	ADS	Recognize current state of user
	2.	ADS	Check if user is not sleeping (followed by user setting)
	3.	ADS	Check if user's state is sitting or standing
	4.	ADS	Check if current state of user does not change
	5.	ADS	Update the duration of current state
	6.	ADS	Check if the duration of current state is over maximum time for sitting and standing (followed by user setting)
	7.	ADS	Notify to user
	8.	User	Click to notification
	9.	App	Open practice screen and show a random exercise tutorial
Alternative flow of events	**No.**	**Actor**	**Action**
	2a.	ADS	Skip if user is sleeping
	3a.	ADS	Skip if user's state is not sitting or standing
	4a.	ADS	Re-assign the duration of current state to 0 if current state of user changes
	6a.	ADS	Skip if the duration of state is not

			over maximum time for sitting and standing
Post-condition	None		

Table 4.3 Use case "Remind user do exercise" specification

4.2 Application design

This section describes the design of the healthcare application from general to detail, including architecture design, use case design, user interface design and database design.

4.2.1 Architecture design

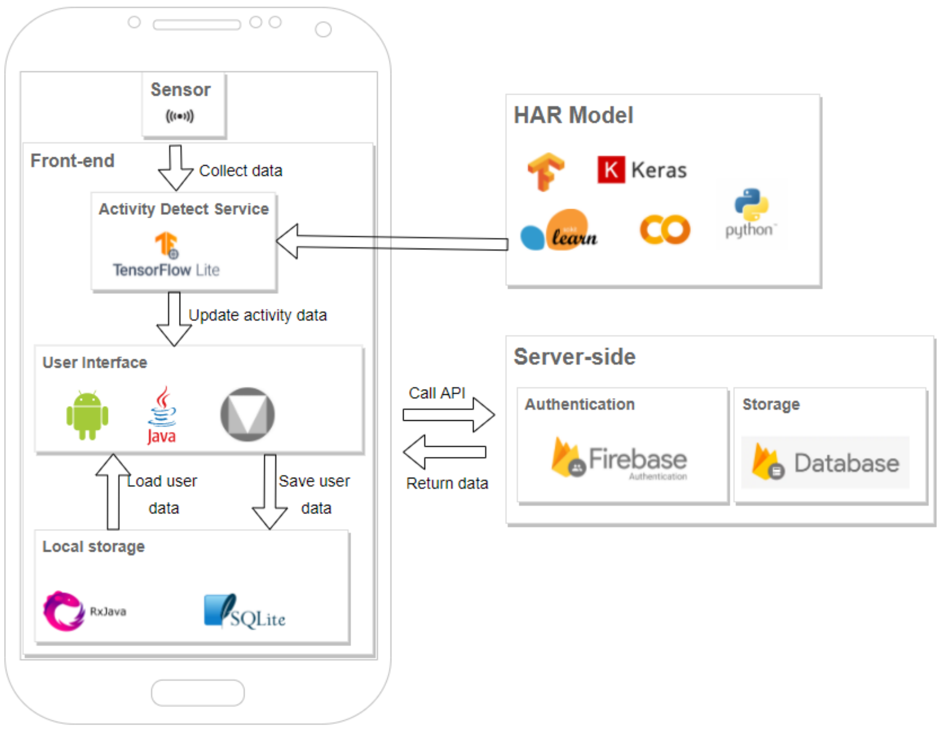


Figure 4.10 The general architecture of the application

Figure 4.10 illustrates the system architecture. It can be seen from the figure that the system is divided into three main parts, including client side, server side and human activity recognition model.

First of all, in the HAR model, I used Pandas, Seaborn, Matplotlib and NumPy for analyzing and processing training data. Because I decided to use the Long Short-Term Memory network model, a kind of deep learning model, TensorFlow and Keras are the common frameworks that support developers build and train deep learning models. The training process of the deep learning model requires the power of hardware, so I chose Google Colab and its virtual machine for storing the training data and performing the training process. After being trained, the HAR model will be saved and stored on mobile.

Because of thesis limitations, I used some services of Google Firebase instead of implementing a server for storing data and processing business logic. Google Firebase provides enough services so that I can implement a server-side quickly. In the system of the healthcare application, I used the authentication service for implementing account registration, authentication and reset account's password through email. I also used the cloud storage and database of Firebase for storing user information and images. Google Firebase is so convenient for developers to implement a back-end for a short time.

Client-side is the application interface of the system, including the interface for users to provide functions related to track activity progress, review daily and monthly reports, remind users to do exercises, and store user's data to local storage. It contains a background service that will be responsible for collecting sensor data from smartphone, recognizing user activity and then updating metrics in user interface. At the client side, the healthcare application stores and loads user's data from local storage with RxJava because it helps to implement asynchronous tasks easily and quickly. To implement the front-end for the system, I chose the native approach of Android platform because Android is open and easy to access. To build the front-end for the system, I implemented Model-View-ViewModel architecture (MVVM).

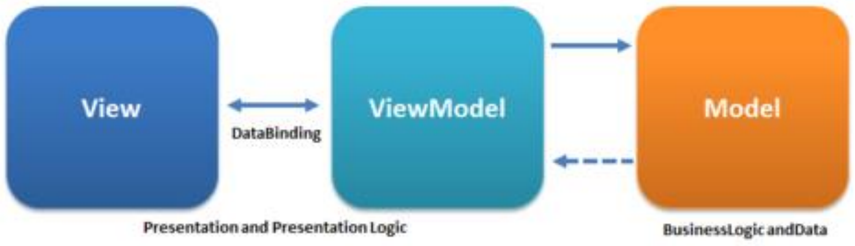


Figure 4.11 The architecture of frontend

Figure 4.11 presents the architecture for the client side of the system. Front-end consists of three parts: View, Model and View Model. Firstly, Model refers either to a domain model, which represents real state content, or to the data access layer, which represents content. View is the structure, layout and appearance of what users see on the screen. It displays a representation of the model and receives the user's interaction with the view and it forwards the handling of these to the ViewModel via the data binding that is defined to link the View and View Model. Finally, View Model is an abstraction of the view exposing public properties and commands whose properties a view directly binds to send and receive updates.

4.2.2 Detailed design

4.2.2.1. Use case design

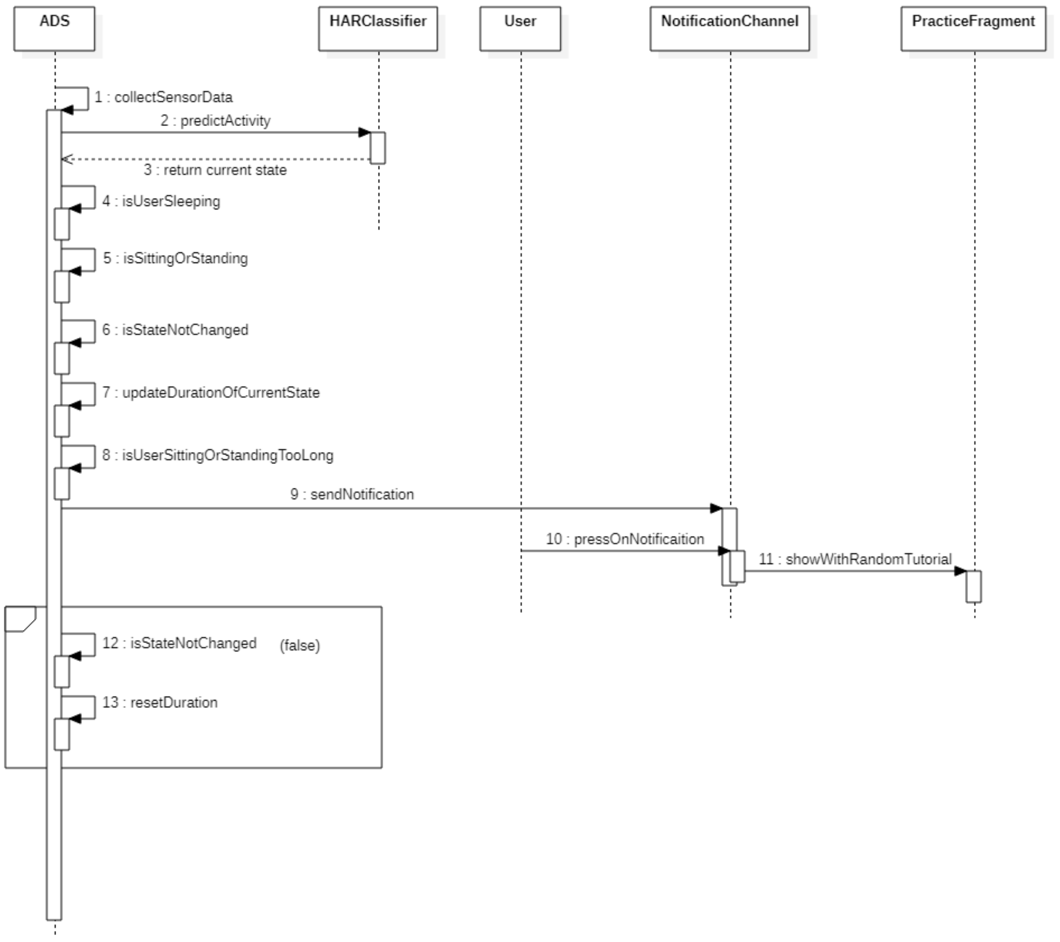


Figure 4.12 Sequence diagram of use case "Remind user do exercise"

Figure 4.12 illustrates the use case "Remind user do exercise" by a sequence diagram. Process of this use case includes three parts. Firstly, Action Detect Service (ADS) collects data from sensors of the smartphone such as accelerometer and gyroscope and predicts the current activity of the user. After that, it will check some conditions of the warning feature. If all conditions are passed, ADS will send a notification to the notification channel. When the user presses on this notification, Practice Screen will be opened with a random exercise tutorial.

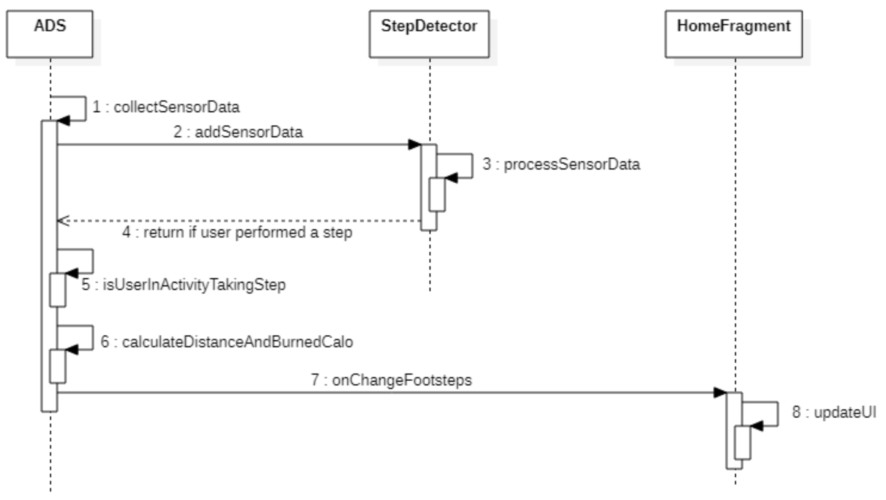


Figure 4.13 Sequence diagram of use case "Count user steps"

Figure 4.13 illustrates the use case "Count user step" by a sequence diagram. It can be easily seen that the process of this use case contains three parts. Firstly, the Action Detect Service (ADS) collects the data from the accelerometer sensor and sends it to Step Detector. After that, Step Detector and ADS will check if the user performed a step. Finally, the number of performed footsteps will be updated and ADS will calculate some metrics from it like total distance and burned calories and send them to Home Fragment to update on screen.

4.2.2.2. Package design

Because I used some services of Google Firebase to implement the server side of the application, I only needed to concern the package design of source code of the client side.

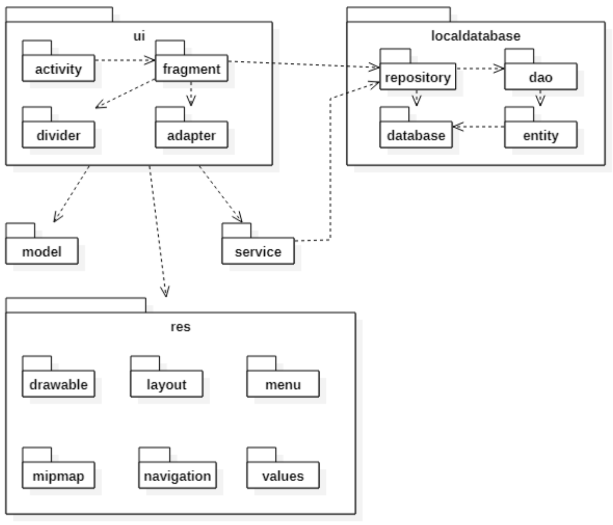


Figure 4.14 Package design for frontend

The Figure 4.14 demonstrates the package design of front-end part. There are five main packages, including ui, res, service, localdatabase and model. Firstly, the "ui" package is responsible for handling views and events of the application. It consists of activities that are considered as the screens of the application, fragments of screen and some special elements of view such as adapter and divider. This package renders the layouts that are designed with XML and stored in "res" package. The "service" package contains services that the application runs. These services may run on foreground or background of the smartphone and do some tasks like recognize user activity and recognize performed footsteps. The rest parts of system are responsible for handling data on local database and Firebase database.

4.2.2.3. User interface design

The user interface of the application is designed and built based on Material Design, a design system created by Google to help teams build high quality digital experiences for Android, iOS, Flutter, and the web. The healthcare application user interface contains 6 parts: (i) authentication screens, (ii) home screen, (iii) report screen, (iv) practice screen, (v) family screen, and (vi) setting screen.

- Authentication screens:

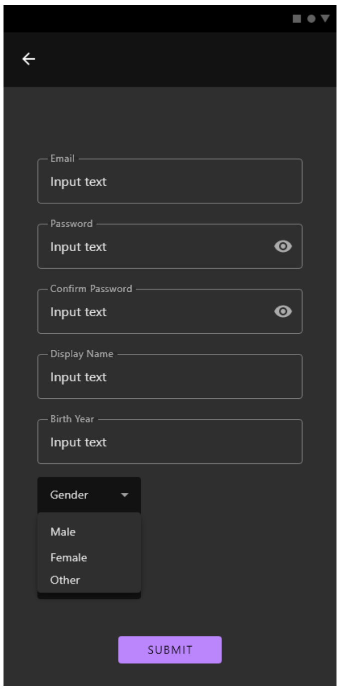


Figure 4.15 Design of sign up screen

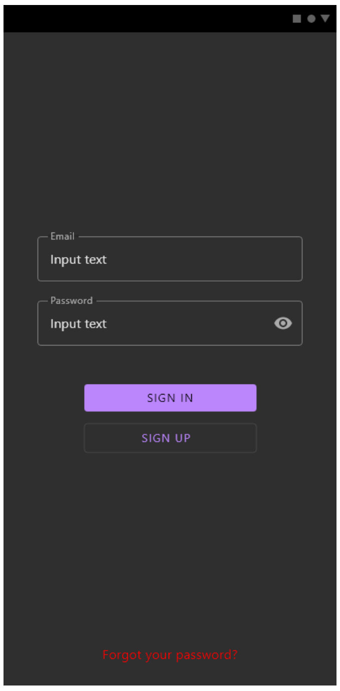


Figure 4.16 Design of sign in screen

For authentication, a user of the healthcare application needs to register an account with his or her email and some basic information such as the name displayed in application, gender and birth year. After having his or her own account, the user can login to the application through email and password. In addition, if a user forgot the password of the account, he or she can press on the text "Forgot your password" and reset the password through the message sent to his or her email.

- Home screen:

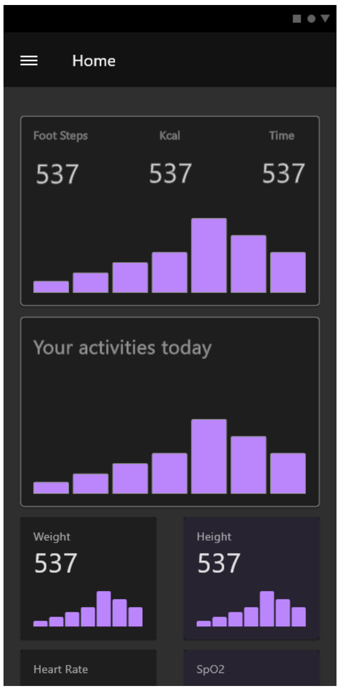


Figure 4.17 Design of home screen

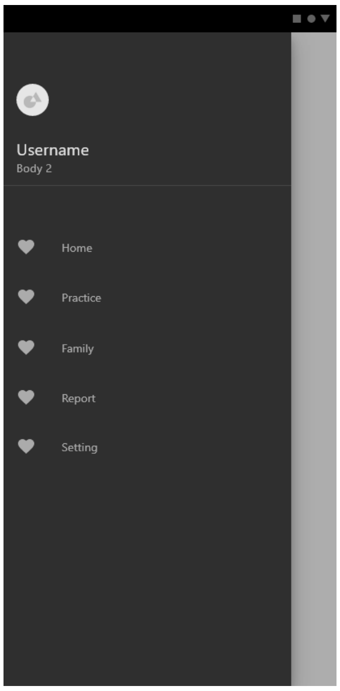


Figure 4.18 Design of application navigation

The Figure 4.17 and 4.18 describe the main screen and the navigation of the application. In Home Screen, the main screen of the healthcare application, there are many metrics such as the total footsteps, the total burned calories, the chart of progress of a user's activities in a day and the recent measured weight and height. Each of the information may be combined or stand alone in a card view so that users can see easily and clearly. With the height and weight information, users can update the last measurement result by pressing on the card view containing them. The Home Screen is also the first screen of the navigation and the rest screens follow it. The navigation also has some information about the user's account like the avatar and the display name.

- Report screen:

When I was designing the Report Screen, I had no idea about how the charts of the user's metric display. However, all the metrics will be displayed in the form of a chart on Report Screen. They also need to be separated into some card views for clear purposes.

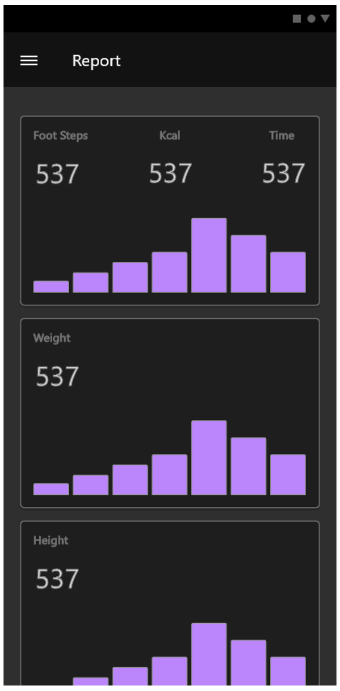


Figure 4.19 Design of report screen

- Practice screen:

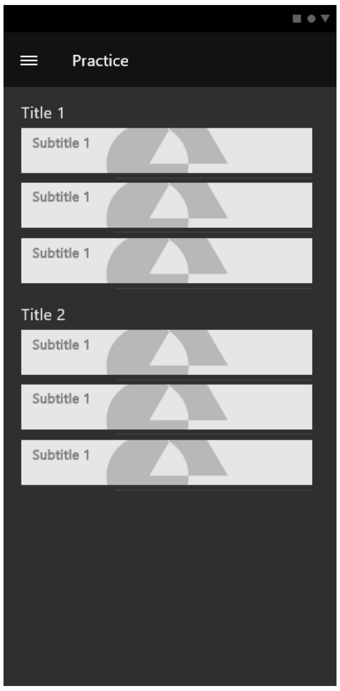


Figure 4.20 Design of practice screen

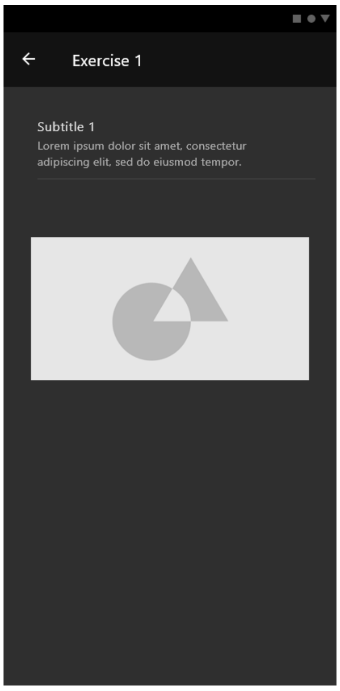

Exercise 1
Subtitle 1
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor.

Figure 4.21 Design of exercise tutorial screen

As shown in Figure 4.20, Practice Screen displays a list of exercise tutorials. When pressing on an item of list, the detail of the exercise tutorial will be displayed.

- Family screen:

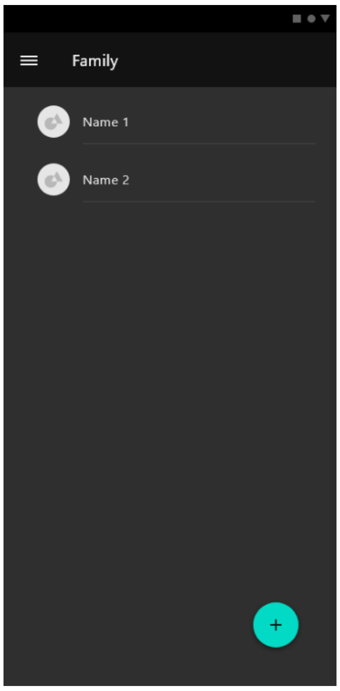


Figure 4.22 Design of family screen

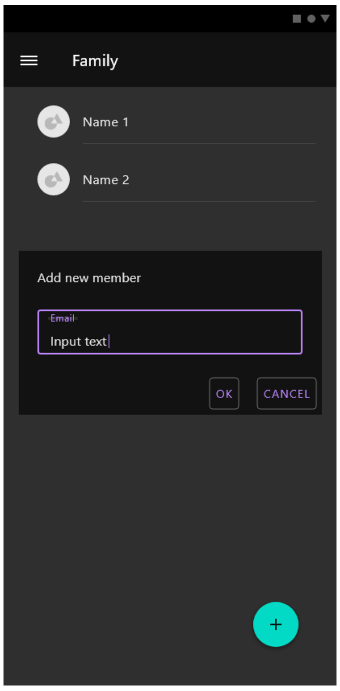


Figure 4.23 Design of family screen with add member popup

As simple as Practice Screen, Family Screen also displays a list of members of the user's family who the user wants to take care of and receive notification when their health status seems to be worse. Users can add more members by pressing the add button at the bottom left of the screen.

- Setting screen:

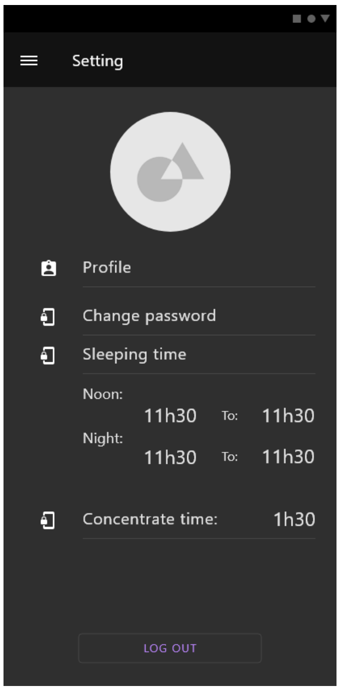


Figure 4.24 Design of setting screen

Finally, in Setting Screen, users can change some information about their account such as password, avatar, and user's profile. Besides, users can also set their own configuration for sleeping time and concentrate time by pressing on the time texts on Setting Screen.

4.2.2.4. Database design

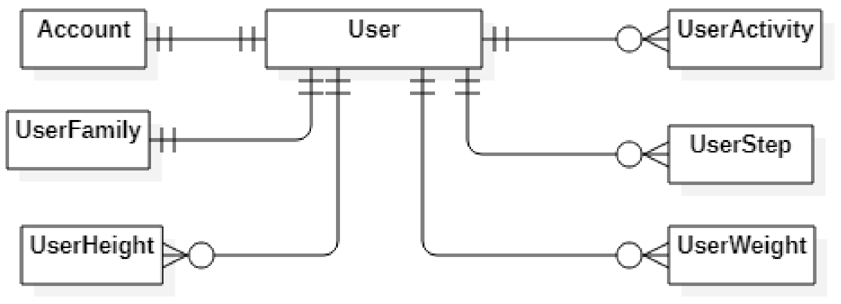


Figure 4.25 Database design of application

It can be seen from Figure 4.25 that the system has seven entities. Account entity is stored in the authentication service of Google Firebase. Each account is corresponding to a user. User and user family will be stored in the database of Firebase, so they are stored in NoSQL. User activity, user step, user weight and user height will be saved in the local database of the smartphone, so they are stored in SQL. Each user will have many members in his or her family, so this connection will be stored in the "user family" table. The metrics of user's activity each day will be stored in user activity and user step and stored local for private purposes. The details of each entity will be shown in Table 4.4.

Table 4.4 Detail database design

Entity	**Field**	**Type**	**Description**
Account	id	String	The id of user
	email	String	Email of user
	password	String	Password of the account
User	id	String	The primary key for document and user
	email	String	The email of user
	display_name	String	The name displayed in application
	gender	String	Male, Female or Other

	birth_year	Integer	The year user was born
	avatar	String	URL of the avatar of user
UserFamily	uid	String	The id of user
	member_ids	List of String	The id of member in user’s family
UserActivity	uid	String	The id of user
	timestamp	Long	The start time of activity in millisecond
	activity	Integer	The id of activity
	duration	Long	The duration of activity in millisecond
UserStep	uid	String	The id of user
	timestamp	Long	The start day time in millisecond
	amount_of_step	Integer	Total footsteps user performed in day
	walking_step	Integer	Number of walking steps
	jogging_step	Integer	Number of jogging steps
	downstairs_step	Integer	Number of downstairs steps
	upstairs_step	Integer	Number of upstairs steps
	total_burned_calos	Float	Total burned calories
	distance	Float	Total distance follows by footsteps
UserWeight	uid	String	The id of user
	timestamp	Long	The start day time in millisecond
	weight	Float	Weight of user
UserHeight	uid	String	The id of user

	timestamp	Long	The start day time in millisecond
	height	Float	Height of user

4.3 Application construction

4.3.1 Used tools and libraries

The following table contains all libraries, frameworks and tools that I used to finish this project.

Table 4.5 List of used tools and libraries

Purpose	**Library / Tool**	**URL**
Code editor	Android Studio	https://developer.android.com/studio
Database, Authentication, Storage	Google Firebase	https://firebase.google.com/
UI library	Material Design	https://material.io/
Android circle image view	hdodenhof/ CircleImageView	https://github.com/hdodenhof/CircleImageView
Load image	Glide	https://bumptech.github.io/glide/
Show chart	MPAndroidChart	https://weeklycoding.com/mpandroidchart/
Event communication	EventBus	https://github.com/greenrobot/EventBus

4.3.2 Main screens and features

In this section, I describe some main screens of the healthcare application and some important functions in them.

4.3.2.1. Home screen

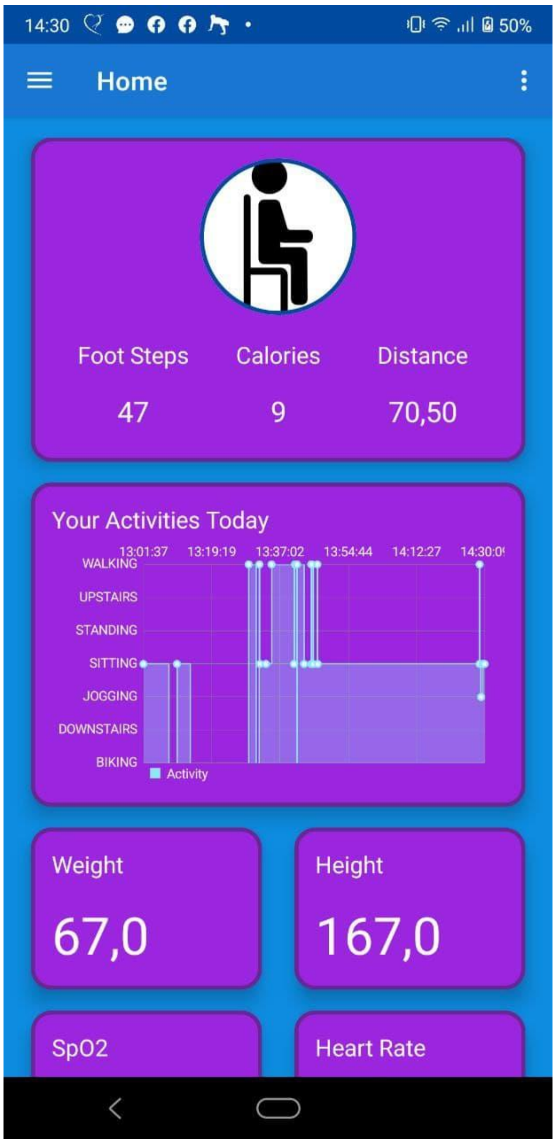


Figure 4.26 Home screen of demo application

The Figure 4.26 describes the Home screen, the main screen of the healthcare application. This screen displays all the information about the activities of the user in the current day. They include the current activity of the user, the number of footsteps the user performed, the total amount of calories burned and the total distance followed by the user's footsteps in a day. They are updated in real-time by the Activity Detect Service that works in the background of the application. With these metrics, users can easily follow the current status and will have a suitable plan for doing exercises or relaxing. In addition, users can also track the progress of their activities each day. From this chart, users can realize some bad habits and have a particular plan for making their life healthier. For instance, by tracking with this chart, users realize that they spend a very long time on sitting and working. This habit is not good for their health, so they can change a little bit, like to spend a little time on drinking water after about one working hour. Besides, users can update their height and weight after they make a measurement. They will not need to remember them.

4.3.2.2. Report screen

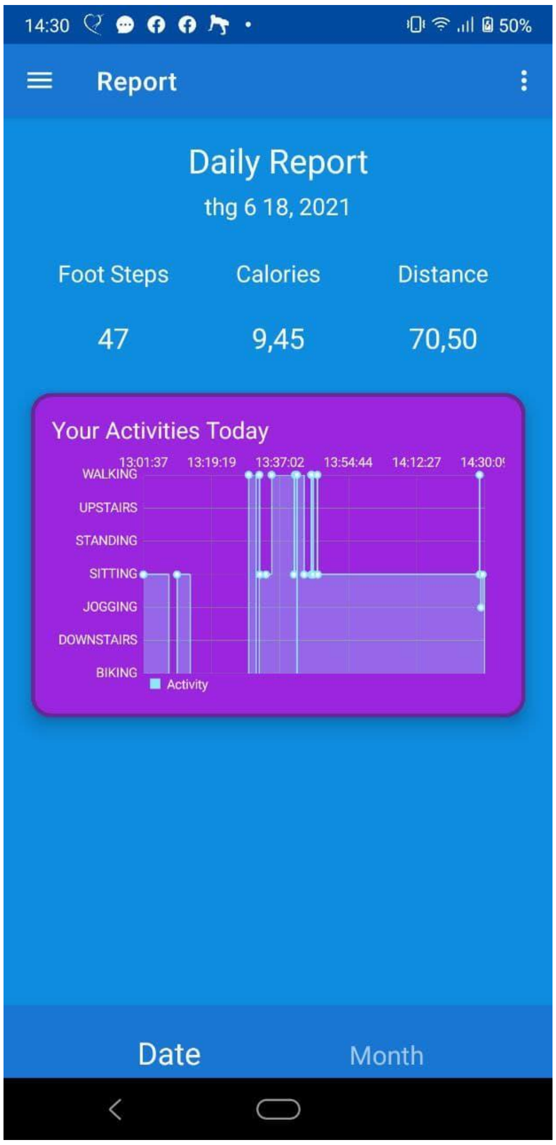
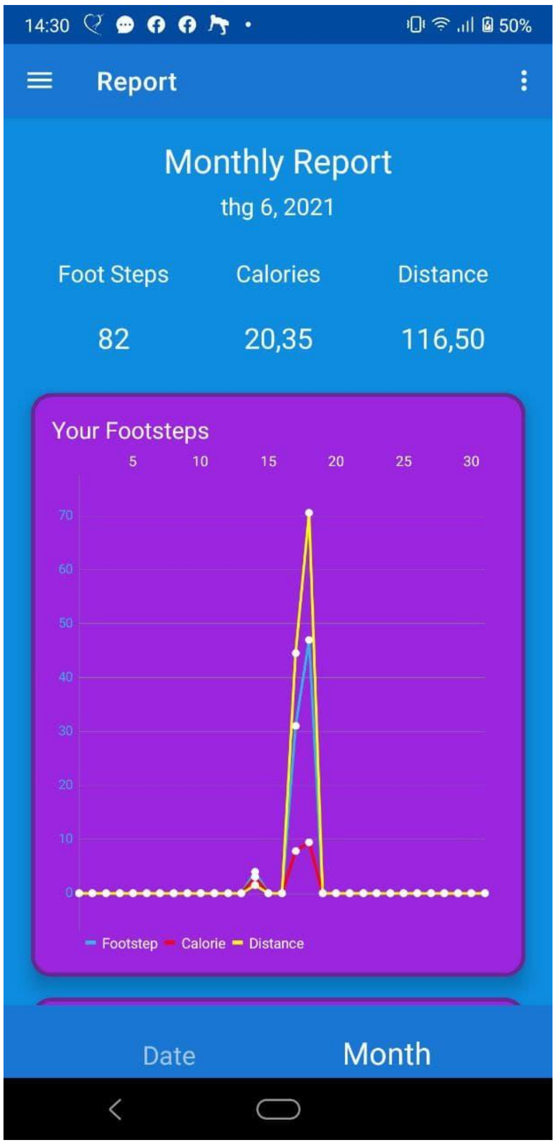


Figure 4.27 Daily report screen of demo application

If users want to review their activity data of previous days, they can navigate the application to the report screen and choose tab “Date”. After that, they need to choose the date by pressing on the time text under title “Daily Report”. After users submit the date they want, they can review the total number of footsteps they performed, the total amount of burned calories, the total distance followed by footsteps and the chart of progress of their activities on that day.

If the daily report is too detailed and users want to have an overview of their activities, they can choose the tab “Month” in the report screen. All the general information that is summarized from all days in a month is displayed in this tab. Users also need to choose the month they want by pressing the time text under title "Monthly Report". After submitting the chosen month, all the metrics will be shown like the following figures.



Monthly Report
thg 6, 2021

Foot Steps	Calories	Distance
82	20,35	116,50

Figure 4.28 Monthly report screen with line chart

Firstly, the monthly report also displays the total footsteps, burned calories and distance that users performed in the whole month. In addition, they can see the change of these metrics day by day in month through the line chart shown in Figure 4.28.

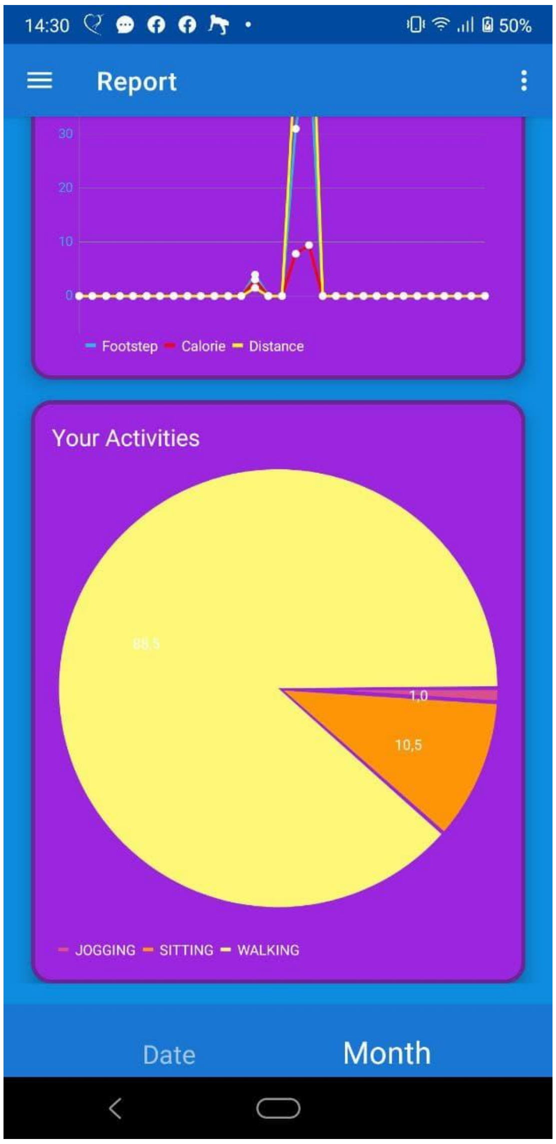
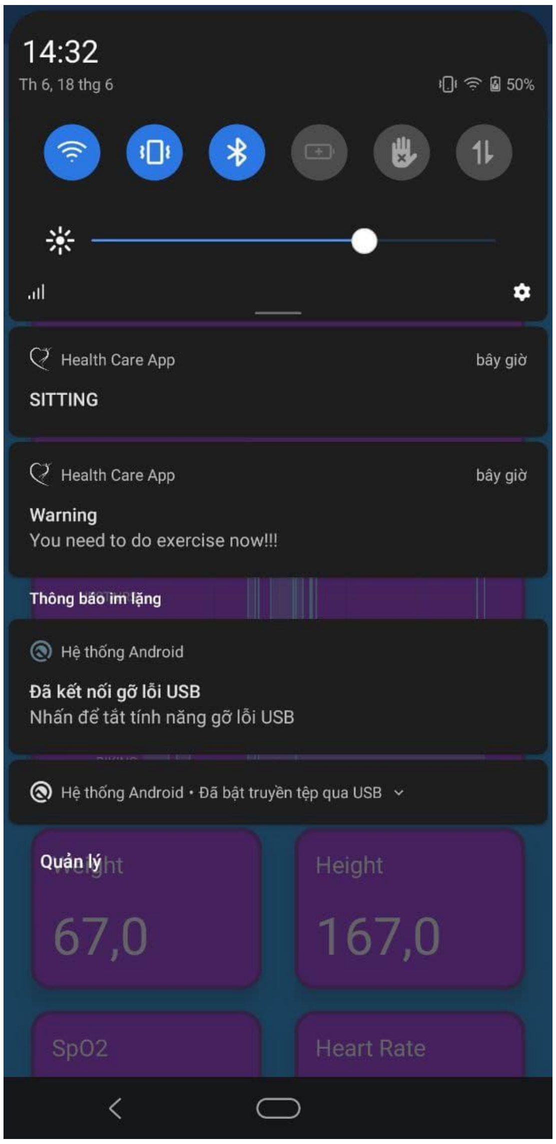


Figure 4.29 Monthly report screen with pie chart

Finally, the monthly report displays the pie chart of all activities of users. Users can see the percentage of each activity they performed in a month, so they will have a suitable plan for health balance.

4.3.2.3. Receive notification when user sat or stood for a long time


14:32
Th 6, 18 thg 6
50%

Health Care App bây giờ
SITTING

Health Care App bây giờ
Warning
You need to do exercise now!!!

Thông báo im lặng

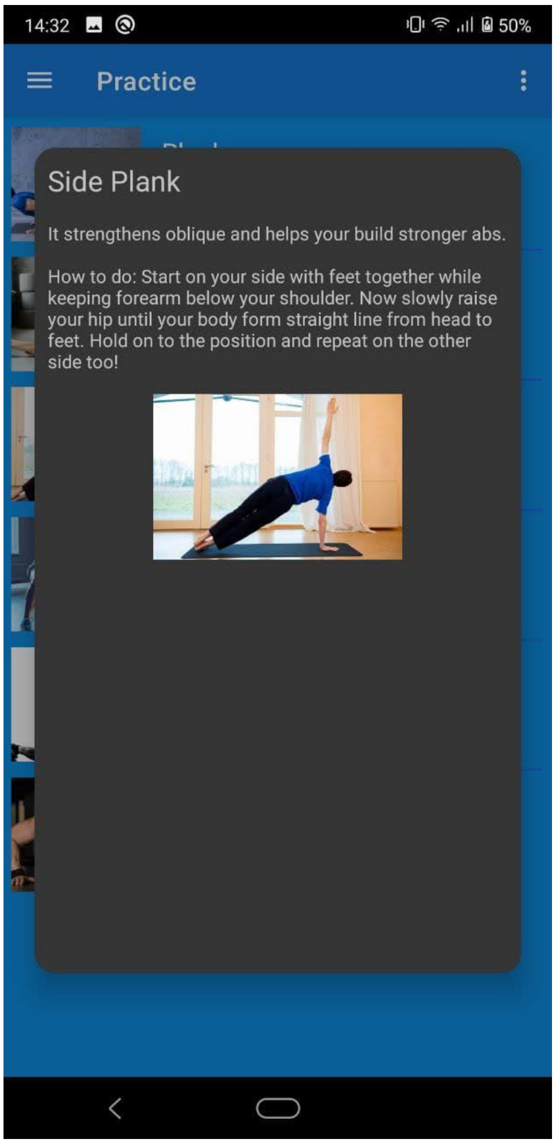
Hệ thống Android
Đã kết nối gỡ lỗi USB
Nhấn để tắt tính năng gỡ lỗi USB

Hệ thống Android • Đã bật truyền tệp qua USB

Quản lý
Weight 67,0
Height 167,0
SpO2
Heart Rate

Figure 4.30 Screen when receiving notification

If users sit or stand for a long time, it is not good for their health. It may cause some diseases like Hemorrhoids and Backache. For this reason, this healthcare application will send a notification to users to remind them to do exercises for a minute.


14:32

Practice

Side Plank

It strengthens oblique and helps your build stronger abs.

How to do: Start on your side with feet together while keeping forearm below your shoulder. Now slowly raise your hip until your body form straight line from head to feet. Hold on to the position and repeat on the other side too!

Figure 4.31 Application screen after pressing on notification

After pressing on the notification, the healthcare application will be opened and navigated to the practice screen. At here, the application will show a random exercise tutorial so that users can follow.

4.4 Testing

4.4.1 Test for "Remind user do exercise"

Pre-condition: user logged in to the application.

Table 4.6 List of test cases of main features

Test case	**Steps**	**Expected result**	**Result**
Successfully warn user when user sat	Put the smartphone on pocket and stand (not in	Current state is STANDING	Pass

or stood for a long time	sleeping time)		
	Stand for a long time (over the maximum sitting or standing configured in application setting)	Receive a notification	Pass
	Press on notification	Open practice screen and show a random exercise tutorial	Pass
Not warn in sleeping time	Put the smartphone on pocket and stand (in sleeping time)	Current state is STANDING	Pass
	Stand for a long time (over the maximum sitting or standing configured in application setting)	Not receive a notification	Pass
Not warn if while user is sitting or standing when user do another activity	Put the smartphone on pocket and stand (not in sleeping time)	Current state is STANDING	Pass
	Stand for a short time (not over the maximum sitting or standing configured in application setting)	Not receive a notification	Pass
	Do another activity	Change state	Pass
	Stand still the total time is over the maximum sitting or standing configured in application setting	Not receive a notification	Pass

4.5 Deployment

The application was built for testing only because it has not been completed. However, it can be installed through this APK: https://bom.to/demo_apk.

CHAPTER 5: MAIN CONTRIBUTION

After introducing the development of the healthcare application, including the survey of mobile applications in healthcare fields, requirements breakdown, designing and building the application, Chapter 5 will present outstanding solutions and contributions of my thesis.

5.1 Build and train a model for recognizing human activity and run it on mobile

Firstly, I will sum up the approach, training dataset, data preprocessing, building and training of the long short-term memory model that I used to build the application. The model is just a basic LSTM model because of the limitations of the thesis.

5.1.1 Approach method

The model follows the approach that human activity is recognized based on that data of wearable sensors. Because the format of data of wearable sensors while a person performs an activity is in time series, Long Short-Term Memory network, a recurrent neural network architecture, is a well-choice for human activity classification. Unlike traditional RNNs, an LSTM network is well-suited to learn from experience to classify, process and predict time series when there are very long time lags of unknown size between important events [5].

5.1.2 Dataset

The dataset used to train and validate the model is "Sensors activity dataset", which is published by Shoaib et al on "Pervasive systems research data sets" topic of University of Twente [6]. It contains 7 physical activities, including Biking, Downstairs, Jogging, Sitting, Standing, Upstairs and Walking. They are the basic motions of human activities in daily life.

This dataset is the performances of 10 participants in each of these activities for 3 to 4 minutes. All the participants are male from 25 to 30 years old. The experiments were carried out indoors in one of the university buildings, except biking. For walking, and jogging, the department corridor was used. For walking upstairs and downstairs, a 5-floor building with stairs was used. Each person was equipped with 5 smartphones in 5 positions, including right and left jean's pocket, belt position towards the right leg, right upper arm and right wrist. The first three positions are commonly used by people carrying smartphones. The fourth position is usually used when activities like jogging are performed.

In these experiments, the orientation of the smartphones was portrait for the upper arm, wrist, and two pockets, and landscape for the belt position. The data was recorded for all five positions at the same time for each activity and it was collected at a rate of 50 samples per second. This sampling rate (50 samples per second) is enough to recognize human physical activities. Data was collected for an accelerometer, a gyroscope, a magnetometer and a linear acceleration sensor in three directions.

5.1.3 Implementation

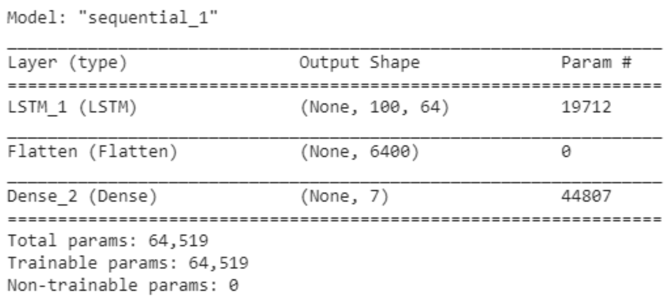
5.1.3.1. Data preprocessing

Firstly, because the concerned problem is to recognize human activity from data sensors of a smartphone, the sensor data at position of left and right pockets seems to be the most suitable. These positions are the most common positions that people usually put their smartphone in when they perform some basic daily activity like walking, jogging, walking upstairs, walking downstairs, sitting and standing. So, the data of sensors at these positions needs to be extracted from the origin dataset. In addition, the data of the magnetometer seems to not have a significant impact on this problem, so it is also skipped. The magnitude of the vector of each sensor is added to the training dataset as a feature. Secondly, since the sensor data is in time series, a common technique used to generate sequences of data is sliding window. With the sampling rate of 50 samples per second, the window will have length of 100 and step size of 50. Finally, I reshaped the training dataset and it would be ready to use.

5.1.3.2. Build and train model

- Model structure

The model contains 3 layers. The first layer is a long short-term memory layer with input shape is a 2D matrix. Each input data is the collection of 12 features in 100 time steps, including the data of accelerometer, linear acceleration and gyroscope in 3 axes and the magnitude of the vector of each sensor. The second layer is a Flatten layer in order to flat the output of the LSTM layer. The rest layer is a Dense layer with activation function of SoftMax so that the output of the model would be the probability of activities. Both the LSTM and Dense layers are regularized with L2 function for all parameters in order to reduce training time. The optimizer used to train the model is Adam, the optimizer that is usually used to train machine learning models.



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
LSTM_1 (LSTM)                (None, 100, 64)           19712     
_________________________________________________________________
Flatten (Flatten)            (None, 6400)              0         
_________________________________________________________________
Dense_2 (Dense)              (None, 7)                 44807     
=================================================================
Total params: 64,519
Trainable params: 64,519
Non-trainable params: 0
```

Figure 5.1 Model summarization

- Parameter selection

I performed training model with some lists of parameter. The Figure 5.2 and 5.3 describe the impact of batch size and number of hidden layer to the accuracy of model. They may not have too more influence on accuracy, so I chose the batch size of 1024 and the number of hidden layer of 64.

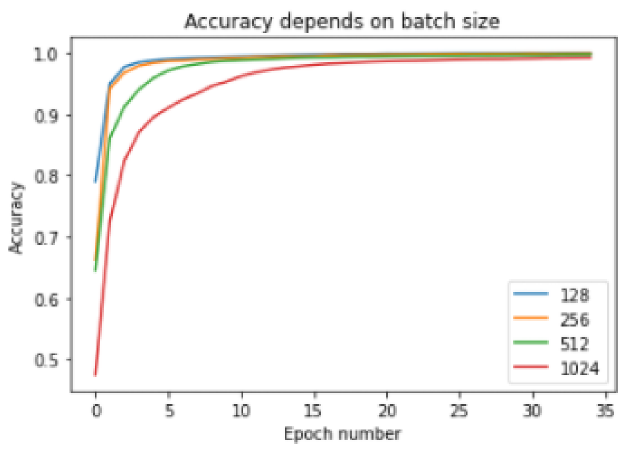


Figure 5.2 Accuracy depends on batch size

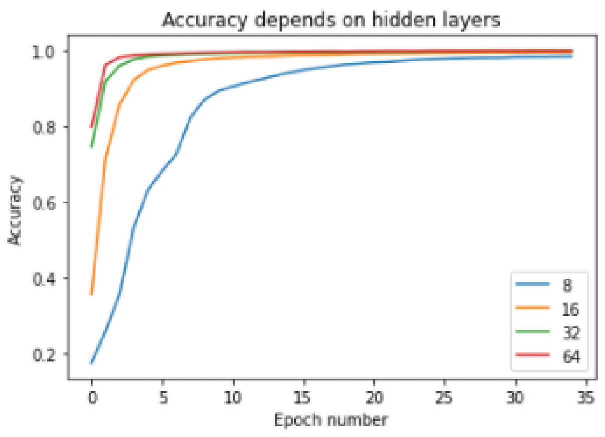


Figure 5.3 Accuracy depends on hidden layer

The Figure 5.4 shows the influence of learning rate on accuracy of model. It is easily to see that the learning rate of Adam optimizer should be small enough. So I chose the learning rate of 0.0001.

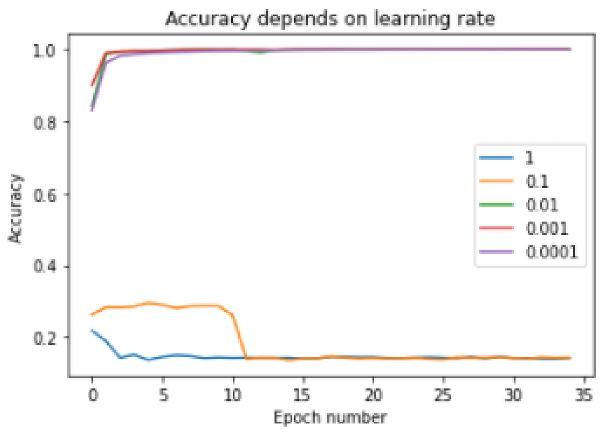
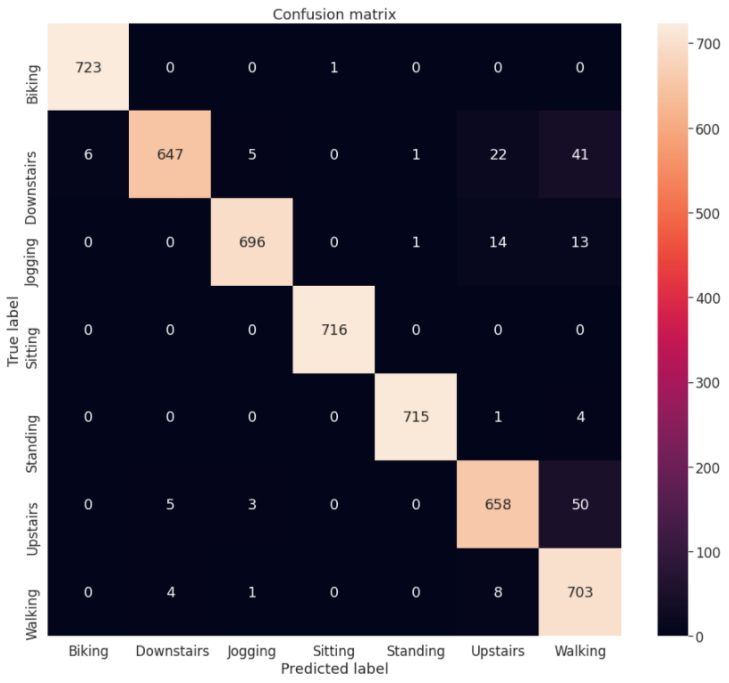


Figure 5.4 Accuracy depends on learning rate

I also chose a small number of epoch in order to reduce the impact of overfitting problem. In my thesis, I only trained model with 10 epochs.

- Test trained model with test dataset

The confusion matrix when testing the trained model with test dataset is displayed in Figure 5.5.



	Biking	Downstairs	Jogging	Sitting	Standing	Upstairs	Walking
Biking	723	0	0	1	0	0	0
Downstairs	6	647	5	0	1	22	41
Jogging	0	0	696	0	1	14	13
Sitting	0	0	0	716	0	0	0
Standing	0	0	0	0	715	1	4
Upstairs	0	5	3	0	0	658	50
Walking	0	4	1	0	0	8	703

Figure 5.5 Confusion matrix when testing model with test dataset

From the confusion matrix, I can realize that the performance of the trained model in the test dataset is quite high. However, this model sometimes predicts the activity is walking when the true activity is walking downstairs or upstairs.

5.1.3.3. Model evaluation

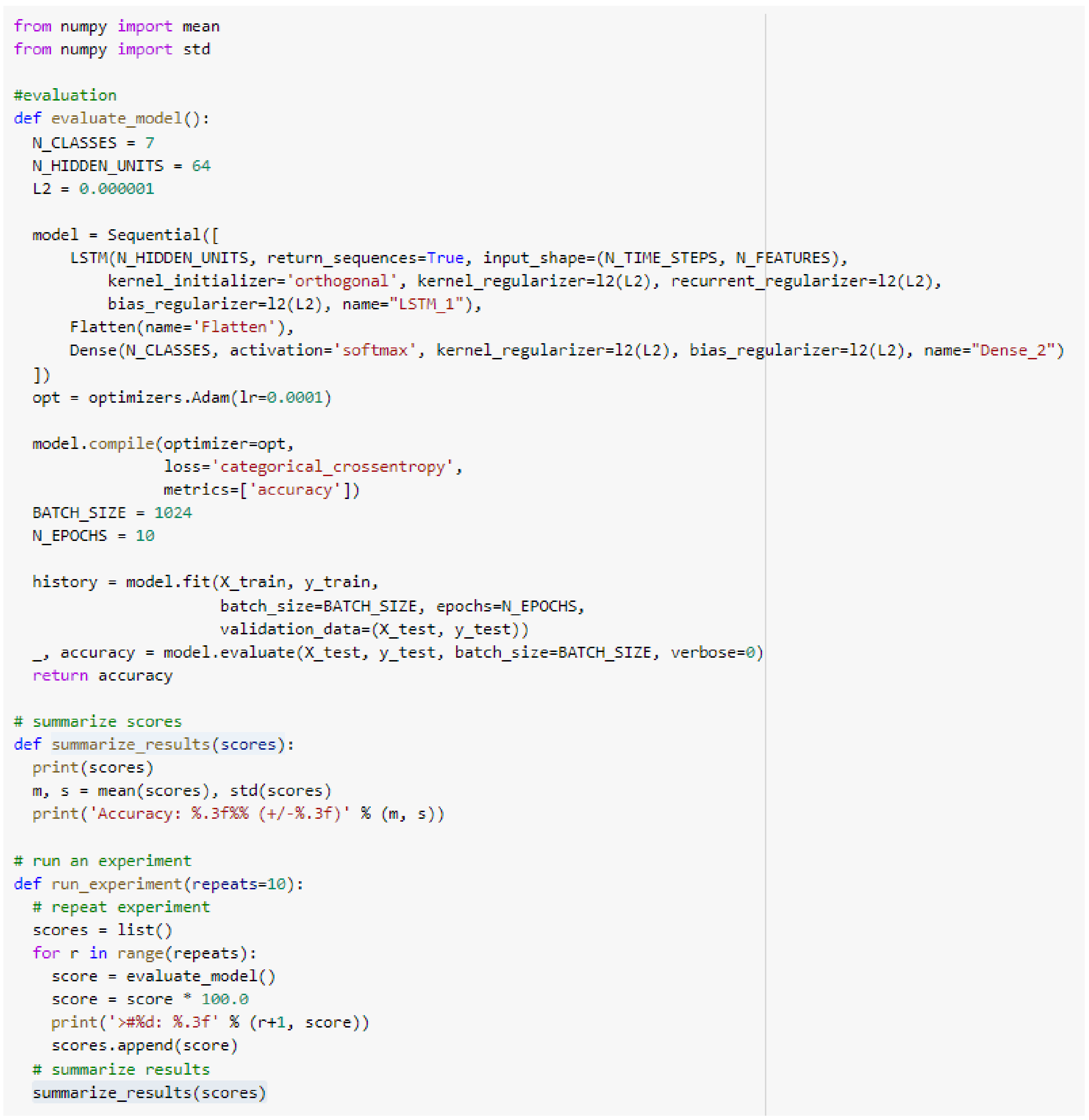
Human activity recognition is a classification problem because the output domain of this problem contains discrete values. In a classification problem, metrics that are important and used to evaluate the performance of machine learning models are accuracy, precision and recall. In my thesis, the considered problem is to recognize human activity with output domain is a set of basic daily activities, so the accuracy of the model is more important than precision or recall. Especially, with the application that recognizes and records the activity of the user in real-time, the more accurate the model is, the more the chart of activity of the user is.

The accuracy of a machine learning model is calculated with the following equation:

$$accuracy = \frac{number\ of\ true\ predictions}{total\ number\ of\ predictions}$$

Equation 5.1 Accuracy of a machine learning model

However, a deep learning model cannot be judged from a single evaluation. The reason for this is that neural networks are stochastic, meaning that a different specific model will result when training the same model configuration on the same data. This is a feature of the network in that it gives the model its adaptive ability, but requires a slightly more complicated evaluation of the model. Therefore, in order to evaluate my LSTM model, I repeated the evaluation of the model multiple times and then summarized the performance of the model across each of those runs.



```
from numpy import mean
from numpy import std

#evaluation
def evaluate_model():
  N_CLASSES = 7
  N_HIDDEN_UNITS = 64
  L2 = 0.000001

  model = Sequential([
      LSTM(N_HIDDEN_UNITS, return_sequences=True, input_shape=(N_TIME_STEPS, N_FEATURES),
          kernel_initializer='orthogonal', kernel_regularizer=l2(L2), recurrent_regularizer=l2(L2),
          bias_regularizer=l2(L2), name="LSTM_1"),
      Flatten(name='Flatten'),
      Dense(N_CLASSES, activation='softmax', kernel_regularizer=l2(L2), bias_regularizer=l2(L2), name="Dense_2")
  ])
  opt = optimizers.Adam(lr=0.0001)

  model.compile(optimizer=opt,
                loss='categorical_crossentropy',
                metrics=['accuracy'])
  BATCH_SIZE = 1024
  N_EPOCHS = 10

  history = model.fit(X_train, y_train,
                      batch_size=BATCH_SIZE, epochs=N_EPOCHS,
                      validation_data=(X_test, y_test))
  _, accuracy = model.evaluate(X_test, y_test, batch_size=BATCH_SIZE, verbose=0)
  return accuracy

# summarize scores
def summarize_results(scores):
  print(scores)
  m, s = mean(scores), std(scores)
  print('Accuracy: %.3f%% (+/-%.3f)' % (m, s))

# run an experiment
def run_experiment(repeats=10):
  # repeat experiment
  scores = list()
  for r in range(repeats):
    score = evaluate_model()
    score = score * 100.0
    print('>#%d: %.3f' % (r+1, score))
    scores.append(score)
  # summarize results
  summarize_results(scores)
```

Figure 5.6 Setup for evaluating model

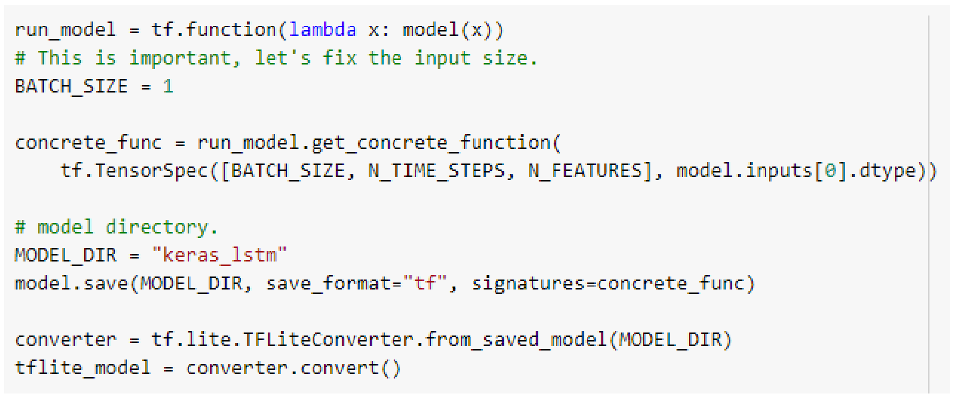
Figure 5.6 illustrates how I implemented evaluation of the model. I built and trained the model a total of 10 times and recorded the accuracy of the model with the test dataset at each time. Finally, I calculated the mean and standard deviation of the performance. The mean gives the average accuracy of the model on the dataset, whereas the standard deviation gives the average variance of the accuracy from the mean. Because the results seem to be too small if I caption it from Google Colab. Hence, I decided to write them in the following lines.

Scores = [95.57363986968994, 95.43469548225403, 92.1992838382721, 94.10480260848999, 98.11432957649231, 92.59626865386963, 93.96585822105408, 91.86185002326965, 94.44223642349243, 93.62842440605164]

Accuracy: 94.192% (+/-1.767)

It can be seen that the model performed well, achieving a classification accuracy of about 94.192% trained on the raw dataset, with a standard deviation of about 1.767.

5.1.3.4. Save model

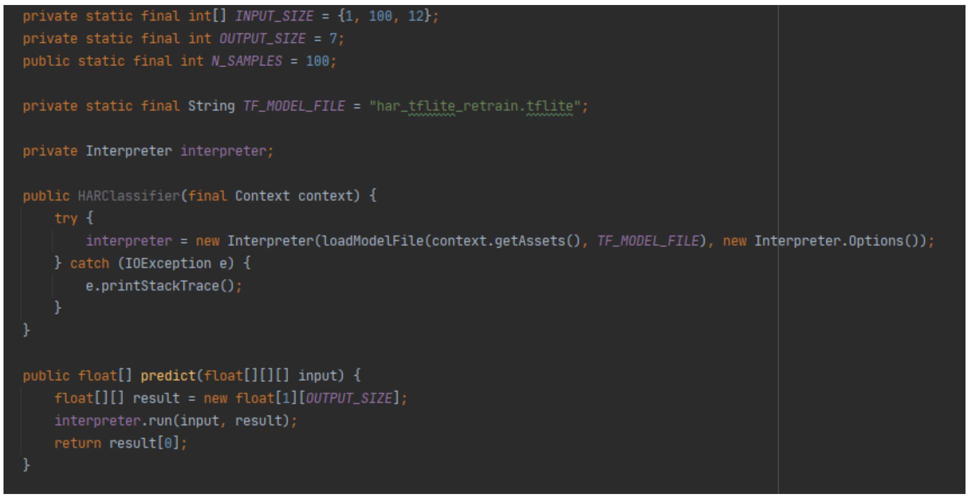


```python
run_model = tf.function(lambda x: model(x))
# This is important, let's fix the input size.
BATCH_SIZE = 1

concrete_func = run_model.get_concrete_function(
    tf.TensorSpec([BATCH_SIZE, N_TIME_STEPS, N_FEATURES], model.inputs[0].dtype))

# model directory.
MODEL_DIR = "keras_lstm"
model.save(MODEL_DIR, save_format="tf", signatures=concrete_func)

converter = tf.lite.TFLiteConverter.from_saved_model(MODEL_DIR)
tflite_model = converter.convert()
```

Figure 5.7 Save model illustration

Figure 5.7 illustrates the process of saving a LSTM model in TensorFlow Lite from Keras. First step is to create a TensorFlow function from the LSTM model. After that, we get the concrete function from the TensorFlow function. In this step, we use TensorSpec to specify the input signature. Then, we save the model to TensorFlow format and use TensorFlow Lite Converter to convert from TensorFlow format to TensorFlow Lite format. Now, the last step is to download the model in format of TensorFlow Lite and use it in a mobile project.

5.1.3.5. Load model and call inference on mobile



```java
private static final int[] INPUT_SIZE = {1, 100, 12};
private static final int OUTPUT_SIZE = 7;
public static final int N_SAMPLES = 100;

private static final String TF_MODEL_FILE = "har_tflite_retrain.tflite";

private Interpreter interpreter;

public HARClassifier(final Context context) {
    try {
        interpreter = new Interpreter(loadModelFile(context.getAssets(), TF_MODEL_FILE), new Interpreter.Options());
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public float[] predict(float[][][] input) {
    float[][] result = new float[1][OUTPUT_SIZE];
    interpreter.run(input, result);
    return result[0];
}
```

Figure 5.8 Load model on mobile illustration

Figure 5.8 describes how we can load and call inference from the model that is saved in the previous step. First of all, we need to read the model file into the format of a mapped byte buffer and then pass it into an interpreter. The rest steps are to define the shapes of input and output and run the interpreter with data in the right shape.

5.1.4 Practice result

For testing the practice performance of the Long Short-Term Memory model, I performed a testing exam with following steps. Firstly, I added Text to Speech library to my application. When the service of my application finishes a prediction, it will speak its prediction. After that, I performed each activity 20 times except Biking since the Corona-virus disease was serious and I did not have a bicycle at that time. The result of exam is shown in following table.

Table 5.1 Result of practice test

Activity	**Number of true prediction**	**Number of false prediction**
STANDING	20	0
SITTING	20	0
WALKING	17	3
DOWNSTAIRS	12	8
UPSTAIRS	16	4
JOGGING	20	0

In detail, the model makes predictions with high performance with some activities such as standing, sitting and jogging. When I was walking, sometimes it predicted "walking upstairs". When I was performing the activity of walking upstairs or downstairs, it often predicted "walking". From the result shown in Table 5.1, the accuracy of the LSTM model in this test is: 87.5%.

5.2 Build an independent module for recognizing human activity

5.2.1 Problem

Nowadays, many mobile applications are developed and published on the application store. Applications about healthcare have been important for development in recent years. Human activity recognition is not only the important problem in the healthcare field, but also in other fields like game and personal fitting. If we must train a model, save it and call it on mobile from scratch when we need to develop an application that needs to be implemented with human activity recognition features, we will waste a lot of time on it. For this reason, I decided to build an independent module for supporting human activity recognition features. In the time limitations, I only published this module for Android platform with JAVA language.

5.2.2 Solution

The independent module I built will support developers implementing the human activity recognition feature with a little knowledge about it. They just need to download the module and use some APIs of it to perform recognizing human activity.

5.2.2.1. Features of module

The module I built is implemented with some following steps in implementing human activity recognition features. Firstly, the module stores the long short-term memory model that is used to predict human activity and support to load it. Developers who use this module just need to initialize an instance of HAR classifier with the context of their application. Secondly, the module also helps to reshape the sensor data into the input shape of the LSTM model, they can perform recognizing human activity by using some APIs of the module with some list of sensor data like accelerometer, linear acceleration and gyroscope in three directions. Finally, the module defined the result of the LSTM model with the HumanActivity enum. It defined basic activities that the model can predict and also determined their index in the result of the model. Therefore, developers do not need to have the knowledge about the model.

5.2.2.2. Build and publish module

In this part, I will illustrate how I built and published my module. Firstly, I created an Android module and implemented storing LSTM model and calling inference. After having the module, I built a very simple application for testing the module in the same project. And then, if the module works well, the next step is to publish it with jitpack.

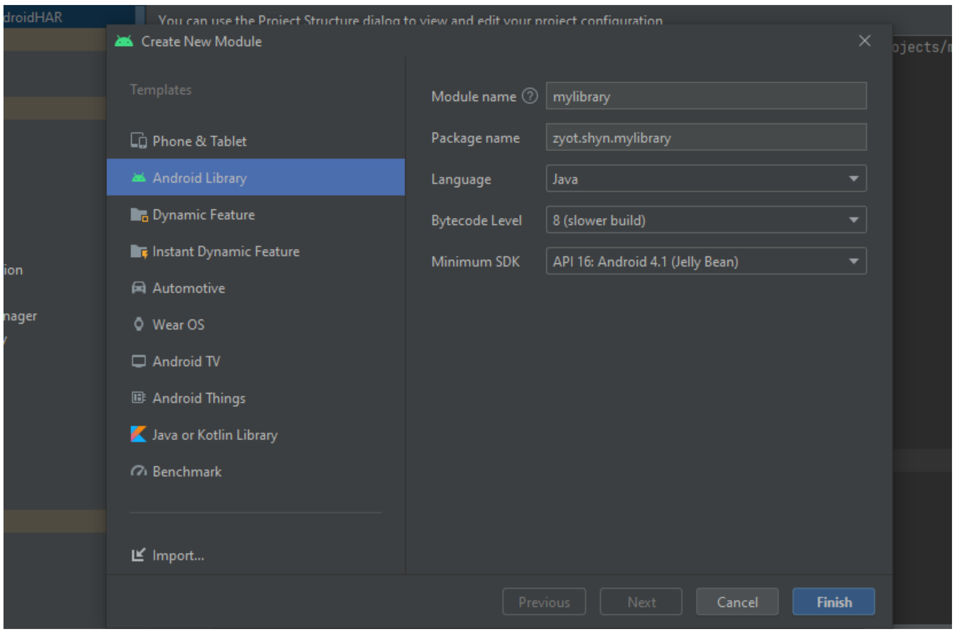
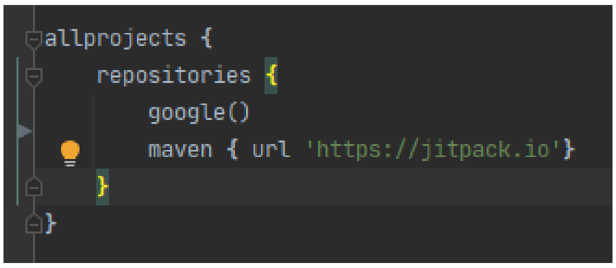

Module name: mylibrary
Package name: zyot.shyn.mylibrary
Language: Java
Bytecode Level: 8 (slower build)
Minimum SDK: API 16: Android 4.1 (Jelly Bean)

Figure 5.9 Create new Android module

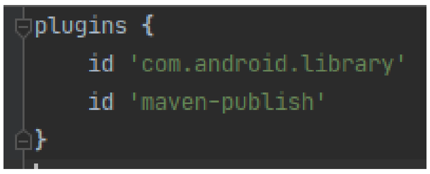
In order to publish the module with jitpack, I accessed the web site jitpack.io and signed in with my github account. After that, I configured the Gradle of the root project (file build.gradle of root project) with jitpack like the Figure 5.9.



```
allprojects {
    repositories {
        google()
        maven { url 'https://jitpack.io'}
    }
}
```

Figure 5.10 Configuration in gradle of root project

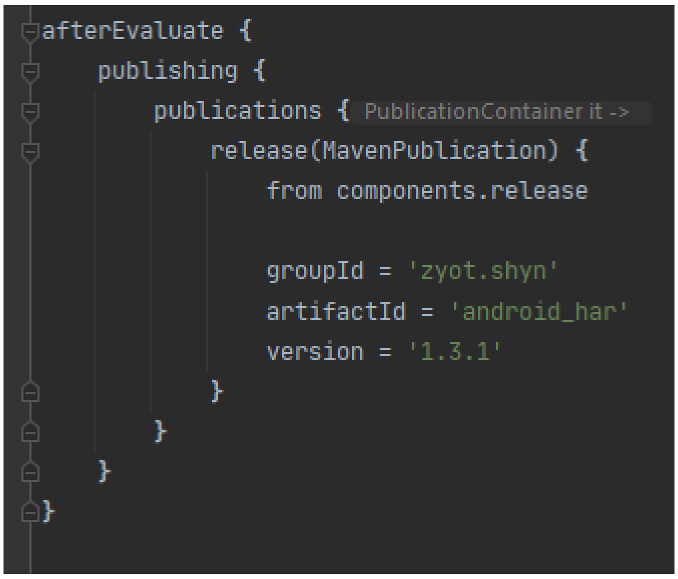
Then, I configured the Gradle of module (file build.gradle of module) with Maven publish like below figures.



```
plugins {
    id 'com.android.library'
    id 'maven-publish'
}
```

Figure 5.11 Add plugin to gradle of module

Firstly, I added the “maven-publish” plugin. This plugin provides the publishing method.



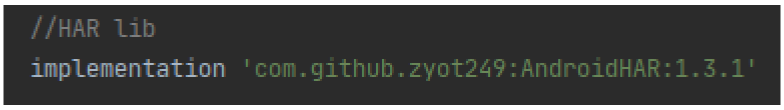
```
afterEvaluate {
    publishing {
        publications { PublicationContainer it ->
            release(MavenPublication) {
                from components.release

                groupId = 'zyot.shyn'
                artifactId = 'android_har'
                version = '1.3.1'
            }
        }
    }
}
```

Figure 5.12 Configuration of publication in gradle of module

Then, I configured the module with some information such as group id, artifact id and version. Maven uses a set of identifiers, also called coordinates, to uniquely identify a project and specify how the project artifact should be packaged. Group id is a unique base name of the company or group that created the project. Artifact id is a unique name of the project. And version is of course the version of the project.

The next step of publishing is to push the source code of the project into github repository. In order to ensure that the configuration is correct, open terminal with the folder of the project and perform the following command: “gradlew publishReleasePublicationToMavenLocal”. If the result shown in terminal is “BUILD SUCCESSFUL”, access “$HOME/.m2/repository” and check the released build. If everything goes well in the previous step, the module is ready to be released. The last step is to create a Github release or add a git tag and now, the module is published.

Finally, in order to import the module into a project, we need to configure the build.gradle of the root project for supporting jitpack like the above step of publishing module and then implement the module like Figure 5.12. The name of the module will have the following format: “com.github.yourGithubName:RepositoryName:Tag”.



```
//HAR lib
implementation 'com.github.zyot249:AndroidHAR:1.3.1'
```

Figure 5.13 Import module in project

CHAPTER 6: CONCLUSION AND FUTURE DEVELOPMENT

6.1 Conclusion

I have presented about the mobile-based application for taking care of the health of users. The system has been applied with a long short-term memory model for solving the human activity recognition problem in the healthcare field. The model has been built and trained on a server by using Google Collab and TensorFlow and saved by using TensorFlow Lite technique. The application has been built on the Android platform using some services of Google Firebase for authentication and storing data.

The project has achieved some good results after a semester of researching and learning new technologies, analyzing the requirements of the system as the desire of end user. The application is completed with main features and can use the core of the system, the human activity recognition module, for many features. The user interface of the application is quite simple and user friendly. Furthermore, an independent module that helps developers implement human activity recognition features is built and available for combining with other modules to solve difficult problems.

However, there are some disadvantages that I hope to improve in the near future I will illustrate in the next part.

6.2 Future work

In the near future, I would like to finish all the features that I have not completed in a time limitation of the thesis. Furthermore, I want to improve the Action Detect Service so that it consumes less power of the smartphone. It seems to work with lower frequency when users do not use the application. I also want to improve the performance of my LSTM model so that the feature using human activity recognition could have higher accuracy. In addition, my model should recognize more types of human activity in daily life. Another weakness of my model is that it can only recognize the activities of users when their smartphone is put in the jean’s pocket. It needs to retrain with other data sets in order to have the ability of recognizing human activity when the smartphone is in other common positions. Finally, I want to improve the user interface of the healthcare application in order to bring a better experience to users when they take care of their health.

REFERENCES

[1] J. Pang, "Human Activity Recognition Based on Transfer Learning," 2018.

[2] "Sensors Overview," Google, [Online]. Available: https://developer.android.com/guide/topics/sensors/sensors_overview.

[3] Bearqst, "Rxjava2 introduction and detailed examples," 04 01 2020. [Online]. Available: https://programmer.ink/think/rxjava2-introduction-and-detailed-examples.html.

[4] Josh Gordon, Sara Robinson, "TensorFlow Blog," 30 3 2018. [Online]. Available: https://blog.tensorflow.org/2018/03/introducing-tensorflowjs-machine-learning-javascript.html.

[5] Yuwen Chen, Kunhua Zhong, Ju Zhang, Qilong Sun and Xueliang Zhao, "LSTM Networks for Mobile Human Activity Recognition," in *International Conference on Artificial Intelligence: Technologies and Applications*, 2016.

[6] "Pervasive Systems Research Data Sets," [Online]. Available: https://www.utwente.nl/en/eemcs/ps/research/dataset/.

[7] "TensorFlow," [Online]. Available: https://www.tensorflow.org/lite.

[8] "ML Kit," Google, [Online]. Available: https://developers.google.com/ml-kit.

[9] "Firebase," Google, [Online]. Available: https://firebase.google.com/.

[10] "Jitpack.io," [Online]. Available: https://jitpack.io/docs/.

[11] M. Hollemans, "Machine learning on mobile: on the device or in the cloud," 16 2 2017. [Online]. Available: https://machinethink.net/blog/machine-learning-device-or-cloud/.

[12] A. Sachan, "A quick complete tutorial to save and restore TensorFlow models," [Online]. Available: https://cv-tricks.com/tensorflow-tutorial/save-restore-

tensorflow-models-quick-complete-tutorial/.

[13] J. Brownlee, "LSTMs for Human Activity Recognition Time Series Classification," 28 8 2020. [Online]. Available: https://machinelearningmastery.com/how-to-develop-rnn-models-for-human-activity-recognition-time-series-classification/.

[14] "An introduction to Machine Learning," [Online]. Available: https://www.geeksforgeeks.org/introduction-machine-learning/.

[15] S. Saxena, "Introduction to Long Short Term Memory (LSTM)," [Online]. Available: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/.