Activity Recognition by Using Neural Network in IoT Environment

Machi Iwata
iwata008@morris.umn.edu
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

Abstract

Internet of Things (IoT) has begun to be a part of our lives recently, and it is used in many fields and is useful for various purposes, such as healthcare, agriculture, and entertainment for home-usage. Using multiple sensors for activity recognition on human activity in the IoT environment can make the system capable of recognizing the user's behavior patterns. That makes the system capable of recognizing and predicting the user's behavior and configuring and adjusting the IoT environment accordingly. The collected data of human activity is sequential, therefore, Park et al. are using a Recurrent Neural Network, that is suitable for sequential data. The goal of Park et al. is to collect and analyze the data about human activity by using a neural network.

Keywords: Internet of Things (IoT), activity recognition, Neural Networks

1 Introduction

The *Internet of Things* (IoT) began to be discussed in 1982 [1]. IoT is a technology that makes our daily life easier today. IoT describes physical objects that are embedded with sensors, processing ability, software, and other technologies. IoT connects and exchanges data with other devices and systems over the Internet or other communications networks [1]. It is widely used in various fields such as agriculture, self-driven cars, and medical care, and future technological developments are highly expected. By integrating IoT into our lives, various ideas are born and new systems are developed.

Smart Home has been attracting attention among IoT fields. Smart Home refers to a convenient home setup where appliances and devices can be controlled remotely from anywhere with an internet connection using a mobile phone or other devices. The device and room allow users to adjust or control functions of home appliances such as temperature and lighting remotely [10]. By using multiple sensors and collecting data on human activity in an IoT environment, it would be possible to predict the user's behavior by using a neural network. With the data of user's usual patterns, we can build a system that configures and adjusts the IoT environment. For example, this user wakes up at the same time every morning, goes to the bathroom, and makes coffee with a coffee maker. If this coffee maker is a smart coffee maker, it would be possible for it to turn on automatically and make coffee for the user at the moment the user wakes up because it knows the user's pattern. On the other hand, with the data of user's unusual patterns, we can do something different, such as a reminder. An example of the unusual patterns is when the user is an elderly person. This user is supposed to take the medicine after every meal, and if they try to take the medicine more than they should be, which is considered unusual, the system reminds the user not to do it.

We would like to introduce the approach of Fahed Alkhabbas et al. for activity recognition. In their paper, they are proposing the architecture and the model for human activity recognition, however, they are not specifically explaining how they are utilizing the machine learning in their structure. Therefore, we complement the discussion with Park et al.'s machine learning.

We begin background information with Section 2, introducing terms that should be mentioned to understand later sections. After that, in Section 3, we present the machine learning architecture by Fahed Alkhabbas. Then we move on to the next section and talk about the activity recognition by using a neural network, and a proposed approach by Park et al. From there, we move to Section 5 where we walk through the test and results of the approaches in Park et al. Finally, we finish the paper with conclusion in Section 6.

2 Background

2.1 Neural Network Overview

Neural networks are also known as Artificial neural network (ANN) and it was inspired by the human brain, allowing computer programs to recognize patterns and solve common problems [8]. Artificial neural networks consist of nodes or neurons which are divided into three major layers; the input layer, one or more hidden layers, and the output layer. Every node in a layer is connected to all the nodes in the next layer, and these connections are called edges. Each edge has an associated weight which represents the strength of the connections.

2.2 Feedforward Neural Networks

Feedforward neural network (FNN) is a type of artificial neural networks, therefore the structure is the same as an artificial neural network. This type of artificial neural network does not form a cycle which means it only works in

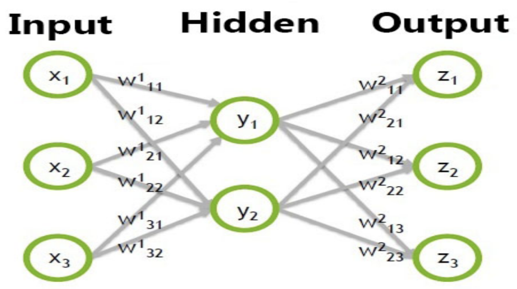


Figure 1. Illustration of Artificial Neural Network [8]

one direction. Feedforward neural networks were the first type of artificial neural network invented and the simplest type among neural networks. The entered values are put in the nodes in the input layer and are multiplied by the value of weights. When the nodes value are passed to the next layer, all the nodes value are summed up. This process occurs among all the layers when there is more than one hidden layer. Lastly, the value goes to the output layer from the hidden layer where the *activation function* is performed both in the hidden layer and the output layer which determines how the value is output to the next layer [8].

2.3 Training Neural Networks

Training neural networks is very important for better performance with accuracy, and usually, backpropagation is used to train FNN. FNN is usually used for *supervised learning* in cases where the data to be learned is neither sequential nor time-dependent. Supervised learning is a method of learning by giving the answer which is also called labeled data or labels, in advance. For supervised learning to work, a labeled set of data is necessary that the model can learn from to make correct predictions. Here is how backpropagation works. Training neural networks starts with assigning a random value of weights. Then we calculate the nodes to get the output value in the output layer and compared these values to the labels. To improve the neural network with high accuracy, we re-assign the value of weights until the output value and the value of labels have fewer differences [8]. This difference between the output value and the value of labels is called loss, and a function called *loss (or error) function* is used to calculate the loss. This process is automatic, and there are various types of loss functions.

2.4 Recurrent Neural Networks

Recurrent neural networks are a class of artificial neural networks, and they are derived from feedforward neural networks. Their key difference from FNN is that recurrent neural networks do form a cycle or loops in the hidden layer because the output of the hidden layer comes back into the hidden layer as an input. When a recurrent neural network makes a decision, it considers the current input and also what it has learned from the inputs received previously, which allows it to be very precise in predicting sequential data. Sequential data is the data that comes in sequences. An example of sequential data is a sequence of words, such as written text, or a sequence of sounds, such as speech or music. Recurrent neural networks are suitable for pattern recognition such as voice recognition, activity recognition, speech recognition, and natural language processing [9]. However, there is an issue with recurrent neural networks, and it is called the vanishing gradient problem. A vanishing gradient is when the gradient shrinks as backpropagation is used through time. The gradients are used to minimize errors or losses while updating the parameters of each layer and are multiplied one after another as it approaches the input layer from the output layer. The problem here is, that if a small value gradient continues in each layer, the gradient near the input layer is regarded as zero, and learning cannot proceed well and it would not contribute much to the learning process. This problem happens when a neural network carries information when a sequence is too long. The solution for this is long short-term memory units (LSTM) and gated recurrent units (GRU) [12].

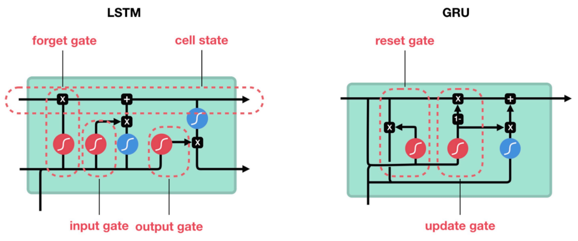


Figure 2. Illustration of LSTM and GRU [12]

2.5 Long Short-Term Memory and Gated Recurrent Units

As mentioned, Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) are good solutions for vanishing gradients. RNN would struggle to carry information from earlier time steps to later steps if a sequence is long enough, which means that if RNN is trying to process a long paragraph of text to make predictions, it might not be able to store important information from the beginning of the sequence.

Both LSTM and GRU have an internal mechanism called gates that can regulate the flow of information. These gates can learn which data should be kept or thrown away in the *hidden state*. The hidden states are a representation of previous inputs when RNN makes loops and it keeps information. LSTM has three gates; forget gate, input gate, and output gate. The key point of LSTM architecture is *cell state*. The cell state keeps information from previous inputs and combines the entire sequence while the hidden state emphasizes the most recent input. The forget gate is in charge of deciding

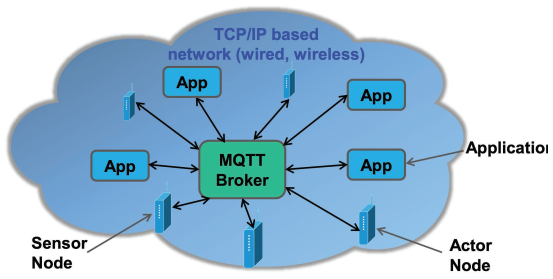


Figure 3. Illustration of the MQTT system [7]

which information to keep or discard. The input gate updates or adds data to the cell state. Lastly, the output gate decides what the next hidden state should be.

On the other hand, GRU has no cell state and has two gates instead of three; update gate and reset gate. The update gate configures in the same way as the forget gate and the input gate in LSTM, so it deals with deciding which information should be kept or thrown away in the hidden state. The reset gate is another gate that decides how much past information to throw away [12].

2.6 MQTT and REST

In the next section, we introduce the architecture of Alkhabbas et al., and they are using *MQTT* and *REST services* as a communication tool among the architecture. MQTT stands for MQ Telemetry Transport and is a lightweight publish-subscribe messaging protocol designed for machine to machine (M2M) telemetry in low bandwidth environments [7]. The "MQ" originally had roots in the IBM MQ product line which is used to stand for Message Queue [2]. MQTT is often used to build the IoT system and is primarily used for the data transmission process. Figure 3 illustrates the system of the MQTT message broker. The sensor node sends the sensor data to the MQTT broker. The application receives sensor data from the MQTT broker and decides to activate equipment in the room, such as the lamp and the curtains. Then the application sends an activation message to the actor node through the MQTT broker.
REST stands for REpresentational State Transfer and it is an architectural style for distributed hypermedia systems [4]. REST can make the systems communicate easier to each other.

3 Approach

In this section, we present Alkhabbas's approach to the smart hotel setting by focusing on the architecture model. We consider a smart hotel with multiple sensors connected to the objects and these sensors collect the data in the same way as Park et al.'s approach. All the sensors were installed in the daily objects such as lights, curtains, laundry machines, etc., and there are various types of sensors; pressure sensors, light sensors, temperature sensors, gas sensors, etc. These sensors are used according to their purpose or daily devices, and these sensors were activated when a movement or activity was detected or the machine started working. The setting is that the user has a tablet, a smartphone, a laptop, and wireless speakers. The user starts using their laptop at a table, and we assume that they are suggested to configure the following services via the available smart TV for a better working experience:

1. To adjust the light level of the room by increasing the the intensity of the lamp and by opening the curtains.
2. To use the available smart TV as an extended monitor to their laptop and the TV's speaker as the sound output device to improve the user's experience.

We consider this smart hotel setting and suggestions for the user and present the proposed model and architecture by Alkhabbas.

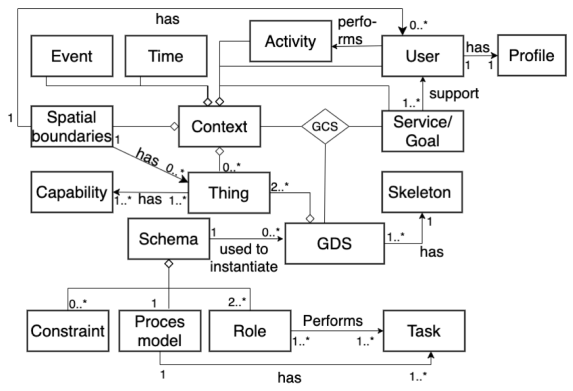


Figure 4. The structure of the proposed model by Alkhabbas et al. [5]

3.1 Model

Figure 4 illustrates a model of the approach by Alkhabbas et al., and we explain the individual role and their relationship. A Goal-Driven IoT System (GDS) is a set of things with their individual functionalities that cooperate and connect temporally to achieve a goal[6]. GDS is a core of this model, and it is formed of at least two things that have individual functionalities, in order to provide a service to a user or achieve their goal in a specific context. Context represents the five Ws which are Who, What, Why, When, and Where. The context or the five Ws is represented by users, the service/goal, activity, time, event, spatial boundaries, and things. Event represents dynamic changes in the context, for example, sudden unavailability of things such as lamps and curtains. Activity is performed by a user, for instance, sleeping, working, eating, and etc. Things simply represent anything in a hotel room, such as lamps, curtains, and TV. A

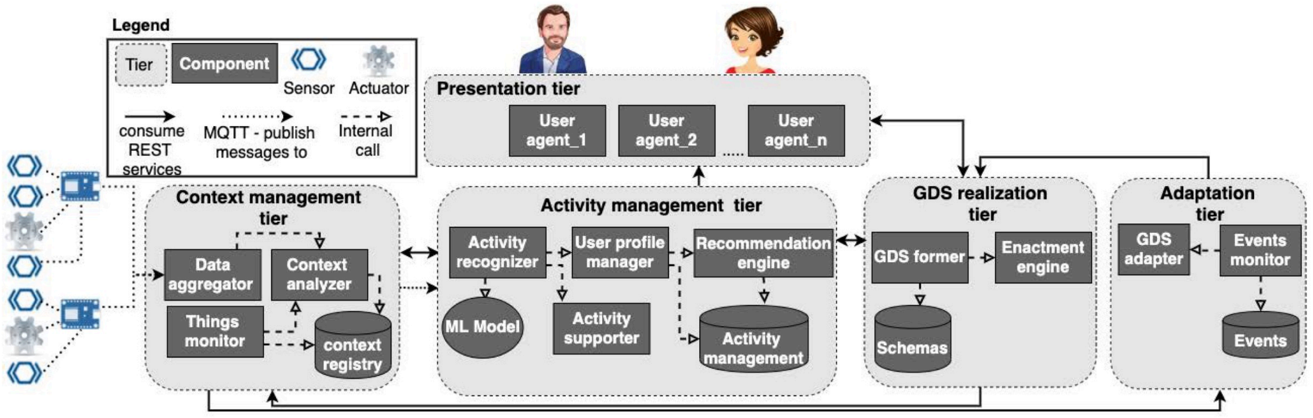


Figure 5. Architecture with five tiers [5]

thing by itself has one or more capabilities which means, for example, a lamp can change the brightness and/or change the color of the light in a room. A capability can have one or more preconditions and effects. For instance, a lamp can be turned on if it is turned off, and turning it on results in increasing the light level in the environment. Spatial boundaries means a hotel room in this case, and it says the spatial boundaries "has" user and things because there should be a person who uses the system and the room, and the room should have some equipment such as lamps and TV. A schema is used to instantiate forming GDS to provide services of a specific type. A schema is represented by constraint, process model, and role. A role represents a stereotype of tasks that the system should perform. A process model represents the order of the execution of the role's tasks. A constraint represents the prerequisites that the system should meet to adopt a role. Instantiating a schema means assigning dynamically the roles it comprises to available things that meet the specified constraints and have capabilities to perform the roles' tasks. Lastly, the authors mention that they consider two types of preferences, default and temporary preferences. The default preference is that a user always has. For example, if the user always uses a stereo speaker to play music from their device while they study, that is the default preference. On the other hand, the temporary preference represents the user's preferences when the default preference cannot be applied. For example, if the stereo speaker is not available, the user plays music on their device while they study, that is a temporary preference.

3.2 Architecture

In the Alkhabbas's proposed architecture, there are five tiers; context management tier, activity management tier, GDS realization tier, adaption tier, and presentation tier. We can think of a tier as a row or layer in a series of similarly arranged objects. Figure 5 shows the architecture and each tier has a role and operates independently. As mentioned earlier, the communication among tiers, they are using REST services and MQTT messages. **The context management tier** manages the context; discovers, monitors, registers, analyzes and stores information; it summarizes and formats the collected data and passes it to the next tier, the activity management tier. **The activity management tier** determines how it can support users to perform their activities. It recognizes the users' activities that are currently happening by using a pre-trained machine learning model. Alkhabbas et al. did not specify the kind of machine learning that could be used here, but the approach is similar to the one is proposed by Park et al. This tier is constituted in a user's profile, for example, their activities and preferences. The profile should be queried and updated. This tier also analyzes the activity so it can learn the user's behavior pattern and what kind of service they tend to choose. **GDS realization tier** realizes *Goal-Driven IoT System (GDS)* to support users to perform their activities. This tier is responsible for communicating with the user and inferring the user's behavior. **The adaptation tier** is event-based and monitors events that are stored in the repository and maintains the provisioning of services. And finally, **the presentation tier** makes the user able to communicate with the system by using the user's devices such as smartphone or tablet.

4 Activity Recognition

4.1 Approach

Park et al. proposed an approach for activity recognition by using neural network, and they specifically used RNN with residual vector, called Residual-RNN. In the architecture, there are residual vectors, and the attention module. Sensor embeddings and feature vectors are used as inputs. Figure 6 shows the structure of the Residual-RNN. In the sensor embedding layer, they use sensor-id to identify individual sensors. One activity involves several sensors; for example,

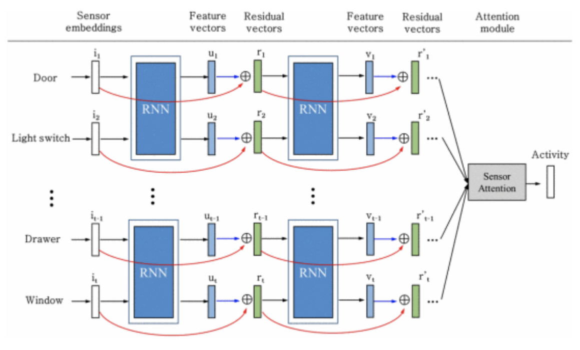


Figure 6. Illustration of Residual-RNN by Park et al. [11]

using the bathroom involves the sensors of a switch of light, toilet, sink, etc, and this provides a set of input values. In the RNN layer, either LSTM or GRU are used. Since there are gates in LSTM and GRU, this layer decides which information to keep or throw away. In the feature vectors, they use the sensor-time label which is a matrix that has sensors and time sequence. This layer makes activity recognition possible. The residual vector skips some layers to deal with the vanishing gradient problem. Finally, there is the attention module layer at the end. In this layer, they are using soft attention. We assume that soft attention is used as the activation function called softmax in this architecture. The softmax is usually used in the output layer of a neural network to predict a probability distribution. The probability distribution is a function that describes all the possible values and likelihood that a random variable can take within a given range. This says that the softmax is used when more than two class labels are required. An example of the probability distribution would be that the input value was "preparing dinner" and after making prediction, the probability of the input value being "preparing dinner" was 70 percent, and the rest of the probability, for example, "preparing breakfast" was 30 percent. This means that the predicted input was "preparing dinner" by treating outputs as probability because the probability of the predicted input being "preparing dinner" was higher than "preparing breakfast."

5 Test and Results

5.1 Dataset

In this section, we would like to briefly describe the dataset and how the data was collected, before going over the test and results of activity recognition by using a neural network.

To evaluate their approach, Park et al. trained multiple sensors and activity data from the Massachusetts Institute of Technology (MIT) laboratory [14]. This dataset is a data of one resident who lives in an apartment, and the resident's activity and movements were recorded for two weeks. As mentioned in the earlier section, resident's and machines' movements were recorded in the same way as Alkhabbas's recording. The data they collected helps in understanding and analyzing the user's life pattern. In the dataset, there are 295 activities, 76 sensors, and 2,823 detected activations.

5.2 Test

Park et al. compared their approach Residual-RNN with LSTM and with GRU models to the general FNN and the general LSTM and GRU. They implemented the Residual-RNN by using the Tensorflow library. Tensorflow library is a free and open-source software library for machine learning and artificial intelligence (AI). Tensorflow specializes in training and inference of deep neural networks [3]. To evaluate the overall performance of the real-world dataset which is collected by MIT, they randomly split the dataset into three sets;

1. Training set 150 sample data (50%) of the total 295 activity dataset
2. Validation set (25%)
3. Test set (25%)

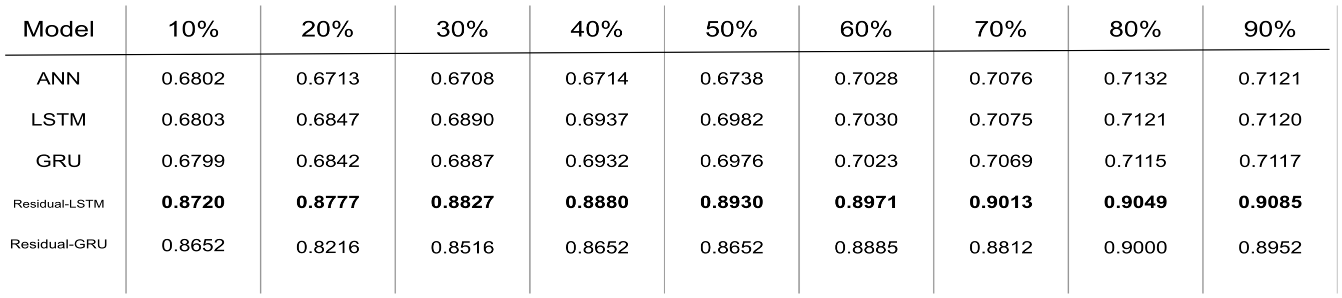
A training set is a first and largest dataset to be used. This set is used to adjust the model's parameters to fit the relation between the inputs and their labels. A validation set is used to tune the model hyperparameters, which are the parameters that control the behavior of the machine learning algorithm. We adopt the model with the best performance to use. The test set is used to check the accuracy of the model, and it is only used for performance testing [13].

In this evaluation, they are using a measurement called Root Mean Square Error (RMSE) to get the accuracy of the results. RMSE is an error expressed by a function that evaluates how different the prediction and the values are. Here is the formula of RMSE:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(S_i - O_i)^2}$$

where O_i are the observations, S_i predicted values of a variable, and n the number of observations available for analysis.

They got the result by repeating this procedure *5-fold cross-validation* 20 times from the random data split process. 5-fold cross-validation is usually described as k-fold cross validation and it is a type of cross-validation. Cross-validation is a method to evaluate the skill of machine learning models. The k-fold cross-validation evaluates the accuracy rate by dividing the data into k pieces and using one of them as test data and the remaining k-1 pieces as training data. This is a method of learning k times so that all k data become test data once, and averaging the accuracy. Therefore, in this case, they divide the data into five pieces and repeat this procedure 20 times.



Model	10%	20%	30%	40%	50%	60%	70%	80%	90%
ANN	0.6802	0.6713	0.6708	0.6714	0.6738	0.7028	0.7076	0.7132	0.7121
LSTM	0.6803	0.6847	0.6890	0.6937	0.6982	0.7030	0.7075	0.7121	0.7120
GRU	0.6799	0.6842	0.6887	0.6932	0.6976	0.7023	0.7069	0.7115	0.7117
Residual-LSTM	**0.8720**	**0.8777**	**0.8827**	**0.8880**	**0.8930**	**0.8971**	**0.9013**	**0.9049**	**0.9085**
Residual-GRU	0.8652	0.8216	0.8516	0.8652	0.8652	0.8885	0.8812	0.9000	0.8952

Figure 7. Table of accuracy on test set [11]

5.3 Result

There are two results from Park et al.'s evaluation. Figure 7, the table, shows the overall accuracy of test data of RNN, general LSTM, general GRU, Park et al.'s Residual-LSTM, and Residual-GRU. Percentages on top of the table show the ratio of the training set, and as we can see, the higher ratio of the training set tends to have higher accuracy. Compared to other models, we can also see that residual-LSTM and Residual-GRU have better performance.

The second one, Figure 8, shows the comparison of the results specifically between Residual-LSTM and Residual-GRU in terms of error rate. The results are very similar to each other, however, Residual-LSTM has slightly better performance than Residual-GRU, and we believe this is because LSTM has one more extra gate than GRU. For this reason, for the beginning process, we can assume that Residual-GRU trains faster than Residual-LSTM.

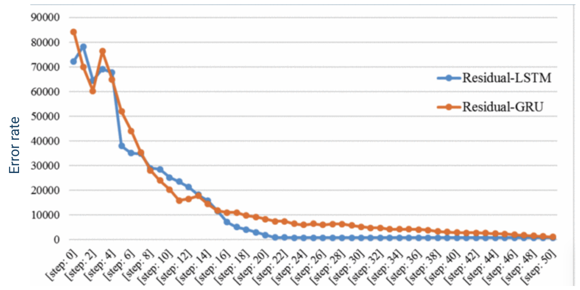


Figure 8. Graph of error rate [11]

6 Conclusion

Park et al. developed activity recognition in an IoT environment with Residual-RNN and compared it to other models. They successfully trained the neural network by using the data. Their proposed approach shows the best performance with high accuracy among the others. Considering that the data was only from one person, more data is needed in order to know if the system works for other users' activity recognition, and the reliability of the findings is low because of that.

Overall, by combining Park et al.'s neural network approach and Alkhabbas's architecture approach, there is a possibility that we can build the predicting system on human activity.

Acknowledgments

I would like to thank my advisor Elena Machkasova, Kristin Lamberty, and Nic McPhee, and Thomas Harren for their feedback and advice.

References

[1] 2022. Internet of things. https://en.wikipedia.org/wiki/Internet_of_things

[2] 2022. MQTT. https://en.wikipedia.org/wiki/MQTT

[3] 2022. Tensorflow. https://en.wikipedia.org/wiki/TensorFlow

[4] 2022. What is REST. https://restfulapi.net/

[5] Fahed Alkhabbas, Sadi Alawadi, Romina Spalazzese, and Paul Davidsson. 2020. Activity Recognition and User Preference Learning for Automated Configuration of IoT Environments. In *Proceedings of the 10th International Conference on the Internet of Things* (Malmö, Sweden) *(IoT '20)*. Association for Computing Machinery, New York, NY, USA, Article 3, 8 pages. https://doi.org/10.1145/3410992.3411003

[6] Fahed Alkhabbas, Romina Spalazzese, and Paul Davidsson. 2018. ECo-IoT: An Architectural Approach for Realizing Emergent Configurations in the Internet of Things. In *Software Architecture*, Carlos E. Cuesta, David Garlan, and Jennifer Pérez (Eds.). Springer International Publishing, Cham, 86–102.

[7] Steve Cope. 2021. Beginners Guide to the MQTT protocol. (2021). http://www.steves-internet-guide.com/mqtt/

[8] IBM Cloud Education. 2020. What are neural networks? https://www.ibm.com/cloud/learn/neural-networks

[9] IBM Cloud Education. 2020. What are recurrent neural networks? https://www.ibm.com/cloud/learn/recurrent-neural-networks

[10] Adam Hayes. 2022. Smart home. https://www.investopedia.com/terms/s/smart-home.asp

[11] Jiho Park, Kiyoung Jang, and Sung-Bong Yang. 2018. Deep neural networks for activity recognition with multi-sensor data in a smart home. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*. 155–160. https://doi.org/10.1109/WF-IoT.2018.8355147

[12] Michael Phi. 2020. Illustrated guide to LSTM's and GRU's: A step by step explanation. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

[13] Tarang Shah. 2020. About train, validation and test sets in machine learning. https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7

[14] Emmanuel Munguia Tapia, Stephen S. Intille, and Kent Larson. 2004. Activity Recognition in the Home Using Simple and Ubiquitous Sensors. In *Pervasive Computing*, Alois Ferscha and Friedemann Mattern

(Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 158–175.