International Journal of Computational Intelligence Systems
Vol. 13(1), 2020, pp. 852–863
DOI: https://doi.org/10.2991/ijcis.d.200611.001; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

Research Article

An Intelligent and Automated Approach for Smart Minimarkets

Talal A. Edwan[1,*], Ashraf Tahat[2], Sara Hammouri[1], Leen Hashem[1], Leen Da'boul[1]

[1] *Department of Computer Engineering, Princess Sumaya University for Technology, Amman, Jordan*
[2] *Department of Communications Engineering, Princess Sumaya University for Technology, Amman, Jordan*

ARTICLE INFO

Article History
Received 08 Jul 2019
Accepted 03 Jun 2020

Keywords
Smart cities
ANN
Smart minimarket
Smart shopping
Networked embedded systems
Queueing analysis

ABSTRACT

This paper presents the design and implementation of a smart and safe minimarket prototype for deployment in busy smart cities to mitigate the overhead of shopping experience. The prototype allows customers to remotely access and browse the available products at the minimarket using a special smart-phone application. The system can intelligently detect the nearby location of customers and subsequently provide location-dependent services such as allowing orders to be placed using the application, predicting weekly customer expenditures based on artificial-neural-network machine-learning approach, and automatically delivering purchased products using a robotic shopping cart. This proposal is believed to support safe shopping which became a critical issue after COVID-19 pandemic. From a service provider view point, the application allows the provider to remotely manage the minimarket by adding/removing product items, keeping track of shortage in products, and getting revenue information. Empirical results show that the average service time of the minimarket is ≈ 60 seconds per customer. However, an analytical model based on queueing theory was used to analyze the performance of the system when customers arrive according to a Poisson random process and get served according to a general-service-time distribution (M/G/1). The case of batch customer arrivals ($M^{[H]}/G/1$) was also analyzed, where batch size is also assumed to be random. Various traffic intensities and the effect of variable service times were studied and cross-validated with simulation results. Worst-case scenario shows that under heavy load of 95%, when customers arrive at the minimarket every 63 seconds on average, the average response time for each customer is ≈ 10 minutes.

1. INTRODUCTION

In recent years, the concept of smart cities was introduced. This concept involves many aspects related to managing urban systems. Such systems include, e.g., managing water resources, energy, waste, transportation and other city-wide services such as supermarkets and shopping centers. City management becomes more critical with the increase of population. It is anticipated by United Nations (UN) that around 60% of the global population is expected to live in large cities [1]. In Jordan however, as for 2018, 42% of a total of around 10 million people live in the capital city of Amman, and 18% live in the next bigger city, Irbid [2]. Under circumstances with the similar growth rates, it is anticipated that the number of people living in Amman will increase by approximately 84% in 2050 [2]. This increase in city population poses many challenges on the city resources and services. One such challenge is how to make the shopping experience in supermarkets, shopping malls, and minimarkets—which are heavily scattered in the city of Amman—easier and more convenient for both, customers and suppliers. Another challenge however is safe shopping. Since the health crisis of COVID-19 pandemic at the beginning of 2020, the Jordanian government allowed citizens to shop through minimarkets (not supermarkets) and within fixed social distance.

A shopping center can offer a variety of services to assist customers in their shopping, e.g., a customer can view the available products and their prices/offers using his/her smart phone before actually arriving at the shopping center, i.e., at home or while a customer is on his/her way to the shopping center. Furthermore, the shopping center can learn customers shopping habits and use that to predict the amount of customer expenditures for the current weekly shopping based on previous data of customer expenditures. In addition, a shopping center can smartly detect the existing of a customer when he/she arrives at the shopping center and provide him/her with additional services, such as allowing the customer to shop using his smart-phone application, make an order and wait for a robotic cart to deliver the shopping to his/her car.

This paper discusses the realization of the aforementioned services using a smart minimarket prototype, which was designed and implemented for experimental purposes. The proposed minimarket supports six types of products and has a robotic cart that automatically carry purchased items to the customer nearby place when an order is placed. The minimarket is able to learn and predict customer expenditures trends using an artificial neural network

[*] *Corresponding author. Email: t.edwan@psut.edu.jo*

(ANN) machine-learning approach and feed this information back to the customer on his/her next weekly shopping. Also, it provides customers with location-dependent services. Both customers and service providers can access the minimarket system remotely with different privileges from a smart-phone application that was developed for this purpose.

The minimarket system has an average service time of approximately 60 seconds per customer. An analytical model based on queueing theory was used to assess the performance of the minimarket system for general service-time distribution and Poisson customer arrivals under various levels of load. Simulation results show that, under heavy load of 95% (when customers arrive at the minimarket every 63 seconds on average), the average response time for each customer is approximately 10 minutes.

The specific contribution of our work consists in the following:

1. Design and implementation—hardware and software—of an intelligent drive-through minimarket system (i.e., novel system design and construct) that has a set of novel features which improves both customer and provider services as follows:
    - Remote and convenient accessibility to a minimarket through a specialized smart-phone application.
    - Robotic drive-through capability.
    - System ability to predict current customer expenditures for each type of products and its ability to link that information with any existing discounts, then feeding this information to the customer.
    - Real-time monitoring of product quantities and revenue using a dedicated database and a specialized smart-phone application.
2. The use of machine-learning approach (ANN) to predict weekly customer expenditures.
3. Providing an analytical model to study the scalability of the minimarket prototype and evaluate its performance before installation.
4. Validation of the analytical model using a simulation model created using the R Language.

The rest of the paper is organized as follows: Section 2 presents related work in the literature, Section 3 gives a general overview of the minimarket prototype. The software and hardware parts of the design are presented in Sections 4 and 5, respectively. The analytical model used to evaluate the performance of the minimarket prototype is described with its necessary background in Section 6 and the discussion of simulation results that cross-validates the analytical results along with the evaluation of the prediction capabilities of the system are elaborated upon in Section 7. Finally, Section 8 presents our conclusion and future work.

2. RELATED WORK

The decline in costs of electronics and sensors, along with the wide spread of smart phones and the popularization of machine-learning techniques can enable supermarkets and shopping centers to be smart and provide adequate services to customers. In [3], researchers presented a communication robot for use in shopping malls to provide shopping information, route guidance and build rapport with customers. Although the robot was human operated, it was able to detect the presence of customers with floor sensors to initiate interaction and was able to identify individuals with Radio-Frequency Identification (RFID) tags. An alternative approach for identification, however, is to use secure customer accounts on a database, where customer information can be stored and identified using unique user Identification (ID) number. On one hand, this allows the supply of data directly from customers and making it accessible to stakeholders, service providers and decision-makers. On the other hand, the stored customer data can be used to provide additional services to customers (e.g. learning customers' shopping habits and offering them services that fit their habits). An experimental mobile, intelligent and interactive shopping assistant for shopping centers was studied in [4]. Two main points were focused on: the first was online building of maps of the shopping center using sonar-based grid maps and vision-based graph maps. The second was detection and tracking of customers during the guide tour using a probabilistic approach; both points were proved reliable. A low-cost smart shopping facilitator was proposed in [5, 6] to aid visually impaired and blind people. The facilitator provides guidance for customers in supermarkets to identify and purchase goods, RFID tags were implemented to identify products. A case study that emphasizes this approach was discussed in [7].

A very interesting attempt was made in [8] to make customers' payments a smooth process by modeling the trade-off between usability and security as an optimization problem. Since systems that allow customers to do payments with little effort (e.g., smart cards) usually do not require explicit approval from the customer, thus making the payment experience less cumbersome. However, other systems may sacrifice the ease of payment in favor of more accurate payment. The researchers proposed a mobile payment scheme that tries to minimize the transaction cost (cost function), i.e., to find the optimal point of the trade-off.

There has been several attempts to improve the functionality of goods-handling devices (e.g., vending machines, refrigerators, coffee machines, etc.) so it can better fit human needs. An experimental Internet of Things (IoT) vending machine which has different levels of customer involvements was designed and implemented in a Future Living Lab environment [9]. The vending machine allows customers to identify themselves so they can access personalized service. However, it was mentioned that identification can be done by means of using different digital tokens, such as regional cards, travel cards and Near-Field Communication (NFC) smart phones. Through smart cards, RFID and NFC devices, the customer can receive contextualized communication and marketing strategies as well as ad hoc interactions with the vending machine. A relatively similar attempt was made in [10]. A smart coffee vending machine was presented in [11]. Similarly, the design and implementation of a smart refrigerator based on neuro fuzzy embedded agent was discussed in [12]. The intelligent refrigerator is able to learn user habits and notify the user in cases where he/she consumes an unusual number of calories.

3. SYSTEM OVERVIEW

An overview of the smart minimarket system is illustrated in Figure 1. The system supports six different types of products. Using a special smart-phone application, a user (customer or provider)

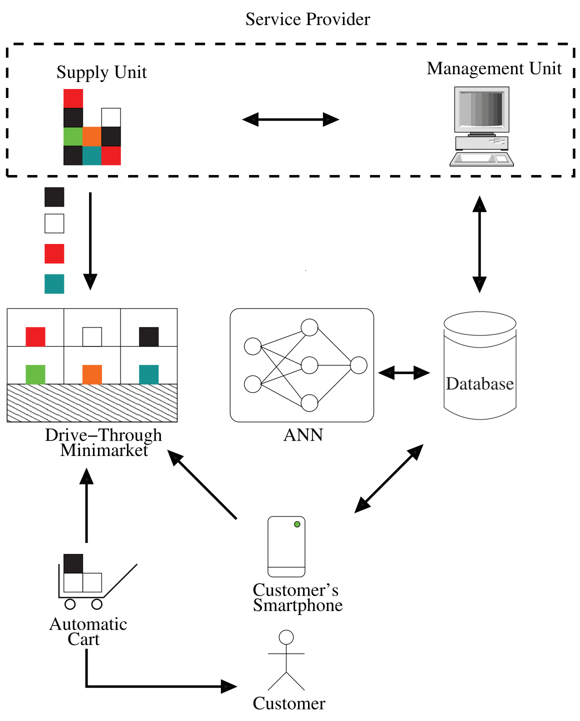


Figure 1 | General design.

can securely connect to server and access the minimarket database to check the available products and their prices. If a user is logged in into the server as a customer, and in case the customer is doing a weekly shopping, the server uses the customer ID to predict the amount of money that the customer will spend in his/her current weekly shopping, then feeds this information back to the customer through the smart-phone application. The server uses machine-learning algorithm based on ANN to predict future customer expenditures. This is based on three features: the amount of expenditures on each of the last week and for the week before, and the amount of expenditures for the week before one year. Using this information, the server predicts the amount of expenditures in the current week and sends this information to the smart-phone application.

If the customer is nearby the minimarket and no other customers are being served, the smart-phone application allows the customer to place an order by tapping on a special button. This in turn initiates a sequence of automatic events, which starts by updating the database entries, then activating the relevant servo motors for the selected product items to drop the purchased quantities into an automatic shopping cart. Then, the shopping cart moves automatically to the customer's location. When the customer lifts all shopping items from the cart, the cart returns to its original position at the minimarket and waits for another customer to serve.

On the other hand, if a user is logged in into the server as a provider, he/she can view the product items and their prices, add new items, add pictures of items, specify their prices, know how much items are left in the stock and get an analysis of profit. This information can be automatically utilized by the "Service Provider" (see Figure 1), to compensate for any shortage in product items by automatically loading the necessary number of product items into the minimarket.

4. SOFTWARE DESIGN

There are three software components in the minimarket system: the software application, the database and the time-series prediction engine. We elaborate on each of these components in more detail in this section.

4.1. Software Application

In desktop software applications, when certain application needs the services of another application it invokes the whole application. In smart-phone applications, however, this behavior is more common. In order to facilitate this paradigm in an efficient way, a smart-phone application is usually broken into a set of *Activities*. Instead of invoking a whole application, a calling application invokes the required activity in the invoked application. In other words, the activity serves as an entry point for the calling application. Broadly speaking, one activity implements one screen in a smart-phone application.

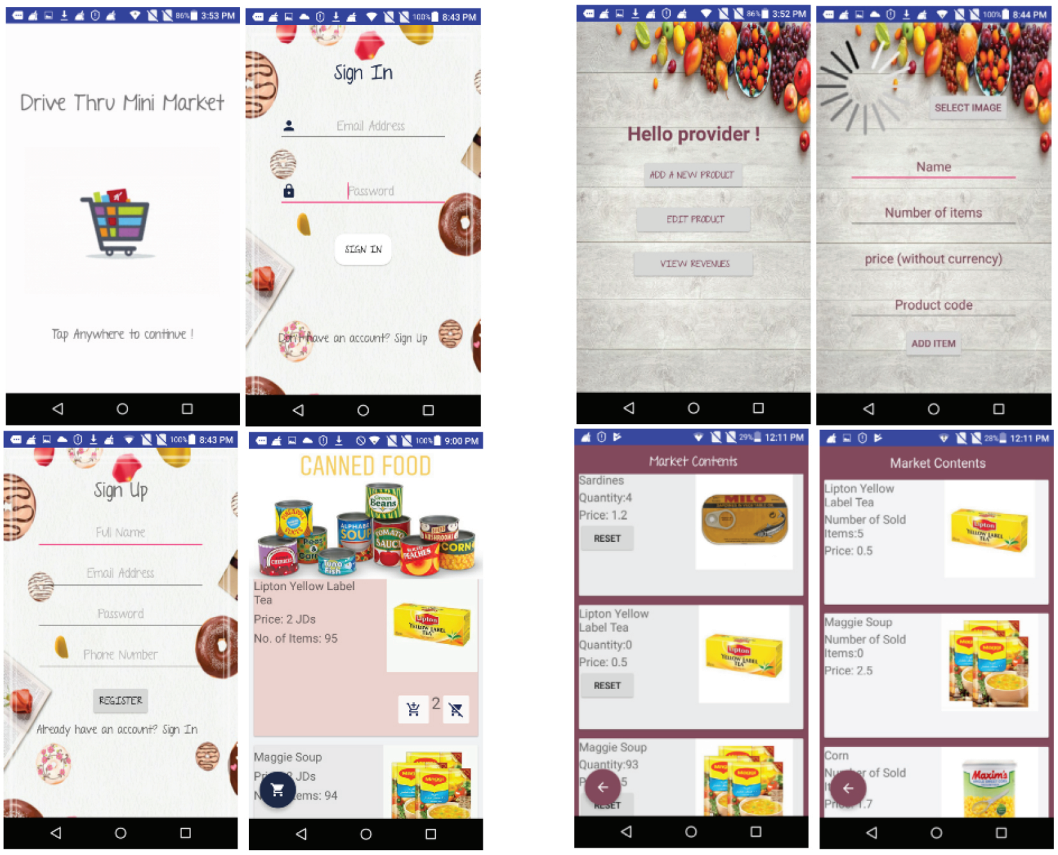
An Android-based smart-phone application was developed to enable the user to interact with the minimarket. The application has 11 activities, we briefly discuss six essential activities of this application.

The Splash-Screen activity is the first activity. When a user opens the application, a splash screen which has the logo of the minimarket appears. The client can proceed to the next activity by tapping anywhere on the screen. The next activity is the Sign-In activity, where the screen prompts the user to fill in his/her personal information: his/her available balance, valid email address and valid password. However, a new unregistered user will be directed to the Sign-Up activity. In this activity, the user can register as a client by creating a new account and entering his/her information on the database of the system. Figure 2a depicts these activities. It is worth noting that only the system administrator can register the user as a provider.

After a successful client authentication in the Sign-In activity, a client is directed to the Client activity (bottom-right screen in Figure 2a). This activity contains an advertisement banner of the products available at the minimarket. Below it, there is a list (card view) where each row in this list is dedicated for a single product. Each row contains the name, picture, price and number of remaining items of one type of products. In addition, that activity has two buttons that allow the user to either add or remove a product from his/her shopping cart. When one of these two buttons is pressed, a variable called ordered_quantity in the database is automatically updated. This variable is used to inform the provider about the exact number of purchased items per product.

The minimarket has six groups of products: Bread, Canned seafood, Tea, Soup, Canned corn and Potato crisps. These can be changed to other types of products as required by the service provider. All products' information are retrieved from the database in real-time. When the client finishes selecting the items he/she wants to buy, he/she can tap on the shopping-cart button at the bottom-left corner of the screen (Client activity). This, in-turn will move the user to a new activity (shopping-cart activity).

Unlike the Client activity, which displays the information of all products available at the minimarket, the Shopping-cart activity



(a) Activities: Splash screen, Sign in, Sign up, and Client.

(b) Activities: Provider, Add new item, Reset product, and Revenue.

Figure 2 Application screens.

displays a list view (card view). It shows only information about the products that the client selected for purchase. At the bottom of the Shopping-cart activity, there are two buttons: the first one is a button on the left (arrow button), if tapped on, it moves the user to the previous activity (the Client activity), before this action takes place, a warning message appears to warn the client that his/her shopping cart will be emptied. The second button is on the right of the screen, when tapping on it, an Alert Dialog appears showing the total price of the client's order. Once confirmed, the application sends a command to the microcontroller unit (MCU) at the minimarket. The command contains the name and number of each purchased product item. The purpose of this command is to activate the relevant motors, which will in-turn drop the items into the shopping cart. Finally, the application moves to the splash screen and waits for another order to be placed.

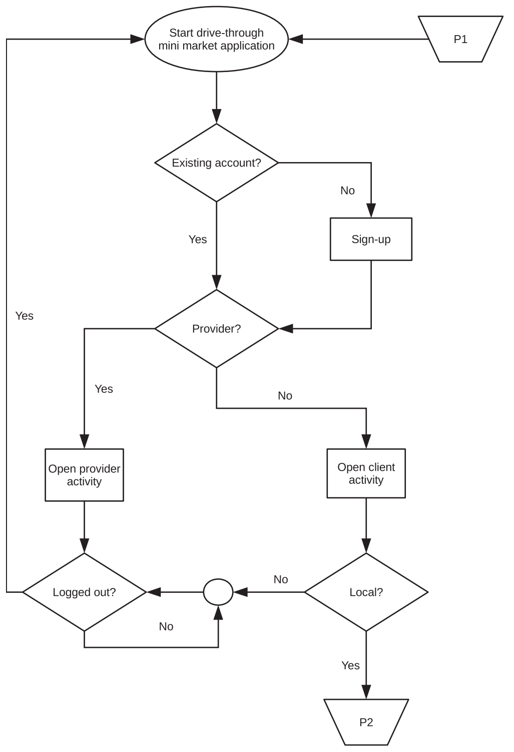
A provider must have a record in the database. If a user signs in as a provider, the application authenticates the email and password. If they match the information in the database, the user is granted access to the Provider activity. The screen of this activity consists of three buttons (see upper-left screen in Figure 2b). The first button (add a new product) allows the provider to add the name, number of items, price, product code and picture for a new product item. The second button (edit product) allows the provider to edit the information of any of the added product items. Finally, when the provider taps on the third button (view revenues) he/she can get information about the revenues of the minimarket.

The flowchart of the smart-phone application is illustrated in Figure 3a and 3b, where, P1 and P2 are off-page connectors, i.e., we start at the top of Figure 3a. When P2 is reached (at the bottom of Figure 3a), the flow is transferred to P2 at the top of Figure 3b. Similarly, when P1 is reached in Figure 3b, the flow is transferred to P1 at the top of Figure 3a. As it can be seen in Figure 3a, the application makes three decisions. The first one is to decide whether the user is a client or provider, and the second one is to decide whether the client is located nearby the minimarket or not. The application is able to determine if a user is still logged in (third decision), otherwise, it returns the first activity.

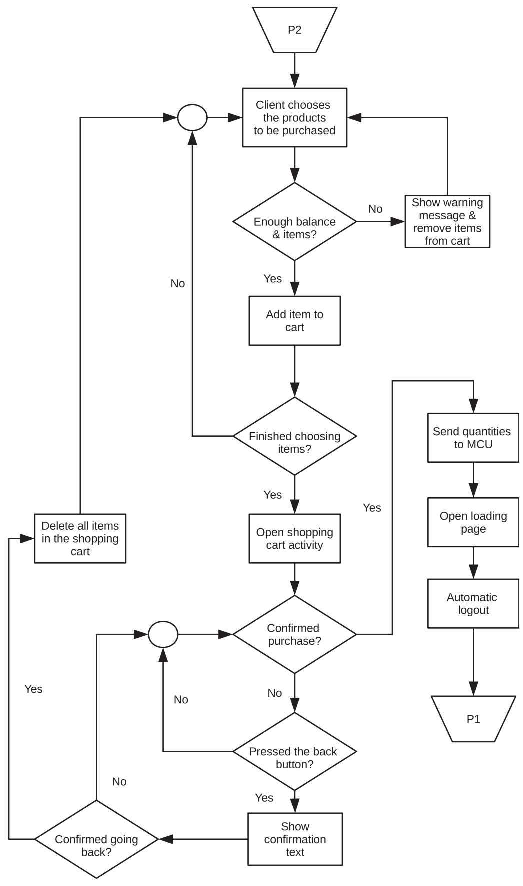
If the client is local (see Figure 3b), has enough balance and the minimarket has non-zero items, he/she can proceed with his/her shopping. Once the purchase is confirmed, the application directs the MCU to handle the order. It is worth mentioning that if the user has zero-added items, he/she can still proceed and confirm his/her purchase, but no commands will be sent to the microcontroller and the application goes back to the first activity (P1 in Figure 3b).

4.2. Database

A database was used in this project to store users' and products' information. Each user (client or provider) has an entry which contains an ID number, balance, email address and password. A client can enter this information remotely using a secure network connection. A provider, however, has a different entry which, contains an



(a) First flowchart.



(b) Second flowchart.

Figure 3 | Flowcharts of the Android application.

ID, email address and password. Unlike a client's entry, a provider's entry needs to be entered by the system administrator.

Furthermore, the database holds entries for products' information, specifically, each product has an entry which contains its name, picture, price and number of items. Also, the database stores information about the number of purchased items, expenditures per client, and total revenue of the minimarket, from which, analysis and prediction of revenue and expenditures per client can be made to adapt the supply with the demand of customers, to better fit customers' needs and maximize provider's profit.

The database used in this project was implemented using a real-time database (Firebase [13]). This database implementation is based on Nonstructured Query Language (NoSQL) cloud database, which uses a different data structure (e.g., key-value, wide column, graph or document) for storage and retrieval of data in contrast to the traditional tabular approach used in relational databases. The approach adopted by this real-time database makes it scalable and allows some operations to be executed faster.

In addition, the real-time database provides an Application Program Interface (API) that allows the smart-phone application data to be stored on a cloud. The database relies on REpresentational State Transfer (REST) web service architecture. Specifically, it has REST API that uses Hyper Text Transfer Protocol (HTTP) requests to GET, PUT, POST and DELETE data. This architecture uses less bandwidth and thus is more suitable for Internet usage. From a security view point, the real-time database used in this project uses Secure Sockets Layer (SSL) to establish secure and encrypted connections between the smart-phone application and the database.

Contrary to the traditional way of storing data in tables, the data in this real-time database is stored as *nodes*, each of which has a number of key-value pairs. Three nodes were used: Shopping-Cart node, User node and Products node. The Products node contains the information about each product, whereas the User node contains information about each user and is updated each time a new user registers in the system. The Shopping-Cart node contains information about the products ordered by the client and is automatically deleted when the user logs out. However, for each order made by a client, the date of the order and the amount of client's expenditure per product in that order is retained in the database.

Nested nodes can be used in the database, i.e., a node can have a set of sub-nodes. For example, the Products node has a sub-node that contains automatically-generated IDs of all products in the minimarket. Each ID sub-node has 8 sub-nodes that defines the picture, name, number of items, ordered price, ordered quantity, price, product code and product ID.

4.3. Time-Series Prediction

The minimarket employs machine-learning algorithms to predict customer expenditures. An ANN was used for that purpose. The

ANN was used as a prediction model due to its various advantages over other time-series prediction approaches. To illustrate, the architecture of ANN makes it non-linear, thus an ANN can predict non-linear time series. On contrary, classical time-series prediction approaches—e.g., Auto-Regressive Integrated Moving Average (ARIMA)—assume that time series is generated from linear processes, however, this might be incorrect assumption if the actual empirical time series (i.e., obtained from real system) is nonlinear, thus a prediction model based on ANN is more general. Furthermore, ANN is self-adaptive and is driven by the collected data sample which is used to train the prediction model, i.e., given a data sample (inputs and their corresponding outputs) it can be shown that ANN can approximate any continuous function to any desired accuracy, therefore ANNs are considered a universal approximators. Finally, ANN is immune to noisy data and can still provide accurate prediction of new unseen data even if the data sample that was used in training is noisy.

To illustrate the concept, we review the basic architecture of an ANN and show how time-series prediction fits into that, then provide an approach to improve the prediction capabilities. The basic information-processing element in an ANN is the artificial neuron, which can be defined as a set of connected links, each of which has a weight of its own. Say, e.g., that we have a set of inputs $\{x_{t-1}, x_{t-2}, \dots, x_{t-i}\}$, $i = 1, 2, \dots, \ell$, each of these inputs is associated with a weight. That is, we have a set of weights for the j^{th} neuron, $\{w_{ij}\}$, $j = 1, 2, \dots, m$. In addition to these inputs, we have a bias value, which can be represented as an input of "+1" with an associated weight of b_1. The inputs and the bias values are multiplied by their weights and summed, then fed into the so-called activation function ($\psi_1(.)$) to produce an output. Thus for the j^{th} neuron we have the following output (u_j):

$$u_j = b_1 + \sum_{i=1}^{\ell} w_{ij} x_{t-i} \tag{1}$$

Assuming that the input is a time series, we have three layers of neurons ($N^{(\ell-m-n)}$), and we need to estimate the output of the series at t from ℓ previous values, then we can write the estimated output ($\hat{x}_t$) as

$$\hat{x}_t = \Psi_3\left(b_3 + \sum_{k=1}^{n} w_k \Psi_2 \left(b_2 + \sum_{j=1}^{m} w_{jk} \Psi_1 \left(b_1 + \sum_{i=1}^{\ell} w_{ij} x_{t-i}\right)\right)\right) \tag{2}$$

It is common to use linear activation function in the last layer (i.e. $\psi_3(.)$) to avoid distortion in the predicted output [14]. In this project, the *rectifier* [15] activation function was used in the three layers for efficient computation.

The problem that ANN tries to solve is: given a set of inputs and their corresponding outputs, what are the weights and the bias values of the neurons that define the relationship between the input and the output? This can be formulated as an optimization problem with an appropriate objective function (loss function). Specifically, the mean squared error can represent an objective function. This can be expressed as

$$U = \frac{1}{n} \sum_{j=0}^{n-1} (x_{t+j} - \hat{x}_{t+j})^2 \tag{3}$$

where, x_{t+j} is the actual output at $t+j$, $\hat{x}_{t+j}$ is its corresponding estimated output, and n, ($n > 0$) is the number of samples (the set of inputs and their corresponding outputs). In the context of ANN, the process of determining the weights and bias values is usually referred to as *training* the network, and the samples used for that purpose are called training samples. The weights and bias values that minimize this function can be determined using an optimization algorithm (e.g., RMSprop algorithm [16] which was used in this project because of its fast convergence to the optimal value).

An alternative approach to time-series prediction is ARIMA model. To illustrate, a time series $\{x_t\}$ can be represented as

$$x_t - \alpha_1 x_{t-1} - \cdots - \alpha_p x_{t-p} = c + \epsilon_t + \theta_1 \varepsilon_{t-1} + \dots + \theta_q \varepsilon_{t-q} \tag{4}$$

where, α_i and θ_i are model's parameters, c is a constant and ϵ_{t-i} is the error term at $t-i$, $\alpha_0 = \theta_0 = 1$. Now, using the Back-Shift (or Lag Operator) Notation [17]), assuming a unit root $(1-B)$ for the polynomial $\left(1 - \sum_{i=1}^{p'} \alpha_i B^i\right)$, and factorizing, the time series can be expressed as

$$\left(1 - \sum_{i=1}^{p\prime} \alpha_i B^i\right) x_t = \left(1 + \sum_{i=1}^{q} \theta_i B^i\right) \epsilon_t + c \tag{5}$$

$$\left(1 - \sum_{i=1}^{p\prime} \alpha_i B^i\right) = \left(1 + \sum_{i=1}^{p\prime - d} \phi_i B^i\right) (1-B)^d \tag{6}$$

where ϕ_i is another model parameter obtained after factorization. After combining (5) and (6), and substituting ($p = p' - d$):

$$\left(1 - \sum_{i=1}^{p} \phi_i B^i\right) (1-B)^d x_t = \left(1 + \sum_{i=1}^{q} \theta_i B^i\right) \epsilon_t + c \tag{7}$$

Equation (7) defines an ARIMA (p,d,q) model for the time series, where, p is the order of the autoregressive part, d is the degree of first differencing involved – usually taken as $d = 1$ for practical purposes – and q is the order of the moving average part. It can be seen from (4) and (7) that the future value in a time series is assumed to be a *linear function* of some past observations and random errors. The autocorrelation function (ACF) and the partial autocorrelation function (PACF) of the sample data are usually used to identify the order of the ARIMA model [18] (for other methods, see [14]).

One way to improve the prediction capabilities of a model is to combine both *linear* and *nonlinear* models in a hybrid model. For example, combining ARIMA and ANN [14]. Particularly, if we let $y_t = (1-B)^d (x_t - c)$ and e_t be the residual error at t, then a two-stage model can be used as follows: first, an ARIMA model is used to generate the residual errors, second, an ANN model is used to capture the nonlinear and linear relationships in both the data and residual errors. Thus, the predicted value of y_t ($\hat{y}_t$) becomes

$$\hat{y}_t = \psi_3\left(b_3 + \sum_{k=1}^{n} w_k \psi_2 \left(b_2 + \sum_{j=1}^{m} w_{jk} \psi_1 \left(b_1 + \sum_{i=1}^{p} w_{ij} y_{t-i} + \sum_{i=p+1}^{p+q} w_{ij} e_{t+p-i}\right)\right)\right) \tag{8}$$

Since the minimarket system is able to store customer expenditures and their IDs, a time series of each customer expenditures on weekly basis can be constructed if the customer used to do his/her weekly shopping on specified day of the week. We slightly abuse the notation of the time-series that we used before and use (X) instead of (x), since this notation is common to denote random variables (used later in Section 6). Thus, we can express this time series as $X = \{X_0, X_1, \dots, X_{t-52}, \dots, X_{t-2}, X_{t-1}, X_t, \dots\}$, where t is the time in weeks. Thus, X_t is the amount of customer expenditures on week t, X_{t-1} is the amount of customer expenditures on week $t-1$, and X_{t-52} is the amount of customer expenditures on the corresponding week before one year of week t. Our aim was to predict the value of X_t given the values of X_{t-1}, X_{t-2}, and X_{t-52}. In the context of ANN, this maps into three inputs (also called features) and one output (also called label).

In order to achieve our time-series prediction aim, we constructed the ANN shown in Figure 4. This network has four inputs (three inputs and one bias value of "+1") represented by small unfilled circles in the figure. The network also has three layers of neurons; the first two have 64 neurons each, and the output layer has only one neurons. Neurons are represented as big circles and weights/bias values were removed from the figure to keep it simple. The small filled circles in the figure are used to indicate that there are many neurons (i.e., up to 64 neurons). The output of the network is the output of the last neuron (layer 3) in Figure 4.

The ANN was trained as follows (see Figure 5):

1. The data set (samples) was divided into training samples (80%) and validation samples (20%).
2. The training samples were divided into batches of size n selected at random.
3. Each batch is then used to train the network, i.e., solving for the weights/bias values that minimize (3). Then the model parameters (weights/bias values) were updated.
4. For the next batch, the previous model parameters were used as initial values, and the process is repeated for all batches in the training samples; this is called one epoch.
5. The process was repeated for a number of epochs. For each new epoch, the training samples were shuffled and new batches were selected at random from the training samples. Model parameters' values propagated to the next epoch.
6. For each number of completed epochs, the mean absolute error ($E = \sum_{j=1}^{k} |x_j - \hat{x}_j|$, where k is the number of training samples) was computed.
7. Also, for each number of completed epochs, the mean absolute error ($E = \sum_{j=1}^{l} |x_j - \hat{x}_j|$, where l is the number of validation samples) was computed and compared with the corresponding training samples' mean absolute error.
8. Training stops automatically if the mean absolute error is less than a predefined threshold value, this is called, early stop.

The number of layers and the number neurons are tunable. The values shown in Figure 4 were sufficient to predict customer expenditures with acceptable accuracy in this project.

5. HARDWARE DESIGN

The hardware design of the minimarket is divided into two parts: the actual structure of the minimarket, where the products are placed, and the electronic circuitry associated with that structure, which controls the operations of the minimarket.

5.1. Minimarket Structure

The structure of the minimarket is similar to that of a vending machine. It is a rectangular box made from foam-core boards with

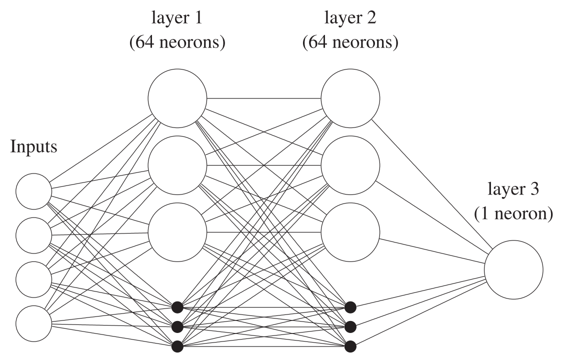


Figure 4 | Fully-connected artificial neural network (ANN).

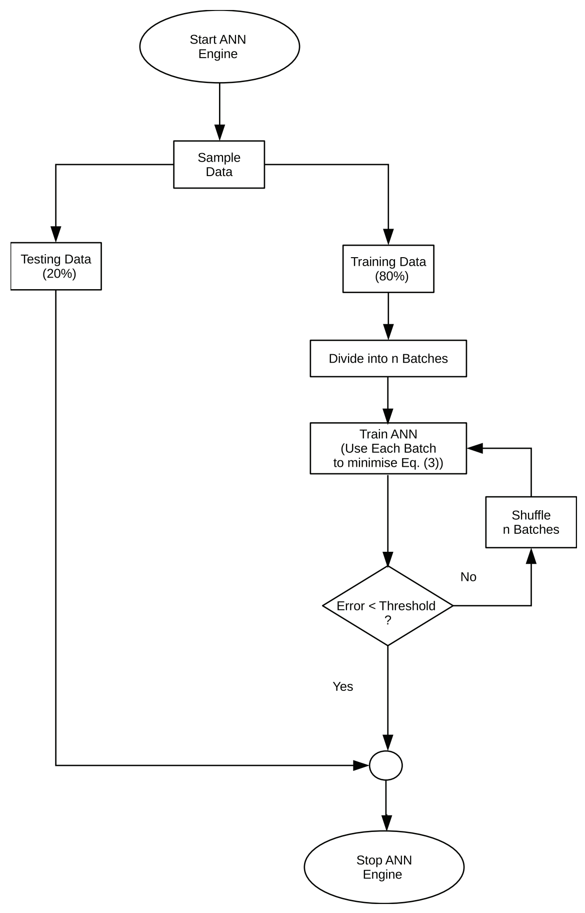


Figure 5 | Artificial neural network (ANN) flowchart.

six compartments organized in a 2 × 3 matrix. Each compartment is designed to hold one type of products and is equipped with a coil attached to the back end of the compartment. Product items are placed between the coil turns, i.e., they are interleaved with the wire rings of the coil. The front end of the compartment is open and allows the product to be dropped when the coil rotates. In addition, each compartment has servo motor which controls the number of coil rotations. This was carefully calibrated such that each rotation drops one item of a product. Figure 6 depicts the structure of the minimarket.

Product items dropped from the aforementioned structure are placed in a robotic shopping cart which uses a dedicated rail to transport customer's purchased items to his/her location (drive-through window).

5.2. Electronic Design

The servo motors in minimarket structure is electronically controlled by a MCU (NodeMCU[19]), which is connected to the user's smart-phone via Wi-Fi (IEEE 802.11) wireless connection. When a client places his/her order, the smart-phone application identifies the product IDs and the number of purchased items of each product then maps this information into row-column pair and number of motor rotations, respectively. Subsequently, the application sends this information to the MCU through its wireless connection.

Another MCU (Arduino Uno[20]) is placed on the shopping cart to control its motors and synchronize its operation with minimarket structure's operations. Particularly, two sensors were implemented; one is an Infra Red (IR) sensor connected to the NodeMCU microcontroller, the other is a Light-Dependent Resistor (LDR) sensor connected to the Arduino Uno microcontroller. The former informs the NodeMCU about the position of the shopping cart, i.e., whether it is at position A (at the minimarket structure) or at position B (at the customer's location). If the shopping cart is at position A, a Light Emitting Diode (LED) is deactivated to indicate that the product items can be dropped into the cart. Once all purchased items are dropped, the LED is activated to indicate that the cart is ready to move to position B. This LED is strategically position in a Line of Sight (LoS) with an on-board LDR sensor. Thus, when activated; the LED triggers the shopping cart to start moving to position B.

The distance between position A and position B was measured and stored in the memory of the on-board Arduino Uno MCU. Given constant preconfigured Direct Current (DC) motor speeds, the on-board MCU computes the time to reach position B, then activates the shopping-cart motors for this period of time.

Finally, when the shopping cart arrives to position B and the client lifts his/her purchased items, he/she can return the shopping cart to position A by pressing an on-board push button, which reverses the direction of rotation of the shopping-cart motors and activate them for the period of time to reach position A.

As it can be seen from the schematic diagram of the NodeMCU microcontroller circuitry (see Figure 7), the micro controller is connected to six servo motors (shown as MG995), each of these servo motors is attached to a unique compartment in the minimarket structure and is responsible for rotating the coil in that compartment.

6. ANALYTICAL MODEL

In order to analyze the performance of the drive-through minimarket system, we created a mathematical model to capture the dynamics of the system. Particularly, we modeled the drive-through minimarket system as a single-queue single-server queueing station with random customer arrivals and random service times. Figure 8 illustrates the queueing model of the system. Given a customer population, customers arrive randomly with an average arrival rate (λ) and get served by a single server. Service time per each customer is random and depends on the number of items he/she orders. We

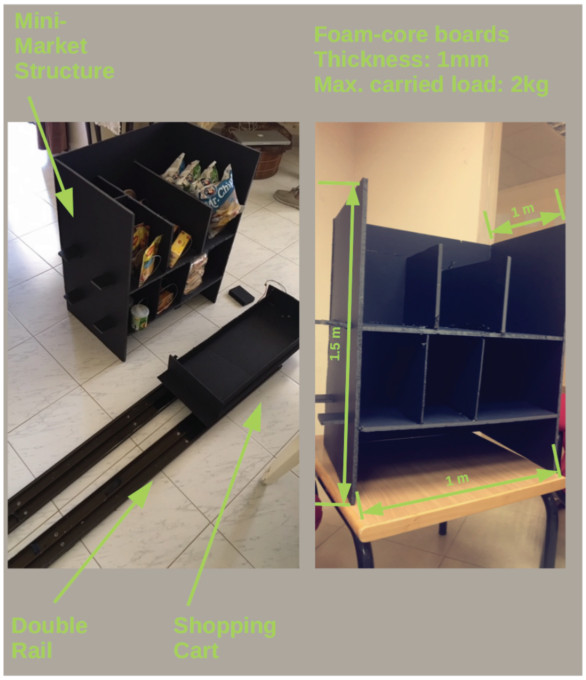


Figure 6 Hardware design.

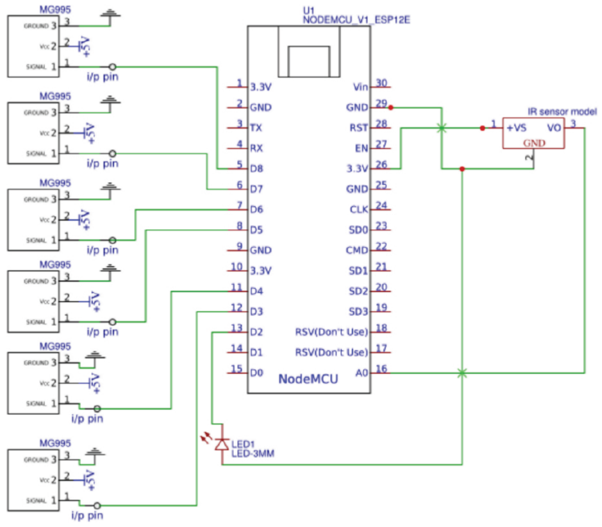


Figure 7 Microcontroller unit (MCU) circuit diagram.

denote by τ the average service time. However, if customers arrive while a customer is being served, they are placed in a queue, which has a First-In-First-Out (FIFO) queueing discipline.

The service time is defined as the time period from the instant a customer makes an order from his/her smart-phone, to the instant he/she receives the order. The average service time was determined empirically, by first measuring the time it takes to serve a customer when he/she orders all the items of only one type of product. The experiment was repeated 30 times for each type of product and the average service time was reported, then divided by the total number items of that product.

We illustrate the process with the aid of Figure 9, where the y-axis shows the average service time of 30 experiments in the case when a customer orders all four items of one type of product. The x-axis, on the other hand, shows the weight in grams of one item of that type of product. We noticed that the values are close to each others. However, the maximum is approximately 20.5 seconds when 4 items of the maximum-weight product are ordered. Thus, we approximated the service time required to serve one item to be 5 seconds.

We denote the service time by the random variable S. Let the total number of items of product i be L_i, where, $i = 0, 1, \dots, k$, and since we have six different product types, then $k = 5$. In addition, because we used the same number of items for each type of product, we let $L_i \equiv L$, where, L is constant ($L = 4$). We denote by X_i the number of purchased items of product i, thus, X_i is a set of independent and identically distributed (i.i.d.) uniform random variables. Note that, $X_i = \{0, 1, 2, \dots, L\}$. Finally, we denote by T_s the service time—in seconds—required to serve one item of product, thus in our case $T_s = 5$ seconds. Thus, the average service time τ can be written as

$$\tau = E[S] = E\left[\sum_{i=0}^{k} T_s X_i\right] = T_s \sum_{i=0}^{k} E[X_i] \tag{9}$$

which can be simplified to

$$\tau = E[S] = T_s \sum_{i=0}^{k} E[X] = (k+1) T_s E[X]. \tag{10}$$

where, $E[.]$ denotes the expected value and $X_i \equiv X$, since all random variables are i.i.d. Now, the average number of purchased items ($E[X]$) can be evaluated as $E[X] = 0(1/5) + 1(1/5) + 2(1/5) + 3(1/5) + 4(1/5) = 2$ items. Thus, substituting in (10) we obtain $\tau = E[S] = 60$ seconds. That is, theoretical calculations show that the average service time of the drive-through minimarket is approximately 1 minute.

Now, we model a single minimarket station first as simple $M/G/1$ queueing station. That is, we assume that customers arrive according to a Poisson random process with an average arrival rate of λ customers per second and get served according to a general service-time distribution with an average service time of $\tau = 60$ seconds.

The response time of the system is defined as the time period from the instant a customer enters the system to the instant a customer departs from the system, i.e., the sum of the queueing time and the service time. The average system response time was used as a quantified measure to evaluate the performance of the minimarket station. This can be determined by first computing the average number of customers in the system using the Pollaczek–Khintchine mean value formula [21]:

$$E[N] = \rho + \frac{\rho^2(1 + c_S^2)}{2(1-\rho)}, \tag{11}$$

where N is the number of customers in the system (queue and in service), $0 \leqslant \rho = \lambda\tau < 1$ is the traffic intensity or server utilization, which can be interpreted as the average number of customers in service (i.e., in the server), and c_S is the coefficient of variation of the service time and is defined ($c_S = \sigma_S / E[S]$), where σ_S is the standard deviation of the service time. The coefficient of variation of a random variable is a measure of dispersion of the probability density function of the random variable, its value is one for an exponential random variable (i.e., completely random) and zero for a deterministic random variable. Other random variables take values in between. Using Little's formula and assuming a FIFO queueing discipline, we can write the average response time of the system as

$$E[T_{\text{FIFO}}] = \frac{E[N]}{\lambda} = E[S] + \frac{\rho E[S](1 + c_S^2)}{2(1-\rho)}, \tag{12}$$

where T_{FIFO} is the system response time under a FIFO queueing discipline and $\lambda > 0$. The expression of $E[T_{\text{FIFO}}]$ in (12) can be simplified to

$$E[T_{\text{FIFO}}] = \frac{E[S]}{1-\rho}\left(1 - \frac{\rho(1 - c_S^2)}{2}\right). \tag{13}$$

Thus, given the average arrival rate of customers, the average service time and the standard deviation of the service time, we can determine the average response time in the minimarket using (13).

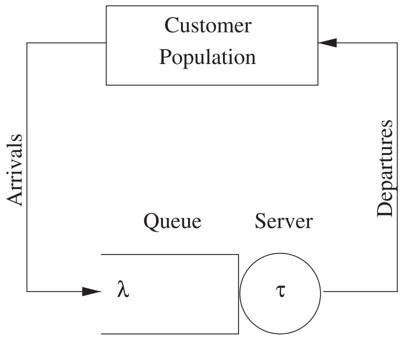


Figure 8 | Queueing model of the minimarket prototype.

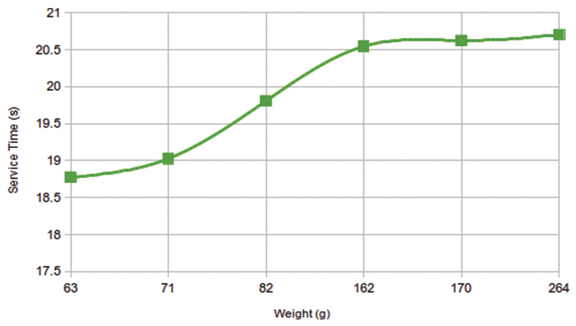


Figure 9 | Service time for each type of product when all four items of that product is ordered.

In addition to random customer arrival, we studied the random arrivals of a random batch/bulk of customers. That is, we let H be a random variable that denotes the size of batch, then the traffic intensity becomes $\rho = \lambda E[H]E[S]$. We state here without proof the average customer's queueing time for $M^{[H]}/G/1$ as [22]:

$$E[W_{\text{FIFO}}] = \frac{\lambda E[H]E[S^2]}{2(1-\rho)} + \frac{E[H(H-1)]E[S]}{2E[H](1-\rho)}, \qquad (14)$$

Consequently, the average customer's response time can be expressed as

$$E[T_{\text{FIFO}}] = E[W_{\text{FIFO}}] + E[S]. \qquad (15)$$

The first term of the right-hand side in (14) is a manipulation of the last term of the right-hand side in (12). The second term of the right-hand size in (14) accounts for the extra waiting time experienced by customers that are not the first in the batch. A proof of (14) is provided in [23].

7. RESULTS AND DISCUSSION

This section is divided into two parts. The first part discusses the results of the performance evaluation of the minimarket system, while the second part discusses the results of the ANN-based time-series prediction for an example data set.

7.1. Performance Evaluation

We cross-validate the performance measure obtained analytically with that obtained using a simulation model. We show in this section that both results agree and can be used to study the performance of the minimarket under various levels of traffic intensity and various levels of variations in the service time.

A simulation model was created using the R Language [24] to simulate the system presented in Figure 8. Customers arrived according to a Poisson random process (i.e., exponential inter-arrival times), and several values for the traffic intensity were used: $\{0.25, 0.5, 0.75, 0.85, 0.9, 0.95\}$ with an average service time of 60 seconds. The standard deviation of the service time was set to zero, i.e., a deterministic service time (i.e., M/D/1). Ten simulation runs were performed for each traffic-intensity value each of which has 10^8 simulation time steps. The average and standard deviation of the response time for each value of the traffic intensity were reported and plotted in Figure 10 along with the corresponding theoretical average response time obtained using (13). It can be seen from Figure 10 that at low traffic intensity (25%), the average response time is approximately 60 seconds, however, as the traffic intensity increases, the average response time increases. It was reported that at worst case (95% traffic intensity), if customers arrive at the minimarket every 63 seconds on average, then the average response time for each customer is approximately 10 minutes.

The Complementary Cumulative Distribution Function (CCDF) for the number of customers in the system (N) is plotted in Figure 11 for traffic intensities near the value of 95%. To illustrate the use of the curves in Figure 11, we see that the probability of having more than 10 customers in the system when the traffic intensity is 90% is approximately 10% (i.e., $P(N > 10) \approx 0.1$), while the probability of having more than 10 customers in the system when the traffic intensity is 95% is approximately 30% ($P(N > 10) \approx 0.3$).

We also studied the effect of variations in service time on the average response time, specifically, we let the service time have a Gaussian distribution and used a set of three values for the standard deviation, $\{0, 10, 20\}$, to obtain a trend on how the average response time varies with the increase of standard deviation at worst case (95% traffic intensity). For each value of the standard deviation values we have run the simulation 100 times, 10^8 time steps each. Figure 12 summarizes the result of these experiments. Obviously, high variation in service times has an adverse effect on system performance, this can be seen from the increase in the average response time as the variance of service time increases.

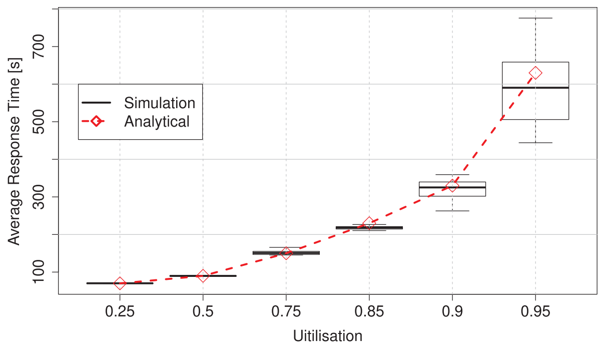


Figure 10 M/D/1: effect of variations in traffic intensity.

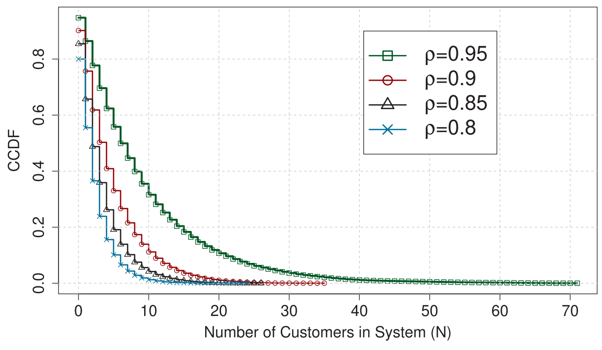


Figure 11 Complementary cumulative distribution function (CDF).

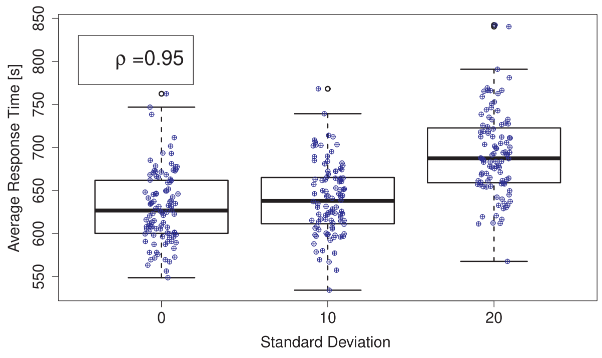


Figure 12 M/G/1: effect of variations in service time.

Finally, we examined the case of batch customer arrivals. Specifically, we considered a geometrically-distributed batch size with an average size $E[H] = \alpha$ and a parameter $1/\alpha$. That is, $P[\text{"}a\ batch\ contains\ h\ customers''] = (1 - 1/\alpha)^{h-1}(1/\alpha)$, where, $h = \{1, 2, \ldots\}$. Figure 13 illustrates the average response time per customer for various traffic intensities over a range of average batch sizes. The case of very heavy traffic (95%) can be considered as an upper bound. The figure shows that for heavy traffic intensity (90%) and average batch size of four customers, the average customer response time is 2000 seconds (≈ 34 minutes).

7.2. Prediction of Customer Expenditures

To illustrate the prediction capability of the minimarket, we relied on the average annual expenditure of Jordanian households on groups of commodities and services by size of household (4-3 members) in Jordanian Dinar (JOD), as obtained from the Department of Statistics in Jordan [2] for the year of 2017. Table 1 shows the average weekly expenditure of middle-size (4-3 members) Jordanian households for a group of commodities commonly consumed by Jordanian families. From Table 1 we can estimate the average sum of expenditures for the given six types of products by approximately 26.56 JOD/week.

We mimicked an arbitrary customer expenditures behavior by creating a time series that has an average value of 26.56 JOD/week and several periodic trends during a total period of 5 years. The minimarket system is able to extract such information for each customer from the database. This time series is fed into the ANN shown in Figure 4 to predict customer expenditures.

The ANN was implemented using the Python deep learning library, Keras [25]. An API was used to integrated the library with the R-Language programming environment. The network was trained according to the steps mentioned in Section 4.3. First, the number of epochs that produce sufficient accuracy was determined, then training samples were randomly grouped into batches. Each batch has a size of $n = 32$ samples. The network was trained using different number of epochs and the mean absolute error for the training and validation data were computed and plotted *versus* the number of epochs. It can be seen in Figure 14 that for epoch values above 100, the error diminishes to small values. Therefore, we trained the ANN using 100 epochs. Second, we used the validation samples (20% of the data set) to validate the predicted values of customer expenditures. The result is shown in Figure 15. We notice that the ANN is capable of predicting weekly customer expenditures with relatively high accuracy.

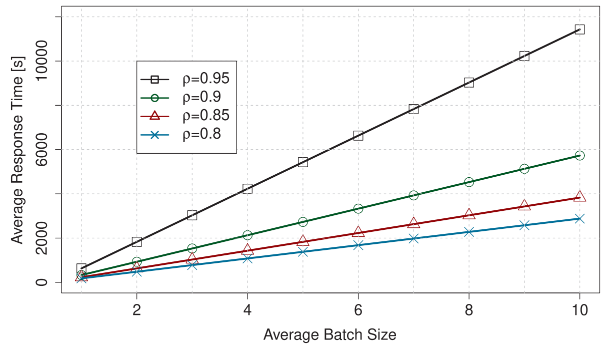


Figure 13 M$^{[H]}$/D/1: effect of random batch arrivals.

Table 1 Average weekly expenditure of Jordanian households of size 4-3 members on group of commodities.

Product	Expenditure (JOD/week)
Cereals and bread	6.83
Sea food	0.9
Sugar	4.36
Soft drinks	3.38
Diary and cheese	8.32

8. CONCLUSION

One of the challenges posed by the growth of population in smart cities is managing supermarkets and shopping centers. In this paper, we discussed the design and implementation of a smart minimarket prototype, which makes the shopping experience more convenient for both, customers and providers. The minimarket provides intelligent services such as the prediction of customer weekly expenditures based on ANN, automatic nearby customer location detection, and automatic delivery of purchased products using a robotic shopping cart. Both customers and service providers can access the minimarket remotely with different privileges. The deployment of such relatively cheap smart minimarkets in smart cities will make the shopping experience more enjoyable and smooth for customers. Furthermore, it gives the service

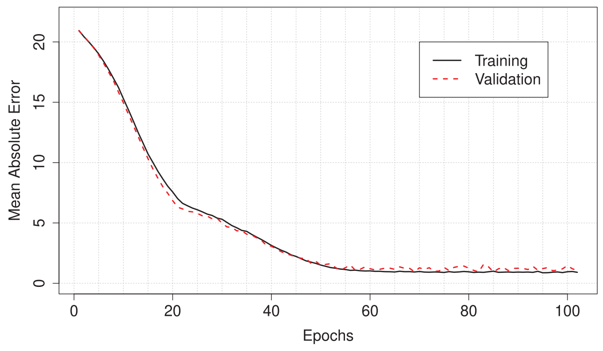


Figure 14 Relationship between accuracy and number of epochs.

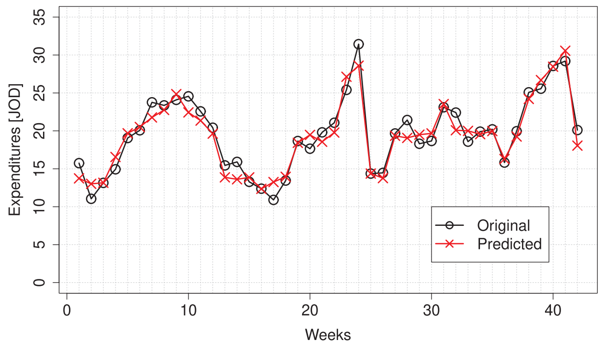


Figure 15 Weekly customer expenditures: predicted using validation samples.

provider the chance to adapt quickly to customer needs, especially in crowded and busy cities where the life pace is fast.

As part of future work, a number of minimarket models can be implemented in a more realistic environment at the shopping malls in the capital city of Amman, where customers' data can be collected on a large scale, and service satisfaction of customers/providers can be measured and used to fine tune motor speeds to increase/decrease the service times, and thus improve the response time of the system.

REFERENCES

[1] T.-h. Kim, C. Ramos, S. Mohammed, Smart city and iot, Future Gener. Comput. Syst. 76 (2017), 159–162.

[2] Department of statistics in Jordan, http://dosweb.dos.gov.jo/

[3] T. Kanda, M. Shiomi, Z. Miyashita, H. Ishiguro, N. Hagita, A communication robot in a shopping mall, IEEE Trans. Robot. 26 (2010), 897–913.

[4] H.-J. Boehme, C. Schroeter, S. Mueller, A. Koenig, C. Martin, M. Merten, A. Bley, Shopbot: progress in developing an interactive mobile shopping assistant for everyday use, in 2008 IEEE International Conference on Systems, Man and Cybernetics, Singapore, 2008, pp. 3471–3478.

[5] M. Mathankumar, N. Sugandhi, A low cost smart shopping facilitator for visually impaired, in 2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Mysore, India, 2013, pp. 1088–1092.

[6] M. Mathankumar, T. Kavitha, Design and implementation of smart supermarket system for vision impaired, Int. J. Eng. Technol. 5 (2013), 215–219.

[7] V. Maike, S. Buchdid, M.C. Baranauskas, A smart supermarket must be for all: a case study including the visually impaired, in XV Brazilian Symposium on Human Factors in Computing Systems (IHC'16), Nice, France, 2016.

[8] V. Lehdonvirta, H. Soma, H. Ito, T. Yamabe, H. Kimura, T. Nakajima, Ubipay: minimizing transaction costs with smart mobile payments, in Proceedings of the 6th International Conference on Mobile Technology, Application & Systems, Mobility '09, ACM, New York, NY, USA, 2009, pp. 1:1–1:7.

[9] S. Vicini, A. Sanna, S. Bellini, A living lab for internet of things vending machines, in International Conference on the Impace of Virtual, Remote, and Real Logistics Labs, Springer, Berlin, Heidelberg, Germany, 2012, pp. 35–43.

[10] R. Chen, L. Peng, Y. Qin, Supermarket shopping guide system based on internet of things, in IET International Conference on Wireless Sensor Network 2010 (IET-WSN 2010), Beijing, China, 2010, pp. 17–20.

[11] K. Kim, D. Park, H. Bang, G. Hong, S. Jin, Smart coffee vending machine using sensor and actuator networks, in 2014 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 2014, pp. 71–72.

[12] B. Bostanci, H. Hagras, J. Dooley, A neuro fuzzy embedded agent approach towards the development of an intelligent refrigerator, in 2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Hyderabad, India, 2013, pp. 1–8.

[13] Firebase real-time database, https://firebase.google.com/products/realtime-database/

[14] M. Khashei, M. Bijari, An artificial neural network (p, d, q) model for timeseries forecasting, Expert Syst. Appl. 37 (2010), 479–489.

[15] Rectifier (neural networks), https://en.wikipedia.org/wiki/Rectifier_(neural_networks)

[16] T. Tieleman, G. Hinton, Lecture 6.5—RmsProp: divide the gradient by a running average of its recent magnitude, COURSERA Neural Netw. Mach. Learn. 4 (2012), 26–31. http://www.cs.toronto.edu

[17] R.J. Hyndman, G. Athanasopoulos, Forecasting: Principles and Practice, OTexts, 2018. https://otexts.com/fpp2/

[18] G.E. Box, G.M. Jenkins, G.C. Reinsel, G.M. Ljung, Time Series Analysis: Forecasting and Control, John Wiley & Sons, Hoboken, NJ, USA, 2015.

[19] NodeMCU: an open-source firmware and development kit, http://www.nodemcu.com/index_en.html

[20] Arduino uno, https://www.arduino.cc/

[21] H. Kobayashi, B. Mark, System Modeling and Analysis: Foundations of System Performance Evaluation, Pearson Prentice Hall, Upper Saddle River, NJ, USA, 2009.

[22] B.R. Haverkort, Performance of Computer Communication Systems, John Wiley & Sons Ltd, Hoboken, NJ, USA, 1998.

[23] N. Kawasaki, H. Takagi, Y. Takahashi, S.-J. Hong, T. Hasegawa, Waiting time analysis of Mˆ X/G/1 queues with/without vacations under random order of service discipline, J. Oper. Res. Soc. Japan. 43 (2000), 455–468.

[24] R Core Team, R: a Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2019.

[25] F. Chollet, *et al.*, Keras, 2015. https://github.com/fchollet/keras