A Cross Platform Contact Tracing Mobile Application for COVID-19 Infections using Deep Learning

Josephat Kalezhi[1]
Department of Computer Engineering
The Copperbelt University
Kitwe, Zambia

Christopher Chembe[3]
Department of Computer Science
ZCAS University
Lusaka, Zambia

Francis Lungo[5]
School of Social Sciences
Mulungushi University
Kabwe, Zambia

Mathews Chibuluma[2]
Department of Information Technology/Systems
The Copperbelt University
Kitwe, Zambia

Victoria Chama[4]
Department of Computer Science and Information Technology
Mulungushi University
Kabwe, Zambia

Douglas Kunda[6]
Department of Computer Science
ZCAS University
Lusaka, Zambia

***Abstract*—The COVID-19 pandemic has remained a global health crisis following the declaration by the World Health Organization. As a result, a number of mechanisms to contain the pandemic have been devised. Popular among these are contact tracing to identify contacts and carry out tests on them in order to minimize the spread of the coronavirus. However, manual contact tracing is tedious and time consuming. Therefore, contact tracing based on mobile applications have been proposed in literature. In this paper, a cross platform contact tracing mobile application that uses deep neural networks to determine contacts in proximity is presented. The application uses Bluetooth Low Energy technologies to detect closeness to a Covid-19 positive case. The deep learning model has been evaluated against analytic models and machine learning models. The proposed deep learning model performed better than analytic and traditional machine learning models during testing.**

***Keywords*—Contact tracing mobile application; coronavirus; COVID-19; deep neural networks**

I. Introduction

In March 2020, the coronavirus disease (COVID-19) was declared a pandemic by the World Health Organization [1]. Since then, relentless efforts were put in place by several nations to understand the virus and how to contain the pandemic. One of the promising approaches is digital contact tracing [2]. Contact tracing has been used to follow the pattern of networks for an individual or population infected by an infectious disease. In the past, contact tracing has been used to combat sexually transmitted diseases, severe acute respiratory syndrome (SARS) and other invading pathogens [3]. Traditionally, there have been various contact tracing models including Individual-based simulation models; Pair approximation models; Models based on branching processes; and Phenomenological approaches [4]. These come with various challenges such as the inability for stochastic simulation-based models to be analysed analytically. Other challenges are associated with contact structure itself, backward- and forward tracing, identification of Super-spreaders, endemic equilibrium and efforts required for contact tracing [4]. The effectiveness of various contact tracing mechanisms has been presented by Klinkenberg et al [5].

In recent years, digital contract tracing has been championed to supplement the deficiencies introduced by traditional contact tracing. For example, a mobile contact tracing application was developed to trace and monitor Ebola epidemic in Northern Sierra Leone and proved to be effective [6]. Similarly, Sacks et al. developed a smartphone based mHealth application using CommCare and business intelligence software Tableau to assist in contact tracing of Ebola epidemic in Guinea [7]. Swanson et al. [8] gives details on the performance of contact tracing in Liberia during the 2014 to 2015 epidemic. Other uses of mobile phones in contact tracing have been used in tracing the spread of Tuberculosis (TB) [9]. Furthermore, the outbreak of COVID-19 and subsequent declaration as pandemic has seen a proliferation of mobile applications aimed at contact tracing to help combat COVID-19 [10] [11]. These applications have proved effective in tracing contacts in order to contain the pandemic [2].

In early days of COVID-19 pandemic, Singapore developed a COVID-19 contact tracing mobile app called "TraceTogether" [12]. This app uses Bluetooth technology to facilitate contact tracing. The app further uses the received signal strength indicator (RSSI) values detected from other mobile devices for distance estimation. The detected RSSI values are then compared against calibrated RSSI values to determine distance between mobile devices. The app notifies users when they are exposed to COVID-19 and are in close contact to other users who are using the same app. It also allows users to access their COVID-19 health status. The app uses a BlueTrace protocol to preserve privacy. The reference

implementation of the BlueTrace is referred to as OpenTrace [13]. Through the Ministry of Health, the app provides guidelines on how to avoid getting infected. In case one tests positive for COVID-19, the data is then shared with Ministry of Health. The Bluetooth data is kept on the phone no longer than 25 days. Despite being useful, the application has been criticised for the potential in being exploited in undertaking criminal investigations by the police.

Similarly, a National Health Service COVID-19 app was developed in England and Wales [14]. The app gave an option to users to enable contact tracing. It was reported that the effectiveness of the app towards reducing infections was dependent on the number of users. Bluetooth RSSI values were used to estimate distance between two close devices. Several RSSI values were taken and then used to determine the distance. Another contact tracing app called Immuni was developed in Italy by the Ministries of Health and Technological Innovation in 2020 [15]. The app used Bluetooth technology to facilitate contact tracing. The distinguishing functionality of the app was the absence of a centralized database to manage contact tracing. For users willing to use the app, they would download the data to support contact tracing in their smartphones and a decision made locally in case the users were exposed. Privacy concerns were also addressed in the development of the app in line with the national laws.

A contact tracing app called "Radar Covid" was developed in Spain in 2020 [16]. Users of the app received notifications when they were in close contact with a COVID-19 positive person. The app used Bluetooth low energy technology to detect if the user is in close proximity to a positive case. The app was voluntary and addressed the privacy and security concerns of users. When the user test positive, the user is presented with an anonymous code that can be entered into the app voluntarily. This in turn facilitated the notification of other users in case they had been in contact with a positive case. Despite being useful the app had a vulnerability that allowed attackers to use fake identities.

In 2020 Apple and Google joined forces to develop application programming interfaces (APIs) to facilitate contact tracing [17]. The technology adopted was Bluetooth owing to its availability in virtually every mobile device. The APIs are used to detect contacts typically within two meters for a period exceeding 15 minutes [18]. As pointed out in [17], the APIs have addressed privacy concerns by preventing access to user profiles by health authorities.

Other technology-based contact tracing mechanisms to fight COVID-19 has been employed previously. For example, a combination of machine learning classification algorithm and data obtained from Wi-Fi signals from users was proposed to determine when two users sharing the same physical space can inform exposure [19]. In [20], a framework based on IoT was proposed for contact tracing. They incorporated symptom-based detection ignored in other tracing models to confirm COVID-19 cases. The work proposed by Sahraoui et al used online social network to trace COVID-19 infections [21].

Despite the many works presented on contact tracing, there is more to be done for digital contact tracing to be appreciated in future. One challenging issue is privacy and protection of user data. In this paper, we present a cross platform contact tracing application that uses Bluetooth Low Energy Generic Attribute Profile (GATT) framework and deep neural network to determine and inform users of exposure to COVID-19. GATT framework is used to mitigate the many concerns regarding data privacy and protection. The data exchanged between Bluetooth devices embedded with GATT framework is encapsulated thereby authorizing only intended recipient. The deep neural network is applied to predict the distance between communicating devices. Using GATT framework to encapsulate packets exchanged between devices and deep neural network to predict distance has not been presented in literature. Hence the proposed approach presents novelty and contribution of this paper.

The rest of the paper is structured as follows. In Section II we review the literature related to this work. Section III presents the Received Signal Strength Indicator (RSSI) obtained from Bluetooth Low Energy (BLE) devices. In Section IV, a logarithmic distance path loss model applied in this work is presented. A decision tree is presented in Section V. The proposed deep neural network model that uses the RSSI is reported in Section VI. A comparison of performances of models is presented in Section VII. Section VIII reports the development of a cross-platform contact-tracing mobile application. The conclusion appears in Section IX.

II. Related Work

In order to aid with the digital contact-tracing process, a number of mobile applications have been developed worldwide [11]. These applications have proved effective in tracing contacts in order to contain the pandemic [2]. However, despite being useful a number of challenges still remain [22]. These include security and privacy concerns by users sharing the data, transparency, the effectiveness of the tracing application, social and cultural issues, legal and ethical issues and many more [23]. Megnin-Viggars et al. [24] identifies other barriers and factors to engaging in contact tracing during an infectious pandemic such as COVID-19. To mitigate some of the challenges and barriers to digital contact tracing, researchers have proposed various solutions. For instance, blockchain technology has been proposed to preserve privacy during contact tracing for COVID-19 pandemic to gain trust by users [25][26][27]. According to [26], it was reported that blockchain technology was able to detect unknown cases of COVID-19. The application was also capable of enabling individuals to use the mobile application to predict the probabilities of being infected. The study paved way for the use of blockchain technology to contain the spread of the epidemic as well as early detection of unknown infections.

The works in [28], proposed a smart contact tracing mobile application that uses Bluetooth Low Energy (BLE) and machine learning techniques. The application determined whether the user was at risk or not depending on whom they came into contact with. An analytic proximity estimation model based on RSSI was particularly used to determine the distance between two devices. Five machine learning classifiers were considered in the estimation of distance

between devices. These were Support Vector Machine, Decision Tree, Naïve Bayes, Linear Discriminant Analysis and K-Nearest Neighbors. It was reported that the Decision Tree classifier yielded the best accuracy compared to other classifiers.

In [29], authors presented a contact tracing application for wearable devices that employs machine learning. The application used BLE for distance estimation whereas machine learning was applied to categorize the risk of possible exposure. Additionally, an appropriate signature protocol was used to guarantee infected user anonymity. The authors studied four supervised-learning classifiers namely Decision Tree, Linear Discriminant Analysis, Naïve Bayes and K-Nearest Neighbours. It was reported that the classifiers performed well and yielded good precision and recall values. The Decision Tree classifier was reported to yield the best performance in terms of precision, recall, accuracy among others.

In [30] authors proposed a BLE application that monitors location patterns of old people indoors. The system relied on RSSI to estimate the positions of these elderly people. To achieve this, BLE beacons were either attached to a person's clothes or worn on wrists. Further the users could also place the beacons in their pockets. The beacons were periodically sending broadcasts. These broadcasts were detected by a number of BLE enabled Raspberry Pi devices that were stationed at fixed known locations. The broadcasts carried the RSSI among others. Each Raspberry Pi then relayed the received data to a server for additional processing. The server ran a machine-learning classifier to determine the location of the person. A path-loss model was used for estimation of distance from RSSI. Further a number of classifiers were used and these are Naïve Bayes, Random Forest, BayesNet, Sequential Minimal Optimization and J48. In overall the performances of classifiers were good for indoor localization.

A dependence of RSSIs on distance for iOS and Android mobile devices was reported in [31]. According to [31], the iOS device was used in that study, the RSSI reached an asymptotic value (where the RSSI appears not to change) earlier than the Android device. It was further reported that the RSSI detected on the Android phone used decreased gradually compared to the iOS device. In terms of temporal RSSI variations, the Android device exhibited more variations compared to iOS. Therefore, according to the study [31], the dependence of RSSI on distance varies between iOS and Android devices.

In a related work [32], an evaluation of a contact tracing mobile application in Norway based on the Google Apple Exposure Notification (GAEN) system was reported. The authors observed variations in Bluetooth attenuation levels, when alerts are generated among others between iOS and Android mobile devices. The Android device was reported to exhibit high variabilities compared to iOS devices. In another related study [33], a number of data mining models have been applied to reveal hidden patterns in patients' data in Zambia. The models include J48 decision classifier, Naïve Bayes, Multilayer Perceptron among others. These models were shown to exhibit good performance compared to baseline results. The COVID-19 cases in Zambia are reported by the Ministry of Health through the Zambia National Public Institute [34]. The contact tracing process used in Zambia prior to this work was manual and time consuming.

It is clear from the works reported above that contact-tracing apps have found their application in mitigating the spread of COVID-19 pandemic. In terms of distance estimation, a number of models have been reported in the literature. These include simplistic path loss models as well as several machine learning models. Nevertheless, no work has considered GATT framework to encapsulate user data in order to mitigate the concerns regarding data privacy and protection. Furthermore, no work has used deep neural network to predict the distance between communicating devices. Thus, in this paper we propose deep learning methods to estimate distance between the devices and the GATT framework to encapsulate the data between communicating devices in order to secure user data. The proposed deep learning model is compared against some models reported in the literature. The deep learning model is further converted to models suitable for use in a contact tracing mobile application. We further developed a cross-platform contact-tracing app for use in Zambia that incorporates these models.

III. Received Signal Strength Indicator

The Received Signal Strength Indicator (RSSI) is a measure of the signal strength detected by a receiving device [35]. The RSSI is manufacturer dependent and can vary even at a fixed separation between the sending and receiving Bluetooth Low Energy (BLE) devices [35]. Factors such as multipath propagation, scattering, shadowing, refraction among others affect BLE signals and this has implications for applications that rely on BLE such as those for contact tracing [36]. The RSSI can be reasonably mapped to a distance from the sending device on iOS devices. However, the quality of RSSI on Android devices varies significantly due to presence of several chip manufacturers [35]. Nevertheless, a variation of this quantity with distance from a sending device can be used as a distance measure [35].

In this work, we relied on BLE RSSI to determine distance between two devices. Measurements were initially taken at several separations between two devices. It was observed that even when the distance was fixed, the RSSI reading was changing as reported in [35]. Measurements were taken at several distances between two devices beginning with an initial separation of 4 m to a final separation of 0.1 m in steps of 0.1m. At each distance, several RSSI measurements were repeated (twenty in this case) and the mean determined. Fig. 1 shows the dependence of the mean RSSI in decibels on actual distance separating two devices in meters. The signal originated from an iOS device (iPhone 7 Plus) and the RSSI was detected on an Android device (Infinix Smart HD). As can be seen from Fig. 1, the mean RSSI follows a particular trend despite the fluctuations as the distance between devices changes.

Similar measurements were undertaken for signals originating from an iOS device (iPhone 11) and detected on another iOS device (iPhone 7 Plus). Measurements were also undertaken for signals originating from an Android device and detected on an iOS device (iPhone 7 Plus). It was observed that

RSSI values decrease in general as the distance increased in all cases. However, variations of RSSI on distance were different for each case. The calibration results of mean RSSI at 2m separation for various devices has been reported for TraceTogether contact tracing app as used in OpenTrace [13]. The results shown in Fig. 1 at a separation distance of 2 m are in agreement with OpenTrace calibration results.

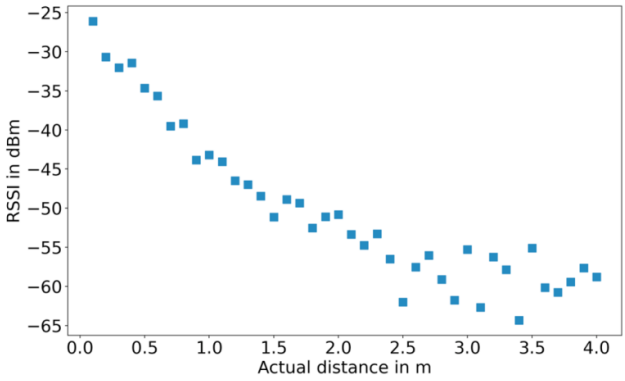

Fig. 1. Dependence of Mean RSSI in dBm on Actual Distance between Two Devices in Metres. The Signal Originated from an iOS Device (iPhone 7 Plus) and Detected on Android Device (Infinix Smart HD).

IV. Logarithmic Distance Path Loss Model

One of the simplest models relating RSSI to distance is the logarithmic distance path loss model [37]. This model is expressed as:

$$RSSI = - 10n \log(d/d_0) + A + X_\sigma \quad (1)$$

where n is an environment dependent path loss parameter, d is the distance from the sending device to the receiving device, d_0 is a distance where the RSSI takes the value A. X_σ is a random variable that follows a Gaussian-distribution. X_σ has zero mean and a variance of σ^2.

Taking the mean of equation (1), one obtains

$$RSSI_{mean} = -10n \log(d/d_0) + A_{mean} \quad (2)$$

where $RSSI_{mean}$ represents the mean RSSI. A_{mean} is the mean RSSI at distance d_0.

According to equation (2) the distance between two devices is then given by

$$d = d_0 10^{(Amean - RSSImean)/(10n)} \quad (3)$$

A. Optimization of Logarithmic Distance Path Loss Model

In order to apply the logarithmic distance path loss model to predict the distance between devices given the mean RSSI, suitable values of A, d_0 and n appearing in equation (3) were needed. A model function was created in python that returned the predicted distance given the mean RSSI, the values A, n and d_0 according to equation 3. As mentioned in Section III, various values of RSSI for measured actual distances were recorded. Some of these values and associated measured actual distances served as a data set to fit the model as shown in equation 3.

A built-in function called curve_fit in the python scipy module was used to fit the model appearing in equation (3). The curve_fit function uses nonlinear squares to fit the model to the data. The inputs to the curve_fit function were the model function as described before, the mean RSSI data, and the corresponding measured actual distances and the parameters d_0, A and n. The curve_fit function then returned the optimized values of n, A and d_0. The optimized values where latter used in the model to predict the distance between two devices.

V. Decision Tree

Decision Trees are a popular supervised learning method that can be applied in classification and regression problems [38]. They are capable of learning decision rules from the training dataset. These rules are usually expressed in form of if-then statements. When used in regression the decision tree model is piecewise smooth. According to [38], a Decision Tree can be set to have a certain maximum depth. However, as the depth increases, the tree rules tend to be complicated and such a tree is prone to overfitting.

In this work a Decision Tree was trained using various RSSI values in order to predict the distance between devices. The scikit-learn library [38] was used to implement the Decision Tree. The original dataset was first split into training and test sets. The training set comprised 80% of the original data whereas the test set comprised 20%. The training features were the RSSI values whereas the training labels were the actual distance between devices. A fit was then done on the training dataset. The fit was undertaken for models with varying maximum depth. It was observed that fitting was poor for models with small maximum depth, typically less than 5. A model with a maximum depth of 5 was therefore chosen. This model was then used to make predictions on the test dataset. It was observed that the larger the maximum depth model, the better the model fits the training dataset. However, models with larger maximum depths could not perform well when applied on the test set.

VI. Deep Neural Network Model

Deep learning is a subset of machine learning that is applied in several fields [39]. As pointed out in [39], in deep learning an artificial neural network consisting of multiple layers is used to model a problem. Among the layers are an input layer, a number of hidden layers and an output layer [39].

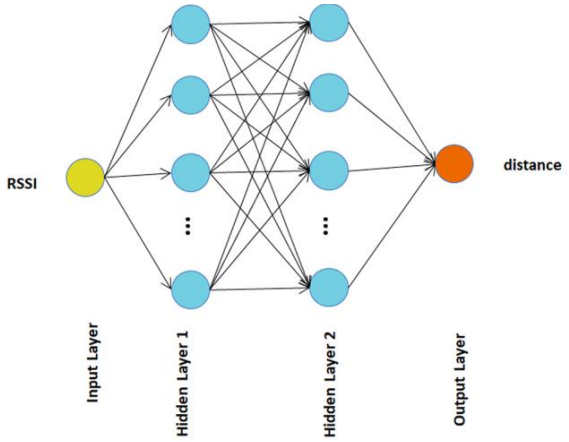

Fig. 2. Deep Neural Network Architecture for Predicting Distance between Two Devices.

In this work we propose a deep neural network (DNN) model to determine the distance between two devices using RSSI levels. Fig. 2 shows the deep neural network architecture for determining the distance between two devices. The input layer comprised one artificial neuron representing the RSSI signal which was subsequently normalized. For normalization, two quantities were first computed from the dataset. These are the mean and standard deviation. The normalized RSSI values were then computed by subtracting the mean from the original values and dividing the obtained result by the standard deviation. Two fully connected hidden layers were used in this study. The output layer predicted the distance between two devices. The programming language used to implement the model was python. The particular library used was tensorflow [40] and keras [41] as the application programming interface. The Rectified Linear Unit was used as the activation function. The Adaptive Moment Estimation (Adam) optimizer was chosen for this work. The learning rate was set to 0.01 as this was found to be appropriate. The loss function considered was the mean squared error. The original dataset for each platform was split into training and test sets where the training set comprised 80 percent of the original dataset. During training using keras, 10% of the training data was used for validation and the number of epochs was set to 200.

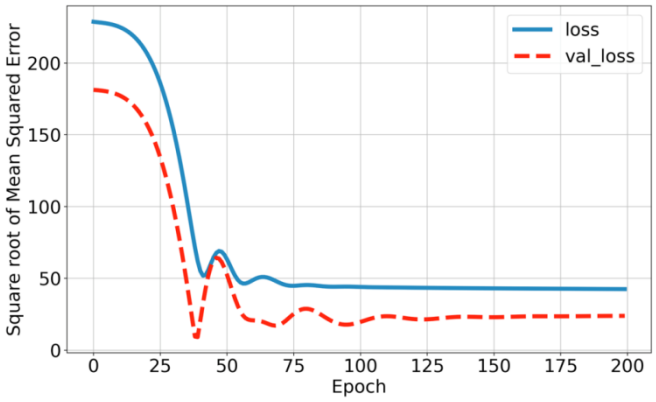

Fig. 3. Dependence of Square Root of Mean Squared Error on Epochs on Training Set (Loss) and Cross-Validation Sets (val_loss).

Fig. 3 shows the dependence of square root of mean squared error loss function on epochs. The loss functions on training set (loss) and cross-validation sets (val_loss) are shown in the Fig. 3.

The deep neural network model for each operating system was later converted to a model to be incorporated in a mobile application. For the iOS operating system, the model was converted to a coreml model using coremltools [42]. For the Android operating system, the model was converted to a tensorflow lite model according to [43]. As reported in [35], the RSSI is manufacturer dependent, therefore even for the same operating system, it varies from a device from one manufacturer to another.

VII. Comparison of Deep Neural Network, Decision Tree and Logarithmic Distance Path Loss Models

The accuracy of the deep neural network (DNN) model predictions was compared with the decision tree and logarithmic distance path loss model (LDPL). The root mean square deviation (RMSD), computed as the square root of the mean squared error (MSE) was used for the comparison. Equation 4 shows how the MSE is computed.

$$MSE = (1/(N-1))\ \Sigma_i\ (y_{predicted,i} - y_{true,i})^2 \quad (4)$$

In equation 4, $y_{predicted}$ represents the estimated target values, y_{true} represents the ground truth (correct) target values and N is the number of elements in the population. The MSE was determined using the mean_squared_error builtin function in a python sklearn.metrics module [44].

The root mean square deviation (RMSD) was then obtained according to equation 5.

$$RMSD = (MSE)^{1/2} \quad (5)$$

Fig. 4 shows the predictions of the deep neural network, decision tree as well as logarithmic distance path loss models for the iOS device. The maximum depth of the decision tree was set to 5. Also plotted is the training dataset. The trained models appear to reasonably fit the training dataset.

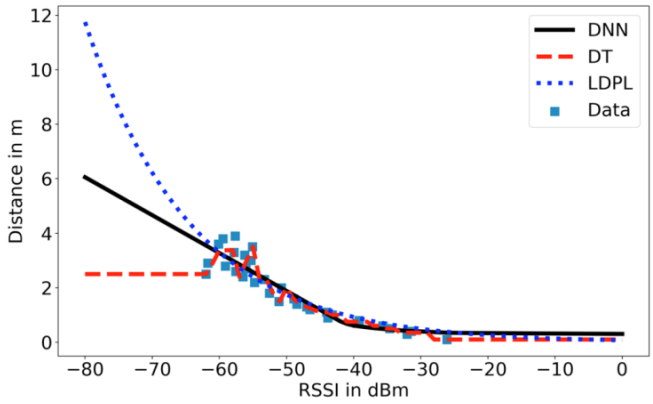

Fig. 4. Comparison of Actual Measured Data against Deep Neural Network (DNN), Decision tree (DT) and Logarithmic Distance Path loss (LDPL) Models Predictions during Training for the iOS Device.

Table I reports the RMSD computed according to equation (5) during training and testing of models. As reported in Table I, the DNN model performed better than the LDPL model during training and testing. However, the DT performed better than the DNN during training but the DNN performed better than DT during testing. This is attributed to overfitting by DT during training. Now, according to [28], it was reported that the DT performed better that the simplistic distance path loss model. This is also in agreement with the results reported in Table I.

TABLE I. Root Mean Square Deviation (RMSD) Calculation for Training and Testing the Deep Neural Network (DNN), Decision Tree (DT) and Logarithmic Distance Path Loss (LDPL) Models

RMSD on training the models in m			**RMSD on testing the models in m**		
DNN	***DT***	***LDPL***	***DNN***	***DT***	***LDPL***
0.41	0.15	0.46	0.44	0.49	0.57

The total processor (CPU) time to make model predictions on the entire training and test datasets used in Table I was

compared for the three models. Table II reports the total CPU time taken by the models in seconds.

TABLE II. TOTAL CPU TIME TAKEN TO MAKE MODEL PREDICTIONS ON THE DATASETS USED IN TABLE I.

CPU time taken during training in s			**CPU time taken during testing in s**		
DNN	***DT***	***LDPL***	***DNN***	***DT***	***LDPL***
0.091	0.00084	0.0008	0.095	0.00047	0.00098

According to Table II, the DT model took the least total CPU time to make predictions on the test dataset compared to LDPL and DNN models. However, as reported in Table I, the DNN model performed better than the DT and LDPL models in terms of predictions on the test dataset.

Owing to the fluctuation nature of the RSSI, a more accurate model for predictions is preferred despite the tradeoff in the CPU time taken to make predictions. As shown in Table II, the reported CPU times are all at sub-second level.

The processing power of mobile devices is generally lower than those of conventional desktop machines. It is worth mentioning that the DNN models have been further optimized to efficiently run on mobile devices as reported in [42] for iOS devices and [43] for Android devices. The DNN model was therefore adopted for prediction of distance between two devices.

VIII. DEVELOPMENT OF CROSS-PLATFORM CONTACT-TRACING MOBILE APPLICATION

In this work, a cross platform contact-tracing mobile application was developed. The targeted operating systems were iOS and Android. The application was developed in C# using Xamarin, a free, open source, cross-platform for building applications targeted at iOS and Android operating systems among others [45].

A number of features to be incorporated in the contact-tracing application were identified. These include the ability to detect the presence of another person running the same application on their device within an accepted range and notifying the users. The capability to notify a user if they have been exposed to a reported positive COVID-19 case in the past fourteen days was also included. In order to achieve this, the system used an already existing repository of COVID-19 patient data that contained unique diagnosis identifiers for patients. Furthermore, in case the user tested positive for COVID-19, the system was expected to provide an option to share their diagnosis identifier. This was an important feature that the application uses to notify others that they have come into contact with a positive case. A self-service feature where the user could query the application whether they have been in contact with a positive case was also included.

In order to meet these requirements, suitable technologies were identified. Bluetooth Low Energy (BLE) [46] was ideal for detecting when one user was closer to another. The Bluetooth LE Received Signal Strength Indicator (RSSI) levels described earlier were used to determine whether users were close to each other, within 2 metres for a period exceeding 15 minutes [18]. The deep neural network models described in Section VI were used to predict the distance between two users. The Global Positioning System (GPS) [47] / Cell Tower Triangulation [48] were used to determine the location in form of Latitude and Longitude coordinates. The location information was used for predicting COVID-19 hotspots.

To facilitate the contact tracing process, each mobile application user was assigned a unique random identifier that was later securely shared with another user who is in close proximity as determined by the deep neural network model. The Bluetooth Low Energy Generic Attribute Profile (GATT) framework was adopted in this case since it enables exchange of data between two devices [49]. According to GATT, the data is encapsulated in services where each service contains one or more characteristics.

The mobile application had GATT client and server capabilities. As a GATT server, the application was able to advertise services. To distinguish a mobile device running the contact tracing application from others, a unique service identifier following a universally unique identifier (UUID) format was generated for each device. Encapsulated in this service was a characteristic whose identifier was in universally unique identifier (UUID) format. The characteristic identifier was unique to the application and was used for contact tracing purposes. As a GATT client, the application was able to scan for a unique service associated with the contact tracing application and read the associated characteristics.

To implement the GATT capabilities in the application, a cross platform framework named Shiny was adopted [50]. Shiny supports BLE client and hosting among several features. This framework was also found to be convenient in that in supports backgrounding which allows an application to continue running even when sent to the background.

Fig. 5 is a top-level algorithm used to scan for BLE services, characteristics, RSSI and sending device manufacturer data among others. As shown in Fig. 5, the mobile application keeps scanning for devices and scan results are kept in a list. For every element of the list, the RSSI, manufacturer data as well as whether the device is connectable are obtained. With the obtained RSSI and sending device manufacturer data, an appropriate deep neural network (DNN) model is invoked to predict distance. In case the device is connectable, a connection was made to the device to discover offered services. The application checked for a particular service unique for contact tracing. If this service was available, the associated characteristics were obtained. The characteristics are used for contact tracing.

```
while scanning for devices:
    store scan results in a list
    for each result in a list:
        obtain RSSI, manufacturer data, IsConnectable value, and others from advertisement
        call appropriate DNN model to predict distance
        if IsConnectable value is true:
            connect to the device
            obtain discovered services
            if serviceUUID matches application UUID:
                obtain characteristics associated with service
                store RSSI, characteristicUUID, distance, manufacturer data, contact time for further processing
            end if
        end if
    end for
end while
```

Fig. 5. Top-Level Algorithm for Scanning for BLE Services, Characteristics and RSSI.

The predicted distance was used in further processing such as alerting the user in case the devices were too close. Datetime information was also captured in addition to Latitude/ Longitude coordinates. This information, together with the predicted distance from RSSI, sending device manufacturer data, characteristic UUID and contact time was stored in a local database in the user's mobile device.

After collecting this information, the contact tracing mobile application then securely transmitted this information to a central database of COVID-19 cases. An application programming interface (API) using DotNet core WebAPI development framework was written that allows for data to be shared from the mobile application to the database server using JavaScript Object Notation (JSON) as an exchange data format and HyperText Transfer Protocol (HTTP) protocol as a transport medium.

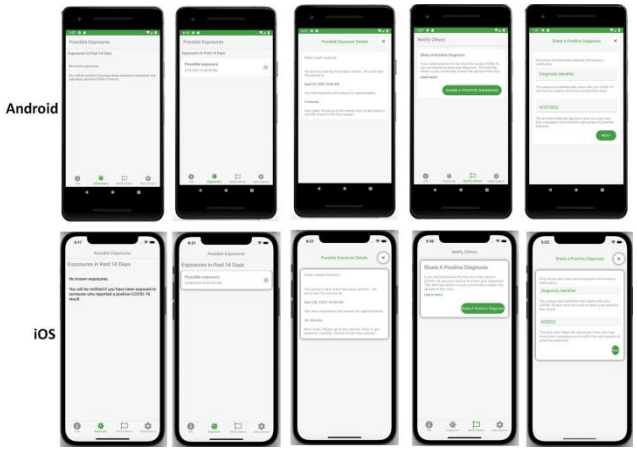

Fig. 6. Some Features of the Contact Tracing Mobile Application as Shown in Android (Top Row) and iOS (Bottom Row) Operating Systems.

A feature was also available that enables a user to determine in real-time whether they have been in contact with a positive COVID-19 case. The stored local contact tracing details for the last fourteen (14) days were then securely sent to the server and a Covid-19 database queried. If one of the contacts was recorded as a positive case in the COVID-19 database, the user was then alerted without revealing further details. To support this feature, the COVID-19 server offered a web service that was accessed through relational state transfer (REST) application programming interfaces (API).

Fig. 6 illustrates some features of the contact-tracing application. The mobile application provides real-time exposure alerts. The application also had a manual contact tracing feature to allow for positive diagnosed individuals to manually input the phone numbers of the contacts they have had met. In order to avoid abuse, a phone number verification feature was added. The application also incorporated location based services to get encrypted coordinates to be used in prediction of epicentres.

Furthermore, the Latitude, Longitude and date time information from contact-tracing applications was sent securely to the Ministry of Health databases. This data can be displayed in real-time in a map for identification of possible hotspots as shown in Fig. 7.

In this section it has been shown how the GATT framework was used to encapsulate and exchange data between Bluetooth devices embedded with GATT framework. Furthermore, the implementation of the deep neural network model in order to predict distance between communicating devices has been reported.

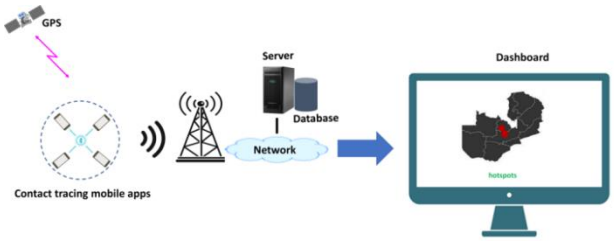

Fig. 7. Using Location and Datetime Information from Contact-Tracing Applications for Identification of Hotspots.

IX. Conclusion

A number of digital contact tracing applications have been applied world-over to facilitate the contact-tracing process. In this work, a cross platform contact tracing application that uses deep neural network models and Bluetooth Low Energy Generic Attribute Profile framework to determine and inform users of exposure to COVID-19 has been developed. The performance of deep neural network models has been evaluated against other models. The reported results show that the deep learning model performs well during testing.

The proposed deep learning model appears to learn the nonlinear relationship between distance and RSSI values better compared to analytic and decision tree models. The analytic model had four parameters according to equation 3. This suggests that the analytic model overlooked some parameters that are useful in determining the distance between communicating devices. As regards to decision trees, a major limitation is their likelihood to overfit the data on the training set as the maximum depth increases. The smaller the maximum depth, the less is the generalizing ability. On the other hand the larger the maximum depth, the larger the likelihood of overfitting on the training set resulting in less accuracy on the test set.

The developed contact-tracing application can be beneficial not only to COVID-19 prediction but also to other pandemics.

References

[1] D. Cucinotta, M. Vanelli, "WHO declares COVID-19 a pandemic," Acta Bio Medica: Atenei Parmensis, vol 91, issue 1, pp. 157 –160, 2020

[2] S. Basu, "Effective contact tracing for COVID-19 using mobile phones: an ethical analysis of the mandatory use of the aarogya setu application in India," Cambridge Quarterly of Healthcare Ethics, vol 30, issue 2, pp. 262 – 271, 2021

[3] K.T. Eames, M.J. Keeling, "Contact tracing and disease control," Proceedings of the Royal Society of London. Series B: Biological Sciences, vol 270, issue 1533, pp. 2565 –2571, 2003

[4] J. Müller, M. Kretzschmar, "Contact tracing-Old models and new challenges," Infectious Disease Modelling, vol 6, pp. 222 –231, 2021

[5] D. Klinkenberg, C. Fraser, H. Heesterbeek, "The effectiveness of contact tracing in emerging epidemics," PloS one, vol 1, issue 1, p. e12, 2006

[6] L.O. Danquah, N. Hasham, M. MacFarlane, F.E. Conteh, F. Momoh, A.A. Tedesco, A. Jambai, D.A. Ross, H.A. Weiss, "Use of a mobile application for Ebola contact tracing and monitoring in northern Sierra Leone: a proof-of-concept study," BMC infectious diseases, vol 19, issue 1, pp. 1 –12, 2019

[7] J.A. Sacks, E. Zehe, C. Redick, A. Bah, K. Cowger, M. Camara, A. Diallo, A.N.I. Gigo, R.S. Dhillon, A. Liu, "Introduction of mobile health tools to support Ebola surveillance and contact tracing in Guinea," Global Health: Science and Practice, vol 3, issue 4, pp. 646 – 659, 2015

[8] K.C. Swanson, C. Altare, C.S. Wesseh, T. Nyenswah, T. Ahmed, N. Eyal, E.L. Hamblion, J. Lessler, D.H. Peters, M. Altmann, "Contact tracing performance during the Ebola epidemic in Liberia, 2014-2015," PLoS neglected tropical diseases, vol 12, issue 9, p.e0006762, 2018

[9] G. Mosweunyane, T. Seipone, T.Z. Nkgau, O.J. Makhura, " Design of a USSD system for TB contact tracing," In IASTED International Conference Health Informatics (AfricaHI 2014), ACTAPRESS, 2014

[10] M. Shahroz, F. Ahmad, M.S. Younis, N. Ahmad, M.N.K. Boulos, R. Vinuesa, J. Qadir, "COVID-19 digital contact tracing applications and techniques: A review post initial deployments," Transportation Engineering, vol 5, p. 100072, 2021

[11] M. Nazayer, S. Madanian, F. Mirza, "Contact-tracing applications: a review of technologies," BMJ Innovations, vol 7, issue 2, pp. 368 – 378 2021

[12] TraceTogether, https://www.tracetogether.gov.sg, date accessed 03 October 2021

[13] J. Bay, J. Kek, A. Tan, C.S. Hau, L. Yongquan, J. Tan, T.A. Quy, " BlueTrace: A privacy-preserving protocol for community-driven contact tracing across borders," Government Technology Agency-Singapore, Tech. Rep, 18, 2020

[14] C. Wymant, L. Ferretti, D. Tsallis, M. Charalambides, L. Abeler-Dörner, D. Bonsall, R. Hinch, M. Kendall, L. Milsom, M. Ayres, C. Holmes, M. Briers, C. Fraser, "The epidemiological impact of the NHS COVID-19 App, " preprint at go.nature.com/2m4scfk, 2021

[15] Italy: Government Implements Voluntary Contact Tracing App to Fight COVID-19. [Web Page] Retrieved from the Library of Congress, https://www.loc.gov/item/global-legal-monitor/2020-11-09/italy-government-implements-voluntary-contact-tracing-app-to-fight-covid-19/.

[16] Spain: Government's Contact Tracing App Now in Operation Throughout Country. [Web Page] Retrieved from the Library of Congress, https://www.loc.gov/item/global-legal-monitor/2020-11-20/spain-governments-contact-tracing-app-now-in-operation-throughout-country/.

[17] Privacy-Preserving Contact Tracing - Apple and Google, https://www.apple.com/covid19/contacttracing, date accessed 03 October 2021.

[18] L. Dyani, "Contact-Tracing Apps Help to Reduce Covid Infections," Nature, vol 591, pp. 18 –19, 2021

[19] A. Narzullaev, Z. Muminov, M. Narzullaev, "Contact Tracing of Infectious Diseases Using Wi-Fi Signals and Machine Learning Classification," In 2020 IEEE 2nd International Conference on Artificial Intelligence in Engineering and Technology (IICAIET), pp. 1 – 5, 2020

[20] A. Roy, F.H. Kumbhar, H.S. Dhillon, N. Saxena, S.Y. Shin, S. Singh, "Efficient monitoring and contact tracing for COVID-19: A smart IoT-based framework," IEEE Internet of Things Magazine, vol 3, issue 3, pp. 17 – 23, 2020

[21] Y. Sahraoui, L. De Lucia, A.M. Vegni, C.A. Kerrache, M. Amadeo, A. Korichi, "TraceMe: Real-Time Contact Tracing and Early Prevention of COVID-19 based on Online Social Networks," In 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC), pp. 893 – 896, 2022

[22] K. Carteri, G. Berman, M. Garcia-Herranz, V. Sekara, "Digital contact tracing and surveillance during COVID-19 General and Child-specific Ethical Issues," https://www.unicef-irc.org/publications/pdf/IRB2020-11.pdf, date accessed 9 October 2021

[23] T. Jiang, Y. Zhang, M. Zhang, T. Yu, Y. Chen, C. Lu, J. Zhang, Z. Li, J. Gao, S. Zhou, "A survey on contact tracing: the latest advancements and challenges," ACM Transactions on Spatial Algorithms and Systems (TSAS), vol 8, issue 2, pp.1 – 35, 2022

[24] O. Megnin-Viggars, P. Carter, G.J. Melendez-Torres, D. Weston, G.J. Rubin, "Facilitators and barriers to engagement with contact tracing during infectious disease outbreaks: A rapid review of the evidence," PloS one, vol 15, issue 10, p. e0241473, 2020

[25] E. Bandara, X. Liang, P. Foytik, S. Shetty, C. Hall, D. Bowden, N. Ranasinghe, K. De Zoysa, "A blockchain empowered and privacy preserving digital contact tracing platform," Information Processing & Management, vol 58, issue 4, p.102572, 2021

[26] M. Torky, E. Goda, V. Snasel, A.E. Hassanien, "COVID-19 Contact Tracing and Detection-Based on Blockchain Technology," In Informatics, vol. 8, issue 4, p. 72, Multidisciplinary Digital Publishing Institute, 2021

[27] H. Xu, L. Zhang, O. Onireti, Y. Fang, W.J. Buchanan, M.A. Imran, "BeepTrace: blockchain-enabled privacy-preserving contact tracing for COVID-19 pandemic and beyond," IEEE Internet of Things Journal, vol 8, issue 5, pp. 3915 – 3929, 2020

[28] P.C. Ng, P. Spachos, K.N. Plataniotis, "COVID-19 and your smartphone: BLE-based smart contact tracing," IEEE Systems Journal, vol 15, issue 4, pp. 5367 – 5378, 2021

[29] P.C. Ng, P. Spachos, S. Gregori, K.N. Plataniotis, "Epidemic Exposure Tracking With Wearables: A Machine Learning Approach to Contact Tracing," IEEE Access, vol 10, pp. 14134 –14148, 2022

[30] L. Bai, F. Ciravegna, R. Bond, M. Mulvenna, "A low cost indoor positioning system using bluetooth low energy," IEEE Access, vol 8, pp. 136858 – 136871, 2020

[31] J. Paek, J. Ko, H. Shin, "A measurement study of BLE iBeacon and geometric adjustment scheme for indoor location-based mobile applications," Mobile Information Systems, 2016

[32] H. Meijerink, C. Mauroy, M.K. Johansen, S.M. Braaten, C.U.S. Lunde, T.M. Arnesen, E.H. Madslien, "The first GAEN-based COVID-19 contact tracing app in Norway identifies 80% of close contacts in "real life" scenarios, " Frontiers in digital health, vol 3, 2021

[33] J. Kalezhi, M. Chibuluma, C. Chembe, V. Chama, F. Lungo, D. Kunda, "Modelling Covid-19 infections in Zambia using data mining techniques," Results in Engineering, vol 13, p. 100363, 2022

[34] Zambia National Public Health Institute, Available: https://znphi.co.zm, date accessed: 19 February 2021

[35] Proximity and RSSI, https://www.bluetooth.com/blog/proximity-and-rssi/ Date accessed: 16 March 2022

[36] L. Flueratoru, V. Shubina, D. Niculescu, E.S. Lohan, "On the High Fluctuations of Received Signal Strength Measurements with BLE Signals for Contact Tracing and Proximity Detection," IEEE Sensors Journal, 2021

[37] G. Li, E. Geng, Z. Ye, Y. Xu, J. Lin, Y. Pang, "Indoor positioning algorithm based on the improved RSSI distance model," Sensors, vol 18, issue 9, p. 2820, 2018

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, "Scikit-learn: Machine learning in Python," the Journal of machine Learning research, vol 12, pp. 2825 –2830, 2011

[39] Y. LeCun, Y. Bengio, G. Hinton, "Deep learning, " Nature, vol 521, pp. 436 – 444, 2015

[40] M. Abadi et al, "TensorFlow: Large-scale machine learning on heterogeneous systems," Software available from tensorflow.org, 2015

[41] F. Chollet et al, "Keras," Available at: https://github.com/fchollet/keras, 2015

[42] TensorFlow 2 Conversion, https://coremltools.readme.io/docs/tensorflow-2 Date accessed: 18 March 2022

[43] tf.lite.TFLiteConverter : TensorFlow Core v2.8.0, https://www.tensorflow.org/api_docs/python/tf/lite/TFLiteConverter Date accessed: 18 March 2022

[44] G. Van Rossum, F.L. Drake, "Python 3 Reference Manual," Scotts Valley, CA: CreateSpace, 2009

[45] Xamarin: Open-source mobile app platform for .NET , https://dotnet.microsoft.com/apps/xamarin, date accessed: 8 October 2021

[46] Bluetooth Technology Overview, https://www.bluetooth.com/learn-about-bluetooth/tech-overview/, date accessed: 8 October 2021

[47] Satellite Navigation - Global Positioning System (GPS), https://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gnss/gps/, date accessed: 8 October 2021

[48] J. Yang, A. Varshavsky, H. Liu, Y. Chen, M. Gruteser, "Accuracy characterization of cell tower localization," In Proceedings of the 12th ACM international conference on Ubiquitous computing 2010 Sep 26, pp. 223 –226, 2010

[49] K. Townsend, C. Cufí, R. Davidson, "Getting started with Bluetooth low energy: tools and techniques for low-power networking, " O'Reilly Media, Inc., 2014

[50] shinyorg/shiny: A Xamarin Framework for Backgrounding & Device Hardware Services, https://github.com/shinyorg/shiny Date accessed: 18 March, 2022.