Encyclopedia of Information Science and Technology, Fourth Edition

Mehdi Khosrow-Pour
Information Resources Management Association, USA

IGI Global
DISSEMINATOR OF KNOWLEDGE

Published in the United States of America by
IGI Global
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA, USA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data

Names: Khosrow-Pour, Mehdi, 1951- editor.
Title: Encyclopedia of information science and technology / Mehdi Khosrow-Pour, editor.
Description: Fourth edition. | Hershey, PA : Information Science Reference, [2018] | Includes bibliographical references and index.
Identifiers: LCCN 2017000834| ISBN 9781522522553 (set : hardcover) | ISBN 9781522522560 (ebook)
Subjects: LCSH: Information science--Encyclopedias. | Information technology--Encyclopedias.
Classification: LCC Z1006 .E566 2018 | DDC 020.3--dc23 LC record available at https://lccn.loc.gov/2017000834

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Recurrent Neural Networks for Predicting Mobile Device State

Juan Manuel Rodriguez
ISISTAN, UNICEN-CONICET, Argentina

Alejandro Zunino
ISISTAN, UNICEN-CONICET, Argentina

Antonela Tommasel
ISISTAN, UNICEN-CONICET, Argentina

Cristian Mateos
ISISTAN, UNICEN-CONICET, Argentina

INTRODUCTION

Smartphones have become a key part of everyday life as an essential tool for their users. People have fully integrated mobile devices into their lives by using them to communicate with friends, check e-mails, play games, record physical activities and take pictures, among other possible uses. Moreover, smartphones are equipped with several features (WiFi, GPS, and Bluetooth among others) that can record activities and contextual information, such as location, application usage, and even messaging and calling behaviour. Hence, smartphones are interesting options for tracking and mining user behaviour in daily life (Do and Gatica-Perez, 2014). This information offers new opportunities to analyse human behaviour aiming at enhancing the user experience with mobile devices and, at the same time, helping to ease the use of smartphones' services (Rios et al., 2014).

Several domains leverage on the prediction of mobile devices' states (Pejovic and Musolesi, 2015; Niroshinie et al., 2013; Ravi et al., 2008). For example, in the development of context aware applications, the predictions could be useful for determining the context in which applications are running. Such context aware applications are encompassed in a concept called "Anticipatory Mobile Computing" (AMC) (Pejovic and Musolesi, 2015). The goal of AMC is deciding which actions should be taken based on predicted future states to improve the outcome. AMC concepts are present in personal assistance technology (such as Google Now, Microsoft Cortana or Siri), healthcare applications and smart cities. Personal assistance technology uses state prediction to provide relevant information to the user before such information is requested. For example, predicting that the hour in which the user goes to work allows personal assistance technology to provide information about traffic and weather, which might be relevant to the user.

Other use of predictions can be found in mobile cloud computing (Niroshinie et al., 2013). One of its key proposals is moving computing from mobile devices to the cloud to reduce battery consumption. However, to effectively reduce battery consumption, it is necessary to predict whether the energy requirements for communicating are lower than those of processing. In addition, if the mobile device is not going to be connected to the Internet when the cloud finishes its work,

DOI: 10.4018/978-1-5225-2255-3.ch577

computation results will be unavailable. This might lead to the repetition of the computation in the mobile device, which would waste more energy than if the mobile device had performed the computation in the first place. These are just a few examples of the developing mobile device state prediction techniques' importance.

The current state of mobile devices can be regarded as the consequence of the previous states. Consequently, future behaviour can be predicted based on how a user has been using his/her device. The generation of predictive models of human behaviour has emerged as a topic of interest in several areas, such as recommendation systems, context-aware services, and personalised and adaptive interfaces. For example, several studies have focused on predicting the probability of users to be at a particular place at a given time. Also, Do and Gatica-Perez (2014) and Liao et al. (2012) aimed at predicting which application the user will use next based on contextual information to reduce the time users spend on searching for a specific application.

Although mobile devices constantly evolve as they provide increasing functionality due to the improvements in processing power, storage capabilities, graphics and connectivity; battery capacities do not experience the same growth (Ravi et al., 2008). Power emerges as a critical resource for battery-powered systems as mobile devices. Hence, battery management becomes a crucial requirement to users. Providing battery management information requires the ability to accurately predict remaining battery life in a dynamically changing system. Interestingly, most of the studies in the literature focus on offering location prediction, or application personalisation, instead of analysing the impact of user behaviour in battery level and life. In this context, this work evaluates the suitability of Recurrent Neural Networks (RNN) for predicting future battery levels of mobile devices based on the users' usage pattern of different features, such as the WiFi connection or screen status, among others.

BACKGROUND

The analysis of human behaviour through smartphone usage has attracted considerable interest in recent years. Works on smartphone mining include physical activity recognition by applying learning techniques to accelerometer data, and personalisation of content and user interface. For example, Do and Gatica-Perez (2014) predicted human behaviour based on smartphone sensors using statistical methods commonly used in signal processing for forecasting time series. Particularly, user location within a ten-minute window was predicted along with the applications users were more likely to use based on the location, time, Bluetooth proximity and communication logs. The approach linearly combined least-squares linear regression, logistic regression and Markov models. Location prediction results showed that the most important predictors were Bluetooth proximity, location and time, and confirmed the dependency between human mobility and social interactions.

Application usage prediction was also studied in (Shin et al., 2012; Liao et al., 2012). Both approaches aimed at presenting users with the most probable applications to be used aiming at reducing the time spent searching for a desired application. Shin et al. (2012) based the prediction on a probabilistic Naïve Bayes model that was personalised to each user according to his/her application usage pattern. Prediction was based on information from GPS, cellular network, accelerometer, calls, WiFi, Bluetooth, screen status, battery status, illumination and running applications. Results showed that the previously used applications, location and hour of day were useful predictors.

Liao et al. (2012) created temporal profiles of application usage and proposed a three step probability-based scoring model to compute the probability of using an application. First, the periodicity of application usage was detected by a Discrete Fourier Transform. Second, captured periodicities were hierarchically clustered to

identify users' behaviour. Clusters represented different usage patterns. For example, one cluster corresponded to weekday behaviour, whereas other comprised weekend behaviour. Finally, the possible application usages were predicted based on the Chebyshev's inequality.

Regarding connectivity prediction, Rios et al. (2014) aimed at predicting Bluetooth and WiFi usage based on mining association rules from information regarding data, WiFi and Bluetooth connections, GPS status changes, ringer status (silent, vibrate, normal) changes and headset connection. The rules were able to accurately capture the relations between WiFi connection habits relative to location and GPS status. The authors highlighted the potential of their approach for assisting users in the activation/deactivation of connectivity features to increase battery life.

Finally, closely related to this work are the studies carried out in (Wen et al., 2003; Ravi et al., 2008), which aimed at predicting battery life. Wen et al. (2003) predicted battery based on information regarding variations in workload, application usage, battery charge rates, the charging-cycle effect and a voltage curve. Then, such information was combined into an online linear curve fitting statistical approach. Ravi et al. (2008) also aimed at predicting the next charging opportunity, call time predictor (i.e. how much call-time will the user require over a determined time period), and application battery consumption. With such information the authors intended to inform users not only when they will run out of battery, but also, which applications should they terminate to increase battery life. The analysis considered location, battery status, list of running processes, id of the cell the smartphone is connected to, past calling behaviour, average number of minutes of call time that users need during each hour of the day separated between weekdays and weekends, and the base curve of battery life.

PREDICTING MOBILE DEVICES STATES USING RNN

This work aims at predicting a mobile device future state based on previous states, i.e. predicting sequences of states. Mobile device state changes as the result of an event in time. For example, a change in the battery level can be defined as an event, which triggers a new state of the mobile device in which the battery level is modified, and all other features remain the same. Particularly, this work defines a state as a combination of the following features:

- **Time Between Events:** Minutes passed in-between the current state and the previous one. Type: Integer.
- **Battery Level:** Mobile device current battery level ranging between 0 and 100. Type: Integer.
- **External Energy Supply:** Whether the mobile device is plugged to an external energy supply, e.g. AC adapter or USB connection. Type: Boolean.
- **Screen On/Off:** Whether the mobile device screen is active. Type: Boolean.
- **WiFi:** Whether the mobile device is connected to a WiFi network. Type: Boolean.

To predict future states, RNN will be used. RNN (Williams and Zipser, 1989) differ from feed-forward Artificial Neural Networks (ANN) in that they can have cycles in the computational networks. These cycles allow the network to have dynamic contextual windows, instead of fixed size windows like the standard feed forward models. This dynamic window contributes to the enormous potential of RNN for successfully predicting sequences of events. In this context, the sequence of events models the sequence of states for a mobile device. Although this work does not evaluate the

feasibility of deploying the proposed predictor in a mobile device, a work (Alsharif et al., 2015) on decoding keyboard gesture patterns into text by means of an RNN shows that it is possible to use RNN within a mobile device.

SOLUTIONS AND RECOMMENDATIONS

The proposed ANN is based on the RNN known as Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997). RNN were used because they are very effective for predicting sequences (Alsharif et al., 2015) as new predictions depends on past predictions. This allows to use an arbitrary number of previously known states to make a prediction.

The proposed ANN needs at least two consecutive states for predicting the following state. Figure 1 depicts the architectural structure of the ANN. In a first step, each state is processed by a multi-layer perceptron (MLP). Each layer linearly combines its input vector using a weighting matrix (W) and an internal bias (B), with the combination being processed via

$$L_n\left(i\right) = tanh\left(\left(i_n^T W_n\right)^T + B_n\right),$$

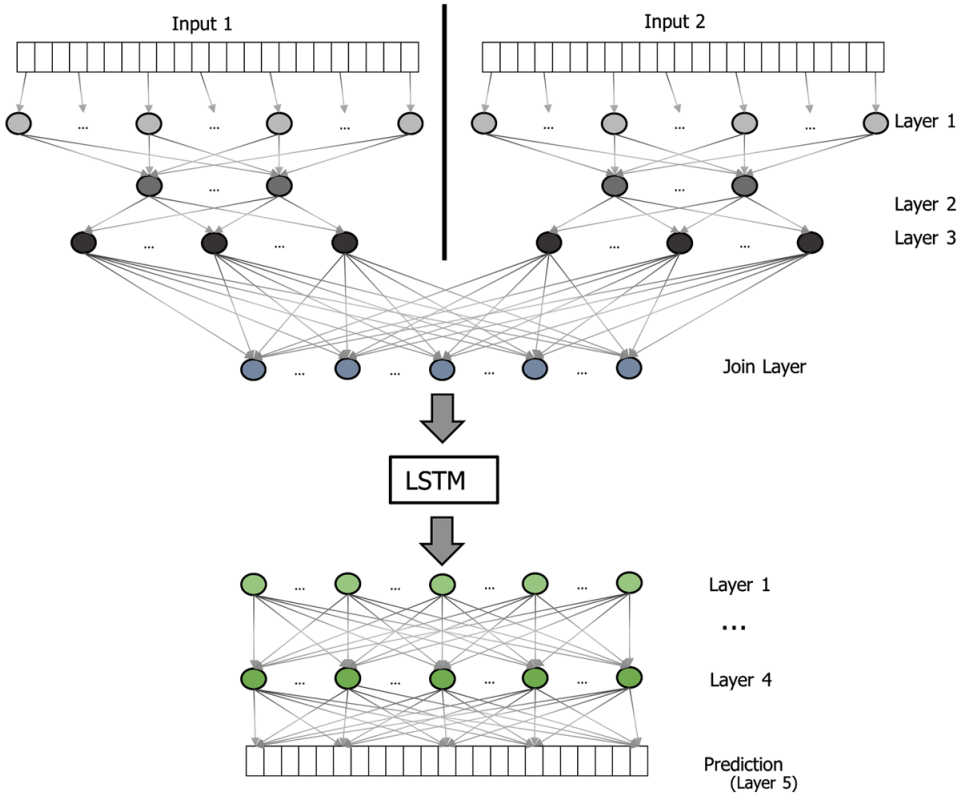
which describes how each layer behaves. Note that functions such as tan$h$ and *sigmoid* are applied element wise to the vectors. Then, the different layers are combined in a MLP via

$$L_n\left(L_{n-1}\left(\ldots L_1\left(x\right)\ldots\right)\right).$$

As a result of these operations, the newly computed vector can be used as input for another layer.

Since the proposed ANN uses two different MLP for processing the two states, the results

Figure 1. Neural Network structure



of each MLP need to be combined. For this, the proposed approach uses a layer similar to the layers in MLP, but that accepts several inputs. In the approach this layer is called Join Layer (*JL*(*i*…*j*)), which can be defined as

$$tanh\left(\left(i_n^T W_{JL-i} + \ldots + j_n^T W_{JL-j}\right)^T + B_{JL}\right)$$

The Join Layer result is fed to the recurrent layer of the ANN, namely the LSTM layer. This layer is recurrent because it has two variables that take their values from previous outputs: the context and the hidden state. Since the output of this layer depends on previous outputs, the prediction of the proposed ANN depends on previous predictions making it an RNN. The context (c_{t-1}) is the LSTM layer previous output, and the hidden state (s_{t-1}) is the LSTM layer previous internal state. For the first prediction, the context c_0 and the hidden state s_0 are vectors full of zeros. The LSTM layer has three inputs: the current state (x_t), the hidden state and the context. The hidden state s_t depends on s_{t-1}, $g_t\left(x_t, c_{t-1}\right)$ and $y_t^{in}\left(x_t, c_{t-1}\right)$, and can be defined as

$$s_t = s_{t-1} + g_t\left(x_t, c_{t-1}\right) \circ y_t^{in}\left(x_{t-1}, c_t\right),$$

where

$$g_t\left(x_t, c_{t-1}\right) = sigmoid\left(\left(x_t^T W_g^x + c_{t-1}^T W_g^c\right)^T + B_g\right)$$

and

$$y_t^{in}\left(x_t, c_{t-1}\right) = sigmoid\left(\left(x_t^T W_{y^{in}}^x + c_{t-1}^T W_{y^{in}}^c\right)^T + B_{y^{in}}\right).$$

After computing the new hidden state, the LSTM layer output (c_t) can be defined as the dot product $h_t \circ y_t^{out\left(x_t, c_{t-1}\right)}$, which takes as input

$$h_t = sigmoid\left(\left(s_t^T W_h\right)^T + B_h\right),$$

and

$$y_t^{out}\left(x_t, c_{t-1}\right) = tanh\left(\left(x_t^T W_{y^{out}}^x + c_{t-1}^T W_{y^{out}}^c\right)^T + B_{y^{out}}\right).$$

Finally, the LSTM layer output is taken as input by other MLP for obtaining the final prediction.

In order to be analysed by the NN, mobile device state features were encoded into a vector of elements, in which each element can adopt either -1 or 1. The selection of such values corresponds to the fact that ANN usually rely on *sigmoid*(*i*) and tan*h*(*i*) functions, which when applied to -1 or 1 result in values close to their asymptotes. Also, applying such functions to 0 results in the mean value of their asymptotes. For example, tan*h*(*i*) has two asymptotes: -1 when $i \to -\infty$ and 1 when $i \to \infty$. Applying the function to the selected values results in: tan*h*(-1)≅-0.7616, tan*h*(1)≅0.7616, while tan*h*(0)=0, i.e. the mean value of the asymptotes. Integer features were encoded using their binary representation. For instance, the "Hour of the day" feature was encoded using the five binary digits needed for representing any number between 0 and $31=2^5-1$. In the case of the "Time between events" feature, the maximum time was set to 3 hours or 180 minutes. Thus, it was encoded using 8 elements. Finally, for binary features, -1 stands for false, and 1 stands for true, e.g. when the mobile device is connected to a WiFi network the "WiFi" feature is 1, otherwise it is -1.

Figure 2 illustrates the application of the approach for predicting the states from time $n+1$ to m using the previous n states. In a first step, the ANN is fed with the known states x_1 and x_2 resulting in a prediction $x_3^{'}$. Such step is not performed for obtaining the prediction, as x_3 is known, but for the sake of the ANN internal state. After this step, the ANN is fed with x_2 and x_3. The ANN is fed repeatedly with the known states until the $x_{n+1}^{'}$ prediction is obtained. At this moment, as no states are known, $x_{n+1}^{'}$ is the first state to be predicted. Unlike the input, in which each element can only adopt two values, the output elements adopt ap-

proximations to the expected values. For instance, if a particular element of x'_{n+1}, is -0.9 it is likely that the real value of x_{n+1} is -1. Hence, x'_{n+1} needs to be pre-processed before it is fed to the ANN. Eq. 1 shows the pre-process function. The sequence prediction algorithm has been proved to be effective in other sequence prediction domains (Sutskever et al., 2014).

$$p(x') = \begin{cases} 1 & x' \leq 0 \\ -1 & otherwise \end{cases}$$

Experimental Settings

The dataset used in the experimental evaluation was gathered from a mobile device using the application described in (Rios et al., 2014). The mobile device owner was a University employee who volunteer for the study. The mobile device was a Samsung Galaxy SII and the dataset was collected between 5th of June of 2015 and 8th of October of 2015. Since the mobile device owner could turn on and off the logging application at will, the dataset does not have information for every day between those dates. The resulting dataset comprises data belonging to 19,067 records of mobile device states collected during 79 days. Records were grouped into chunks, where each chunk contains states from a single day. However, days in which states were separated by more than 3 hours were split into several chunks. In total, 83 chunks were created, which were split into training and validation splits. The training split comprised 70% of the chunks, which were randomly selected. Table 1 summarises the dataset characteristics.

Table 1. Dataset description

	Full Dataset	Training Split	Validation Split
Days	79	56	23
Chunks	83	59	24
Events	19,067	12,809	6,258
Average	230	217	261
St. Dev.	168	124	246
Min. Size	63	63	75
Max. Size	979	638	979

The proposed ANN has 546,862 trainable parameters, which are denoted by the weights matrices (W), and the biases vectors (B) denote the ANN parameters to be trained. Table 2 shows the size of the matrices and vectors involved in each of the first MLPs. The parameters of the join layer and LSTM are detailed in Table 3 and Table 4, respectively. Finally, the parameters of the MLP that produces the final prediction are presented in Table 5.

To fit the ANN parameters, different approaches were evaluated. Firstly, two error functions were considered, namely Mean Squared Error (MSE) and Cross-Entropy Error (CEE). MSE consists in adding the element wise squared difference between the expected value and the predicted value, as shown in Eq. 2 CEE is used in logistic regression and represents the entropy between the obtained prediction and the expected value. For the expected values being $y \in \{0,1\}$, Eq. 3 presents the cross entropy formula. Since the elements of the expected values are $y_i \in \{0,1\}$, the correction formula $c(x)=(x+1)/2$ was applied element wise

Figure 2. State prediction

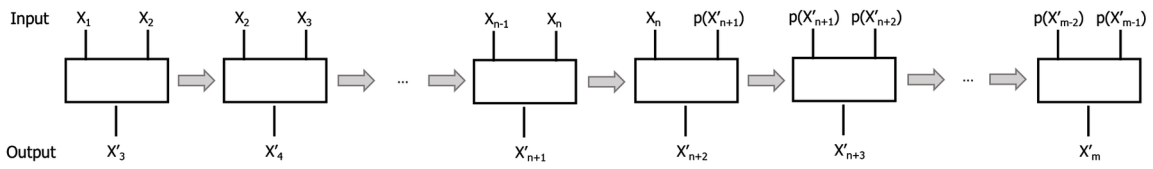


Table 2. First MLP parameters

Parameter	Dimensions	Parameter	Dimensions
W_1	23 x 23	B_1	23
W_2	23 x 40	B_2	40
W_3	40 x 50	B_3	50

Table 3. Joint layer parameters

Parameter	Dimensions
$W_{JL\text{-}1}$	50 x 100
$W_{JL\text{-}2}$	50 x 100
B_{JL}	100

Table 4. LSTM parameters

Parameter	Dimensions	Parameter	Dimensions
W_g^x	100 x 1000	$W_{y^{out}}^x$	100 x 100
W_g^c	100 x 1000	$W_{y^{out}}^c$	100 x 100
B_g	1000	$B_{y^{out}}$	100
$W_{y^{in}}^x$	100 x 1000	W_h	1000 x 100
$W_{y^{in}}^c$	100 x 1000	B_h	100
$B_{y^{in}}$	1000		

Table 5. Final MLP parameters

Parameter	Dimensions	Parameter	Dimensions
W_1	100 x 40	B_1	40
W_2	40 x 23	B_2	23
W_3	23 x 40	B_3	40
W_4	40 x 23	B_4	23
W_5	23 x 23	B_5	23

for both expected values and predictions before computing the Cross Entropy Error.

$$J\left(y, y'\right) = \frac{\sum\left(y_i - y_i^{'}\right)^2}{N}$$

$$J\left(y, y'\right) = \frac{-\sum\left(y_i \cdot \log\left(y_i^{'}\right) + \left(1 - y_i\right) \cdot \log\left(1 - y_i^{'}\right)\right)}{N}$$

To avoid overfitting, the impact of adding a regularisation factor to both MSE and CEE was also assessed. The regularisation factor was calculated as the sum of the parameters' L_1-norm. Finally, two different techniques were evaluated to approximate the ANN parameters. The first one was the well-known Stochastic Gradient Descent. The second one was ADADELTA (Zeiler, 2012), a variation of Stochastic Gradient Descent in which the learning rate is computed based on the historic values of the gradient. The two error functions,

regularised or not, and the techniques for fitting parameters can be freely combined resulting in eight different settings.

Experimental Results

The first experiment aimed at assessing all potential training settings described above. For each setting, the evolution of the ANN up to 50 epochs was evaluated. One-state-in-the-future prediction stands for predicting the state x'_{n+1} using all the states between x_1 and x_n. Since the ANN requires at least two previous states for generating a prediction, the first possible prediction is x'_3 using x_1 and x_2. In a chunk of m states, the last prediction is x'_m using (x_1, x_2, …, x_{m-1}).

Firstly, the training dataset was evaluated using MSE, as shown in Figure 3. Within this figure, Figure 3a and Figure 3b present the errors obtained when training the ANN using MSE or CEE, respectively. The settings using Gradient Descent and MSE achieved the lowest errors. Furthermore, the setting without regularisation performed slightly better, which is expected as regularisation is intended to reduce overfitting the training dataset. However, these results alone are not sufficient for evaluating as overfitting is not completely avoided. Hence, the same evaluation was performed using the validation dataset. Figure 4 shows the results for the validation dataset. Particularly, Figure 4a shows the errors for the ANN trained using MSE, and Figure 4b shows the errors for ANN trained using CEE. As expected, the obtained MSE are higher than the errors obtained for the trained dataset. The shape of the curves in Figure 4a and Figure 4b are similar to the ones in Figure 3a and Figure 3b, respectively. This implies that the ANN did not overfit the training data. Moreover, Gradient Descent and MSE without regularisation also achieved the lowest error.

The second experiment was designed for evaluating the sequence prediction capabilities of the ANN. Thus, predictions of up to 10 states into the future were generated using the validation dataset and the best performing ANN from the previous experiment. Such ANN was the one obtained after the 34th epoch using Gradient Descent and MSE without regularisation. Note that for predicting the n state into the future, the n–1 previous predictions were used. For instance, suppose that m states are known and that the state $m+n$ is to be predicted. In this context, the ANN would use as input the following state sequence

$$\left(x_1,\ldots,x_m,p\left(x'_{m+1}\right),\ldots,p\left(x'_{m+n-1}\right)\right).$$

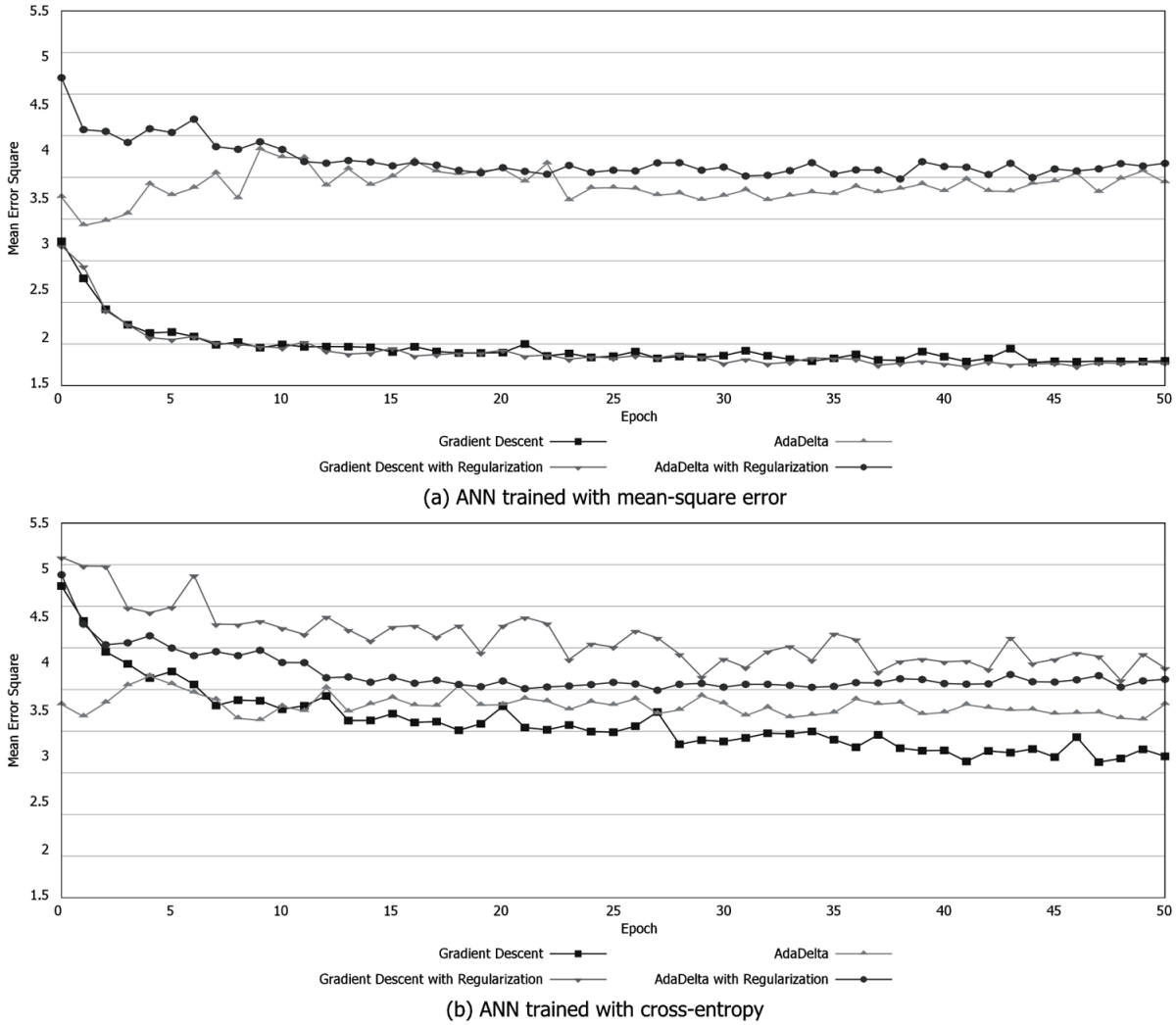
The average time between 10 states is 48'26.82" with a standard deviation of 58'47.27". The minimum registered time between 10 states was 10 minutes and the maximum was 697 minutes.

For the integer type features, namely "Hour", "Time between events", and "Battery", the error was calculated as the Mean Absolute Error (MAE) as defined in Eq. 4. Figure 5 presents the obtained MAE. For "Hour" and "Battery", the MAE increases as the number of states in the future increases, whereas for "Time between events" remains invariant. This is expected due to the fact that "Hour" and "Battery" values are correlative with previous values, while "Time between events" values are not. For instance, if "Battery" has a value of 60% and the battery is not in charging state, it is very unlikely that the next state has a value for "Battery" of 10% or 90%. Hence, an error predicting a "Battery" value would have a heavy impact on further predictions.

$$MAE = \frac{\sum \left| expected\,value - predicted\,value \right|}{\#\,prediction}$$

Since the error in predicting the time between n states is given by the sum of each individual prediction error for "Time between events", the mean accumulated absolute error (MAAE) is defined in Eq. 5. MAAE accounts for the accumulated error in a prediction sequence. In this context, the MAAE for "Time between events"

Figure 3. MSE for training dataset



was 121 minutes. Considering these results, it can be concluded that the ANN did not performed effectively for predicting "Time between events", but performed with a low error rate for predicting "Hour" and "Battery".

$$MAAE = \frac{\sum_{i=1}^{N} \left| expected\,value_i - predicted\,value_i \right|}{\#\,prediction}$$

The relation between the "Hour" and "Battery" features' Absolute Error (AE) and the "Time between events" feature Accumulated Absolute Error (AAE) for the 10-state-in-the-future prediction was studied to determine whether failing at predicting the "Time between events" is correlated to errors in the prediction of other features. Table 6 presents the Pearson correlation between "Hour" and "Battery" features' AE, and the "Time between events" feature AAE. There is a strong correlation between the errors of predicting "Hour" and "Time between events" probably because both are related to the prediction of the time between states. However, there is no strong correlation with the "Battery" prediction, i.e. errors in "Battery" prediction are independent from errors in the time interval prediction.

Finally, a True Positive (TP) and True Negative (TN) analysis for the prediction of the binary fea-

Figure 4. MSE for validation dataset

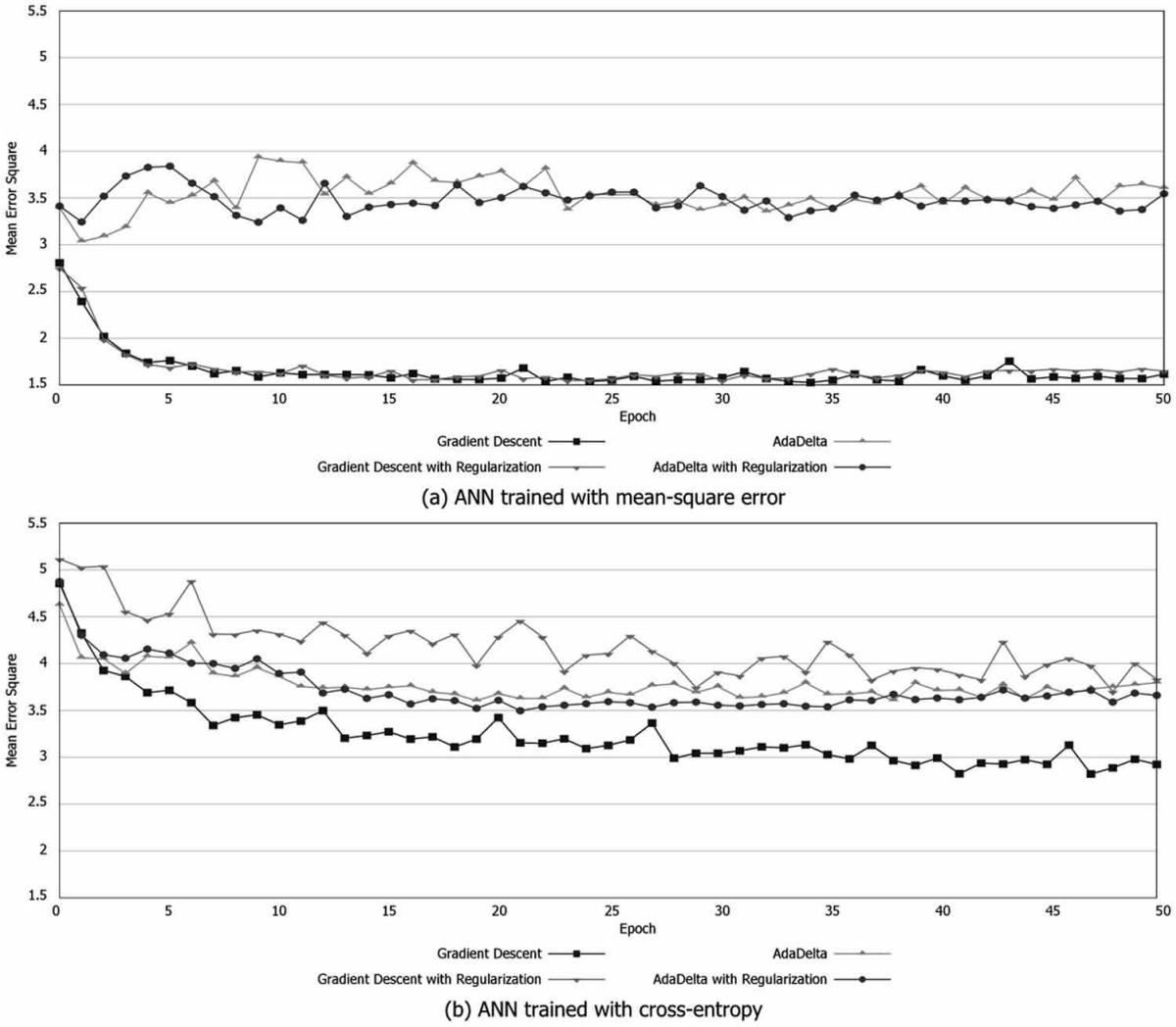


Figure 5. Mean Absolute Error up to 10-state-in-the-future prediction

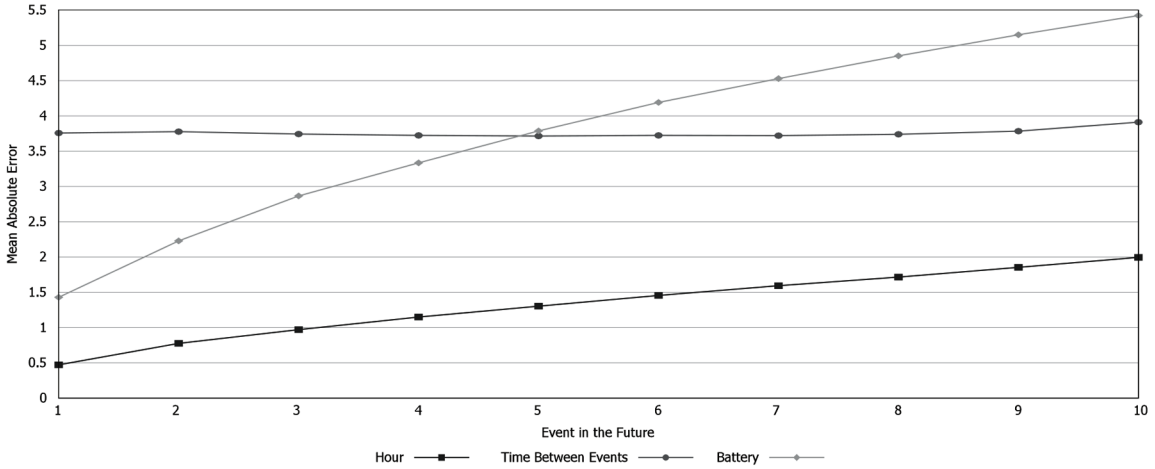


Table 6. Pearson correlations with "Time between events"

Feature	Correlation	p-value
Hour	0.400	>0.001
Battery	0.005	0.719

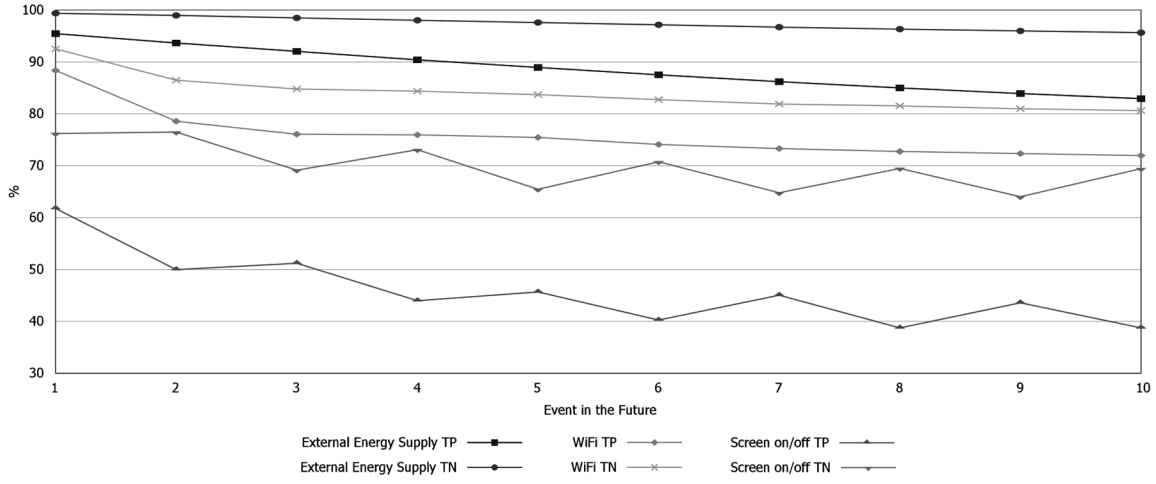
tures ("External Energy Supply", "Screen on/off" and "WiFi") was performed, as Figure 6 reports. The best results were obtained when predicting "External Energy Supply". Although the TP and TN decrease as more states into the future are predicted, they are never below 80% and 95%, respectively. The ANN also achieved good performance when predicting the "WiFi" feature with a TP rate of 70% and a TN rate of 80%. However, predicting the "Screen on/off" feature resulted in low TP and TN rates. Although the performance when predicting this feature was relatively good when considering a 1-state-in-the-future prediction, it rapidly decreased as more predictions into the future were performed. For example, predicting that the screen would be on in the second future state provided no real information about whether the screen would be on or off, as the TP rate was about 50%. Furthermore, after three states into the future, predicting that the screen would be on meant that the screen would be most likely off. This was probably a result of a lack of information about user incentive for activating the screen, such as application notifications or user location.

FUTURE RESEARCH DIRECTIONS

Since this paper defines mobile device states using only a small subset of all the possible features, future studies will consider additional features, such as location, Bluetooth connection, incoming/outgoing calls, cellular connection, or 3G signal strength. The main limitation of the current approach is the high number of parameters. Future research should evaluate other ANN configurations with fewer parameters.

Future works will further validate the proposed approach using a larger dataset, such as the Cambridge Device Analyser dataset (Wagner et al., 2014). Additionally, the possibility of using just one ANN for modelling a number of users could be studied. This could be useful in domains in which group behaviour is more important than individual behaviour. Another line of work is comparing ANN based state prediction with other machine learnings techniques. Finally, more complex

Figure 6. True-Positive/True-negative analysis



ANN models should be evaluated in the future. Such models would not only include other type of ANN layers, but also expand the definition of the mobile device state by adding other features, such as location, CPU usage, and day of the week.

CONCLUSION

This study aimed at evaluating the suitability of ANN for predicting mobile device future states. The obtained evidence suggests that RNN have potential for predicting future states of mobile devices. However, the results were mixed, as the ANN was able to effectively predict with low error rates the state of features, such as "Battery" or "External Energy Supply", while it was ineffective for predicting other features, such as "Screen on/off" or "Time between events".

REFERENCES

Alsharif, O., Ouyang, T., Beaufays, F., Zhai, S., Breuel, T., & Schalkwyk, J. (2015). Long short-term memory neural network for keyboard gesture decoding. *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, 2076–2080. doi:10.1109/ICASSP.2015.7178336

Do, T. M. T., & Gatica-Perez, D. (2014). Where and what: Using smartphones to predict next locations and applications in daily life. *Pervasive and Mobile Computing*, *12*, 79–91. doi:10.1016/j.pmcj.2013.03.006

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. doi:10.1162/neco.1997.9.8.1735 PMID:9377276

Liao, Z.-X., Lei, P.-R., Shen, T.-J., Li, S.-C., & Peng, W.-C. (2012). Mining temporal profiles of mobile applications for usage prediction. *Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on*, 890–893. doi:10.1109/ICDMW.2012.11

Musolesi, M., Piraccini, M., Fodor, K., Corradi, A., & Campbell, A. T. (2010). *Pervasive Computing: 8th International Conference, Pervasive 2010, Helsinki, Finland, May 17-20, 2010. Proceedings*. Springer Berlin Heidelberg.

Niroshinie, F., Seng, W. L., & Wenny, R. (2013). Mobile cloud computing: A survey. *Future Generation Computer Systems*, *29*(1), 84–106. doi:10.1016/j.future.2012.05.023

Pejovic, V. and Musolesi, M. (2015). Anticipatory mobile computing: A survey of the state of the art and research challenges. *ACM Comput. Surv.*, *47*(3), 47:1–47:29.

Ravi, N., Scott, J., Han, L., & Iftode, L. (2008). Context-aware battery management for mobile phones. In *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*, (pp. 224–233). IEEE. doi:10.1109/PERCOM.2008.108

Rios, C., Rodriguez, J. M., Godoy, D., Schiaffino, S., & Zunino, A. (2014). Usage pattern mining for smartphone use personalization. In *Proceedings of the 13th Brazilian Symposium on Human Factors in Computing Systems*, (pp. 377–380). Porto Alegre, Brazil: Sociedade Brasileira de Computação.

Shin, C., Hong, J.-H., & Dey, A. K. (2012). Understanding and prediction of mobile application usage for smart phones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, (pp. 173–182). New York, NY: ACM. doi:10.1145/2370216.2370243

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Proc. NIPS*.

Wagner, D. T., Rice, A., & Beresford, A. R. (2014). Device analyzer: Large scale mobile data collection. *SIGMETRICS Perform. Eval. Rev.*, *41*(4), 53–56. doi:10.1145/2627534.2627553

Wen, Y., Wolski, R., & Krintz, C. (2003). Online prediction of battery lifetime for embedded and mobile devices. Lecture Notes in Computer Science, 3164, 57–72.

Williams, R. J., & Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, *1*(2), 270–280. doi:10.1162/neco.1989.1.2.270

Zeiler, M. D. (2012). *ADADELTA: An adaptive learning rate method.* CoRR, abs/1212.5701

ADDITIONAL READING

Burbey, I., & Martin, T. L. (2012). A survey on predicting personal mobility. *International Journal of Pervasive Computing and Communications*, *8*(1), 5–22. doi:10.1108/17427371211221063

Campbell, A., & Choudhury, T. (2012). From smart to cognitive phones. *IEEE Pervasive Computing / IEEE Computer Society [and] IEEE Communications Society*, *11*(3), 7–11. doi:10.1109/MPRV.2012.41

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. doi:10.1038/nature14539 PMID:26017442

Rosa, F. D., Malizia, A., & Mecella, M. (2005). Disconnection prediction in mobile ad hoc networks for supporting cooperative work. *IEEE Pervasive Computing / IEEE Computer Society [and] IEEE Communications Society*, *4*(3), 62–70. doi:10.1109/MPRV.2005.55

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85–117. doi:10.1016/j.neunet.2014.09.003 PMID:25462637

Zambonelli, F. (2015). Engineering self-organizing urban superorganisms. *Engineering Applications of Artificial Intelligence*, *41*, 325–332. doi:10.1016/j.engappai.2014.10.004

KEY TERMS AND DEFINITIONS

Artificial Neural Network: A kind of machine learning algorithms loosely based on how biological neural networks work.

Epoch: A single training pass through the entire training set.

Error Function: A function used for assessing how well a machine learning method performs.

Gradient Descent: Technique for fitting parameters of a function.

Mobile Device: A computational device that people carries, such as smartphones or tables.

Mobile Device State: A set of variables that describe the mobile device's current conditions.

Recurrent Neural Network: ANN that uses previous states for making new predictions.