A Deep Neural Network Approach for Customized Prediction of Mobile Devices Discharging Time

Mattia Gentil, Alessandro Galeazzi, Federico Chiariotti, Michele Polese, Andrea Zanella, Michele Zorzi
Department of Information Engineering, University of Padova – Via Gradenigo, 6/b, 35131 Padova, Italy
Email: {gentilma, galeazzial, chiariot, polesemi, zanella, zorzi}@dei.unipd.it

***Abstract*—The role of mobile devices, like smartphones or tablets, is becoming more and more important in everyday life, at the point that their unavailability due to early or unexpected battery discharge is perceived as a serious issue. Therefore, there is an urge for smart and efficient battery management algorithms that can prolong the duration of the battery charge. To this end, a reliable prediction of the battery discharging process would represent a precious tool to enable energy-efficiency optimization mechanisms. In this paper, we address this challenge by considering different machine learning techniques to provide an accurate and user-dependent prediction of the discharging time of a mobile device and, eventually, we propose a Deep Neural Network model that provides the best performance. Unlike previous solutions proposed in the literature, our method exploits space-time data from the device operating system (Android) to learn the specific battery usage pattern of the user, thus offering a customized prediction of the discharge process. We show that such model outperforms the other machine-learning methods considered in this study, and achieves much better performance than the deterministic linear fitting methods widely used in commercial devices.**

I. Introduction

Internet access from mobile devices has been rising in the last years, with an eightfold increase projected by 2020 [1]. Furthermore, the variety and complexity of the services offered to mobile users are also constantly growing. These factors put a strain on the duration of the battery charge of mobile devices, which usually have to be recharged during the day to provide the required dependability [2]. While there is a huge research effort to increase the battery capacity and recharging speed, another approach looks at techniques to improve the energy efficiency of the devices, e.g., implementing smart energy-saving policies [3]. In this respect, an accurate estimate of the residual charge duration can be useful both to drive the energy-saving policies implemented by the operating system of the device and to let the user adopt energy-preserving strategies when using his/her mobile device.

As of today, most devices predict the battery depletion time by taking into account the mean duration of a full battery charge and the most recent battery-draining trend, disregarding time and location information [3]. In this work, we propose a novel method for the prediction of the battery depletion time that makes use of a Deep Neural Network (DNN) to draw a personalized battery-usage pattern from time and location information provided by the mobile device. We apply our method to a dataset that includes mobility data collected by the LifeMap monitoring system at Yonsei University in Seoul [4]. Results show that our prediction method is by far more accurate than a popular method, currently employed in commercial mobile devices to predict the residual duration of the battery charge, and outperforms other machine-learning methods. We also provide insights on the importance of location information to the accuracy of such a prediction, and on the computational complexity of the proposed approach.

The rest of the paper is organized as follows. Sec. II overviews the state of the art on prediction techniques in mobile networks, with specific attention to the prediction of the battery charge duration. Sec. III introduces the preprocessing and model selection procedures, while Sec. IV presents the performance of our model in comparison with other baseline methods. Finally, Sec. V concludes the paper with some remarks and possible future extensions.

II. Related Work

Predicting the state evolution of mobile systems is an interesting research topic, which has received much attention in the last decade. The prediction of a device trajectory in a mobile network, for example, is a fundamental problem that has been addressed in many research papers, and with different approaches [5]–[9]. In particular, [7] and [9] use Neural Networks (NNs) to predict the next location of a device from the past trajectory. Another network parameter that is suitable for the prediction of the device position is the wireless channel gain, which is considered, for example, in [10] and [11] by using Support Vector Machines (SVMs) and Bayesian regressors, respectively. These systems provide accurate estimates, but they do not consider the energy consumption, which is a fundamental parameter in mobile device applications.

Machine learning techniques have also been applied to predict the battery charge duration of mobile devices. Most of the proposed solutions based the forecast on the current state of the device together with the battery-discharge history. For example, Zhao *et al.* [3] use the multiple linear regression prediction method: the discharge rate is calculated by comparing the current device status (e.g., CPU load, LCD brightness and I/O device usage) with the discharge rates that were observed in the past.

In [12], two NNs are trained in order to predict the discharging curve of batteries. Starting from some considerations on the electro-chemical nature of rechargeable batteries, the authors derive a non-linear discharging pattern. In order to model this behavior, they feed the NNs with some key parameters of the battery and device state, such as time, battery level, battery temperature, and resource usage in the past. The obtained predictor can estimate the remaining working time with an accuracy of 3%. However, their model was implemented and

evaluated on a digital multimeter under controlled research conditions and no testing was performed on real world data. Wen *et al.* [13] propose to predict battery lifetime by using an online and offline calculation. First, a reference discharge curve is derived from a one-time, full-cycle, voltage measurement of a constant load, and is suitably transformed to reduce prediction complexity. Then, the forecasting is performed by mapping the current discharging data on the reference curve, using linear fitting or Least Squares methods.

An empirical approach to battery lifetime prediction is proposed in [14]. After analyzing a large dataset containing the discharging patterns of several users and training a general model, they built a tool to predict smartphone charge levels by classifying smartphone users based on the comparison of their discharging patterns with those in the dataset.

In [15], Rakmatov *et al.* study the problem of battery discharge time prediction from a theoretical point of view. Starting from the physical working principles of a lithium-ion electrochemical cell, a high-level model of the battery is built. The discharging coefficients of the model are estimated by simulation and statistical fitting of empirical data, and variable and constant loads are taken into account to improve the estimate of the residual operating lifetime of the device.

Most of the techniques mentioned above do not take individual users' discharging patterns and personal behavior information into account, but instead exploit the depletion discharging curve of a generic smartphone usage or focus on a short-term prediction mainly employing low-level information. Although current battery level and resource usage are fundamental to predict the battery discharge time, we advocate that a much more reliable and long-term estimation of the battery charge can be obtained by considering some space-time features of the device usage patterns. Our model will be able to capture time- and location-dependent user habits which can have a heavy impact on the battery lifetime [3]. Moreover, it is highly scalable to different types of devices and batteries and does not require manual tuning or reference parameters, unlike [13] or [3].

III. System Model

In this section we discuss the proposed estimation method. We start by describing the dataset used in the study and the selection of input and output parameters. Then, we explain the preprocessing operations performed on the data to ease the learning task of the DNN.

A. Dataset and features

The dataset from the LifeMap project [4] includes a wide collection of mobility data that was gathered from the smartphones of 6 graduate students from Yonsei University over 6 months. More specifically, data about the battery level, position and connectivity level of each smartphone were retrieved from the operating system and stored every 10 minutes. Data clearly span many charge and discharge cycles, but the battery charge reached a critically low level only on a few such cycles, since in many occasions the smartphones were recharged during the day, before the battery charge dropped below critical levels. Therefore, given this limitation of the dataset, we rely on the expected lifetime of the battery, as we will explain in the following paragraphs.

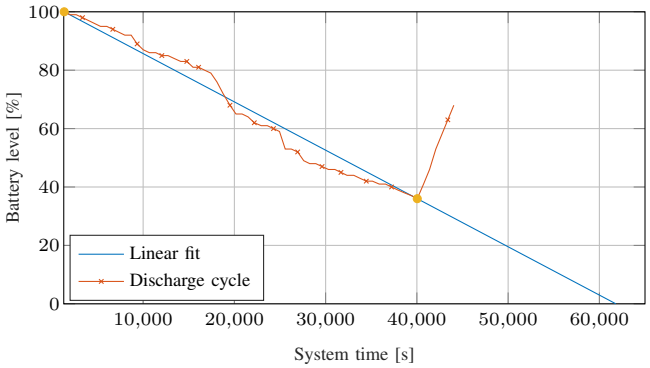


Figure 1: Example of battery discharge rate estimation. The read line with cross markers shows one empirical discharge curve taken from the dataset, while the blue line is the linear interpolation, whose intersection with the x-axis provides the estimated depletion time. Note that the last marker of the red line corresponds to the battery charge at the first sampling instant of the recharge phase.

To design our predictor, we first need to identify the *features*, or *input* parameters, that have to be extracted from the dataset, and the *targets*, or *output* values, of the machine learning model. The choice of these parameters is discussed in the following.

a) Target parameter: Since our purpose is to predict the battery discharge process over time, the output of the estimator should be a projection of the residual lifetime of the device in the future. To build the training set for our machine learning algorithm, hence, we need to find the (expected) residual lifetime of the battery charge in each of the time instants of the discharge phases where data were collected. For the discharge phases that end with a non-negligible residual battery level, the expected depletion time is estimated by linear extrapolation of the charge levels during that cycle. More specifically, let $(t_0, t_1, \dots t_n)$ be the set of sample times in a given discharge period, and $b(t_i)$ be the value of the battery level (on a 0-100 scale) measured at time t_i, $i = 0, 1, \dots, n$. If we take the whole discharge cycle into consideration, the expected depletion time is obtained as:

$$t_{\text{dep}} = t_0 - b(t_0)\frac{t_n - t_0}{b(t_n) - b(t_0)}\,. \tag{1}$$

Finally, the outputs y_i used to train our prediction model are obtained for each discharge cycle as

$$y_i = t_{\text{dep}} - t_i, \quad 0 \le i \le n\,. \tag{2}$$

An example of this analysis is shown in Fig. 1. We observe that this estimate is more accurate when $b(t_n)$ is small, i.e., when the discharge cycle ends with an almost complete discharge of the battery, which is quite common with modern devices [2]. The linear estimation of the depletion time is a limit to the prediction accuracy, but the limit is inherent in the dataset.

Moreover, we observe that the estimate of the residual battery charge duration is more accurate when it becomes more useful, i.e., when the battery level is low.

Architecture	Activation Function	α	Training time [s]	Evaluation time [s]	R^2 validation set	R^2 training set
(100, 100, 100, 100)	identity	0.01	2.479	0.030	0.036	0.040
(300)	logistic	1e-05	2.212	0.045	0.036	0.040
(100, 100, 100, 100)	logistic	0.01	1.064	0.076	0.000	0.000
(300)	ReLU	0.001	40.875	0.023	0.580	0.738
(100, 100, 100)	ReLU	0.01	54.795	0.044	0.656	0.842
(100)	tanh	0.01	17.947	0.031	0.568	0.699
(300)	tanh	0.01	57.476	0.041	0.529	0.642
(100, 100, 100)	tanh	0.01	76.726	0.066	0.754	0.853
(200, 150, 100, 50)	tanh	0.12	111.439	0.108	0.806	0.904

Table I: Performance of the DNN for different parameter settings. The architecture column represents the number of nodes in the hidden layers of the DNN. The identity activation function is $f(x) = x$, the logistic is $f(x) = 1/(1 + e^{-x})$, and the rectified linear unit (ReLU) is $f(x) = x$ if $x > 0$, $f(x) = 0$ otherwise.

b) Input features: Our model considers multiple inputs in order to estimate the remaining battery lifetime, namely:

- Battery level: the current status of the battery is clearly essential for a meaningful prediction.
- Time of the day: this parameter is essential to allow the model to learn and recognize specific battery usage patterns during the day. For instance, a user may make a more intensive use of the smartphone while resting in the evening, rather than during the day, or *vice versa*. To capture these aspects, battery usage data need to be coupled with time information.
- Day of the week: along the same rationale, this feature can make the DNN able to recognize patterns of battery usage that depend on the day of the week. For example, sports activities regularly practiced during the week may be associated to a lower usage of the smartphone (e.g., when left in the locker) or to a higher usage (e.g., due to activity-tracker applications), depending on the user's habits.
- Location: this feature can also discriminate different battery consumption patterns, which are often correlated with the user's location. The dataset includes data regarding the user movements and labels for each place that was visited. The locations are classified in 12 categories, such as home, workplace, leisure place, and so on.
- Movement: finally, we consider a boolean feature indicating whether the user is moving from one place to another or is static. This piece of information can make it possible to identify activities typically performed by users while moving on foot or with public transportation, e.g., emailing, phone calls, or web/social network surfing.

We advocate that such features should be sufficient to discriminate among the specific usage patterns of the different users, thus making it possible to customize the prediction engine.

B. Preprocessing

To efficiently train the machine learning models, we first need to preprocess the input data [16]. In particular, we standardized features and outputs by removing the mean and scaling the amplitudes to get unit variance. By doing so, the distribution of individual features is close to the normal distribution with zero mean and unit variance.

We explored several machine learning techniques, namely SVMs [17], K-Nearest Neighbors [18] (KNN), linear regression [19] and NNs [20]. The performance of these methods has been evaluated in terms of the *coefficient of determination metric* R^2, which is a very popular criterion to measure and analyze the performance of machine learning models [21]. Denoting by y_i and $\hat{y}_i$ the actual and predicted values of the ith output, respectively, and by $\bar{y}$ the mean of the actual values, then the coefficient of determination is defined as

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \,. \tag{3}$$

Note that $R^2 \leq 1$ and the fraction gives the mean squared estimation error over the variance of the outputs. A perfect predictor will yield $R^2 = 1$, while a "dummy" prediction that always returns the expected value of the output, disregarding all inputs, would yield $R^2 = 0$.

In our tests, a DNN with at least one month worth of training data always outperformed all other methods and thus turned out to be the most suitable tool for battery discharge prediction.

The parameters of the DNN were chosen by performing an exhaustive search in the parameter space; each setting was evaluated by cross validation, and successive rounds refined the search space until convergence to the optimal settings. The explored parameters were the size and shape of the hidden layers of the network, the activation function of hidden neurons and the regularization parameter α. We always used the "L-BFGS" [22] solver, an optimizer in the family of quasi-Newton methods, with 200 training iterations. In Tab. I some meaningful iterations are shown. We also report the R^2 score on the training and validation sets. In particular we can notice that the activation function and network structure strongly impact both the network prediction performance (i.e., R^2 validation score and R^2 training score) and the time required to train the network and to perform the prediction. We found that a distribution of the neurons across several layers allowed the model to detect more complex battery usage patterns, yielding in general better performance with both the `ReLU` and the `tanh` activation functions, which showed the best performance, as reported in the last 6 rows of Tab. I. On the other hand, increasing the number of neurons does not always have a positive impact on the test score, and usually requires longer training phases (see rows 6 and 7 in Tab. I).

The model chosen after cross validation consists in a DNN with 4 hidden layers of 200, 150, 100 and 50 neurons each, respectively, shown in the last row of Tab. I. The best performing activation function was the hyperbolic tangent function, i.e., $f(x) = \tanh(x)$. In order to avoid overfitting, the regularization parameter α has been set to the quite high value of 0.12.

Data processing, model training and performance evaluation have been carried out by using the Scikit-learn Python frame-

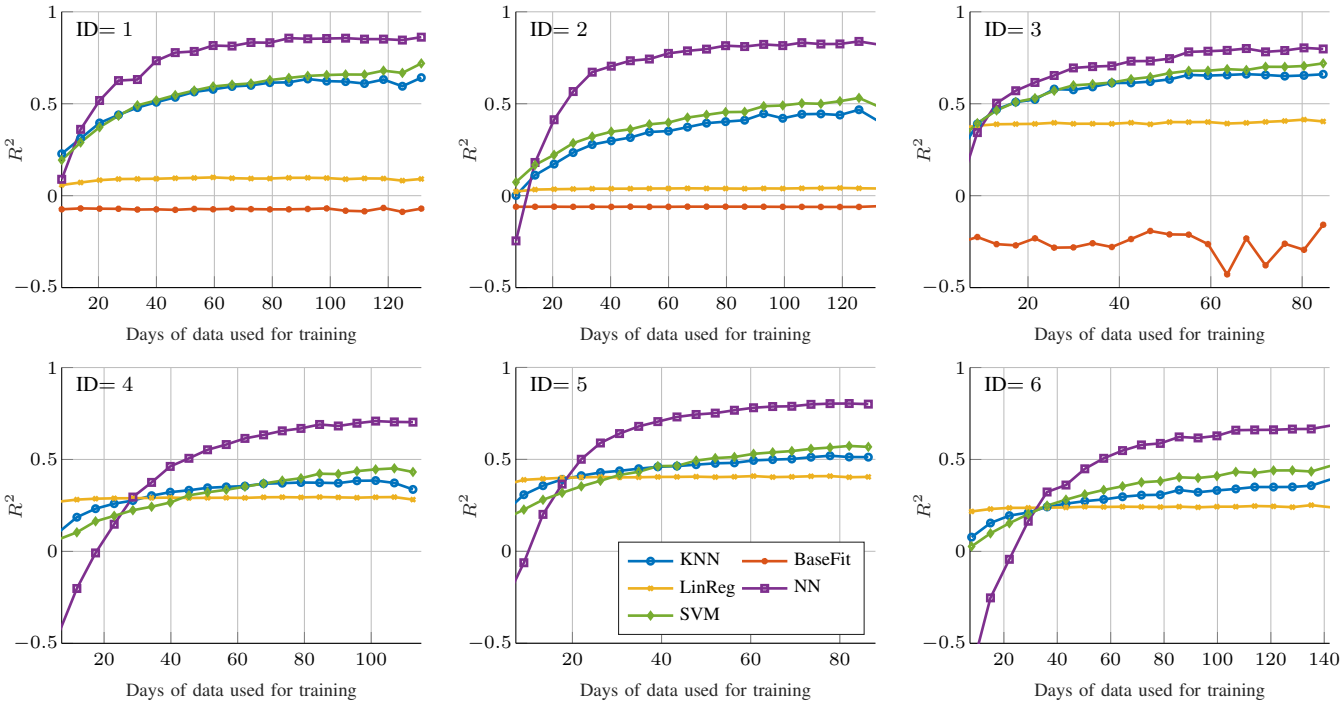


Figure 2: Comparison of the performance of several models by averaging results from 20 random training/testing subsets. The legend is shared by all the plots. The commonly employed deterministic baseline "BaseFit" is reported in orange and is the worst method in general; for the last 3 plots the R^2 metric of the BaseFit algorithm is below -0.5, therefore the curve does not appear in the specified range. The yellow curve "LinReg" displays the performance of a simple linear regression fitting, while the blue line reports the accuracy of a KNN framework. Finally the efficiency of an SVM is reported in green while the performance of our Deep NN is purple.

Train user \ Test user	1	2	3	4	5	6
1	0.643	-0.0733	-0.293	-2.130	-3.508	0.642
2	-0.268	0.645	-0.581	-4.083	-6.479	-0.048
3	-0.195	-0.074	0.687	-2.490	-3.865	0.514
4	-0.121	-0.068	-0.220	0.664	-3.173	0.714
5	-0.110	-0.067	-0.223	-1.715	0.703	0.729
6	-0.070	-0.101	-0.423	-0.335	0.670	0.761
All	0.308	0.371	-0.167	-4.693	-3.725	-0.711

Table II: R^2 values of DNNs with about 3 months worth of training data. Each column reports the results of the testing on a different user, while the rows specify the user on which the training is performed. For the last row, the training is carried out by mixing data from all six students.

work [23]. The test set corresponds to 10% of the available data for each user.

IV. Results

We compare our prediction model with a baseline method that is very similar to the estimation that is adopted in today's smartphones. This deterministic method consists of taking the last 5 samples of the battery level history and obtain a linear approximation of the discharge pattern via Least Squares linear fit [13]. The linear approximation is then used to estimate the remaining charge duration (basically, the same method we used to extrapolate the charge lifetime in our training dataset). We also compare the performance of our DNN with SVMs, KNN regression and Linear Regression using the same input features and training/test sets. The plots shown in Fig. 2 confirm that, for all the 6 users of the considered dataset, the machine learning models in general have better performance than the deterministic baseline method and, among the considered machine learning techniques, the DNN achieves the best performance.

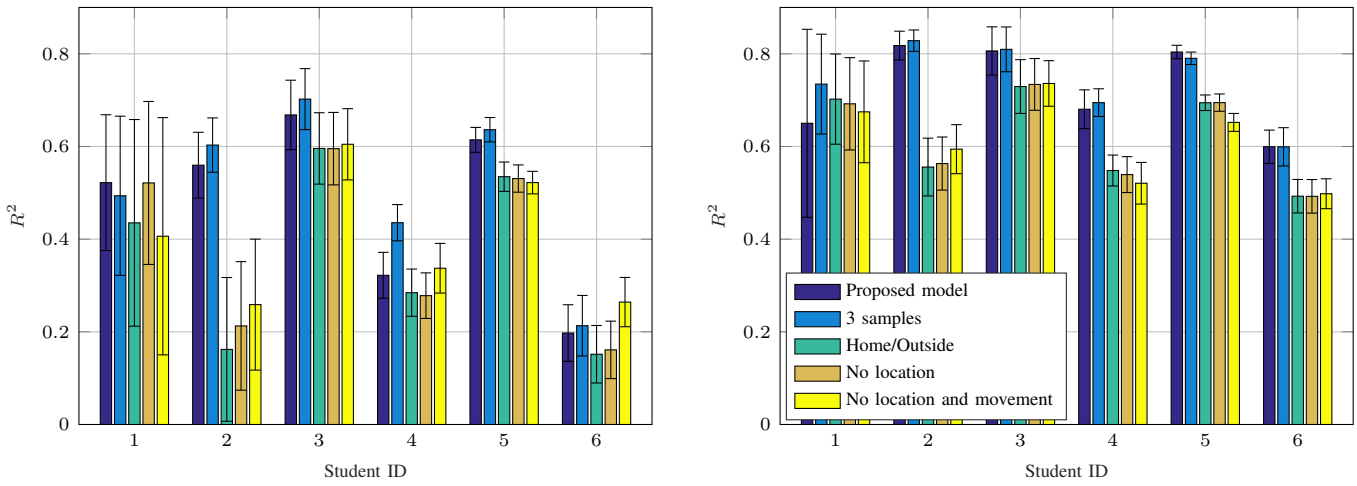
A. Prediction performance and generalization

The fully trained DNNs reached a mean coefficient of determination of 0.7835, with a peak of 0.8620 for a user whose dataset was particularly clean, with fairly regular battery usage patterns.

As Fig. 2 shows, the deterministic battery estimation paradigm is inefficient, as it bases its prediction only on the recent battery history, without considering more general patterns which can instead be accounted for by using location and time-based information that is available at almost no cost in every modern mobile device [24]. The clear advantage of such a model, however, is that it does not require any training.

In order to test the generalization/discrimination capabilities of our model, we trained the DNN using the data of a single user and tested its prediction performance on the data of any other user. The results of this test are shown in Tab. II, where each row reports the R^2 obtained for the different users when the network was trained with the data of the user in the corresponding row. The last row reports the results obtained when training the network with a mix of the data of all the users. It is clear that the DNN generally performs poorly if trained on a user and tested on a different one, while the performance is good when training and test data are from the same user[1]. Moreover, the low performance values reported in the last row of the table confirm that our model must be trained only on personal data, otherwise it cannot learn accurately a user's personal patterns nor provide reliable predictions. This proves that our model can learn a user's behavior properly

[1]Disjoint portions of each dataset are used for training and testing.



(a) One month of training data

(b) Three months of training data

Figure 3: R^2 score and 95% confidence intervals on test sets of NNs with different input vectors for every user. Performance with 1 month (left plot) and 3 months (right plot) worth of training data. Bars refer to different choices of the input vectors: "Proposed model" refers to the input vector described in Sec. III-A; "3 samples" adds the last 3 battery level samples to the input; "Home/Outside" replaces the 12 location categories with a boolean input indicating whether or not the user is at home; "No location" disregards location inputs; "No location and movement" disregards both location and motion inputs.

but can hardly generalize to other users. In other words, the learnt model is strictly personal and dependent on the user's specific habits. As a consequence, the model needs to be trained with local information that can be collected only by the smartphone of each user, so that it is not possible to pre-train the network and deploy it on a general device, rather the DNN must be trained while the device is being used by its owner and this process requires some time, as discussed later. On the other hand, the DNN does not need to share information with other users or with service providers in order to be trained or provide the prediction. Considering the growing concerns for the user's privacy in location-based services [25] and the decreasing willingness of the users to share location information [26], these features of the proposed approach can be appreciated by the users. Of course, during the training phase of the DNN, the prediction of the battery charge lifetime can be obtained with other approaches, in order to guarantee the availability of the service to the user: when the training is complete, the prediction will become more precise, being tailored to the user's habits.

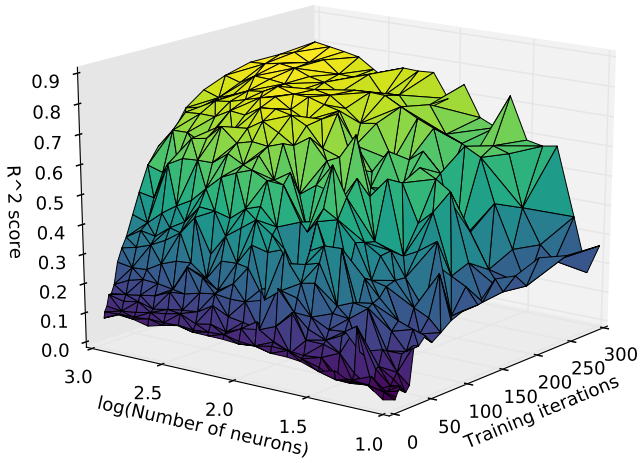


Figure 4: R^2 metric for the DNN as a function of model complexity, expressed as the logarithm of the total number of neurons and the training time in epochs.

B. *Effect of incomplete input on the prediction*

Our "Proposed" input vector, described in Sec. III-A, consists of the current battery level, the time of the day, the day of the week, the location of the user, and the motion information. In order to gain insights on the relevance of the different input features on the prediction accuracy of the DNN, we considered other input vectors, namely a larger input vector obtained by enriching the Proposed one with the last three samples of the battery level; then a smaller input vector with a boolean value "home/outside" in place of the full 12 location categories; then an input vector, named "No location", without any location-based data; and finally an input vector without location and movement data, called "No Location and movement." Fig. 3 shows the R^2 score obtained for the different test users and for each of these input vectors, when considering one month (left-hand side plot) or three months (right-hand side plot) of training data.

We observe that, with one month of training data, the location-based DNNs outperform the models with incomplete or no location data available for students 2, 3 and 5, while for the other subjects the different input vectors yield comparable results. However, with three months of training data, the advantage of a location-based input vector becomes more evident for all the subjects. Without location information, the "movement" feature becomes useless or detrimental: if the DNN has no information on users' location, it should just disregard movement information. We can also notice that, in general, considering more than the last sample of battery level only brings small improvements in terms of R^2. Furthermore, from the confidence intervals in Fig. 3 we can see that a larger amount of training data improves the stability of the prediction and reduces its sensitivity to noisy input data, as expected.

C. Computational complexity of the prediction

Finally, we investigate the impact on the R^2 metric of the number of neurons and the number of training iterations to reach convergence. We found that a network with 500 neurons is a good tradeoff between computational complexity and reliability of the model. Fig. 4 shows the results of one of our experiments. Note that with as few as 10 neurons and 100 training iterations, the DNN already outperforms the common predictors of mobile devices that always yield negative R^2 score during our experiments (see Fig. 2). However, increasing the size of the model makes it more robust to variations and noise in the training step and improves its performance with respect to the other ML algorithms tested in this paper. On the other hand, the increased computational costs of very large DNNs have diminishing returns, as Fig. 4 clearly shows: the DNN we used in this paper offers a good tradeoff between prediction accuracy and computational complexity.

The computational complexity of the model is overall reasonable. A training step of 200 epochs on 144 points, that represent a day's worth of collected data, requires 2 seconds of CPU time for training and this can be easily performed while a device is charging or may be relegated to a moment when the phone is idle. A single evaluation of the remaining battery lifetime requires less than 100 μs on a modern CPU, which makes this model highly portable. Therefore, the proposed model can be used locally, on the users' smartphone, and does not need computational offloading on a remote server, confirming its privacy-oriented character.

V. Conclusions

In this work, we proposed a model for battery lifetime prediction based on a DNN framework. We proved that such model is able to learn and predict patterns of battery usage that are strictly correlated to a user's mobility trends. Thus, a learning process leads to a highly customized battery prediction model that can increase the accuracy of current battery life estimation methods and can be integrated in optimization frameworks to further increase the energy efficiency of the device. Moreover, the proposed approach is self-contained and does not need to disclose personal information to third parties, thus preserving the user's privacy.

As future work, we plan to implement and test the model on a physical Android smartphone and experimentally verify its performance on a larger number of users. We will also consider whether there exist pre-training methods that can speed up the learning process, thus providing acceptable performance even with limited training.

References

[1] Cisco, "Cisco Visual Networking Index, Forecast and Methodology, 2015–2020," Market Report, June 2016.

[2] G. P. Perrucci, F. H. P. Fitzek, and J. Widmer, "Survey on energy consumption entities on the smartphone platform," in *IEEE 73rd Vehicular Technology Conference (VTC Spring)*, May 2011, pp. 1–6.

[3] X. Zhao, Y. Guo, Q. Feng, and X. Chen, "A system context-aware approach for battery lifetime prediction in smart phones," in *ACM Symposium on Applied Computing*, ser. SAC '11, 2011, pp. 641–646.

[4] J. Chon and H. Cha, "LifeMap: A Smartphone-Based Context Provider for Location-Based Services," *IEEE Pervasive Computing*, vol. 10, no. 2, pp. 58–67, April 2011.

[5] Y. Chon, E. Talipov, H. Shin, and H. Cha, "Mobility prediction-based smartphone energy optimization for everyday location monitoring," in *9th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '11, 2011, pp. 82–95.

[6] L. Ghouti, T. R. Sheltami, and K. S. Alutaibi, "Mobility prediction in mobile ad hoc networks using extreme learning machines," *Procedia Computer Science*, vol. 19, pp. 305–312, June 2013.

[7] J. Biesterfeld, E. Ennigrou, and K. Jobmann, "Neural networks for location prediction in mobile networks," in *International Workshop on Applications of Neural Networks to Telecommunications (IWANNT'97)*, 1997, pp. 207–214.

[8] J. Capka and R. Boutaba, "Mobility prediction in wireless networks using neural networks," in *IFIP/IEEE International Conference on Management of Multimedia Networks and Services*. Springer, 2004, pp. 320–333.

[9] S. Parija, R. K. Ranjan, and P. K. Sahu, "Location prediction of mobility management using neural network techniques in cellular network," in *International Conference on Emerging Trends in VLSI, Embedded System, Nano Electronics and Telecommunication System (ICEVENT)*, Jan 2013.

[10] F. Chiariotti, D. Del Testa, M. Polese, A. Zanella, G. M. Di Nunzio, and M. Zorzi, "Learning methods for long-term channel gain prediction in wireless networks," in *International Conference on Computing, Networking and Communications (ICNC2017)*. IEEE, January 2017.

[11] Q. Liao, S. Valentin, and S. Stańczak, "Channel gain prediction in wireless networks based on spatial-temporal correlation," in *IEEE 16th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, June 2015, pp. 400–404.

[12] O. Gérard, J.-N. Patillon, and F. D'Alché-Buc, "Discharge prediction of rechargeable batteries with neural networks," *Integr. Comput.-Aided Eng.*, vol. 6, no. 1, pp. 41–52, January 1999.

[13] Y. Wen, R. Wolski, and C. Krintz, "Online prediction of battery lifetime for embedded and mobile devices," in *International Workshop on Power-Aware Computer Systems*. Springer, 2003, pp. 57–72.

[14] E. A. Oliver and S. Keshav, "An empirical approach to smartphone energy level prediction," in *13th International Conference on Ubiquitous Computing*, ser. UbiComp '11, 2011, pp. 345–354.

[15] D. N. Rakhmatov and S. B. Vrudhula, "An analytical high-level battery model for use in energy management of portable electronic systems," in *IEEE/ACM international conference on Computer-aided design*. IEEE Press, 2001, pp. 488–493.

[16] S. B. Kotsiantis, D. Kanellopoulos, and P. E. Pintelas, "Data preprocessing for supervised leaning," *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 1, no. 12, pp. 4091 – 4096, December 2007.

[17] D. Basak, S. Pal, and D. C. Patranabis, "Support vector regression," *Neural Information Processing-Letters and Reviews*, vol. 11, no. 10, pp. 203–224, October 2007.

[18] M. Maltamo and A. Kangas, "Methods based on k-nearest neighbor regression in the prediction of basal area diameter distribution," *Canadian Journal of Forest Research*, vol. 28, no. 8, pp. 1107–1115, August 1998.

[19] S. Weisberg, *Applied linear regression*. John Wiley & Sons, 2005.

[20] D. F. Specht, "A general regression neural network," *IEEE Transactions on Neural Networks*, vol. 2, no. 6, pp. 568–576, November 1991.

[21] R. Anderson-Sprecher, "Model Comparisons and R^2," *The American Statistician*, vol. 48, no. 2, pp. 113–117, April 1994.

[22] G. Andrew and J. Gao, "Scalable training of L1-regularized log-linear models," in *24th International Conference on Machine Learning*. ACM, 2007, pp. 33–40.

[23] 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, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, October 2011.

[24] Y. Chon, H. Shin, E. Talipov, and H. Cha, "Evaluating mobility models for temporal prediction with high-granularity mobility data," in *IEEE International Conference on Pervasive Computing and Communications*, March 2012, pp. 206–212.

[25] A. Acquisti, L. Brandimarte, and G. Loewenstein, "Privacy and human behavior in the age of information," *Science*, vol. 347, no. 6221, pp. 509–514, January 2015.

[26] H. Almuhimedi, F. Schaub, N. Sadeh, I. Adjerid, A. Acquisti, J. Gluck, L. F. Cranor, and Y. Agarwal, "Your location has been shared 5,398 times!: A field study on mobile app privacy nudging," in *33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2015, pp. 787–796.