Edge computed NILM: a phone-based implementation using MobileNet compressed by Tensorflow Lite

Shamim AHMED†
Research and Development
FLUDIA
Suresnes, Hauts-de-Seine, France
shamim.ahmed@fludia.com

Marc Bons
Research and Development
FLUDIA
Suresnes, Hauts-de-Seine, France
marc.bons@fludia.com

ABSTRACT

In the context of residential Non-intrusive load monitoring (NILM), the usual service deployment process consists of collecting data from a metering device to the cloud, run algorithms on the cloud and then display results in a Web front or in an App. This approach comes with two major problems: on the one hand, important resources are allocated to the cloud process (including maintenance) while selling the solution on a substantial subscription basis is still a challenge. On the other hand, end-users are more and more reluctant to see their personal data being uploaded. In order to propose an alternative, this research has focused on edge computed NILM, namely the possibility to run NILM algorithms on existing devices on the end-user side, such as a smart phone. A two-stage model development has been carried out to obtain good disaggregation accuracy with lower model size. In the first stage, an efficient deep learning algorithm (MobileNet) has been adopted to obtain an accurate and light weight model. In the second stage, TensorFlow Lite has been used to compress further, in order to reduce edge device memory usage and computing time. To deal with real-life diversity, we have built large and diverse training and testing sets based on a combination of HES, UKDALE and REFIT datasets. Disaggregation performance has been assessed for both models: before and after TensorFlow Lite compression. Comparative analysis has been performed to facilitate implementation choices.

CCS CONCEPTS

• Computing methodologies • Machine learning • Learning paradigms • Supervised learning • Supervised learning by regression

KEYWORDS

NILM, Energy Disaggregation, Edge Computing, Deep Learning, MobileNet, TensorFlow Lite.

ACM Reference format:
Shamim Ahmed and Marc Bons. 2020. Edge computed NILM: a phone-based implementation using MobileNet compressed by TensorFlow Lite. In *Proceedings of NILM'20, November 18, 2020, Virtual Event, Japan, 5 pages.* https://doi.org/10.1145/3427771.3427852

NILM'20, November 18, 2020, Virtual Event, Japan

ACM ISBN 978-1-4503-8191-8/20/11…$15.00
https://doi.org/10.1145/3427771.3427852

1 INTRODUCTION

Application of deep learning models for detecting appliance operation [1-3] from aggregated data is becoming more popular. A use case of such energy disaggregation (known as NILM [4]) is household energy management where individual appliance energy consumption is estimated from single meter reading of the whole house. The main goal is to reduce energy consumption as well as monthly energy bill by focusing on the main individual appliance energy consumption. This helps the end-users reducing their bill and carbon footprint by changing hours of appliances usage from peak hour to off peak hour or by reducing abnormal energy consumption for some appliances. Traditional deep learning-based energy disaggregation approaches require deployment of trained models on a dedicated server having high computational power to make the inference. End users can access estimation of appliance energy consumption by connecting the server through web or dedicated API. Almost all the computation is placed on the sever end. This requires high scalability and regular server maintenance when dealing with large number of end users. In a competitive market, it is also a challenge to convince end-users to use such solution based on a subscription fee. On the other hand, due to data privacy issues, end-users are becoming more concerned to share personal data (e.g., energy consumption) in cloud-based services. To deal with these issues, this paper discusses a step by step approach for developing edge computed energy disaggregation solution. Real life applications need to deal with highly diverse scenario: differences in appliance brand, user's habit for appliance operation, varieties of available appliances in different houses etc. To deal with such variability, a combination of HES [5], UKDALE [6] and REFIT [7] datasets are used for training, validation and test sets generation. Sequence-to-point [2] approach has been utilized for features generation. A two-stage model development has been carried out to obtain efficient training model but lower model size for faster inference in the edge device. At the beginning a lightweight training model has been obtained by applying modified version of MobileNets [8]. The obtained model has been compressed in later stage by using TensorFlow Lite [9]. Finally, the compressed model has been integrated into android device.

The paper is organized as follow: Section 2 describes the formulation of NILM and training set development. Model development and compression are discussed in Section 3. Section 4 presents the experimental setup with data treatments. Finally, Section 5 is dedicated for performance evaluation of the applied approach followed by some concluding remarks.

2 PROBLEM FORMULATION

This section summarizes the formulation of NILM strategy. Suppose the mains $\boldsymbol{x(t)}$ represents the aggregation of the active power of all individual appliance in a household at time *t*, then the main can be represented by

$$\boldsymbol{x(t)} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{I}} \boldsymbol{y_i\,(t)} \quad (1)$$

where $\boldsymbol{y_i(t)}$ represents the power consumption of *i*-th appliance at *t* time from *I* number of available appliances. Knowing the aggregated signals $\boldsymbol{X} = \{\boldsymbol{x(t)}\}_{\boldsymbol{t=1}}^{\boldsymbol{T}}$ for *T* duration, the goal is to obtain the energy consumption of the *i*-th appliance $\boldsymbol{Y_i} = \{\boldsymbol{y_i(t)}\}_{\boldsymbol{t=1}}^{\boldsymbol{T}}$ $(\boldsymbol{i = 1, \dots, I})$. By applying sequence-to-point strategy [2], a training dataset $\boldsymbol{D_{N,i}}$ for the *i*-th appliance of $\boldsymbol{N}$ samples is built. Here $\boldsymbol{D_{N,i}} = \left\{\left(\boldsymbol{\psi_i^{(n)}}; \boldsymbol{y_i^{(n)}}\right); \boldsymbol{n = 1, \dots, N;\ i = 1, \dots, I}\right\}$, $\boldsymbol{\psi_i^{(n)}}$ is the feature map obtained from aggregated signals $\boldsymbol{X}$ and $\boldsymbol{y_i^{(n)}}$ represents the corresponding appliance power of the of $\boldsymbol{n}$-th sample. By using the training dataset $\boldsymbol{D_{N,i}}$, a supervised training model is obtained for each *i*-th appliance by applying deep convolutional network (i.e., MobileNet).

3 MODEL DEVELOPMENT

In the first phase of the model development, a modified version of MobileNet (deep convolutional network) has been used. MobileNet architecture is built on depth wise separable convolution [8]. The depthwise separable convolution factorizes a standard convolution into a depthwise convolution and a 1×1-point wise convolution. The most important benefit of depth separable convolution is the requirement of lower computation time for large and complex convolutional network with respect to standard convolution. For example, due to the use of 3×3 depthwise separable convolution applied in MobileNet, the computation becomes 8 to 9 times faster with respect to the use of standard convolution [8]. Unlike original MobileNet architecture, we have used ReLU-6 non-linear activation instead of ReLU activation in all convolution layers and feeds into a sigmoid layer for doing regression task. The input and output size of the network are 359 and 1, respectively. Usage of water heater is very common in French residents. Most of the programmed water heaters are usually active from 2 hours to 6 hours during off peak hour. In order to capture a full signature of a water heater like appliance, we have used 6 hours long input (i.e., 359) size at 1-minute interval. Dealing with large amount of training data, deep learning model is likely to be affected by overfitting. As a result, the model may not be generalized well for unseen data. To deal with this problem, kernel and bias regularization penalties are used in each convolution block through L2 regularization. The definition of this L2 regularization is very crucial for increasing model accuracy for unseen data. A fine tuning of L2 parameters is done to choose the optimal one within a range [10-1, 10-8]. The batch size and epoch have been chosen for 128 and 200, respectively with enabling early stopping option while monitoring validation error. Adam optimization technique has been used with the optimization parameters, learning rate=10-6, beta_1=0.9, beta_2=0.999, epsilon=10-8. The development was done on Keras library with TensorFlow Backend. The training and testing have been done on a desktop computer having configuration Intel Core i5-8400 CPU @2.80GHz×6, 64GB RAM, GPU GeForce GTX 1080 TI.

In the second phase of the development, the trained models need to be deployed on the edge device. This step requires the compression of the model size. Smaller size model takes smaller storage size in the users' device. It reduces usage of RAM and lowers inference time. Moreover, latency reduction by mean of inference time reduction has impact on power consumption of the device. These benefits can be obtained by using TensorFlow Lite, which includes post training quantization functionality. This post training quantization is done under the hood of optimization by lowering the precision of the learning parameters (e.g., MobileNet weights) from their training-time 32-bit floating-point representations into much smaller and efficient 8-bit integer ones. For example, each of the obtained trained model size from first stage is compressed from 39.1MB to 12.8MB after the quantization process. Performing such quantization may potentially have some impacts on the model accuracy depending on the model architecture development and the application domain.

Finally, the compressed model has been integrated in the Android platform in order to access performance of the developed model on the edge device. All android simulation results presented in this work have been performed on android SDK using an android emulator Nexus 6 API 26. The emulator has 2CPU with 1.5GB RAM hosted on a MAC having configuration 1,8 GHz Dual-Core Intel Core i5 and 8GB RAM. In Section 5, we have analyzed the performance evaluation of developed model before and after the quantization process (i.e., on android device).

4 EXPERIMENTAL SETUP

For developing effective and most accurate training model, the necessity of larger dataset is unavoidable. On the other hand, the training data sets should contain enough diversity to be generalized for the unseen data sets from real households. Therefore, a large training dataset consisting publicly available data sets HES, UKDALE and REFIT has been built for treating model development. All these 3 data sources have meter data in different time intervals. In order to make a common data sampling framework, 1-minute sample interval has been chosen by down sampling HES data set and up sampling UKDALE and REFIT data sets. The availability of plug&play sensors (e.g., FM432e [9]), such as the IoT sensors developed by FLUDIA that can accurately read existing electricity meters with 1-minute interval, motivates us to

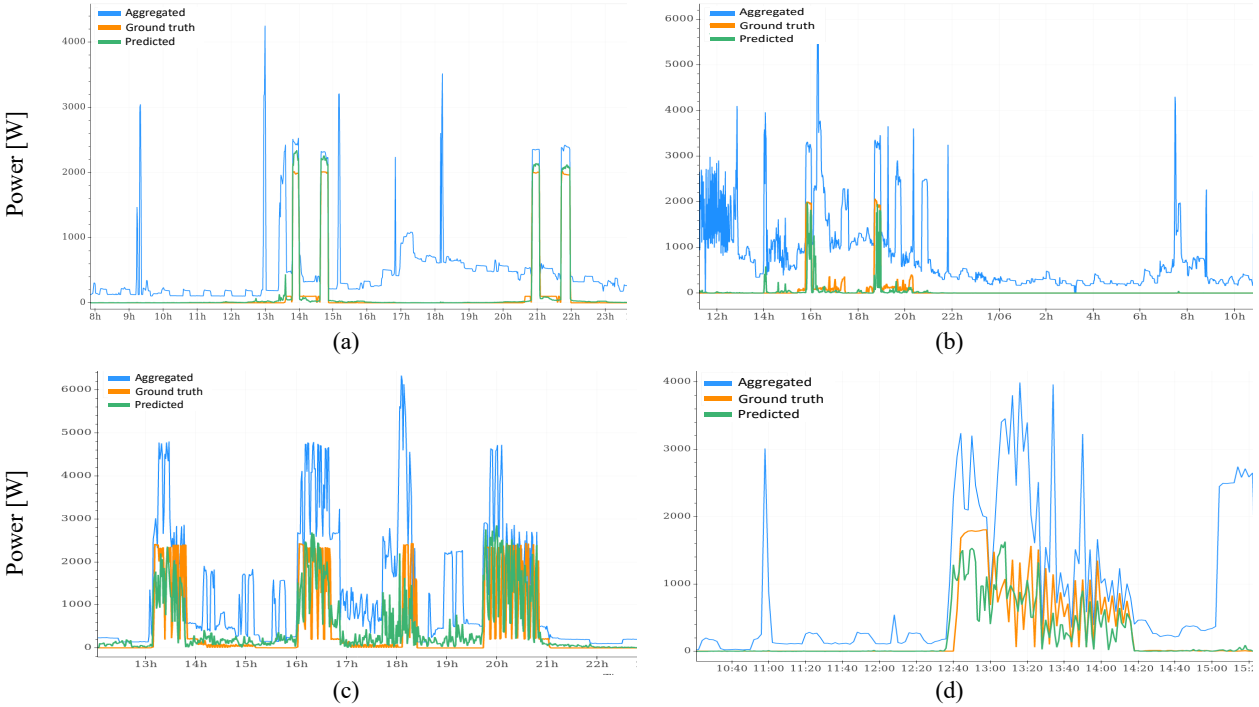

Figure 1: Disaggregation results obtained by MobileNets for a) dishwasher, b) washing machine, c) dryer and d) cooker.

Table 1: Number of houses used for obtaining training, validation and test set

Appliance	Training Set			Validation Set			Test Set		
	HES	UKDALE	REFIT	HES	UKDALE	REFIT	HES	UKDALE	REFIT
Dishwasher	103	3	5	3	0	0	5	1	6
Washing Machine	156	1	7	5	0	0	4	1	5
Dryer	65	0	4	2	0	0	2	0	4
Cooker	113	0	0	4	0	0	10	0	0

choose 1-minute data interval. HES data source contains 250 household's meter and submeter readings. However, a subset of the 250 houses are used for our analysis. Houses having more than 15 individual sub meter readings with better quality data (e.g., presence of appliance activations) are chosen for our analysis. In this work, four separate models have been trained for four appliances "of interest": washing machine, dishwasher, dryer, cooker. Due to the limitation of number of available cooker data samples, stove, oven and cooker are combined to increase the size of the training data set and named as cooker. Table 1 shows the number of different houses used for creating training, validation and test sets. We mainly focused on the presence of individual appliance for treating different houses. For each appliance, training and validation sets contain meter readings from different houses. Test set data samples from the houses are fully unseen. That means, we trained on the data samples from a collection of houses and tested on data samples from a collection of totally different houses.

5 PERFORMANCE EVALUATION

One of the complexities of our approach is the treatment of data set at 1-minute interval. By doing so, detailed characteristics of the appliance signature are lost compare to higher frequency datasets (< 1-minute data interval). Similarly, by down sampling REFIT and UKDALE data sets, we reduced the size of the training set. Thanks to the large HES dataset that we have used for minimizing this short coming with higher diversity. The disaggregation results on the unseen test data have been shown in Figure 1. Most of the appliance signatures are well predicted. According to our experience, estimation of cooking energy consumption is the most complicated one among the four targeted appliances. The different cooking elements used in different houses, the number of available persons in the household and user habit of cooking (number of times and duration of cooking) are as many factors that contribute to create a very large variety of different cooking signatures. However, MobileNet shows reasonable prediction performance for estimating cooking energy consumption, thanks to its learning capabilities obtained from very deep convolutional network. Due to the model compression step by TensorFlow Lite, we are expecting performance degradation from actual training model. We have differentiated model performance based on before and after model compression by means of actual prediction through MobileNet and prediction on android device by compressed MobileNet. Figure 2 shows the actual vs. predicted appliance energy consumption comparison on different houses.

Due to the variability of the treated data duration of each test house, the results show per week energy consumption of each appliance. Due to the robustness of the trained model, disaggregation performance shows promising results on estimating energy consumption. At the same time, model compression couldn't have large impact on model accuracy while running on android device. For quantitative analysis, average relative error ($\boldsymbol{RE_{avg,i}}$) in total energy consumption for the *i*-th appliance has been used.

$$\boldsymbol{RE_{avg,i}} = \frac{\mathbf{1}}{\boldsymbol{H}} \sum_{\boldsymbol{h=1}}^{\boldsymbol{H}} \left(\frac{|\boldsymbol{E_{acual}} - \boldsymbol{E_{predicted}}|}{\max(\boldsymbol{E_{acual}}, \boldsymbol{E_{predicted}})} \right)_{\boldsymbol{h,i}} \quad (2)$$

where $\boldsymbol{E_{acual}}$ and $\boldsymbol{E_{predicted}}$ are actual and predicted total energy consumption, respectively, $\boldsymbol{H}$ is the number of available houses in the test set. Table 2 represents the comparison of $\boldsymbol{RE_{avg}}$ on different appliances. As expected, actual MobileNet model shows 3.7% to 12.6 % $\boldsymbol{RE_{avg}}$ for estimating dishwasher, washing machine, dryer and cooker energy consumption. While running the models on android device, $\boldsymbol{RE_{avg}}$ ranges from 8.39% to 16.69%. Compare to actual MobileNet prediction, prediction error increases for all the appliances except washing machine on android device. At UKDALE house 2 (Figure - 2b), washing machine energy consumption was slightly improved while running on android device. This contributed for overall prediction accuracy at android device for estimating washing machine energy consumption. Moreover, running the compressed models on android device deviate up to around ~ 9.4% error from original model's accuracy.

6 CONCLUSIONS

In this paper, we have presented a comprehensive overview of energy disaggregation solution running on edge device. At first, we have developed efficient deep learning model using modified version of MobileNet convolution network. Then, TensorFlow Lite has been used for post training model compression. The compressed model has been deployed on android device. The developed solution has been trained and tested on publicly available datasets. Finally, performance evaluation for energy disaggregation are shown for both before and after model compression. Due to the model compression, disaggregation accuracy deviates up to ~ 9.4% from original disaggregation model, but, on average, remains satisfactory. Such deviation can be accepted when providing cheaper and scalable energy disaggregation solution computed on android device. Moreover, this approach will also give us the opportunities for developing edge computed disaggregation solution for other type of edge devices (e.g., IOS, microcontroller etc.).

Table 2: Comparison of average relative error $\boldsymbol{RE_{avg}}$ in (%) for estimating appliance energy consumption.

Appliance	Actual prediction	Prediction on android
Dishwasher	3.71	11.54
Washing machine	12.6	8.39
Dryer	11.04	16.69
Cooker	6.49	15.87

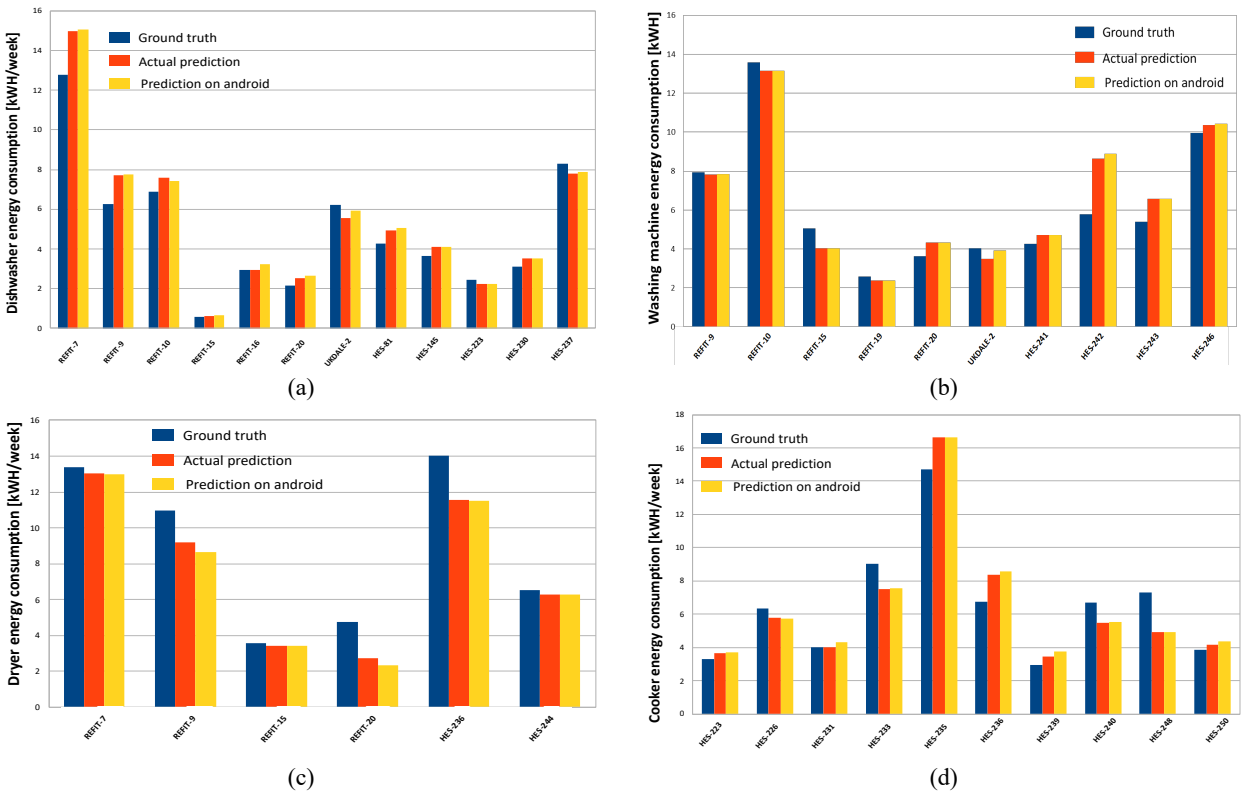

Figure 2: Per week energy consumption comparison for a) dishwasher, b) washing machine, c) dryer and d) cooker.

ACKNOWLEDGEMENTS

This work has been partially supported by “Agence de l'environnement et de la maitrise de l'energie (ADEME)” under the project FUSINI.

REFERENCES

[1] Kelly J., and Knottenbelt W. 2015. Neural NILM: Deep neural networks applied to energy disaggregation. In Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments, 55–64. DOI: 10.1145/2821650.2821672

[2] Zhang C., Zhong M., Wang Z., Goddard N., and Sutton C. 2018. Sequence to-point learning with neural networks for nonintrusive load monitoring. In The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), 2018.

[3] D’Incecco M., Squartini S., and Zhong M. 2020. Transfer Learning for Non-Intrusive Load Monitoring. IEEE Trans. Smart Grid 2020, 11, 1419–1429, DOI: DOI: 10.1109/TSG.2019.2938068

[4] Hart. G. W. 1992. Nonintrusive appliance load monitoring. In Proceedings of the IEEE, 80(12):1870-1891(Dec, 1992). DOI: 10.1109/5.192069

[5] Zimmermann J.-P., Evans M., Griggs J., King N., Harding L., Roberts P., and Evans C. 2012. Household Electricity Survey. A study of domestic electrical product usage. Technical Report R66141, DEFRA, (May 2012).

[6] Kelly J., and Knottenbelt W. 2014. The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes. ArXiv:1404.0284v3 [cs.OH] (Mar 2015). DOI:10.1038/sdata.2015.7

[7] Murray D., Stankovic L., and Stankovic V., 2017. An electrical load measurements dataset of United Kingdom households from a two-year longitudinal study. Sci. Data 4, (January 2017). DOI: https://doi.org/10.1038/sdata.2016.122

[8] Howar A.G., Zhu M., Che b., Kalenichenko D., Wang W., Wey T., Andreetto M., Adam H. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. (April 2017), ArXiv: 1704.048861

[9] TensorFlow Lite | ML for Mobile and Edge Devices. 2020. TensorFlow. [online] Available at: https://www.tensorflow.org/lite [Accessed 21 August 2020].

[10] Fludia.com. 2020. Fludia - Smart Energy Monitoring Solutions - ELECTRICITY. [online] Available at: https://www.fludia.com/-ELECTRICITE-188-.html?lang=en [Accessed 21 August 2020].