Edge Machine Learning for Energy Efficiency of Resource Constrained IoT Devices

Dawit Mengistu
Department of Computer Science
Kristianstad University
Kristianstad, Sweden
Email: dawit.mengistu@hkr.se

Fredrik Frisk
Department of Computer Science
Kristianstad University
Kristianstad, Sweden
Email: fredrik.frisk@hkr.se

***Abstract*— The recent shift in machine learning towards the edge offers a new opportunity to realize intelligent applications on resource constrained Internet of Things (IoT) hardware. This paper presents a pre-trained Recurrent Neural Network (RNN) model optimized for an IoT device running on 8-bit microcontrollers. The device is used for data acquisition in a research on the impact of prolonged sedentary work on health. Our prediction model facilitates smart data transfer operations to reduce the energy consumption of the device. Application specific optimizations were applied to deploy and execute the pre-trained model on a device which has only 8 KB RAM size. Experiments show that the resulting edge intelligence can reduce the communication cost significantly, achieving substantial savings in the energy used by the IoT device.**

Keywords- Edge intelligence; IoT; Smart Sensors; RNN.

I. INTRODUCTION

Several IoT applications have emerged in healthcare with advances in wearable electronics [6], [11]. Miniaturized devices having sensing, computing and communication capabilities transformed the healthcare sector, enabling the realization of new services. Wearable devices can collect data for monitoring patients remotely to get insights on symptoms or trends, and provide better treatment.

Typical healthcare IoT services use wearable devices in combination with smartphones. Physical and physiological data collected by the wearable sensors is sent to the smartphones where it is aggregated and transferred to backend applications for further processing. The backend often consists of a number of Cloud services for data storage, analytics and machine learning needed to provide actionable information to physicians and patients [10].

The motivation for this work comes from an ongoing research aimed at mitigating Musculo-Skeletal Disorders (MSD) problems in sedentary work environment. In an effort to establish a large dataset for this research, participating subjects were identified for collecting posture data. The data collection is performed continuously for several hours a day, over a long period. In order to build a comprehensive dataset, out of work activities requiring sedentary postures (such as driving) will also be included in the data collection [4].

Energy efficiency is a major challenge in the adoption of wearable IoT for such studies because most devices used in these applications are energy constrained, often running on low capacity power sources. In our case, multiple, coin-cell battery operated wireless devices equipped with inertial sensors are worn by the subjects. Communication between the devices and the smartphone takes place via a Bluetooth Low Energy (BLE) interface. However, the batteries of the devices last few hours only because of the volume of data they transfer. For example, a wearable motion sensor with 9 channels reading 50 samples per second generates over 100MB of data per day.

In this paper, we shall present an approach to improve energy efficiency of wearable sensors through Edge Machine Learning techniques. Our goal is to reduce the volume of communication between the sensor devices and the smartphones to the minimum needed. The machine learning implementation shall recognize eventful data and transfer it to the smartphone only when it is necessary. Implementing machine learning algorithms on resource constrained devices is often not practical because the algorithms require adequate computing power and large storage memory, both of which are not available on most wearable devices. However, with the emergence of edge computing, it has been possible to handle most of the computational and storage burden of machine learning far away from the source of the data.

We investigated different machine learning algorithms to identify the ones that suit our task. Our findings show that RNNs can be implemented on a resource-constrained edge device and give the desired accuracy in real-time. As a proof-of-concept, we evaluated the execution performance and accuracy of a pre-trained RNN model on an Atmega640 microcontroller. The Atmega640 is an 8-bit microcontroller with 16MHZ clock, 64KB boot (code) memory and 8KB data memory (static RAM). This microcontroller has lower specifications (processor speed and memory) than typical devices used for such applications. It can therefore be said that the results of our experiment can be applied to devices already adopted by the wearables industry.

A Python based Machine Learning library was used to build and pre-train our RNN model. Experiments were run to determine the optimal set of model parameters that fit in the device without sacrificing accuracy significantly.

Posture data collected for the research is used to train the model. We then developed a program in C to implement the pre-trained RNN and deployed it on the sensor device for evaluation. The model's real-time performance on the edge device is found to be satisfactory for posture monitoring in sedentary work environment.

The rest of this paper is organized as follows. Section II gives a brief background on MSD research and the state-of-art in the area. Section III discusses Edge Machine Learning, its challenges and contemporary research in the field and the approach proposed for the task at hand. Section IV describes the experimental setup for this study. Section V discusses the results of the experiments followed by analysis of the results. Section VI discusses related work in Edge Machine Learning research. We conclude this paper by highlighting important results and citing directions of future work.

II. MSD RESEARCH AND APPLICATION OF IOT

A. A Brief Background to the Research Problem

Sedentary work environment was recognized as one of the major causes of MSD. Studies on the issue found that prolonged seating and poor body postures can reduce blood flow in the cervical region and cause inflammation of muscles and tissues [1]. Poor postures and work positions result in back, neck and shoulder pain. MSD problems may also lead to chronic ailment and even complete immobility. Studies explain the need to maintain the right body posture in work and different daily-life situations, owing to the fact that bad posture and prolonged sitting in one position cause back, neck and shoulder pains that can get worse and develop to chronic diseases with age [3]. Studies have shown the link between sedentary life and the risk of obesity, diabetes, cardiovascular disease, and all-cause mortality.

Increasing healthcare costs, absence from work and the associated negative psychosocial complications are some of the problems caused by MSD with severe impact at societal level. According to the UK National Health Services, the country lost 31 million days in 2014 alone, due to sickness related to back, neck and muscle pain [4]. Recognizing the severity of MSD problems, the National Institute for Occupational Safety and Health in the United States (NIOSH) identified the problem as an important research agenda [8]. NIOSH identified several research directions, among which mechanisms for reducing the impact of MSD is a priority area.

B. IoT in MSD Research

The dataset created in earlier MSD researches were either incomplete or inaccurate due to the method of data collection they employed. In many cases, the data gathering process was based on physical observation or self-reported information [2]. Later researches made use of video recording and tagging [7]. With advances in sensor and communication technologies, it was possible to set up body sensor networks (BSN) that connect multiple devices worn on the subject's body to detect and label movements and postures [5]. Different types of sensors are used today to collect physical and physiological data to capture information on movements, postures, spinal loads, sit-stand frequency, metabolic processes, etc. [1].

The emergence of IoT transformed the field of healthcare by facilitating real-time data collection, monitoring and analysis with greater convenience and ease of use. Cables that were once used to connect wearables to a central data acquisition unit are now replaced with a wireless interface, such as BLE integrated into the devices.

A schematic depicting the setup of an IoT environment for acquisition and storage of data is shown in Figure 1.

The devices needed for this specific study are placed on the center of the upper back area and on the upper part of the left leg. This placement is sufficient to identify sedentary postures and detect whether the subject is sitting or standing. The devices have inertial sensors, (accelerometer, magneto-meter and gyro) and a BLE unit for communication with the smartphone.

The data gathered is stored in a Microsoft Azure Cloud storage as a time series consisting of 9 features (along x, y, z axes for each inertial sensor unit) per sensor node or device. Every row of data is timestamped and contains posture labeling as well.

III. MACHINE LEARNING AT THE EDGE

Edge devices used for sensors are often resource constrained and therefore not capable of running classical machine learning applications on their data. For such devices, both training the model and inference are often carried out far from the origin of the data, in the Cloud. The main innovation in Edge Machine Learning (Edge ML) is that inference can take place on the edge device itself, at the source of the data. Edge ML has several benefits, such as reduction of communication latency, improving energy efficiency, security, personalization and customization of services [9]. Achieving energy requires reducing the energy cost of communication between the sensor edge and the smartphone, which in turn requires reducing the flow of redundant data from the edge device (sensor).

There are important steps that should be investigated to gain from the mentioned benefits of Edge ML. First, identifying the right machine learning algorithm for the dataset at hand requires expertise and skills. Selecting the right algorithms often comes with the dilemma to choose between performance and accuracy.

Second, the capability of the target device to support the algorithm is not a trivial problem. In order to address this issue, several models have to be tested on the target device until a satisfactory one is found. One is often forced to sacrifice the accuracy if the target hardware does not have sufficient processing power or memory.

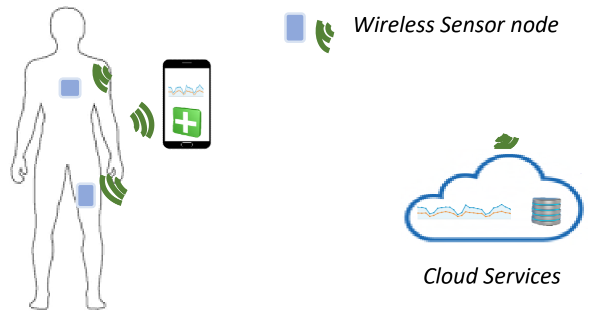

Figure 1. Placement of inertial measurement sensors

Finally, the availability of development tools for machine learning is also a major challenge. Embedded software for many low-end edge devices are written in C. Development environments of some microcontrollers are proprietary with limited flexibility, making importing available libraries very difficult.

Machine learning often requires complex software packages and libraries that can only be executed on powerful processors. Until recently, deep learning was outside the realm of low-end processors on which many IoT devices are based. Because edge devices are resource constrained, they require customized implementations for most machine learning algorithms. The following techniques can be applied in special cases to realize edge ML:

- Offloading the computational work to more powerful devices, for example by performing the training and validation phases on the Cloud;
- Reducing the precision of model parameters and approximating computations with more efficient arithmetic operations wherever possible [20];
- Using lookup tables for activation functions instead of run-time computations.

A. Deep Learning for Temporal Data

One limitation of classical sensor data analysis is the need for manual feature engineering work. Most supervised-learning algorithms are not computationally efficient for deployment on resource constrained devices. Algorithms, such as K-Nearest-Neighbour (KNN) have large storage requirements that can only be met by desktop computers or servers. Furthermore, these algorithms are not suited to detect patterns or contexts hidden in the temporal data collected by the sensors.

RNNs are effective for data with temporal or contextual sequence, such as natural language processing and time series prediction [16]. Their ability to read variable-length sequences of input samples and merge the prediction for each sample into a single prediction for the entire window makes RNN suitable for the posture monitoring application under consideration. A generalized schematic of the RNN architecture is shown in Figure 2. Current hidden states are generated using the input and the previous hidden state. This cyclic behavior in the hidden layer gives the network the ability to learn temporal sequences.

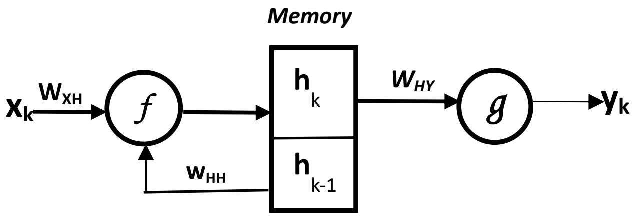

Figure 2. Representation of a recurrent neural network

The mathematical model of the network is represented with the following equations:

$$\mathbf{h}_k = f(\mathbf{W}_{XH}\mathbf{x}_k + \mathbf{W}_{HH}\mathbf{h}_{k-1} + \mathbf{b}_H) \qquad (1)$$

$$\mathbf{y}_k = g(\mathbf{W}_{HY}\mathbf{h}_k + \mathbf{b}_y) \qquad (2)$$

where

k represents time sequence;

x, **y**, and **h** are the input output and hidden state vectors respectively;

$\mathbf{W}_{XH}$, $\mathbf{W}_{HH}$ and $\mathbf{W}_{HY}$ are the input-to-hidden, hidden-to-hidden and hidden-to-output weight matrices respectively;

$\mathbf{b}_H$ and $\mathbf{b}_y$ are the bias vectors for the hidden and output states respectively (not shown in the figure);

f and g are non-linear activation functions.

RNN models can often be inaccurate and unstable for long input sequences and time series, due to the exploding and vanishing gradient problems [17]. The Long Short-Term Memory (LSTM) variant of RNN was proposed as a solution to overcome the problem. LSTM has achieved impressive results with sequential and time series data in applications, such as text generation, sequence prediction and anomaly detection [16][21].

B. Realization of RNN on Constrained Edge Devices

Deploying a deep learning model on resource constrained edge device, such as an 8-bit microcontroller requires significant optimizations. We shall explore where these optimizations can be applied for our specific use case. Since the number of input features and outputs is already decided by the application, one has to identify other areas to look into. Several models have to be built and tested to arrive at an acceptable one.

One optimization measure is to determine the number of neurons in the hidden layer because the computational complexity and memory requirements for neural network grow exponentially with it. Models of different sizes should be evaluated experimentally for acceptable accuracy and matching the edge device's resource capabilities. It is also possible to achieve a lower count on model parameters by pruning edges with negligibly small weights [12] [19].

Further optimization is achieved through input data reduction. The sampling rates of the sensors are often too high for the microcontroller to make inference from the acquired data within a sampling interval. Applying computationally inexpensive low pass filters helps to reduce the volume of data, to improve the quality of the data and get sufficient time interval for making inference.

Another optimization opportunity is simplifying the computation of activation functions. If the microcontroller does not have built-in floating point capabilities, evaluation of functions, such as ***sigmoid*** and ***tanh*** is expensive. Sacrificing the accuracy of these functions to a reasonable level reduces the time needed for inference significantly.

IV. EXPERIMENTS

We planned two specific tasks in this experiment. The first is to realize an optimized implementation of the algorithm and the model parameters that can fit into the available memory of the target microcontroller. Several models are built to evaluate the tradeoff between accuracy and model size. The second task is to evaluate the performance tradeoff between inference accuracy and the achievable saving in energy on the edge device.

A. Data Collection

A smartphone app is also developed for this experiment, to take care of the data received from sensors, as shown in Figure 1. The main functions of the app are labelling, time stamping, aggregating and uploading data to Cloud storage.

A sampling frequency of 50HZ is used as the base rate. We decided not to increase the sampling rate beyond this as the sensor node's microcontroller would not be able to make predictions in real time if the rate is increased. The sensors measurements are different, owing to the local movement of the body area they are placed on.

The data used in this experiment was collected over a period of 30 minutes with the subject assuming different predetermined postures alternately. This data is used to train a machine learning model which should detect the temporal instances of posture transitions. When training and testing is completed, the model can be deployed on the target.

One of the investigators supervised the subjects to wear the sensors at the correct position and guided them to change postures every 2 to 3 minutes. During a transition, i.e., when the subject changes her posture, a label for the new posture is entered via the app's user interface so that all data received from this time on will automatically have the new label until the next posture change occurs.

The postures assumed are labelled as follows:

- *Sit -upright*
- *Sit -lean left*
- *Sit -lean right*
- *Sit -lean forward*
- *Sit -lean backward*
- *Stand*

B. Training the Model

We implemented our machine learning model using the Tensorflow deep learning framework in Python. The model has 9 feature inputs, one hidden layer (LSTM) and one scalar output, for each sensor node. Since we are interested in posture transition only, a binary classifier is sufficient to detect local posture changes. There are 90000 records in the dataset, split into train (90%) and test (10%). A 70:30 train-to-test ratio was also used later for comparison. The python code was executed for different number of neurons in the hidden layer. The model size depends on the number of input features and the number of neurons in the hidden layer. The experiment shows that the number of parameters can be obtained from the equation:

$$P = (4n+1)(n+1) + 4nk \qquad (3)$$

where

P = number of model parameters (weights)
k = number of input features
n = number of neurons in the hidden layer

The storage requirement for the model is 4P bytes with each parameter represented as a 4-byte floating point value.

Another parameter of interest in the experiment is the duration of the time lag window used by the LSTM layer for prediction. Larger window width is not practical for resource constrained devices as the model prediction time becomes unacceptably long. Smaller width on the other side compromises the accuracy of the prediction. The analytical computation of optimum window size is complex because it depends on several factors that cannot be easily quantified. We therefore determined this value empirically.

C. Deploying the Model on Target Device

The microcontroller version of the RNN code was written in C. The compiled version of the code takes 34KB of flash memory. The model parameters were combined with the source code and compiled. However, they were stored in the SRAM.

Modifications were made in the data acquisition part to include a low pass moving average filter and use a sampling rate of 50HZ. The filter serves the purpose of reducing the volume of data processed by the model in addition to stabilizing the data (against noise). The time taken to execute the model's inference task for different number of neurons in the hidden layer is evaluated to determine whether inference can be achieved in real time. Similar experiments are also run for different window sizes.

The optimum number of neurons *n* depends on the amount of SRAM available and the number of input features. Having fewer neurons is not desired as it would compromise the accuracy of the model. The results are summarized in the next section.

V. Results and Discussion

A. Evaluation of Model Training

Different hidden layer sizes were tested in the experiment. However, owing to the limitations in the target device, it is not practical to deploy large models. For example, the number of parameters for a model containing 50 neurons in the hidden layer is about 11,851. This requires about 44KB of memory on the target. The Atmel640 microcontroller has only 8KB static RAM that can be used for all temporary data. After accounting for the memory required to store a few seconds of sensor data, working memory and stack for intermediate computations, the available memory for the model parameters is just under 3.5KB. Applying equation (3), we can train a maximum of 11 neurons in the hidden layer.

Limiting the model size has also the additional benefit of eliminating the risk of overfitting. Our evaluation also shows that larger models are not suitable for the data. We got satisfactory performance with models having as few as 8 neurons. Figures 3 and 4 demonstrate this by comparing the Mean Square Error (MSE) losses for 50 and 8 neurons respectively in the hidden layer. These results show that the smaller model (8 neurons) has in fact a better accuracy because it is a close match to the number of input features.

B. Evaluation of Model Execution on the Target

The performance of the inference model was evaluated on the microcontroller for different sizes of time lag window. It is clear that the inference accuracy improves if a larger time window is used. However, this incurs a large computation cost. For a rate of 50 samples per second, the 20 milliseconds interval is very short to perform data acquisition and inference (prediction). As can be seen in Figure 6, it takes about 35 milliseconds to execute the inference step alone for a window size of 3 seconds.

It is necessary to find an acceptable tradeoff between inference accuracy and real-time response. We applied a low pass filter to stabilize the data by averaging 4 samples at a time, instead of feeding the entire sensor data stream to the ML model. The overhead of this filter is low compared to the inference operations. To our surprise, we got impressive accuracy even when a smaller window size is used. As can be seen in Figure 5, the difference in accuracy between a window size of 2 seconds and 3 seconds is not significant (both have over 95% accuracy). However, the 2 seconds window takes much lower time for inference.

It is possible to deploy a larger model in the flash memory since the SRAM would not be enough. Although the flash memory has a slower access time than that of SRAM, its performance is still acceptable.

C. Analysis of Energy Efficiency

Larger window sizes do not improve the model accuracy significantly. In fact, the improvement, if any, is outweighed by the overhead of the inference step. Because the inference overhead grows exponentially with the size of the window, the solution is not feasible for higher sampling rates. This can be seen in Figure 6.

The energy saving is calculated as the difference between the reduction in data transfer costs and the extra computation incurred by the inference step to achieve this reduction.

It follows that this saving is significant if the frequency of posture changes is low as is the case with sedentary work. The excess computation depends on the sensor data rate and the window size. In our experiment, a sampling rate of 50HZ and averaging every 4 samples is used. This gives the model an interval of 80 milliseconds per inference.

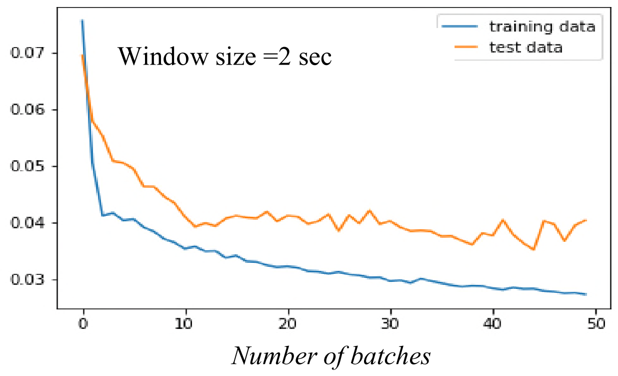

Figure 3. MSE loss (%) for number of neurons=50

As can be seen in Figure 6, if a 2 seconds window is applied, it takes about 9 milliseconds to execute the model (inference code). This achieves the desired result with about 11% increase in computation.

The BLE interface draws an average current of 8.53mA over its connection interval of 2.675 milliseconds with an empty payload [15]. According to the datasheet, the device draws a current of 17.5mA for a full payload data transfer. Android phones support a maximum of 4 packets with a-payload of 20 bytes for a minimum connection interval of 7.5 milliseconds [13].

To get an estimate of the energy saving, we can consider a case where posture changes occur every 10 seconds on average. If edge intelligence were not applied, 4.5KB of data would have to be transferred in the 10 seconds interval (from 9 channels at 50 samples per second and 1 byte per sample). With the above BLE throughput information, a data transfer period of 420 milliseconds over the 10 second duration at an average current of 17.5mA.

With the proposed approach, however, it would be enough to transfer only 225 bytes, the data for one time window only. This achieves a 95% reduction in data transfer costs. The energy consumed for the additional computational overhead is quite modest with the microcontroller drawing less than 2mA, about 10% increase.

VI. RELATED WORK

The emergence of low cost, yet powerful devices has brought machine learning to the edge. Encouraged by this development, researchers in the field have managed to achieve interesting outcomes in edge intelligence in the last few years. In most of the studies we found, the investigators tested their learning algorithms on powerful devices that are not suitable for wearable sensors.

Yazici et. al. investigated porting three different machine learning algorithms to Raspberry Pi running an embedded version of the Android OS [9]. They evaluated the performance of the algorithms for speed, accuracy and power consumption. However, their solution is realized only on powerful edge devices, not on resource-constrained 8-bit microcontrollers.

Gupta et. al. developed a KNN implementation that can run on 8-bit devices such as Arduino [18].

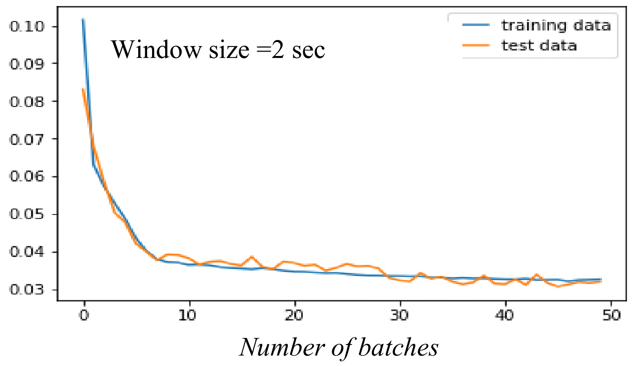

Figure 4. MSE loss (%) for number of neurons=8

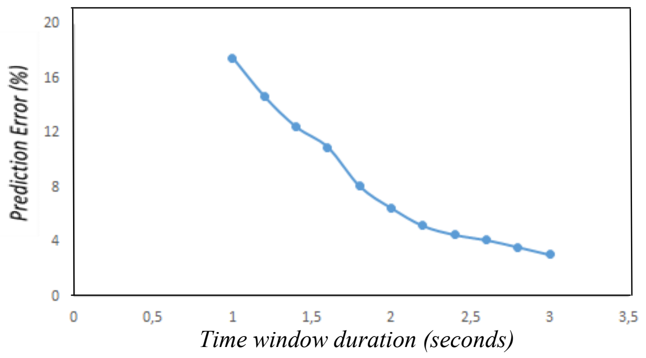

Figure 5. Model accuracy versus time window size

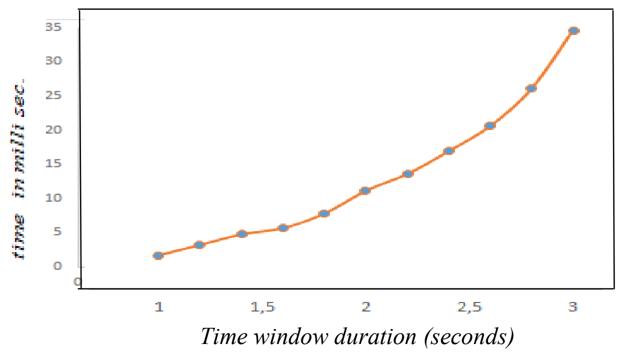

Figure 6. Execution time for inference model vs time window size

KNN is not, however, suitable for cases like ours where large datasets are required for training because the device does not have sufficient storage for the data.

Malhotra et. al. presented stacked LSTM networks for anomaly detection in time series. They evaluated their algorithms on different sensor data sets [16]. However, their algorithm is not ported to edge devices.

VII. Conclusions and Future Work

This study has shown that an LSTM-based Edge Machine Learning can bring about substantial improvement in energy efficiency of resource constrained IoT devices. Advanced machine learning platforms have significantly simplified the practical application of deep learning models by facilitating rapid prototyping and testing.

Due to physical and physiological differences in human beings, the models should be trained on an individual's own data. In our next study, we shall investigate personalized models rather than one-size-fits-all generic ones.

Though not empirically validated yet, we see that further optimizations can be made on the model. Pruning edge weights, applied by Han et al. [19], and exploiting inherent data types of sensor values can result in reduction in the computational overhead of the model. Utilizing binary neural networks [20] could also give interesting results. With this, it can be possible to deploy inference models on devices with even lower capabilities than the one used in this experiment.

References

[1] A. A. Thorp, "Prolonged sedentary time and physical activity in workplace and non-work contexts: a cross-sectional study of office, customer service and call center employees", International Journal of Behavioral Nutrition and Physical Activity Vol. 9, 2012.

[2] M. Graf, U. Guggenbuhl, and H. Krueger. "An assessment of seated activity and postures at five workplaces", International Journal of Industrial Ergonomics Vol. 15, pp. 81-90, 1995.

[3] P. Côté et. al., "The Burden and Determinants Of Neck Pain In Workers", Journal of Manipulative and Physiological Therapeutics Vol. 32, Iss. 25, 2009.

[4] J. Lennon, "Flexible study to fit around work", Occupational Health, Sutton, Vol. 67, Iss. 6, pp. 27-29, 2015.

[5] S. Kozey-Keadle et al., "The Feasibility of Reducing and Measuring Sedentary Time among Overweight, Non-Exercising OfficeWorkers", Hindawi Publishers, Journal of Obesity, 2012.

[6] S. M. R. Islam et al., "Internet of Things for Health Care: A Comprehensive Survey", IEEE Access, vol. 3, pp. 678-708, 2015.

[7] C. Marinac et al., "The Feasibility of Using SenseCams to Measure the Type and Context of Daily Sedentary Behaviors" International SenseCam Conference, Sandiego USA, 2013.

[8] W. S. Marras et al., "National occupational research agenda future directions in occupational musculoskeletal disorder health research", Elsevier Applied Ergonomics, 2009.

[9] M. T. Yazici, S. Basurra, and M. M. Gaber "Edge Machine Learning: Enabling Smart Internet of Things Applications", Big Data Cogn. Comput. 2018, 2, 26

[10] S. B. Baker, W. Xiang and I. Atkinson, "Internet of Things for Smart Healthcare: Technologies, Challenges and Opportunities", IEEE Access, vol. 5, pp. 26521-44, 2017.

[11] A. Godfrey et al., "From A to Z: Wearable technology explained", Elsevier, Maturitas, Vol. 113, pp. 40-47, 2018.

[12] I. Goodfellow et al., "Deep Learning", MIT Press, USA, 2016.

[13] C. Gomez, J. Oller, and J. Paradells. "Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology", Sensors Vol. 12, pp.11734-53, 2012.

[14] A. Kumar, S. Goyal and M.Varma. "Resource-efficient Machine Learning in 2 KB RAM for the Internet of Things", Proceedings of the 34 th International Conference on Machine Learning, Australia, 2017.

[15] "Measuring Bluetooth Low Energy Power Consumption", Texas Instruments http://www.ti.com/lit/an/swra347a/swra347a.pdf visited June 2019.

[16] Malhotra et al., "Long Short Term Memory Networks for Anomaly Detection in Time Series", Proc.s of 25th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Belgium, 2017.

[17] F. Chollet,"Deep Learning with Python", Manning Publ, 2018

[18] C. Gupta et al., "ProtoNN: Compressed and Accurate kNN for Resource-scarce Devices", Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 2017.

[19] S. Han, H. Mao and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding", International Conference on Learning Representations, San Juan, Puerto Rico 2016.

[20] M. Courbariaux et al., "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1", arXiv preprint arXiv:1602.02830, 2016.

[21] W. Yin, K. Kann, M. Yu, and H. Schütze. "Comparative Study of CNN and RNN for Natural Language Processing", arXiv:1702.01923, 2017.