Basman M. Hasan Alhafidh, et. al. International Journal of Engineering Research and Applications
www.ijera.com
ISSN: 2248-9622, Vol. 10, Issue 11, (Series-I) November 2020, pp. 31-37

RESEARCH ARTICLE OPEN ACCESS

Design and Implementation of Home Autonomous System Based on Machine Learning Algorithms

Basman M. Hasan Alhafidh[1], Amar I.Daood[2], Modhar A. Hammoudy [3]
[1]*Dept. of Comp. Engineering* College of Engineering *University Mosul, Iraq*
[2]*Dept. of Comp. Engineering* College of Engineering *University Mosul, Iraq*
[3]*Alhammoudy Dept of Comp. Engineering* College of Engineering *University Mosul, Iraq*

ABSTRACT
Home automation systems are cutting edge tech- nologies to monitor and control a smart home environ- ment to produce an efficient system that accurately predicts the needs of the human occupants. Past research has focused on the accuracy of prediction of a users future action. However, a focus on prediction accuracy often comes at the cost of slower processing time. additionally, a need of hardware implementation is necessary to assure the consistency of the simulation results. Finally, much of that work uses synthetic datasets which do not always reflect the real-world interactions that occur between an individual and the home environment. This paper focuses on the prediction of future human actions in an intelligent environment with the goal of achieving both high prediction accuracy and response times that are appro- priate for a real-time application environment. Using several different machine learning algorithms, both the simulation experiments and hardware implementation were accomplished using the MavPad dataset which was gathered from a fully-instrumented home environ- ment. In the first stage of this study investigates which machine learning algorithm will satisfy the conditions of real-time application. The findings show that neural network technique provides a feasible solution in term of accuracy. Going deeper, the authors use simulation to investigate the performance of a multilayer neural network that predicts future human actions. In the second stage of this work the authors present a hardware implementation of the deep learning model on a FPGA. The results showed that the hardware implementation demonstrated similar accuracy with significantly improved performance compared to the software- based implementation due to the exploitation of parallel computing and using optimization tech- niques to map the designed system into the target device. Furethermore, our implementation of FPGA- based neural network system supports its future uti- lization for other applications.

Keywords: Smart Home System, FPGA, Autonomous Sys- tem, Machine Learning Algorithms, Prediction System

---

Date of Submission: 28-10-2020 Date of Acceptance: 09-11-2020

I. INTRODUCTION

Consumers seek home automation systems that monitor and control a smart home environment to produce an efficient system that accurately predicts the needs of the human occupants. Some recent researches have presented the use of Machine Learning Algorithms (MLAs) to predict the present activity of an individual depending on sensors reading, and other works focused on the accuracy of prediction of next user action on home appliances. However, those studies used synthetic datasets that were generated based on assumptions which may not reflect the real-world interactions between an individual and their home environment. Also, there was often a focus on using MLAs that presented greater accuracy in prediction without considering the time constraints of real-time applications.

In this paper, we will not focus on the creation of models of a user's current activities or on generalizing occupant behavior inside the environment. Instead, we will focus on the prediction of next human actions on changes to the state of actuators inside an intelligent environment. Hence, the prediction process in such an environment needs a fast response speed from the designed prediction system with a high level of accuracy to satisfy the quality of the service based on real-time application domains. To assist the mentioned criteria, several simulation experiments were conducted to compare the average accuracy and average time of prediction results between different approaches of Nave Bayes (NB), Support Vector Machine (SVM), and Neural Network (NN). The implemented MLAs were applied to the MavPad dataset. This dataset was collected from sensors and actuators which distributed across a real-

world environment over seven weeks period. The experiments were divided into two zones, local zone and global zone. NN showed superior performance in term of accuracy and speed. On the other hand, NN showed adequate accuracy and response time in the global zone. The second stage of this work is FPGA implementation of our network to improve the response time. Both the preliminary results of Matlab simulation and FPGA hardware implementation show that a neural network classifier was accepted to meet the constrains of a real time system for the human prediction of a smart home system. Based on that observation, we went deeper by increasing number of layers in our NN hoping to see if we can enhance the accuracy under the mentioned constrains. Besides, the paper presents a performance analysis for the prediction model via the comparison between Software and Hardware Implementation. The comparison is an essential process to assure a real-time response not only in S.W. implementation experiment but also in a real hardware implementation experiment when the configured system predicts the needs of human occupants in a real-time basis.

The remainder of this paper is organized as follows. Section II presents a literature review. Section III presents our experi- mental design using the MavPad dataset, which was collected from a real-world environment [1]. Section IV discusses the simulation results using Matlab simulation program. In Section V, presents the results of FPGA implementation of multilayer NN (Deep Learning) with a comparison between software and hardware results. Section VI, discuss the results, and Section VII, summarizes our conclusions.

II. LITERATURE REVIEW

An environment that possesses ambient intelligence and automatic control has been described as a smart home [2][3] which is defined as the concentrator and disseminator of information and services and is closely related to IoT synergic networks and principles [4] [5]. Continuous improvement for such systems will be necessary to delegate most of the occupant's needs to this type of intelligent system. Therefore, a smart home system must have the capability to be self- autonomous, adaptable, and have an optimized interface.

Predicting the user needs and then implementing an au- tomated action on behalf of the user is a core thread in such a smart environment to ensure the quality of services. Many researchers use MLAs as the basis for prediction of user behavior. In [6], the researchers propose an Artificial Neural Networks (ANN) and a priori algorithms that support user preferences depending on the current context. Other researchers use time series prediction, such as evolving the Fuzzy Predictor model [7] [8]. Semi Markov Model as a probabilistic model was by [9] to predict the user actions inside a real-world environment that was represented by a 60-day dataset. The researchers manage appliances through an energy management system. A pattern matching and reinforcement learning approaches were used to predict next user action by applying mathematical methods on a simple synthetic data. The researchers showed that their proposed algorithm improved the accuracy but did not mention the time of prediction needed to predict next user actions. Most of these researches were dedicated to optimizing energy consumption in addition to enhancing the convenience of the occupant's daily life. Accuracy measurements have been used to compare the performance analysis of different types of MLAs which are used to predict a user's future actions, based on past events.

In this research, the authors investigate the accuracy and prediction time needed by three types of MLAs by comparing the performance of a range of a well-known MLAs. Besides, this research goes deeper to investigate the use of deep learning not only by simulation experiment but also with a FPGA hardware implementation to assess the simulation result and make sure the applicability of using the system under real- time constrains. We list several candidates of machine learning algorithms to be an essential part of learning phase in the BUTLER system [10] which plays an important role in decision making. In this paper, three classifiers were used: Naive Bayes (NB), Support Vector Machine (SVM), and Neural Network (NN). The goal of this work is to determine the best MLA choice between these types of classifiers. Hence, the selected algorithm should assure the quality of services represented by reaching maximum accuracy and fast response performance in such a real-time application system.

The *Naive Bayes* (NB) approach is a commonly-used probabilistic classifier [11]. Both the Multinomial (mn) and Multi-Vibrate Multinomial (mvmn) distributions were used in our experiments.

A *Support Vector Machine* (SVM) is a supervised machine learning algorithm which classifies data by dividing points by using hyperplanes [12].

The *Neural Network* (NN) performs classification by training neural networks and determining the optimum weights of the designed network.

III. EXPERIMENTAL DESIGN

In this section, we describe the experiments used to determine the optimal machine learning

technique to be used for building rules during the learning phase [10]. These experiments used the MavPad dataset [1] which was collected by observing an individual living in an apartment; thus, it consists of observa- tions of human activities in a real environment. The apartment consists of a living/dining room combination, a kitchen, a bathroom, a large bedroom, and a walk-in closet. There were 127 nodes of sensors and actuators in the environment. The complete dataset was collected over 49 days in 49 data files sequentially, with a total of seven weeks. A separated data file for each day was recorded by the system. In each original data file, there is one sensor event on each row of the data file, and the event syntax is represented by date, time, zone number, state, level, and source information.

The MATLAB2017 environment was used to convert the source data file from 49 data type files (.dat) to one matrix data type file (.mat). After the conversion process, all the data will be stored in one matrix for instance: the first column in the matrix file, consists of the names of each node in the first row followed by the status values of that node in the other rows.

The MATLAB was also used to process the data by removing unnecessary details. After filtering the noise, we transfer the data to one Matlab matrix file called (OP49DAYS.mat). This file contains 86 columns that represent the status of the sensors, referred to as predictors. In addition to the predictors, there were 41 columns which represent the actuator's values or the output (Vector Y). As a result, there were more than (73 10^3) rows or events for the first day alone and more than (4 10^6) rows of observations during seven weeks. Each row has a status value for each node type which represents a predictor value or an attribute at a specific time. After the data conversion process, we applied the three different types of MLAs, described above, to predict the next user action in the environment using the MLAs that have been discussed in Section II. Two different combinations of sensors were used to predict the next stakeholder's action. The first case applies all the seven sensors that existed inside the restroom. The second case uses all the 86 sensors inside all the zones of the environment. The predicted action represents an actuator with binary outputs (0/1) inside the restroom labeled in (B5). The actuator B5 is used to turn the light over the mirror ON or shut it OFF. In our experiment, we decide to take the first four weeks (28 days) for training dataset and the fifth week as the test dataset.

IV. SOFTWARE RESULTS

In this section, we evaluate the accuracy and performance of each algorithm by comparing the performance of the MLAs that have been used in our experiments. Initially, the experiments focus on the behavior of the occupant at one specific zone (Restroom Zone) in the apartment.

In our restroom zone experiments, we used two different constructions of sensors to predict the user's action. The first case uses seven sensors, labeled as (V21, V22, V23, S137, S138, S139, S140), that represents the predictors and located in the restroom zone, and the second case uses all 86 sensors at all zones in the apartment concurrently. Moreover, the predicted action is labeled in (B5) used to turn the light over mirror ON or OFF. So, for each day, two cases that use a different number of sensors were taken as input variables to predict the user's action. For training and testing data size, we decide to take one month for a training dataset for all the MLAs followed by one week as testing data.

In this investigation, we seek to compare MLAs by determin- ing their accuracy and relative execution time for the prediction process. We determine the accuracy and prediction time needed for the two combinations, one with seven predictors in the local zone and the second with 86 predictors in the global zone, as processed by each of the MLAs. Tables I and II, the three accuracy results for each type of classifier were presented for each grouping of sensors. The best results of classifier are presented in bold font.

1) Local Zone Experiment: This experiment uses seven input sensors to predict the occupant's use of the B5 actuator. By comparing the experimental results in Table I, we can see that the SVM approach produces the highest accuracy. Since the goal of a real-time prediction system is to predict user actions with maximum accuracy and minimum delay, we can assure that SVM provides the best performance for accuracy but with slow response time (0.896 sec) comparing with NN. Meanwhile, the NN approach produced about 95% accuracy but with a fastest response time 0.244 sec comparing with SVM algorithm. Therefore, we decide to determine whether a Neural Network-based deep learning approach might deliver the best overall performance in such a real-time application environment.

2) Global Zone Experiment: In this experiment, we use all of the sensors in the environment (86 sensors) to predict the operation of the B5 actuator. The results were different compared with the first experiments' results that use only the local seven sensors. The best accuracy result was produced

TABLE I: Accuracy and execution time results for the local zone (7 sensors)

MLAs Type	Accuracy	Time/Sec
NB	0.9810	1.3571
SVM	0.9916	0.8961
NN	**0.9499**	**0.2446**

TABLE II: Accuracy and execution time results for the global zone (all 86 sensors)

MLAs Type	Accuracy	Time/Sec
NB	0.7448	5.824
SVM	0.5445	4.824
NN	**0.9023**	**6.4556**

by the Neural Network, but with a higher time (6.5 sec) of prediction process comparing with the first case which is too slow to be accepted for real-time applications. In this configuration, the accuracy of the binary classifier of SVM dropped down to 54% which is not applicable for a fast predicting process. That result is due to the nature of SVM's algorithm. Predicting hyperplanes of seven inputs is a relatively easy task when estimating only eight parameters to define the hyperplane's equation. On the other hand, when processing 86 inputs, estimating the hyperplane's equation requires the process to accommodate all sensors, which makes the calculation far more complicated. As a result, the system is more vulnerable to over-fitting, which degrades the performance. Regularization and feature selection could be used to improve the performance of SVM since not all the sensors have high correlations with the current B5 output.

V. NEURAL NETWORK-BASED DEEP LEARNING APPROACH

The results mentioned in IV-1 encourage the authors to a deep investigate the use of multilayer NN or Neural Network- based deep learning approach which might deliver a best overall performance as discussed previously. Therefore, the following subsections introduce NN-based deep learning and discuss the results in both software and hardware implementation experiments ending with a clear comparison using the two methods.

The strength of the neural network comes from the mathemat- ical model representation. As a matter of fact, NN is inspired by the biological nervous system of the human brain. It tries to mimic the way of the human brain processes and learns patterns. A Neural Network consists of interconnected nodes (neurons) that process the input data in a certain way to perform a specific task. Theoretically, the NN can represent many different kinds of complex function. Recently, data revolution, parallel architectures, and GPU designs have been developed drastically. Therefore, the neural network has become an efficient tool in the machine learning discipline. Neural network's nature offers

TABLE III: The accuracy and prediction time results using local zone's sensors for multilayer neural network

No. of Hidden Layers	Accuracy	T (Sec)
One hidden layer	0.9585	0.245
Two hidden layers	0.9911	0.387
Three hidden layers	0.9917	0.768

very beneficial characteristics such as learning adaptation, self- organization, real-time output, and implementation ease.

1) NN Software Implementation Results: In this section, we describe the software implementation of our network. Network configuration, such as network depth (i.e., number of layers) and the number of neurons of each layer determine computational speed. Although increasing the depth of NN improves the recognition rate (in case of having enough data), it consumes more CPU and memory resources. In this work, network depth, the number of neurons for each layer, and the training window size (i.e., the number of samples used in the training process of each step to update networks parameters) were determined experimentally by maximizing the classification performance using the available resources.

First, we started our implementation with one hidden layer. Then, we increased the number of neurons in the hidden layer to find the best representation experimentally. After that, we increased the depth of the network by adding a second layer. By fixing the number of neurons for the first layer, we increased the number of neurons for the second layer to come up with the best representation. We repeated this approach for the third layer in our network to optimize the number of nodes. Additionally, we used drop-out layers to reduce the effect of the overfitting problem. We added a drop-out layer between every two fully connected layers by a factor of 0.5. Removing some units of a network during training prevented excessive parameter updating. This drop-out technique may help reduce overfitting effect. The results of NN experiments are shown in Tables III and IV concurrently.

A Deep Learning technique was used to facilitate the power of neural network via the implementation of the multilayer approach. As

shown in Table IV, the use of the local sensors with only one layer will produce an accuracy of 95.85% with 245 milliseconds needed for the prediction process. Supporting the first layer with a second one can significantly enhance the accuracy performance to be 99.11% with an acceptable time of 387 milliseconds in the scope of real-time application. While adding a third layer to the designed model, doesn't enhance the accuracy. Also, the response time of prediction process using three layers is about the double of what we possess using two layers. The mentioned results using local sensors proves that adding more layers in deep learning model doesn't always assure the best performance: Therefore, layers optimization process needs to be considered when adding further layers to the system architecture.

Similarly, Table IV discusses the performance of adding

TABLE IV: The accuracy and prediction time results using global zone sensors for multilayer neural network

No. of Hidden Layers	Accuracy	T (Sec)
One hidden layer	0.9017	6.1023
Two hidden layers	0.9536	9.108
Three hidden layers	0.9611	13.706

a second and a third layer to the prediction system. Since this experiment use 86 sensors, which distributed in the entire environment, we can see that the response time is much higher than what we have in the first experiment that uses only seven sensors. A significant enhancement in accuracy can be noticed when adding a second layer to the deep learning model with 9.1 seconds of prediction time. A similar result of accuracy values have been seen when adding a third layer. In other words, support the model with a third layer doesn't facilitate better performance in the model; Therefore, the authors decided to consider designing the deep learning model to have the first two layers only which possess a maximum average accuracy and minimum average prediction time.

2) NN Hardware Implementation Result Using FPGA: The mathematical operations of neural network models are simple. However, the massive number of operations needs intensive computing resources. Therefore, a fast and efficient realization is required to achieve the benefits of neural models. The FPGA based system allows designers to create digital designs, test them, make modification very quickly, and reduce development time significantly [13]. Also, the Hardware implementation of nonlinear activations, e.g., the sigmoid function, is one of the challenges due to the complexity of implementing division and exponential regarding time and hardware resources. Therefore, the approximate-based approach has been presented to realize sigmoid function efficiently and maintain an acceptable level of accuracy, such as using Lookup Table LUT [14].

In this paper, we utilize an FPGA platform to realize reconfigurable hardware-based neural networks for smart home systems. Digital designs are usually modeled using hardware description languages like Very high speed integrated circuit Hardware Description Language (VHDL) and verified by simulation. In this paper, instead of using low-level coding, such as VHDL, we used a high-level programming language, LabVIEW, to realize the neural network. National Instruments LabVIEW FPGA module uses LabVIEW embedded technology to extend LabVIEW graphical development to target FPGAs on NI reconfigurable I/O (RIO) hardware or some Xilinx boards. This module enables users to create custom hardware without low-level hardware description language coding or board design experience. Moreover, it allows a user to executes multiple tasks simultaneously and deterministically, and also expands the functionality of LabVIEW solutions, including unique timing and triggering routines, ultrahigh-speed control, interfacing to digital protocols, digital signal processing (DSP) virtual instruments (VIs).

The implementation of the trained weight data, the synaptic coefficients which are determined offline in a computing environment, is done using signed fixed-point representation 16-bit total length. Fixed point arithmetic is used for NNs realization, which is implemented as one of the available data types in LabVIEW FPGA module. Therefore, NN coefficients are used in the hardware design without additional pre- calculation. Also, LabVIEW FPGA module provides nonlinear functions, which are used to implement the nonlinear activation functions of each neuron. The other important feature in using LabVIEW software is the ability to import weight coefficients of NNs from Matlab to the NN structure implemented in the FPGA, specifically to the block RAMs that store these coefficients.

Three optimization techniques are used in this paper to optimize and improve the performance of the FPGA-based neural network. The first one is loop pipelining to achieve high throughput by organizing the overlap in the sequence of operation of neural network systems. Single Cycle Timed Loop (SCTL) was used to reduce the required hardware resources and improve the execution speed of our proposed neural network circuit. SCTL is an optimization technique available in LabVIEW FPGA module to eliminate the unnecessary resources exploited by the standard while loop

function. Due to limited computation resources in FPGA platforms and the massive computation required in realizing neural network models, loop unrolling has been exploited to efficiently utilize the resources and avoid complex hardware connections. We used this strategy to unroll the independent data and avoid complex connection topologies.

VI. DISCUSSION

The hardware implementation of the neural network has been done by LabVIEW FPGA module and downloaded to Xilinx XC3S500E FPGA. The whole neural network system fits in a low-cost Xilinx Spartan-3E FPGA platform and uses block RAMs as on-chip buffers and a DRAM as external storage. The Spartan-3E family of field-programmable gate arrays is specifically designed to meet the needs of high volume, and cost-sensitive consumer hardware digital systems, where the cost must be lower than the general purpose processors. The five-member family offers densities ranging from 100,000 to 1.6 million system gates. The XC3S500E FPGA has 4,656 slices, almost 10,476 logic cells, twenty 18x18 hardware multipliers, as well as twenty 18 Kbits modules of dedicated dual-port RAM. In this section, we report the performance of our proposed reconfigurable FPGA-based neural network architecture and compare it with the software implementation. Then, we provide the hardware efficiency compared with the existing FPGA implementations.

Our proposed FPGA realization of neural network has the same accuracy results as what we have in Matlab imple- mentation, and the accuracy is not compromised due to the usage of fixed-point computing units. The characteristic feature of using hardware platforms is performing the mathematical calculations of neural networks in parallel. This feature cannot

TABLE V: The prediction time results using local and global zone's sensors for hardware-based multilayer neural network

Zone Name	No. of Hidden Layers	T (Sec)
Local zone's sensors	One hidden layer	0.091
	Two hidden layers	0.199
	Three hidden layers	0.485
Global zone's sensors	One hidden layer	0.329
	Two hidden layers	0.658
	Three hidden layers	1.841

be achieved with the software-based implementation of neural networks because of the sequential execution of the code. Table V illustrates the prediction time results for local and global zone's sensors with different network structures. The results show that the hardware implementation significantly improves the prediction time on average by factors of two times and thirteen times compared with simulation results in the local and global zone respectively. The significant improvement in average prediction time commits substantial evidence that our designed autonomous system applies to be used in such a real-time application paradigm.

The hardware utilization summary is shown in Table VI and compared with other FPGA realization approaches [15], [14], [16]. The proposed architecture consumes 2,828 slices out of the available 4,656 slices, which is about 60% of the total number of slices. Specifically, the re-configurable hardware realization utilizes 3,938 slice LUTs or 42% and 2,862 slice registers or 30%. Connections per second (CPS) metric, the number of operations to be performed in a second, is used to compare hardware-based neural network architectures due to using different FPGA platforms for realization. The second metric is connection primitives per second per LUT (CPPSL), which takes the hardware resource utilization into account. CPPS can be calculated by multiplying CPS by the bit width of inputs and weights. Table VI shows the performance comparison between our proposed architecture and other implementations using CPS and CPPSL metrics. Our proposed architecture achieves three times more CPPSL compared to the best of existing FPGA implementations.

TABLE VI: FPGA-based neural network resource utilization and a comparison with other FPGA implementations

Reference	Platform	Slice LUTs	Slice Registers	DSP48Es/ Multipliers	CPS	CPPSL
Gomperts et al. 2011	XC5VSX50T	8043	2243	70	536 M	9.6 M
Zhai et al. 2013	Virtex-4 LX40	4346	N/A	8	1.2 M	0.07 M
Zhai et al. 2016	XC7Z010T	4032	2863	28	72.3 M	10.3 M
Proposed	XC3S500E	3938	2862	20	481.3 M	31.2 M

VII. CONCLUSION

This paper presents the design and implementation of a reconfigurable hardware-based neural network for smart home systems. The proposed architecture outperformed the performance of software-based implementation regarding speed due to exploiting parallel computing and some optimization techniques. LabVIEW software enables developers to imple- ment digital systems without the need for low-level HDL language knowledge and reduces the usage of FPGA hardware resources. It also eliminates the need for pre-processing the neural network weights and maps them to FPGAs storage units. The implementation of FPGA-based neural network system allows its future utilization for other applications.

REFERENCES

[1]. M. Youngblood, D. J. Cook, and L. B. Holder, "Seamlessly engineering a smart environment," in *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, vol. 1. IEEE, 2005, pp. 548–553.

[2]. L. C. De Silva, C. Morikawa, and I. M. Petra, "State of the art of smart homes," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 7, pp. 1313–1321, 2012.

[3]. B. M. Alhafidh and W. Allen, "Smart homes based on smart cities design patterns," in *Internet of Things and Big Data Analysis: Recent Trends and Analysis*. United Scholar, 2016, ch. 7.

[4]. M. B. I. Reaz and M. Marufuzzaman, "Pattern matching and reinforce- ment learning to predict the user next action of smart home device usage," *Acta Technica Corviniensis-Bulletin of Engineering*, vol. 6, no. 3, p. 37, 2013.

[5]. B. M. Alhafidh and W. Allen, "Design and simulation of a smart home managed by an intelligent self-adaptive system," *International Journal of Engineering Research and Applications*, vol. 6, no. 8, pp. 64–90, 2016.

[6]. B. Naouar, G. Nesrine, S. A. Adil, and M. Bouchaib, "Proactive intelligent home system using contextual information and neural network approach," 2016.

[7]. M. J. Akhlaghinia, a. Lotfi, C. Langensiepen, and N. Sherkat, "Occupant behaviour prediction in ambient intelligence computing environment," *Journal of Uncertain Systems*, vol. 2, no. 2, pp. 85–100, 2008.

[8]. A.-M. Vainio, M. Valtonen, and J. Vanhala, "Proactive fuzzy control and adaptation methods for smart homes," *IEEE Intelligent Systems*, vol. 23, no. 2, 2008.

[9]. K. Bao, F. Allerding, and H. Schmeck, "User behavior prediction for energy management in smart homes," in *Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on*, vol. 2. IEEE, 2011, pp. 1335–1339.

[10]. B. M. H. Alhafidh and W. H. Allen, "High level design of a home autonomous system based on cyber physical system modeling," in *Distributed Computing Systems Workshops (ICDCSW), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 45–52.

[11]. T. L. van Kasteren, G. Englebienne, and B. J. Kro¨ se, "Human activity recognition from wireless sensor network data: Benchmark and software," in *Activity recognition in pervasive intelligent environments*. Springer, 2011, pp. 165–186.

[12]. Y. Li, S. Gong, and H. Liddell, "Support vector regression and classifica- tion based multi-view face detection and recognition," in *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*. IEEE, 2000, pp. 300–305.

[13]. F. H. Ali, H. M. Mahmood, and S. M. B. Ismael, "Labview fpga implementation of a pid controller for d.c. motor speed control," in *2010 1st International Conference on Energy, Power and Control (EPC- IQ)*, Nov 2010, pp. 139–144.

[14]. X. Zhai, F. Bensaali, and R. Sotudeh, "Real-time optical character recognition on field programmable gate array for automatic number plate recognition system," *IET Circuits, Devices & Systems*, vol. 7, no. 6, pp. 337–344, 2013.

[15]. A. Gomperts, A. Ukil, and F. Zurfluh, "Development and implementation of parameterized fpga-based general purpose neural networks for online applications," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 1, pp. 78–89, Feb 2011.

[16]. X. Zhai, A. A. S. Ali, A. Amira, and F. Bensaali, "Mlp neural network based gas classification system on zynq soc," *IEEE Access*, vol. 4, pp. 8138–8146, 2016.