ICIC Express Letters
Volume **16**, Number **10**, October **2022**

ICIC International ©2022 ISSN 1881-803X
pp. 1071–1078

AN INTELLIGENT TIRE WEAR PREDICTION USING PARALLEL HARDWARE ARCHITECTURE

Atheer Akram AbdulRazzaq[1], Mohammed A. Fadhel[2,*]
and Omran Al-Shamma[3]

[1]Businesses Informatics College
[3]Scientific Affairs Department
University of Information Technology and Communications
AlNidhal Campus, Baghdad 10001, Iraq
{ athproof; o.al_shamma }@uoitc.edu.iq

[2]College of Computer Science and Information Technology
University of Sumer
Rifai 64005, Iraq
*Corresponding author: Mohammed.a.fadhel@uoitc.edu.iq

Received November 2021; accepted January 2022

Abstract. *Nowadays, several advanced control systems are developed to assist drivers and enhance vehicle comfort and safety. Among these sophisticated systems technologies, the tire-condition and its status predictors showed significant progress. In this paper, a neural network model to detect tire wear for motorsport vehicles is implemented. The system used a hardware parallel processor (FPGA) to run it on a real-time basis. Also discuss a method for efficiently running neural networks on FPGAs by utilizing the FPGA's ability to parallelize a large portion of the execution. It concludes that an FPGA is the most suitable hardware structure for building a neural network. However, there are two major limitations in the design's flexibility as it is currently implemented. First, the neural network layout is fixed in the FPGA design. Second, for practical reasons, the weights from the second hidden layer to the output were hard-coded.*

Keywords: FPGA, Tire wear, Neural network, Parallel processing

1. Introduction.

Generally, the tire has a ring shape. It surrounds the rim of the wheel to offer traction on the road. It transports the vehicle mass from the axle to the ground via the wheel [1]. Tires are designed in such a way that they can bear the load of the vehicle. The tire comprises various materials, including carbon black, wire, fabric, natural rubber, synthetic rubber, and different chemical components. It is composed of body and tread. The body offers inclusion for the compressed air mass, while the tread provides traction [2].

However, the tire is the only part of the vehicle touching the road [3]. Therefore, the driver should timely monitor the tire performance to maintain safety. In contrast, the layman might not know when a tire is going to wear out. If the tire is unreplaced in time, there might be fire chances, heavy damage, and might cause accidents. A device or a mobile application must be developed to avoid such fatal situations, known when a tire is worn out [4]. The tire contains tread, which is one of the essential elements to be considered to check whether the tire is worn out or not. Treadwear happens due to the usual touch with the road [5]. Abnormal tread wear has several types that yield owing to poor alignment of the wheel, gravel roads, rocky terrains, rough terrains, over-inflation, under-inflation, unbalanced wheels, rubber degradation, and other parameters [1]. A 4-digit number that represents the date of tire manufacture is printed on the tire. For

DOI: 10.24507/icicel.16.10.1071

example, the tire will expire after six years from that date unless the rubber degrades or results in accidents. If the tire is punctured 4 or 5 times, it is considered to wear out and replaced. The tire should be properly fitted, inflated, and balanced on to the car. If the tire is not parked correctly or rubber degrades, cracks or bulges occur, which causes tire burst and leads to fatal accidents [1,3].

Vehicle efficiency, comfort, steering, braking, and acceleration are adversely impacted due to improperly maintained tires. Besides, defective or inadequately maintained tires contribute to the death of about 6,000 people annually [6,7]. Furthermore, harsh environments and time cause the rubber to lose compliance and become inelastic. This tire condition leads to loss of control and increases the risk of catastrophic failure [8]. Unfortunately, just a few drivers regularly check their tire condition.

In contrast, the risks of improper inflation are aware by vehicle owners and continuously drive their vehicles with low inflated tires [9]. Furthermore, they forego tire inspection, and they did not know risks such as cracking, oxidation, and tire aging due to lesser-known issues [6,7]. Therefore, using easy-to-use tire-inspection systems reduces driver awareness and the risk of a blowout or sudden material separation and enhances compliance with proper tire-replacement timing.

For example, Formula One racing is a form of motorsport that takes place, managing tire conditions and race strategies play a vital role in the success of teams during a race weekend. With different tire compounds of choice that vary in expected durability and speed at each weekend, teams must plan how they will utilize the different tire compounds to be the quickest they can. This can lead to differences between one-stop pit strategies, where drivers utilize only two sets of tires as required by the Formula 1 racing regulations, and two-stop pit strategies, where drivers push harder on each tire set but stop and utilize an additional set of tires. Being able to bring the most out of the tire compounds is important for gaining these precious seconds from strategy alone; thus, it is important for teams to track the condition of their tires throughout the race [10].

In this paper, a tire degradation neural network model is developed for motorsport teams to be able to evaluate tire conditions during a race for all drivers when considering race strategy.

The following summarizes the arrangement of this article. The second section provides an introduction to the history of neural networks. The hardware implementation on FPGA is detailed in the third and fourth parts. The fifth part focuses on the outcomes of our proposal network. The sixth part discusses the conclusion and future efforts.

2. **Neural Network.**

Neural networks are a machine learning method for attempting to learn how to predict an output based on a series of linear functions with associated parameters which can be learned based on training examples [11-16].

A neural network consists of multiple layers: an input layer, an output layer, and one or more hidden layers. Each layer consists of nodes where values are evaluated and accumulated. For the input layer, the nodes consist of the input values provided for the neural network. For the hidden and output layers, the value at the node is a result of a linear function on the values of the nodes from the previous layer. Consider the following Figure 1 for a single node [17].

Each node in the previous layer holds a value x_i and for each connection to the next layer a weight associated with the pair of nodes w_i. Additionally, the node is currently looking at having an associated bias value b. To evaluate the value at the current node, the sum of the product $w_i \cdot x_i$ for all of the nodes in the previous layer and add the bias value b.

Hidden layers of the neural network additionally tend to have activation functions associated with the nodes. Activation functions are an additional operation on the previous sum to obtain the final value at the node. In the case of Figure 1, it is shown the image

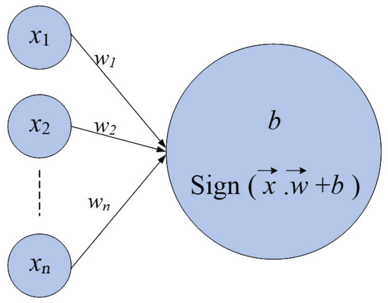


FIGURE 1. Image of a single node and the connections from the previous layer to the current node

that a sign activation function is utilized at this layer. Based on the previously calculated value, the sign function would return a 1 if the value was positive and 0 otherwise [18]. This type of activation function would likely be utilized in a binary classification problem at the output node. Another type of activation function commonly utilized is a ReLU function. These activation functions are often used to capture more complex behavior of equations with the neural network layout.

For the purpose of this work, we attempt to present a method for running neural networks on FPGAs efficiently using the ability of the FPGA to parallelize a large portion of the execution.

3. **Neural Network on FPGA.** The FPGA type ALTERA Cyclone V equipment is a single-die system on a chip (SoC) that is composed of two portions – a hard processor system (HPS) part and an FPGA part [19]. Neural network usually consists of three layers, input, middle (hidden) and output layer [11]. The second layer was far wider than the other layers, so all calculations are parallelized along this middle layer. Each node in the second layer, referred to as the Middle Layer, was responsible for multiplying the weights for all the corresponding inputs and summing these values with a bias value. Then this sum needed to be multiplied by all the weights from the Middle Layer node to the third layer node. To corral all this calculation for each Middle Layer node we merged into one module called the neural net middle (NNM). NNM effectively takes in as many inputs as there are nodes in the first layer and outputs as many signals as there are nodes in the third layer. Each NNM module consisted of a fixed-point multiplier that was responsible for computing all the multiplications. A single multiplier for each NNM module is utilized because that unsure of how many nodes would be in the Middle Layer [20-22].

The multipliers and the rest of the calculations on the FPGA were conducted with multi fixed point because the DSP blocks that held the multipliers were limited to a resolution of 27 bits. All the calculations happened with the help of a state machine that covered shortly in the next section.

One of the significant challenges to implementing a neural net in hardware is having a fast solution for storing and reading weights for the neural net. Each NNM module is responsible for storing one 1 bias value, the weights corresponding to the first layer nodes, and the weights from the Middle Layer node to the third layer nodes. To account for all these weights, an M10K RAM block is implemented within each module to store all the nodes [23]. The address counter was implemented in such a way that the counter would overflow after reaching a value that corresponded to the index of the last weight in the M10K block. Writing to the M10K block happens in the first state and the values are read from the M10K block at every subsequent state.

As can be seen in Figure 2, the state machine inside NNM consists of seven separate parts. The write weights stage is where the weights are written into the RAM module. Stall Cycle I and Stall Cycle II are used to handle the two-cycle read delay for the M10K

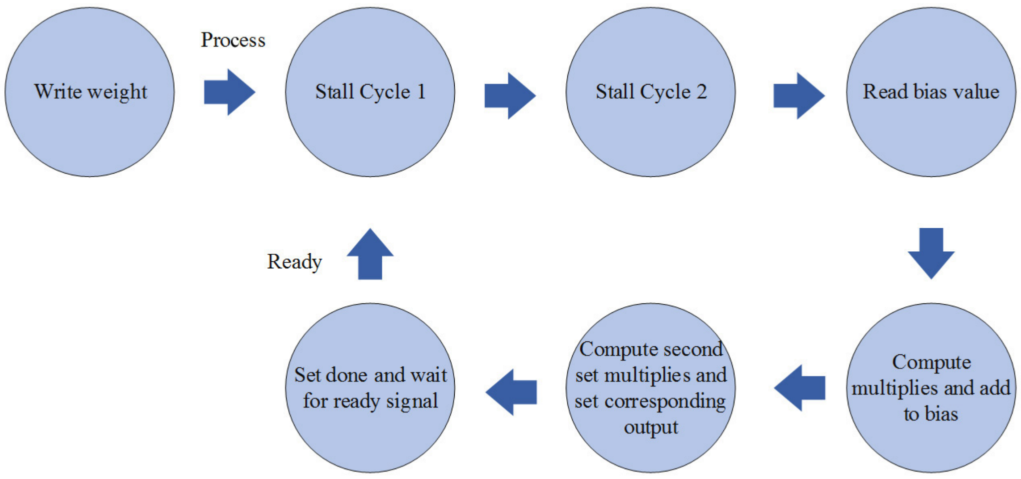

Write weight
Process
Stall Cycle 1
Stall Cycle 2
Read bias value
Compute multiplies and add to bias
Compute second set multiplies and set corresponding output
Set done and wait for ready signal
Ready

FIGURE 2. The state machine used in the NNM module

RAM. Note that address counter is at zero in Stall Cycle I and is incremented in both Stall Cycle I and Stall Cycle II. So, the data at address 0 can be read the cycle after Stall Cycle II, and every weight can be read in order in all the subsequent cycles as well because address counter is incremented every cycle. Read bias value is the stage right after Stall Cycle II and that is where the data at address 0, which holds the bias is read into a register. The compute multiplies and add to bias part of the diagram consists of as many cycles as there are inputs to the neural network. In each of these cycles, one of the multiplies corresponding to the input and its weight is computed in a single cycle using the multiplier. The product is also added to the running sum that includes the bias value within the same cycle and stored in the register that initially held the bias value. Once all the sums are computed, the second set of multiplies corresponding to the third layer is computed in multiple cycles, but each multiply is written to an output that corresponds to each node in the third layer. Within the first cycle of these calculations however, the ReLU operation for the Middle Layer is computed. If the sum is less than zero, then the register holding the sum has a zero written to it, which will be used in all subsequent calculations. Once the calculations are complete, a done signal is raised and the neural net waits for the next set of actions.

The NNM module was initially unit tested by using the address counter to write the value of the address at each address in the write weights stage. Then the outputs of the NNM module were checked on ModelSim for varying number of first layer and third layer nodes to ensure that the outputs corresponded to the values written into the RAM.

4. **HPS to Neural Net.** One advantage of using this technique to stream values in from the HPS is that the FPGA counters can be controlled from the HPS while the counters still run on the FPGA clock. Using a rising edge to enable allowed us to precisely trigger exactly when each operation in the FPGA would happen without having to mix or gate clock signals. The ready signal shown in Figure 3 is used as a rising edge triggered signal. When the FPGA writes output to the data line and raises the valid signal, the HPS can go in and read the data, set new inputs, and raise the ready signal. And when the ready signal is raised, the FPGA triggers all the state machine to go back to their initial state and begin computing on a new value [24,25].

Finally, the 16-bit color graphics primitives' example available on [14] was used as the base for all Verilog implementation. This example came with the SDRAM and VGA subsystem setup in the Qsys tool and writing to the VGA display involves first writing to the SDRAM. The VGA subsystem reads data written to the SDRAM and outputs it on the VGA display. The color information of each pixel was represented over two bytes,

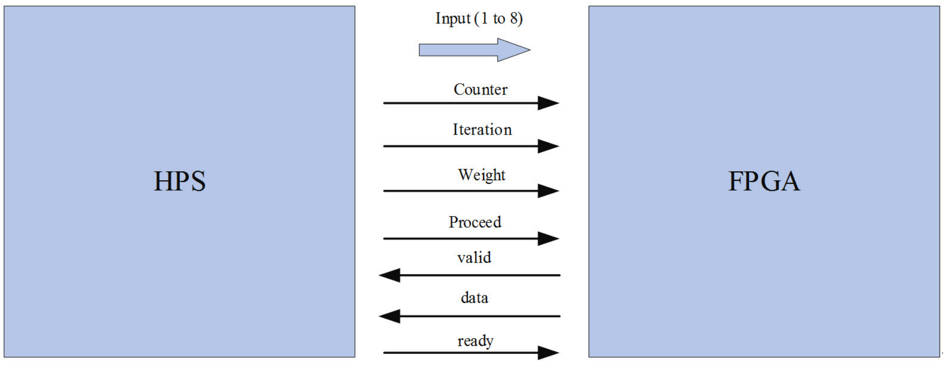
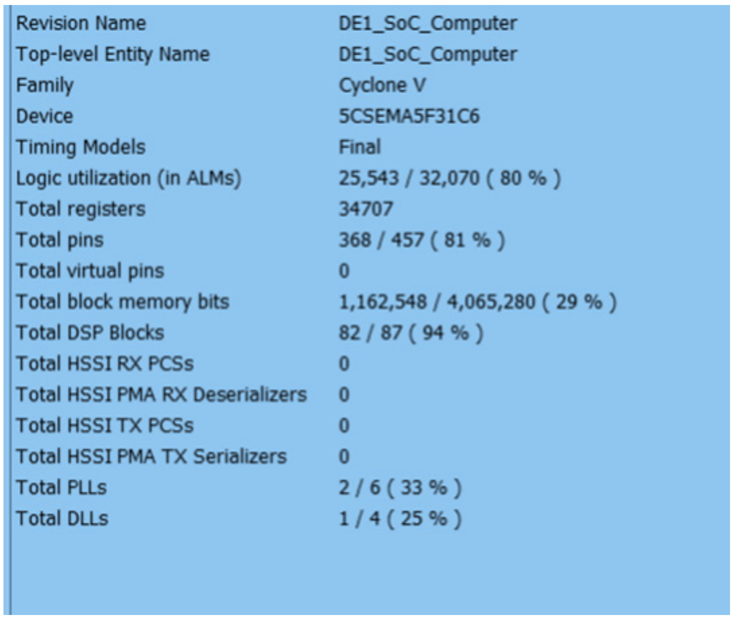


Figure 3. The signals travelling between the HPS and the FPGA

and each pixel's data is stored sequentially in the SDRAM. Since the VGA display is a 640×480 resolution display, the location of a specific pixel's data within the contiguous chunk of memory is determined by the expression: $((640 * y + x) << 1)$ where y and x represent the pixel coordinates.

5. **Results.** Figure 4 shows the compilation report for the final design implemented on the FPGA. The most apparent thing from the report is that the number adaptive logic module (ALM) usage is almost at eighty percent. Second iteration, which consisted of a large design, used approximately the same amount of ALMs. If relied on a more aggressive design, it is likely that used to overextended the number of ALMs available on the Cyclone V SOC. Nevertheless, the design was able to properly read in the various weights and provide the necessary computations for the neural net. The output of the FPGA neural network matched the MATLAB program to six decimal places, so we were able to reach a high level of precision with this system. Also, the design computed values at a deterministic thirty-seven clock cycles, which on a 50 MHz system is certainly in the realm of real-time computation. Any measurements did not take of the runtime on the FPGA because the HPS interactions themselves would have contributed to most of the noise in the system, which in and of itself is a testament to the speed of the design.

While the design for the neural network model is developed with processing speed of the inputs, the time calculation has also been taken into account to process inputs.



Revision Name	DE1_SoC_Computer
Top-level Entity Name	DE1_SoC_Computer
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	25,543 / 32,070 (80 %)
Total registers	34707
Total pins	368 / 457 (81 %)
Total virtual pins	0
Total block memory bits	1,162,548 / 4,065,280 (29 %)
Total DSP Blocks	82 / 87 (94 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	2 / 6 (33 %)
Total DLLs	1 / 4 (25 %)

Figure 4. The FPGA compilation report

This paper was designed around returning results on a lap-by-lap basis rather than real-time. Rather, the speed of the FPGA model is throttled by delaying when the inputs are fed into the neural network such that the results of the simulation are viewable at an appropriate rate. As long as the results are returned to us in a timely manner, for the purposes of this work this was sufficient. The important target not only chases for perfectly optimal execution of inputs, but rather a sufficiently parallel and functional design for implementing the neural network on the FPGA.

The system displayed the results on two locations: the terminal and the VGA. As a result of the simulation being run, every lap is displayed with information about the current lap number, if it is raining or not, and for the driver being followed: their previous tire degradation, if they took a pit stop, and if they did not finish the race. An example of a lap with a driver taking a pit stop is shown in Table 1.

Table 1. A pit stop has been taken for different drivers.

Driver	Lap number	Lap time (s)	Total time (s)	The percentage of tire degradation
MOH	5	94.045	470.225	99%
OMR	5	98.453	492.265	98%
SUZ	5	115.092	575.46	83%
ALI	5	93.213	466.065	99%
JUL	5	100.878	504.39	89%
LAI	5	91.332	456.66	99%
SAM	5	105.766	528.83	82%

When the selected driver does not finish, a flag is set to display the user that the driver is out of the race.

As seen in Table 2, the scoreboard displays the driver initials, the lap number, the lap time, total time, and the current tire degradation. In addition to this terminal output that updates every lap, the monitor is updated from graphing on the VGA the tire degradation of the driver selected as seen here. Note that whenever a pit stop is taken, tire degradation immediately jumps to 100% as the driver now has fresh new tires. The tire degradation was verified with the expected tire degradation of the MATLAB model and was found to be a match for the columns we selected for different tires in different laps, demonstrating correct functionality.

Table 2. Different drivers do not finish the race.

Driver	Lap number	Lap time (s)	Total time (s)	The percentage of tire degradation
DAW	16	71.872	1149.952	28%
RAN	16	72.563	1161.008	20%
YAS	16	71.443	1143.088	55%
JAF	16	70.505	1128.08	35%
ZAI	16	70.550	1128.8	58%
LAM	16	71.112	1137.792	25%
JWA	16	71.496	1143.936	54%

Another important note is that the plot shows tire degradation being negative here. This can be attributed to the fact that tire degradation estimations predict when tires should be changed and 0% corresponds to when pit stops are expected to be taken, not when tires are fully worn. This discrepancy means that sometimes drivers may continue extra laps even on subpar tires based on differences in strategy from NNM model.

6. **Conclusion.** At the end of this work, there are some point summarizing this manuscript.

1) Able to predict labeling equation for tire degradation much better than originally expected.
2) Both the ability of the neural network model in MATLAB to be able to reach such a high accuracy in prediction level for a regression model and the FPGA predictions being essentially equal to those from the MATLAB model.
3) Reduce the serial portions of state machine significantly by using many and more multipliers. That concludes, an FPGA the most suitable hardware structure for building a neural network.
4) Rather, a programmable hardware system with a large array of multipliers and adders would have been optimal for this work. While ASICs that are capable of such a design exist, they are often hardwired not as extensible as an FPGA.
5) However, there are two main limitations in the flexibility of this design in how this was currently implemented. First, the layout of the neural network is fixed in the FPGA design; therefore, the activation functions and number of nodes or layers cannot be adjusted without also adjusting the FPGA implementation. Second, the weights from the second hidden layer to the output were hard-coded for practical purposes, as the design intended to be static once implemented. While a new model could be trained on a different set of races and new weights can be obtained, the hard-coded weights in the FPGA implementation would have to be adjusted accordingly.

REFERENCES

[1] H. Allipilli and S. Samala, Convolutional neural network and OpenCV based mobile application to detect wear out in car tyres, *Journal of Computing Research and Innovation*, vol.6, no.1, pp.97-110, 2021.

[2] P. Taylor, Tires and the tire market, *Tire Waste and Recycling*, pp.15-25, DOI: 10.1016/B978-0-12-820685-0.00022-3, 2021.

[3] T. Kuraishi, K. Takizawa and T. E. Tezduyar, Space-time computational analysis of tire aerodynamics with actual geometry, road contact, tire deformation, road roughness and fluid film, *Computational Mechanics*, vol.64, no.6, pp.1699-1718, 2019.

[4] L. Chen, J. Liu, H. Mousavi and C. Sandu, Evaluation of tire traction performance on dry surface based on tire-road contact stress, *SIAR International Congress of Automotive and Transport Engineering: Science and Management of Automotive and Transportation Engineering*, pp.138-152, 2019.

[5] B. Baensch-Baltruschat, B. Kocher, F. Stock and G. Reifferscheid, Tyre and road wear particles (TRWP) – A review of generation, properties, emissions, human health risk, ecotoxicity, and fate in the environment, *Science of the Total Environment*, vol.733, DOI: 10.1016/j.scitotenv.2020.137823, 2020.

[6] J. A. Cowley, S. Kim and M. S. Wogalter, People do not identify tire aging as a safety hazard, *Proc. of the Human Factors and Ergonomics Society Annual Meeting*, vol.50, pp.860-864, 2006.

[7] M. J. Kalsher, M. S. Wogalter, K. R. Laughery and R. W. Lim, Consumer knowledge of tire maintenance and aging hazard, *Proc. of the Human Factors and Ergonomics Society Annual Meeting*, vol.49, no.18, DOI:10.1177/154193120504901817, 2005.

[8] J. M. Baldwin and D. R. Bauer, Rubber oxidation and tire aging – A review, *Rubber Chem. Technol.*, vol.81, no.2, pp.338-358, DOI: 10.5254/1.3548213, 2008.

[9] R. Sivinski, *Evaluation of the Effectiveness of TPMS in Proper Tire Pressure Maintenance*, Technical Report, NHTSA, 2012.

[10] D. P. Ferguson, *The Science of Motorsport*, 1st Edition, Routledge, London, 2018.

[11] L. Alzubaidi, Y. Duan, A. Al-Dujaili, I. K. Ibraheem, A. H. Alkenani, J. Santamaría, M. A. Fadhel, O. Al-Shamma and J. Zhang, Deepening into the suitability of using pre-trained models of ImageNet against a lightweight convolutional neural network in medical imaging: An experimental study, *PeerJ Computer Science*, vol.7, DOI: 10.7717/peerj-cs.715, 2021.

[12] L. Alzubaidi, A. A. Abbood, M. A. Fadhel, O. M. R. A. N. Al-Shamma and J. Zhang, Comparison of hybrid convolutional neural networks models for diabetic foot ulcer classification, *Journal of Engineering Science and Technology*, vol.16, no.3, pp.2001-2017, 2021.

[13] L. Alzubaidi, M. A. Fadhel, O. Al-Shamma, J. Zhang, J. Santamaría and Y. Duan, Robust application of new deep learning tools: An experimental study in medical imaging, *Multimedia Tools and Applications*, pp.1-29, 2021.

[14] L. Alzubaidi, M. Al-Amidie, A. Al-Asadi, A. J. Humaidi, O. Al-Shamma, M. A. Fadhel, J. Zhang, J. Santamaría and Y. Duan, Novel transfer learning approach for medical imaging with limited labeled data, *Cancers*, vol.13, no.7, DOI: 10.3390/cancers13071590, 2021.

[15] L. Alzubaidi, M. A. Fadhel, O. Al-Shamma, J. Zhang, J. Santamaría, Y. Duan and S. R Oleiwi, Towards a better understanding of transfer learning for medical imaging: A case study, *Applied Sciences*, vol.10, no.13, DOI:10.3390/app10134523, 2020.

[16] L. Alzubaidi, O. Al-Shamma, M. A. Fadhel, L. Farhan, J. Zhang and Y. Duan, Optimizing the performance of breast cancer classification by employing the same domain transfer learning from hybrid deep convolutional neural network model, *Electronics*, vol.9, no.3, DOI: 10.3390/electronics9030445, 2020.

[17] A. Heilmeier, A. Thomaser, M. Graf and J. Betz, Virtual strategy engineer: Using artificial neural networks for making race strategy decisions in circuit motorsport, *Applied Sciences*, vol.10, no.21, DOI: 10.3390/app10217805, 2020.

[18] L. Alzubaidi, J. Zhang, A. J. Humaidi et al., Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions, *J. Big Data*, vol.8, no.53, DOI: 10.1186/s40537-021-00444-8, 2021.

[19] O. Al-Shamma, M. A. Fadhel, R. A. Hameed, L. Alzubaidi and J. Zhang, Boosting convolutional neural networks performance based on FPGA accelerator, *International Conference on Intelligent Systems Design and Applications*, pp.509-517, 2018.

[20] M. A. Fadhel, O. Al-Shamma, S. R. Oleiwi, B. H. Taher and L. Alzubaidi, Real-time PCG diagnosis using FPGA, *International Conference on Intelligent Systems Design and Applications*, pp.518-529, 2018.

[21] L. Alzubaidi, M. A. Fadhel, S. R. Oleiwi, O. Al-Shamma and J. Zhang, DFU_QUTNet: Diabetic foot ulcer classification using novel deep convolutional neural network, *Multimedia Tools and Applications*, vol.79, no.21, pp.15655-15677, 2020.

[22] A. R. Nasser, A. M. Hasan, A. J. Humaidi, A. Alkhayyat, L. Alzubaidi, M. A. Fadhel, J. Santamaría and Y. Duan, IoT and cloud computing in health-care: A new wearable device and cloud-based deep learning algorithm for monitoring of diabetes, *Electronics*, vol.10, no.21, DOI: 10.3390/electronics 10212719, 2021.

[23] V. Cyclone, *Device Overview*, https://www.altera.com/enUS/pdfs/literature/hb/cyclone, 2016.

[24] R. F. Molanes, J. J. Rodríguez-Andina and J. Farina, Performance characterization and design guidelines for efficient processor-FPGA communication in Cyclone V FPSoCs, *IEEE Trans. Industrial Electronics*, vol.65, no.5, pp.4368-4377, 2017.

[25] L. Alzubaidi, M. A. Fadhel, O. Al-Shamma, J. Zhang and Y. Duan, Deep learning models for classification of red blood cells in microscopy images to aid in sickle cell anemia diagnosis, *Electronics*, vol.9, no.3, DOI: 10.3390/electronics9030427, 2020.