Neural Network based ECG Anomaly Detection on FPGA and Trade-off Analysis

Matthias Wess, Sai Manoj P. D., and Axel Jantsch
Institute of Computer Technology
TU Wien, Austria
Email: e0926401@student.tuwien.ac.at, {sai.dinakarrao,axel.jantsch}@tuwien.ac.at

***Abstract*—This paper presents FPGA-based ECG arrhythmia detection using an Artificial Neural Network (ANN). The objective is to implement a neural network based machine learning algorithm on FPGA to detect anomalies in ECG signals, with a better performance and accuracy, compared to statistical methods. An implementation with Principal Component Analysis (PCA) for feature reduction and a multi-layer perceptron (MLP) for classification, proved superior to other algorithms. For implementation on FPGA, the effects of several parameters and simplification on performance, accuracy and power consumption were studied. Piecewise linear approximation for activation functions and fixed point implementation were effective methods to reduce the amount of needed resources. The resulting neural network with twelve inputs and six neurons in the hidden layer, achieved, in spite of the simplifications, the same overall accuracy as simulations with floating point number representation. An accuracy of 99.82% was achieved on average for the MIT-BIH database.**

I. Introduction

An Electrocardiogram (ECG) tracks the electrical activity of the heart over time. It represents the physiological state of the heart and therefore is the most important method for diagnosing heart diseases [1]. Some of the major challenges to detect anomalies in ECG signal are: noise from measuring electrodes or loose contacts and mechanical disturbances; symptoms of anomalies might not show up all the time; taking measurements for long time may not be feasible.

Advancement and efficiency of artificial neural networks (ANN) led to its habituation in various applications. Several different ANN models like multi-layer perceptron (MLP) [2], [3], [4], modular neural networks (MNN) [5], general feed forward neural networks (GFFNN) [5], [6], radial basis function neural networks (RBFNN) and probabilistic neural networks (PNN) [4] have been implemented for ECG classification. In addition to the neural network, the optimization of preprocessing and feature selection holds most potential for improvement. In [2], a combination of fuzzy C-means clustering and principal component analysis (PCA) was successfully implemented with significant improvement of accuracy and dimensionality reduction of the input vectors.

For neural network based ECG anomaly detection on FPGA, a low number of input and hidden-layer neurons is crucial for an effective implementation. Several successful implementations exist, including optimized designs like [7]. As full implementations of ANN are computationally intensive, an optimized approach is needed for hardware implementation and is presented in this paper. It is of paramount importance to understand the impact of ANN parameters while designing the system.

This paper proposes optimization of neural network with piecewise linear approximated transfer functions and fixed point arithmetic. For selection of the most suitable fixed point precision, an extensive trade-off study was performed, discussing the merits and demerits of high/low precision datatypes, and detailing their influence on accuracy and required resources. The paper provides information on how to effectively reduce the amount of required resources implementing ANN on FPGA, for the task of ECG anomaly detection.

II. Implementation

For an FPGA implementation of an ANN, an architecture was developed which allows for a fast development process and high flexibility. Employing High Level Synthesis (HLS) offers the flexibility to study the effects of data type precision and size of the neural network.

A. System Architecture

Before presenting the used system architecture setup, we describe the basic steps and workflow involved in ECG anomaly detection. This includes basically three phases: feature extraction, principal component analysis (PCA), optimizing input data for processing and ANN for anomaly detection. Feature extraction involves detection of turning points in the ECG signal. To reduce the computational costs, the extracted feature set is reduced to a lower dimension using PCA and this data is provided to a multi-layer perceptron (MLP) for anomaly detection. A MLP is a fully connected ANN, with each node connected to every node in the next and previous layers with at least one hidden layer. The hidden layers enable the MLP to perform a nonlinear mapping between an input vector and an output vector [8]. The whole data flow is presented in Figure 1.

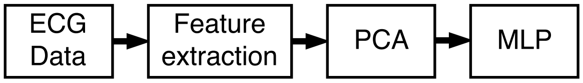

Fig. 1. Data flow for ECG anomaly detection

As shown in Figure 2 the data is preprocessed in Matlab on the host PC and then transferred to the FPGA which implements the MLP. The feature vectors are transferred via Ethernet to the Zynq Processing System. Training and testing of the neural network is entirely controlled by the integrated ARM processor of the Zynq. Classification is performed by the hardware implementation of MLP. To ensure fast data transfer between the ARM processor and the MLP, Block RAM of

978-1-4673-6853-7/17/$31.00 ©2017 IEEE

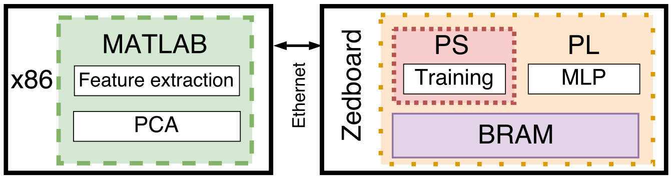

Fig. 2. Hardware/Software architecture, the feature vector is computed in MATLAB, transmitted via Ethernet onto the Zynq FPGA which implements the MLP and performs its training

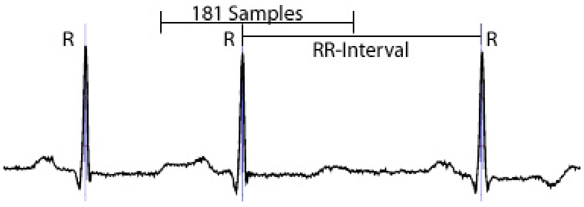

Fig. 3. For every heartbeat 181 Samples around the R-peak and RR-Interval lengths are used

the FPGA is used. Input and output of the entire process is controlled by a MATLAB GUI.

B. Dataset

For verification of the algorithm, the MIT-BIH arrhythmia database [9] was used. The database contains forty-eight 30 minute ambulatory ECG recordings, which include also less common but clinically significant arrhythmias. The Database is therefore suitable to evaluate performance and accuracy of the developed hardware for a wide spectrum of heart diseases [10]. For classification, a variety of feature vectors have been compared in terms of best classification accuracy. As illustrated in Figure 4, 181 samples around the R-peak were used. Additionally for every heartbeat, the RR-intervals to the preceding and the succeeding beats were derived from the MIT-BIH arrhythmia database annotations files. To obtain the feature vector, the selected samples are reduced with Principal Component Analysis (PCA), a statistical procedure to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables. Implementing Fuzzy clustering as in [2] did not prove efficient, as in this work the heartbeats were classified into only two classes.

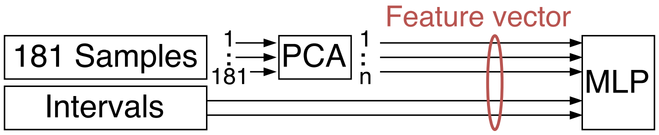

Fig. 4. The feature vector consists of n principal components and the RR-Interval lengths to the preceding and succeeding R-peaks

C. Training Algorithm

In general there exists no training algorithm that fits every task. Problems of backpropagation techniques are slow convergence and the chance for the algorithm to terminate in a local minimum. Paulin et al. [11] compared the performance of training algorithms for Feed forward Artificial Neural Networks for classification of breast cancer. Levenberg Marquardt (LM) performs slightly better than resilient backpropagation [12] (RPROP) and Conjugate Gradient (CG), in terms of accuracy of diagnosis. In [13] a comparison of RPROP, CG ad LM is performed for the tasks of streamflow forecasting and determination of lateral stress in cohesionless soils. In the two case-studies RPROP outperforms the Levenberg-Marquardt algorithm in terms of accuracy during the testing phase. Even though the two studied tasks differ from ECG anomaly detection, the study demonstrates that RPROP offers better generalization ability. Therefore for this paper RPROP was chosen as neural network training algorithm.

D. Activation Functions

For artificial neural networks, several activation functions are available. Choosing the best fitting activation function, secures the best result for the problem. In classification problems the sigmoid activation function for hidden layer neurons and the softmax function [14] for output neurons are commonly used. These functions also meet all the requirements for classifying heartbeats.

a) Hidden Layer: In neural networks, the hidden layer activation function is one of the most frequently used functions, and therefore a reduction of computation time was desired for hardware implementation. As shown in [15], Piecewise Linear Approximation (PLA) performs superior to exchanging the desired tansig activation function with logsig or ramp characteristics. Considering that the final implementation was planned with fixed point arithmetic, piecewise linear approximation was performed with gradients that reduced the required multiplications to simple shift operations [16]. While previous works used a maximum of seven ranges, to reduce the error of the approximated function we propose the following symmetrical function. This leads to a better approximation of the function and a better overall performance of the neural network compared to other proposed implementations. The piecewise linearly approximated hyperbolic tangent function (PLAtanh) is defined in (1), with the borders given in (2)

$$PLAtanh(x) = \begin{cases} 1 & x \geq a \\ x/4096 + 0.9986377 & a \geq x > b \\ x/32 + 0.905 & b \geq x > c \\ x/8 + 0.715625 & c \geq x > d \\ x/4 + 0.53125 & d \geq x > e \\ x/2 + 0.25 & e \geq x > f \\ x & f \geq x > g \\ x/2 - 0.25 & g \geq x > h \\ x/4 - 0.53125 & h \geq x > i \\ x/8 - 0.715625 & i \geq x > j \\ x/32 - 0.905 & j \geq x > k \\ x/4096 - 0.9986377 & k \geq x > l \\ -1 & l \geq x \end{cases} \tag{1}$$

$$\begin{aligned} &a = 5.5799959, b = 3.02, c = 2.02, d = 1.475, \\ &e = 1.125, f = 0.5, g = -0.5, h = -1.125, \\ &i = -1.475, j = -2.02, k = -3.02, l = -5.5799959 \end{aligned} \tag{2}$$

Due to the selected restrictions on the gradient of the linear functions, the biggest error occurs at $x = 0.5$ with approximately 0.03788.

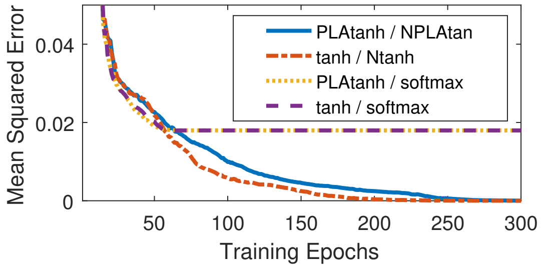

Fig. 5. Mean square error for training with different activation functions

b) Output Layer: To normalize output layer results, the hyperbolic tangent function is used with

$$Ntanh(x) = \frac{tanh(x) + 1}{2} \tag{3}$$

An implementation in this form not only allows to also make use of the already implemented simplified hyperbolic tangent function, but also reduces complexity. To evaluate the performance of this activation function in comparison to the exact hyperbolic tangent function, several neural networks were trained once with exact functions and once with piecewise linear approximations, in both hidden and output layers. As a typical example Figure 5 shows the mean square error for one specific record in the database for different activation functions. The error was logged after every training iteration with RPROP algorithm. The figure shows that using the piecewise linear approximated activation function for hidden layer neurons, does not worsen the results in comparison to exact implementation. Replacing the softmax function with Ntanh, given in (3), as activation function for output layer neurons, leads to faster convergence during the training phase and to better fitting due to the small number of output layer neurons. This difference occurs, because in backpropagation the influence of all output signals on the results of the softmax function increases the complexity of the algorithm and was therefore simplified in our application. A fixed point implementation of PLANtanh reduces the number of Flip-Flops by 90% and the number of LUTs by 80% compared to a floating point implementation. Moreover, the latency was reduced from 31 to 2 clock cycles. (Table I). Consequently, we used PLAtanh and Ntanh as activation functions in our experiments for the hidden layers and the output layer, respectively.

TABLE I. Comparison of PLANtanh 32-bit fixed point / tanh floating point implementation

Implementation	DSPs	Flip-Flops	LUTs	Latency
tanh (floating point)	18	1916	3697	31
PLAtanh (fixed point)	0	183	705	2
Reduction	100%	90.5%	80.9%	93.5%

III. Experimental Results

An ANN with twelve input neurons, one hidden layer with six nodes with PLAtanh as activation function and Ntanh for the output layer has been implemented on FPGA using the Vivado HLS tool. With 24 bit data size this approach achieves 99.82% accuracy. To analyze the trade-off between data type precision, size, latency and accuracy of the neural network, several differently configured MLPs have been implemented.

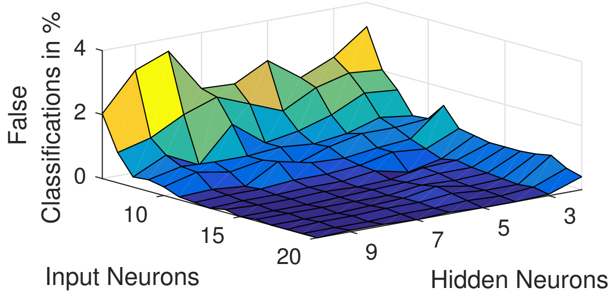

Fig. 6. Accuracy with 16-bit data (8 Fraction bits) for record 104

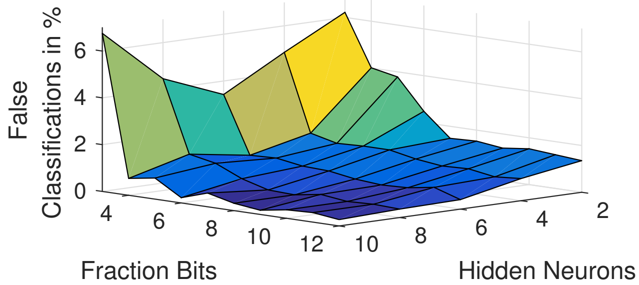

Fig. 7. Accuracy for 8 input neurons for record 104 MIT-BIH

A. Classification Accuracy

To optimize the hardware implementation of the Artificial Neural Network, an extensive analysis of implementation with fixed point data types has been performed by varying the number of input neurons, hidden layers, and the bit width of the fixed point implementation. The amount of fraction bits for fixed point data was set to half of the bit width of the selected data type. Figures 6 and 7 illustrate the amount of false classifications for record 104 of the MIT-BIH database as functions of three parameters. This particular record was selected as it represents a task with typical complexity and allows a good demonstration of how classification accuracy is influenced by change of parameters. However, for some records 100% classification accuracy is not achievable. Training the fixed point implementation of neural network, the training algorithm adapts weights and bias according to the selected precision. Whereas training the neural network in a floating point implementation and transferring weights and biases later onto a fixed point implementation increases false classifications, since weights and biases are trained for floating point precision. In figure 6 it can be seen that independent of the amount of inputs, more than three hidden layer neurons do not increase accuracy. For input neurons this boundary lies between ten and thirteen, depending on the amount of hidden layer neurons. For figure 7 the number of input neurons was set to eight. It shows that for the selected number of inputs a precision of seven fraction bits and eight neurons in the hidden layer already achieve 100% accuracy. For a higher number of inputs the required amount of neurons in the hidden layer and bit width decreases, while the neural network still achieves the same classification accuracy. It can be concluded, that for every number of hidden layer neurons there are certain thresholds for minimum precision and amount of inputs. Above the thresholds, increasing the number of input neurons improves the accuracy more than increasing the precision of the data type. For the selected record the smallest neural network with 100% accuracy consists of eight input neurons, eight hidden layer neurons with 14 bit fixed point data width. The optimal configuration for classification of the entire database has to be determined separately.

B. FPGA Performance

With respect to accuracy, the amount of required resources have a higher dependence on optimal implementation during High Level Synthesis. Table II shows the synthesis results for four different configured neural networks implemented with three different data types. In comparison to a floating point implementation, the proposed optimizations lead to a significant reduction of required resources and latency. For ANN with each one input, hidden and output-layer, concerning the different parameters it can be noted, that for bit widths of 12 and 16, Vivado HLS assigns one DSPs for each input and hidden neuron. For 24 bit width two DSPs are required for every input and hidden-layer neuron. When increasing the amount of hidden layers from four to six, Vivado HLS changes the routing and usage of DSPs, therefore the amount of used Flip-Flops and LUTs is reduced. As a drawback, the maximum frequency of the module decreases by 20%. In general the required resources increase with the number of input and hidden layer neurons. In summary, the amount of input and hidden layer neurons alongside with precision of the selected data type influence ECG classification accuracy of neural networks. While increasing the data width does not significantly increase the amount of required resources, a higher number of input and hidden layer neurons requires more LUTs and FIFOs in the FPGA implementation. With respect to previously simulated accuracies, it can be concluded that only after increasing data precision, the number of input and hidden layer neurons should be increased. Considering these findings an ANN with twelve inputs, six hidden layer neurons and 24 bit bit width was implemented, classifying the entire MIT-BIH database with 99.82% accuracy.

TABLE II. REQUIRED RESOURCES AND ACCURACY FOR RECORD 104

floating point					
Input/Hidden	**DSPs**	**FFs**	**LUTs**	**Latency**	**Accuracy**
8/6	42	9295	15163	1208	99.81%
12-bit fixed point					
Input/Hidden	**DSPs**	**FFs**	**LUTs**	**Latency**	**Accuracy**
8/4	12	1729	3945	62	98.56%
8/6	14	1551	1958	85	99.14%
10/4	14	1977	4399	71	99.28%
10/6	16	1613	1963	97	99.64%
16-bit fixed point					
Input/Hidden	**DSPs**	**FFs**	**LUTs**	**Latency**	**Accuracy**
8/4	12	1801	3653	60	98.87%
8/6	14	1561	1703	83	99.37%
10/4	14	2017	4045	68	99.19%
10/6	16	1646	1708	95	99.77%
24-bit fixed point					
Input/Hidden	**DSPs**	**FFs**	**LUTs**	**Latency**	**Accuracy**
8/4	24	2056	3910	63	99.05%
8/6	28	1772	1895	87	99.59%
10/4	28	2280	4366	71	99.28%
10/6	32	1926	1932	99	**100%**

IV. CONCLUSION

This paper presents a study of the impact of a NN architecture, activation functions, and fixed point precision on accuracy, performance, and area usage for FPGA implementations of an ECG anomaly detection algorithm. Piecewise linear approximation of activation functions was found an effective approach to reduce required resources and latency. The final implementation of a 12-6-2 24-bit multi-layer perceptron classifies the entire MIT-BIH database with 99.82% accuracy. Accuracy can be improved by increasing the number of inputs, hidden layer neurons and fixed point precision. All implementations were performed with High Level Synthesis, leaving potential for manual optimization.

REFERENCES

[1] M. S. Thaler, *The only EKG book you'll ever need*. Lippincott Williams & Wilkins, 2010.

[2] R. Ceylan and Y. Özbay, "Comparison of FCM, PCA and WT techniques for classification ECG arrhythmias using artificial neural network," *Expert Systems with Applications*, vol. 33, no. 2, pp. 286–295, 2007.

[3] V. Dubey and V. Richariya, "A neural network approach for ECG classification," *International Journal of Emerging Technology & Advanced Engineering*, vol. 3, 2013.

[4] A. E. Zadeh, A. Khazaee, and V. Ranaee, "Classification of the electrocardiogram signals using supervised classifiers and efficient features," *computer methods and programs in biomedicine*, vol. 99, no. 2, pp. 179–194, 2010.

[5] S. M. Jadhav, S. L. Nalbalwar, and A. A. Ghatol, "ECG arrhythmia classification using modular neural network model," in *IEEE EMBS Conference on Biomedical Engineering and Sciences*, 2010.

[6] S. L. N. Jadhav, Shivajirao M. and A. A. Ghatol, "Generalized feedforward neural network based cardiac arrhythmia classification from ECG signal data," in *Advanced Information Management and Service (IMS), 2010 6th International Conference on*. IEEE, 2010, pp. 351–356.

[7] Y. Sun and A. C. Cheng, "Machine learning on-a-chip: A high-performance low-power reusable neuron architecture for artificial neural networks in ECG classifications," *Computers in biology and medicine*, vol. 42, no. 7, pp. 751–757, 2012.

[8] M. W. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)a review of applications in the atmospheric sciences," *Atmospheric environment*, vol. 32, no. 14, pp. 2627–2636, 1998.

[9] R. Mark and G. Moody, "Mit-bih arrhythmia database directory," 1988.

[10] G. B. Moody and R. G. Mark, "The impact of the MIT-BIH arrhythmia database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, 2001.

[11] F. Paulin and A. Santhakumaran, "Classification of breast cancer by comparing back propagation training algorithms," *International Journal on Computer Science and Engineering*, vol. 3, no. 1, pp. 327–332, 2011.

[12] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The rprop algorithm," in *Neural Networks, 1993., IEEE International Conference On*. IEEE, 1993, pp. 586–591.

[13] Ö. Kişi, "Comparison of three back-propagation training algorithms for two case studies," *Indian journal of engineering & materials sciences*, vol. 12, no. 5, pp. 434–442, 2005.

[14] W. Duch and N. Jankowski, "Transfer functions: hidden possibilities for better neural networks." in *ESANN*. Citeseer, 2001, pp. 81–94.

[15] H. Amin, K. M. Curtis, and B. R. Hayes-Gill, "Piecewise linear approximation applied to nonlinear function of a neural network," *IEE Proceedings-Circuits, Devices and Systems*, vol. 144, no. 6, pp. 313–317, 1997.

[16] H. Hikawa, "A digital hardware pulse-mode neuron with piecewise linear activation function," *IEEE Transactions on Neural Networks*, vol. 14, no. 5, pp. 1028–1037, 2003.