Surrogate Model based Co-Optimization of Deep Neural Network Hardware Accelerators

Hendrik Wöhrle
Department of Information Technology
Dortmund University
of Applied Sciences and Arts
Dortmund, Germany
hendrik.woehrle@fh-dortmund.de

Mariela De Lucas Alvarez
Department of Mathematics
and Computer Science
University of Bremen
Bremen, Germany
delucas@uni-bremen.de

Fabian Schlenke
Department of Information Technology
Dortmund University
of Applied Sciences and Arts
Dortmund, Germany
fabian.schlenke@fh-dortmund.de

Alexander Walsemann
Faculty of Electrical Engineering
Dortmund University
of Applied Sciences and Arts
Dortmund, Germany
alexander.walsemann@fh-dortmund.de

Michael Karagounis
Faculty of Electrical Engineering
Dortmund University
of Applied Sciences and Arts
Dortmund, Germany
michael.karagounis@fh-dortmund.de

Frank Kirchner
Department of Mathematics
and Computer Science
University of Bremen
Bremen, Germany
frank.kirchner@uni-bremen.de

2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS) | 978-1-6654-2461-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/MWSCAS47672.2021.9531708

***Abstract*—In this paper, we present an ASIC based on 22FDX/FDSOI technology for the detection of atrial fibrillation in human electrocardiograms using neural networks. The ASIC consists of a RISC-V core for supporting software components and an application-specific machine learning IP core (ML-IP), which is used to implement the computationally intensive inference. The ASIC was designed for maximum energy efficiency. A special feature of the ML-IP is its modular, generic and scalable design of the ML-IP which allows to specify the quantization of each computational operation, the degree of parallelization and the architecture of the neural network. This in turn allows the use of ML-based optimization techniques to perform co-optimization for hardware design and architecture of the neural network (NNs). Here, a multi-objective optimization of the overall system is performed with respect to computational efficiency at a given classification accuracy and speed by using a multi-objective optimization, which is carried out using a probabilistic surrogate model. This model tries to find the optimal neural network architecture with a minimum number of training, simulation and evaluation steps.**

***Index Terms*—FDX/FDSOI, hardware acceleration, deep learning, bayesian optimization**

I. Introduction

Machine learning (ML), in particular Deep Learning (DL) has achieved many successes in a variety of application domains. In the last couple of years, applications of ML and DL in embedded systems gained momentum. However, other properties besides accuracy, such as energy efficiency and model size, remain relevant. In many cases, the use of dedicated hardware is the only way to meet the requirements. Here, either generic hardware accelerators for neural networks can be used, which support a variety of NN types, or it is possible to use dedicated hardware developed for the specific application in the form of an ASIC. In most cases, much more demanding requirements can be met by an ASIC.

The performance of ML algorithms depends on many factors. A particularly important one is the set of chosen hyperparameters [1]. Specifically for NNs, there are many hyperparameters to be determined related to architecture, regularization, learning rate etc. All these impact the accuracy and efficiency of the network. Classical optimization methods to determine hyperparameters such as Grid Search or Random Search [2] are inefficient for hyperparameter determination for DL, because the evaluation of the objective function, e.g. training of a NN, is very expensive.

Another case where the evaluation of the objective function is very expensive is the design space exploration of an ASIC. Due to the high complexity of the ASIC design process, methods that support the development of application-specific NN accelerators are urgently needed. It is essential to determine the configuration of the NN in such a way that conflicting requirements can be fulfilled, such as high accuracy and energy efficiency. Afterwards, the ASIC design and NN hardware accelerator are automatically generated for evaluation. Bayesian Optimization (BO) [3] models an objective function using a probabilistic surrogate function and can be used to reduce the number of objective function evaluations. Hence, it is well suited for expensive optimization problems.

The contributions of this paper are twofold. First, we developed a highly configurable hardware architecture to generate hardware representations of neural networks for time series classification. Second, we present an approach that integrates NN hyperparameter optimization and training, ASIC generation and evaluation in order to find a accelerator configuration that is at the same time optimal regarding NN classification

This work was supported by the Federal Ministry of Education and Research (BMBF) in the KI-Sprung POMAA project under the Grants 16ES1139K and 16ES1140.

accuracy and energy efficiency. We evaluate the proposed approach in a biomedical application, where the goal is to detect atrial fibrillation in human electrocardiograms.

II. State of the Art

A. Bayesian Optimization

BO [3] models an objective function based on already seen data points, $(\mathbf{x}_0, y_0), ..., (\mathbf{x}_{i-1}, y_{i-1})$ with the help of a surrogate function, where $X = X_1 \times ... \times X_n$ is an n-dimensional decision space, and y_i are the usually noisy observations of an expensive objective function $f : X \to \mathbb{R}$. The surrogate function can be modeled using probabilistic methods, such as Gaussian processes [4], with the drawback that they use complex and computationally expensive approximation methods and are primarily suitable for modeling continuous parameter spaces. An alternative is the Tree Parzen Estimator (TPE) [5], which can also model discrete, categorical, and conditional parameter combinations. The TPE uses a kernel density estimator to model the densities over the input configuration space rather than the objective function for a single objective directly by $p(\mathbf{x}|y)$.

Multi-objective optimization can be performed with an approach that integrates BO adapted to address multiple goals [6]. This problem is formulated as

$$\text{minimize } \mathbf{y} := f(\mathbf{x}) = (f_1(\mathbf{x}), ..., f_m(\mathbf{x})) \text{ subject to } \mathbf{x} \in X \tag{1}$$

where $f_i : X \to \mathbb{R}^m$ is a m-dimensional objective function. In many applications, objectives conflict with each other. In our case, e.g., we aim to optimize for minimal model loss and minimal number of computations. In this manner, we aim to obtain a set of Pareto-optimal solutions with a minimal number of evaluations of f. The Multi-Objective TPE (MOTPE) extends the TPE with the following probability density functions,

$$p(\mathbf{x}|y) = \begin{cases} l(\mathbf{x}) & \text{if } \mathbf{y} \succ Y^* \cup \mathbf{y}||Y^* \\ g(\mathbf{x}) & \text{if } Y^* \succeq \mathbf{y} \end{cases} \tag{2}$$

where Y^* is a set of objective vectors that satisfy $p(\mathbf{y} \succ Y^* \cup \mathbf{y}||Y^*) = \gamma$. The probability density function $l(\mathbf{x})$ refers to the observed decision vectors $\{\mathbf{x}^{(i)}\}$ that satisfy $\mathbf{y}^{(i)} (= f(x^{(i)}) \succ Y^*$ or $\mathbf{y}^{(i)}$, while $g(\mathbf{x})$ models the remaining vectors.

These probability functions are tree-structured hierarchical processes constructed using adaptive Parzen estimators [5]. The observations are split into D_l and D_g by sorting them based on the $\mathbf{y}$ values. The TPE employs the following Expected Improvement (EI) function as the acquisition function,

$$EI_{y^*}(\mathbf{x}) \coloneqq \left(\gamma + (1-\gamma)\frac{g(\mathbf{x})}{l(\mathbf{x})}\right)^{-1} \tag{3}$$

Multiple candidates are sampled from $l(\mathbf{x})$, the best candidate $\mathbf{x}$ with the largest EI value is selected in each iteration. By not using all observations TPE is more computationally more efficient. Theoretically, MOTPE constructs models in the same way the standard TPE does by implementing an algorithm that splits observations. This algorithm uses Y^* such that $(\mathbf{y} \succ Y^* \cup \mathbf{y}||Y^*) = \gamma$ to split the observations into subsets for $l(\mathbf{x})$ and for $g(\mathbf{x})$. Practically, the observations are directly split for a specific γ in a greedy two step algorithm. The first step appends better non-domination ranked data points to the largest extent possible to D_l. The second step appends the set obtained as a result of the Hypervolume Subset Selection Problem (HSSP) [7] to D_l. HSSP solves the problem of finding a subset of a specific size that maximizes the hypervolume indicator with a given reference point. HSSP is known to be a sub-modular maximization and therefore possible to obtain a $1 - 1/e$ approximation via the greedy method [8].

B. Hardware Accelerators for Neural Networks

Designing accelerators for NNs has recently received increasing interest in industry and academia [9] [10]. However, in general these accelerators are domain specific accelerators that target a whole class of neural networks, while the optimization of an NN accelerator architecture for a specific problem is rarely used. Since many embedded systems are designed for a single application, our goal is to provide a hardware architecture that is highly optimized for the target problem.

III. Neural Network Template Architecture

First, it was important to identify a suitable NN template architecture for ECG sequence classification that obtains a high classification performance and has a comparatively simple structure. We used a Fully Convolutional Network (FCN) [11], which consists entirely of one-dimensional convolutional (Conv1D) layers without local pooling. The FCN was modified for use in the ASIC as follows (see also Fig. 1). First, the input Conv1D layer was replaced by a dilated convolution to subsample the data and reduce the memory requirements and computations. Second, the global average pooling was replaced by a global max pooling to avoid division operations. Third, to avoid unnecessary computations, the Conv1D layers and subsequent Batch Normalization (BN) [12] can be combined after the training of the network. The new weights and biases of the Conv1D layer can be calculated as $w_{fold} := \frac{\gamma w}{\sqrt{\sigma^2+\varepsilon}}$ and $b_{fold} := \gamma\frac{b-\mu}{\sqrt{\sigma^2+\varepsilon}} + \beta$. Finally, the number of filters, size of convolution kernels and dilation factor were used as configurable hyperparameters in the architecture optimization.

IV. System architecture

A. High-Level Architecture of the System

As shown in Fig. 2, the system has been implemented as a microcontroller unit containing a generic RISC-V CPU and a dedicated NN IP-Core. The CPU controls system operation and performs general software tasks. The CAECO NN-IP (see Section IV-B) is connected to the RISC-V core memory bus and can be accessed via memory-mapped registers. An additional data path has been integrated, to pass data directly to the CAECO via a data buffer. This buffer is populated with classification data and weights via a JTAG interface that is part of the RISC-V core, which is then transferred into the CAECO via an AXI-Stream interface. The same JTAG interface is also

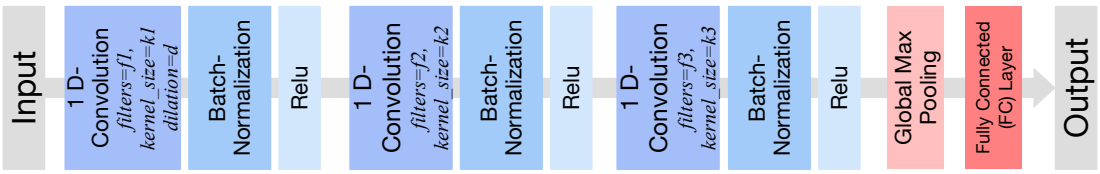

Fig. 1. Template NN-architecture based on the 1D Fully Convolutional Network [11].

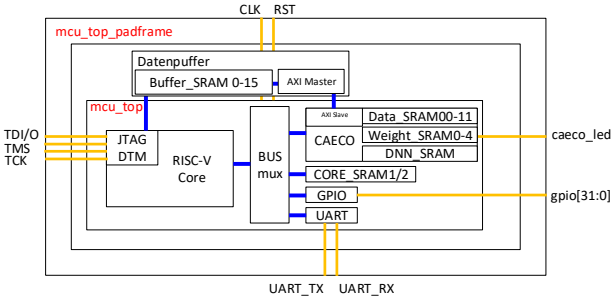

Fig. 2. Block diagram of the system architecture

used to program the RISC-V core. SRAM memory is provided to the RISC-V core to store data and instructions using a von-Neumann approach and is multiplexed onto the core's data and instruction buses. In addition, the system has 32 GPIOs and a UART interface. The software reports system availability after programming via an UART message. Upon completion of a classification, the CAECO triggers an interrupt that causes the RISC-V core to send a new message over the UART interface. The entire system is clocked at 10 MHz. All components relevant for the evaluation of the energy efficiency of the CAECO were grouped under the *mcu_top* instance at the top design level. All other components, such as the I/O and supply pads the SRAM modules of the data buffer and the master of the AXI interface are located outside the *mcu_top* instance on the top hierarchy level of the design *mcu_top_padframe*. The used RISC-V core is a customized version of the Z-scale project [13], which has low complexity and power consumption. Accordingly, the RISC-V core offer the RV32IM basic instruction set with multiplication expansion and has a three-stage single-issue in-order instruction pipeline. The core supports 3 different interrupt sources and allows the integration of periphery devices and the instruction and data memories via AHB-Lite buses.

B. Architecture of the Hardware Accelerator

To implement the NN IP-Core, we developed a generator that can create configurable hardware architectures for 1D convolutional neural network, called Configurable Accelerator Engine for Convolution Operations (CAECO).

1) Base architecture: The CAECO consists of the following main components (see Fig. 3). A streaming interface based on AXI-Stream is used to input data and weights. Registers can be used to set operating mode (configuration or classification) and read results. Directly connected to the streaming interface is the unit for implementing the dilated convolution of the first layer and the normalization of the data. A data buffer contains the data that will be reused during a convolution operations at a given time. Four Processing Elements (PE) are shown as PE 0…PE3), as shown in Fig. 3, are used to compute the network layer activations via multiply-accumulate operations. In total $p + 1$ memories with 1024 words and 8 bit word length are used to store the weights of the neural network, from which p memories store the filter weights of the Conv1D layers and a single memory stores the weights of the fully connected layer and the bias values of all layers. The memories can be configured with new weights via the streaming interface. The activations computed by the PEs are serialized in the Activation-Unit via a shift register. Then, the bias values are added and and a Rectified Linear Unit (ReLU) activation function is applied. The intermediate results of one layer of the network are buffered in a Data Storage, consisting of 4 storage elements with 4096 32 bit words each. The coordination of the different elements is controlled by the Control Finite State Machine (FSM).

2) Resolution optimization: All operations in CAECO use fixed-point numbers with minimal word width. For this, we used the Dynamic Fixed Point (DFP) [14] format that was based on a similar approach that exists for classifying image data using 2D Convolutional Networks [15]. The DFP format uses a different quantization for each layer of the FCN, for input values as well as for weights and biases. One of the main changes to this approach is a change to the ReLU activation function. The output activations of the ReLU were limited to a range of 0 to 1, for which we did not observe a substantial decrease in classification accuracy. This allows to use the same fixed-point quantization in the activation of each layer. However, the value range of the weights and biases is larger and differs between layers (as shown in Table I). In this example, to cover the complete range of values, 4 integer bits are needed for the first Conv1D layer and the FC layer and 3 integers for the other layers. The DFP quantization allows us to reduce the word width of the data and weights to 8 bits without a major loss in classification performance compared to a floating-point TensorFlow model. For the intermediate values 16 bits are used so that the larger range of values resulting from the multiplication can be covered. The fixed point format used for these values depends on the format used for input values and weights. After the ReLU activation the values are reduced to 8 bits.

For further improvements, we applied the following opti-

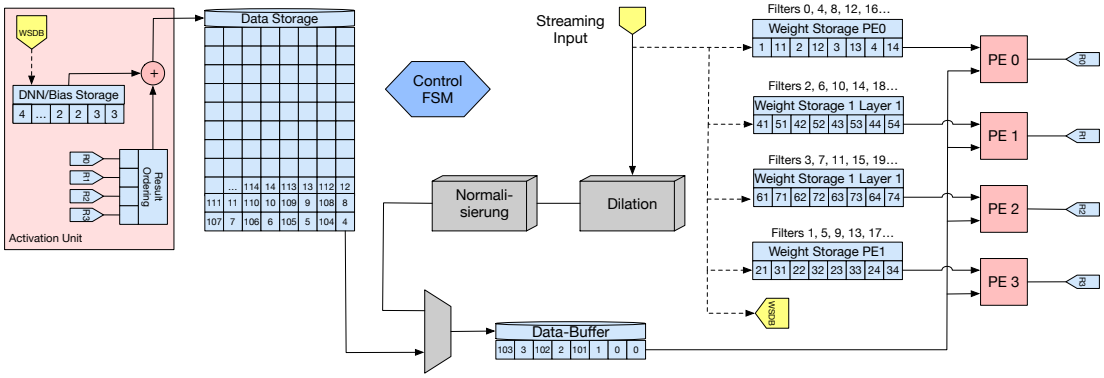

Fig. 3. Schematic representation of the CAECO

Layer	w_{min}	w_{max}	b_{min}	b_{max}
1^{st} Conv1D	-7.4727	6.3586	-1.2847	0.9688
2^{nd} Conv1D	-2.5471	2.5258	-1.2987	0.6000
3^{rd} Conv1D	-1.7824	1.0335	-2.3894	2.5535
FC	-4.0998	4.5852	-0.6542	0.6542

TABLE I
VALUE RANGES OF THE WEIGHTS ON ALL LAYERS OF THE FCN

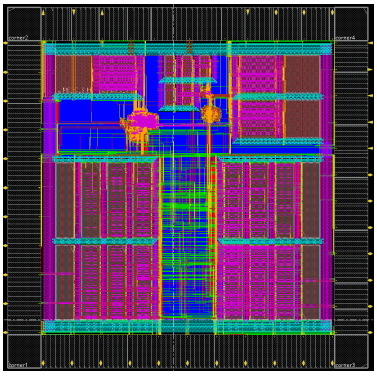

Fig. 4. Layout representation of the chip die in Innovus implementation tool

mization procedures. First, the input data was limited to a range between -1 and 1 by clipping. Second, in addition to the normalization of the data, the BN also performs centering and scaling, which increases the value range and hence the necessary number of integer bits for a fixed-point representation. Since a scaling of the values is already performed by the ReLU activation, additional scaling can be omitted.

C. Implementation Details and Simulation Methodology

To evaluate the energy efficiency of the hardware architecture, the ASIC shown in Fig. 4 has been developed in 22nm FDSOI CMOS technology in Globalfoundries. It uses the 10M_2Mx_5Cx_1Jx_2Qx_LB metal stack and has an area of $1.5mm \times 1.5mm$, according to the properties of Mini-ASICs for MPW production at Europractice. The Synopsys Designware 6.75 track 116m pitch standard cell libraries gf22nsdllogl20-36edp116a with transistor channel lengths between 20 and 36 nm were used for implementation. ThRedwood City, CAe 3.3V I/O cells and supply pads from the Synopsys Designware Area Efficient GPIO library are used in the pad ring (dwc_io_in_3p3v_ae_gpio_fs). Memory for the data buffer was generated using the DesignWare SiWare Automotive Grade 1 Single Port High Density and Performance Leakage Control SRAM 2M Sync compiler. The SRAM modules in the RISC-V core and CAECO have been generated with the Synopsys Ultra-Low-Leakage Single-Port SRAM, Low-Leakage Periphery (S1H) compiler. The RISC-V core uses two 4096×32Bit SRAM modules as a 32kByte shared data and instruction memory. In the CAECO, a total of four 4096×32bit SRAM modules are used for the classification data and five 1024×8bit SRAM modules are used for the weights. The data buffer consists of sixteen 4096×32bit SRAM modules. All memories go into a sleep or power gate mode while maintaining data to reduce leakage current consumption. As can be seen from the floor plan, most of the chip area is occupied by the memory modules. The design consists of 57293 standard cells, which occupy a total area of 22029,423 μm^2. The chip density is 58% while the standard cell density is just 3.303%. Synthesis with Genus was performed with highest leakage optimization effort and clock gating enabled. With a count of 5884 registers, 437 clock gates were inserted, resulting in an average clock tree power consumption saving of 43.14%. The conditions used in synthesis, place & route (P%R) and power simulation are listed in Tab. II and III. In power simulation of the post P%R netlist using the Xcelium simulator, the program memory of the RISC-V core, the buffer memory and the weight memories are preloaded. The input data is transferred from the buffer memory to the CAECO for classification. The simulation ends when the result is available. The signal delay times of the implementation at nominal conditions are exported as an SDF file from Innovus and used in Xcelium to annotate all signals. The signal waveforms are exported in TCF format and imported into Voltus for power consumption calculations. Voltus Static Power Engine was used following the Vector-based Average Power Calculation method. The vectors are taken from the simulation results recorded in the TCF file.

View	Corner	VDD /mV	Temp(°C)	Parasitics
av_typ	TT	0.80	25	nominal
av_max	SSG	0.72	125	FuncRCmax
av_min	FFG	0.80	-40	FuncRCmin

TABLE II
USED PLACE & ROUTE VIEWS

Tool	Corner	VDD /mV	Temp(°C)
Synthesis	SSG	0.72	125
Power	TT	0.80	25

TABLE III
PARAMETERS USED IN SYNTHESIS AND POWER SIMULATION

Consumption Type	Mean(mW)	Standard Deviation (mW)
Internal	0.133831579	0.000224904
Switching	0.027555789	0.00015274
Leakage	0.017816842	$7.98 * 10^{-6}$
Total	0.179194737	0.000373425

TABLE IV
MEAN VALUES AND STANDARD DEVIATION OF POWER CONSUMPTION BY TYPE IN mW.

V. EVALUATION RESULTS

The dataset used for the evaluation is an ECG study provided by the Charite hospital [16]. It contains of 16,000 recordings of 2 min duration, recorded at 512 Hz. 8,000 examples correspond to sinus rythm and 8,000 to atrial fibrillation. For evaluation, the dataset was randomly split into a training dataset (80%) and validation dataset (80% of the data).

A. Multi-Objective Optimization

The following results were obtained in a MOPTE study for the following three criteria:

$$\min_{\mathbf{x}} FLOPs(\mathbf{x}), \max_{\mathbf{x}} TPR(\mathbf{x}), TNR(\mathbf{x}) \tag{4}$$

where $\mathbf{x}$ are the configurable hyperparameter values shown in Fig. 1. During the training of the network, the dilation was extended to support data augmentation by generating new training examples that contain all data points that that would otherwise be discarded by the dilation operation. Fig. 5 shows the Pareto plot of a total of 160 finished trials. We safety margins of TPR$> 95\%$ and TNR$> 0.9\%$, i.e. desired minimal classification performances. As observed, the MOTPE strategy was successful in finding NN's with acceptable objective values. The best MOTPE configuration corresponds to filters $\{8, 8, 8\}$ and kernel sizes $\{3, 3, 5\}$ for each layer respectively. This network achieved 97.18%, 96.56%, 90.84% and $1.92M$ for TNR, TPR, Accuracy and FLOPs, respectively.

B. Final Classification and Functional Verification

A new neural network was trained on the entire dataset with the selected hyperparameters. The weights of the NN were extracted and converted using the steps shown in Section IV-B2 for fixed-point arithmetic. Subsequently, RTL simulations of the CAECO were performed on all $16,000$ given data sets and achieved a TPR of 95.07% and a TNR of 98.04%. These values differ slightly from those obtained before, because they represent the reproduction error of the NN trained on the entire data after fixed-point conversion and using RTL simulation. For the chosen network configuration and clock frequency, each classification requires exactly 30.3343 ms for completion.

C. Energy efficiency

The final evaluation concerns the determination of energy efficiency on 20 different ECG examples. For this purpose, post-place and route simulations were performed and power consumption was determined using Voltus. The data includes the values of the power consumption of the RISC-V core and the CAECO as well as all subcomponents, while the power consumption of the required data buffer and the pad ring was not considered. The results for the estimation of the power consumption are shown in Tab. IV. The total power consumption are 180 µW. In first place are the memories for the weights of the network, whose power consumption is in turn significantly determined by the Internal Power. It can be concluded from this that the data transfer of the weight data from the SRAMs into the logic of the accelerator is the determining factor of the power consumption of the CAECO. This is immediately followed by the SRAM in which the instruction data of the RISC-V core is stored and constantly accessed during execution and the 4 SRAMs in which the intermediate results of the CAECO are stored. After the SRAMs take drivers of the global clock tree (CTS*) and driver cells of the data RISC-V AHB-Lite data bus. The next instance assigned to the CAECO is only at the sixtieth position in the Voltus Report sorted by descending power consumption and is a driver for the data signals which are fed from the data buffer into the CAECO.

VI. CONCLUSIONS AND IMPROVEMENT OPTIONS

In this paper, we showed that high-level optimization using machine learning methods enables the identification of accurate and computationally efficient NN architectures, which are also suitable for ML hardware accelerators. Moreover, a highly configurable hardware accelerator for time series classification has been developed with CAECO, which directly translates such an NN architecture into an RTL hardware description that could be successfully used in an ASIC. However, a number of further optimization possibilities arise.

A. Further improvements to the CAECO

Currently, it is possible to specify the CAECO architecure before synthesis with respect to numerous properties (number of channels, PEs, ...), while the weights can also be changed later. In future, we want to make the structure of CAECO itself configurable (e.g. number/sizes of layers and size filters) within certain limits (e.g., limited by internal memories sizes).

Fig. 5. Multi-Objective Tree Parzen Estimator trials pareto plot for TPR and TNR.

B. Future Improvements of the ASIC architecture

A possible improvement of the ASIC architecture would be to introduce a second independent supply domain with its own power pad for all components to be able to validate the predicted power consumption after production of the chip. Furthermore, it should be verified whether the system can be implemented at lower nominal voltage than 0.8V in sign-off quality to reduce the power consumption. The exploitation of the body bias feature of the technology is currently not used and power management based on dynamic frequency and voltage scaling was not integrated into the design. Here, the supply voltage and the clock frequency can be dynamically controlled so that the system is supplied with the minimum power required to perform its tasks. Initially, a design of a voltage regulator including band-gap voltage reference and biasing network for a maximum load current of 200mA and quiescent current $< 1\%$ was created at schematic level. However, it was not integrated into the system since initial power simulations that the quiescent current of this block was above the average current draw of the current system.

C. Improvement of Multiobjective Optimization

The optimization approach used shows that it is possible to find NN architectures that are energy efficient while retaining good classification results. Meaningful next steps would be the integration of the MOTPE optimization into the full ASIC design flow to include further parameters of the ASIC design into the optimization. Here, we expect further significant improvements in energy efficiency.

D. Further Potential Applications

Altough the experiments in this work were performed using ECG data, the relevance of this technology is transferable to other applications, like EEG and EMG processing [17]. An application in the field of rehabilitation robotics would be in prosthetics and exoskeletons [18], where local processing of biosignals is mandatory to guarantee the mobility of a patient.

References

[1] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in Neural Information Processing Systems*, 2012.

[2] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," Tech. Rep., 2012.

[3] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the Human Out of the Loop: A Review of Bayesian Optimization," Tech. Rep., 2016.

[4] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT Press Cambridge, MA, 2006, vol. 2, no. 3.

[5] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for Hyper-Parameter Optimization," Tech. Rep., 1994.

[6] Y. Ozaki, Y. Tanigaki, S. Watanabe, M. Onishi, and M. . Onishi, "Multiobjective Tree-structured Parzen Estimator for Computationally Expensive Optimization Problems," *Genetic and Evolutionary Computation Conference (GECCO)*, p. 9, 2020.

[7] J. Bader and E. Zitzler, "HypE: An algorithm for fast hypervolume-based many-objective optimization," *Evolutionary Computation*, vol. 19, no. 1, pp. 45–76, 2011.

[8] A. P. Guerreiro and C. M. Fonseca, "Greedy Hypervolume Subset Selection in Low Dimensions," *Evolutionary Computation*, vol. 24, no. 3, pp. 521–544, 2016.

[9] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[10] M. Capra, B. Bussolino, A. Marchisio, M. Shafique, G. Masera, and M. Martina, "An updated survey of efficient hardware architectures for accelerating deep convolutional neural networks," *Future Internet*, vol. 12, no. 7, p. 113, 2020.

[11] Z. Wang, W. Yan, and T. Oates, "Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline," in *International joint conference on neural networks,*, 2017, pp. 1578–1585.

[12] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[13] Y. Lee, A. Ou, and A. Magyar, "Z-scale: Tiny 32-bit RISC-V Systems," in *2nd RISC-V workshop*, 2015.

[14] P. Gysel and S. Ghiasi, "Hardware-oriented approximation of convolutional neural networks," 04 2016.

[15] C. Y. Lo, F. C. M. Lau, and C. Sham, "Fixed-point implementation of convolutional neural networks for image classification," in *2018 International Conference on Advanced Technologies for Communications (ATC)*, 2018, pp. 105–109.

[16] F. Koehler, S. Winkler, M. Schieber, U. Sechtem, K. Stangl, M. Böhm, H. Boll, S. S. Kim, K. Koehler, S. Lücke *et al.*, "Telemedical interventional monitoring in heart failure (tim-hf), a randomized, controlled intervention trial investigating the impact of telemedicine on mortality in ambulatory patients with heart failure: study design," *European Journal of Heart Failure*, vol. 12, no. 12, pp. 1354–1362, 2010.

[17] H. Wöhrle, M. Tabie, S. K. Kim, F. Kirchner, and E. A. Kirchner, "A hybrid FPGA-based system for EEG- and EMG-based online movement prediction," *Sensors (Switzerland)*, vol. 17, no. 7, jul 2017.

[18] S. Kumar, H. Wöhrle, M. Trampler, M. Simnofske, H. Peters, M. Mallwitz, E. A. Kirchner, and F. Kirchner, "Modular design and decentralized control of the RECUPERA exoskeleton for stroke rehabilitation," *Applied Sciences (Switzerland)*, vol. 9, no. 4, feb 2019.