Neuro-inspired Hardware Acceleration for Efficient Biomedical Signal Processing in Mobile Sensing Devices

Von der Fakultät für Elektrotechnik und Informationstechnik der Rheinisch-Westfälischen Technischen Hochschule Aachen zur Erlangung des akademischen Grades eines Doktors der Ingenieurwissenschaften genehmigte Dissertation

vorgelegt von

Johnson Loh, M.Sc.

aus Aachen, Deutschland

Berichter: Univ.-Prof. Dr.-Ing. Tobias Gemmeke

Univ.-Prof. Dr.-Ing. Anke Schmeink

Tag der mündlichen Prüfung: 20.03.2025

Diese Dissertation ist auf den Internetseiten der Universitätsbibliothek online verfügbar.

Abstract

The processing of biomedical signals in mobile sensing devices enables the continuous monitoring of health parameters for early detection of threatening arrhythmia in the population through convenient wearable devices, such as smartwatches. The design of processing modules, which are feasible in this resource-constrained environment, is subject to multiple constraints for their deployment in-field. High quality classification is desired for accurate detection to trigger treatment by trained personnel. Robust classification beyond available training data is necessary to generalize system feasibility across the general population. Low-power operation is necessary for long-term screening for sparse features indicating abnormal health conditions. The co-optimization of neuro-inspired algorithms on dedicated hardware shows the promise to address all desired specifications in an application-specific device.

This work explores neuro-inspired concepts for low-power digital processing of biomedical signals. Artificial neural networks has shown superior classification capabilities in the machine learning domain. Using an artifical neural network as a baseline, a systematic design space exploration methodology is applied to design an ECG classifier and co-optimize the system from the algorithm level down to a hardware design for ultra-low power consumption and high classification quality. Then, the system is extended with a domain generalization method for robust classification across multiple datasets. The method is designed for direct integration into a pre-trained neural network with low overhead regarding inference and training. At last, the temporal coding of information in spikes is adopted from the human brain as a data processing mechanism for low power processing. The investigated temporal coding method shows equivalent numerical values after processing with reduced operations compared to conventional fixed-point arithmetic. In the end, the neuro-inspired concepts show promising directions to improve specialized ANN hardware accelerators for biomedical signal processing both for low-power processing and robust high-quality classification.

Acknowledgements

First, I would like to thank my advisor Prof. Tobias Gemmeke for the opportunity to get an insight into academic research and all its diverse components.

Then, I am grateful for the thesis jury for their time and comments to critically review this work and discuss its contents.

During my PhD journey, I could always count on the support of my family and friends, without whom this journey would not have been possible. In particular, I would like to thank my mother for being my constant motivator through my professional career, and my sister for giving me the help and support in critical times of need.

Contents

List of Figures

List of Tables

List of Abbreviations

ADC	analog-to-digital converter
AF	atrial fibrillation
AFE	analog front-end
ANN	artificial neural network
ASIC	application specific integrated circuit
BLE	Bluetooth Low Energy
CC	computational complexity
CL	correction layer
CMOS	complementary metal-oxide-semiconductor
CNN	convolutional neural network
CVD	cardiovascular disease
CWT	continuous wavelet transform
DG	domain generalization
DNN	deep neural network
DSP	digital signal processing
DUT	device under test
DWT	discrete wavelet transform
ECG	electrocardiogram
EDA	electronic design automation

fdSOI	fully-depleted silicon-on-insulator
FIFO	first-in first-out
FIR	finite impulse response
FPGA	field programmable gate array
FSM	finite state machine
GPU	graphics processing unit
GRU	gated recurrent unit
HDL	hardware description language
IF	integrate-and-fire
IC	integrated circuit
ICG	integrated clock gating
ICM	implantable cardiac monitoring
IID	idependent and identically distributed
IoT	internet of things
KPI	key performance indicator
LSB	least significant bit
LSTM	long-short term memory
MAC	multiply-and-accumulate
MEMS	micro-electromechanical systems
ML	machine learning
MLP	multi-layer perceptron
OOD	out-of-distribution
PCA	principle component analysis

PE	processing element
PPA	power-performance-area
QoS	quality of service
RAM	random-access memory
ReLU	rectified linear unit
RISC	reduced instruction set computer
RMSE	root mean square error
ROM	read-only memory
RTL	register-transfer level
SGD	stochastic gradient descent
SIMD	single instruction multiple data
SIPO	serial-in parallel-out
SNN	spiking neural network
SoC	system-on-chip
SRAM	static random-access memory
TCN	temporal convolutional network
TTFS	time-to-first-spike

Author's Publication List

[JL1] **J. Loh**, J. Wen, and T. Gemmeke, "Low-cost DNN hardware accelerator for wearable, high-quality cardiac arrythmia detection," in *2020 IEEE 31st International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, IEEE, Jul. 2020. DOI: 10.1109/asap49362.2020.00042.

[JL2] C. Latotzke, **J. Loh**, and T. Gemmeke, "Cascaded classifier for pareto-optimal accuracy-cost trade-off using off-the-shelf ANNs," in *Machine Learning, Optimization, and Data Science*, Springer International Publishing, 2022, pp. 423–435. DOI: 10.1007/978-3-030-95470-3_32.

[JL3] **J. Loh** and T. Gemmeke, "Dataflow optimizations in a sub-uW data-driven TCN accelerator for continuous ECG monitoring," in *2022 IEEE Nordic Circuits and Systems Conference (NorCAS)*, IEEE, Oct. 2022. DOI: 10.1109/norcas57515.2022.9934591.

[JL4] **J. Loh** and T. Gemmeke, "Lossless sparse temporal coding for SNN-based classification of time-continuous signals," in *2023 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, IEEE, Apr. 2023. DOI: 10.23919/date56975.2023.10137112.

[JL5] C. Lanius, J. Lou, **J. Loh**, and T. Gemmeke, "Automatic generation of structured macros using standard cells – application to CIM," in *2023 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, IEEE, Aug. 2023. DOI: 10.1109/islped58423.2023.10244608.

[JL6] Y. Chen, J. Lou, C. Lanius, F. Freye, **J. Loh**, and T. Gemmeke, "An energy-efficient and area-efficient depthwise separable convolution accelerator with minimal on-chip memory access," in *IFIP/IEEE 31st International Conference on Very Large Scale Integration (VLSI-SoC)*, IEEE, Oct. 2023. DOI: 10.1109/vlsi-soc57769.2023.10321918.

[JL7] **J. Loh** and T. Gemmeke, "Stream processing architectures for continuous ecg monitoring using subsampling-based classifiers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 1–11, 2023, ISSN: 1557-9999. DOI: 10.1109/tvlsi.2023.3329360.

[JL8] A. Ayad, M. Barhoush, **J. Loh**, T. Gemmeke, and A. Schmeink, "PEACE: Private and energy-efficient algorithm for cardiac evaluation on the edge using modified split learning and model quantization," in *2023 14th International Conference on Information and Communication Systems (ICICS)*, IEEE, Nov. 2023. DOI: 10.1109/icics60529.2023.10330472.

[JL9] M. Lahr, **J. Loh**, M. Schwarz, M. C. Lemme, and T. Gemmeke, "Vehicle surroundings perception using micro-electromechanical systems inertial sensors," *Advanced Intelligent Systems*, Apr. 2024, ISSN: 2640-4567. DOI: 10.1002/aisy.202300679.

[JL10] **J. Loh**, L. Dudchenko, J. Viga, and T. Gemmeke, "Towards hardware supported domain generalization in dnn-based edge computing devices for health monitoring," *IEEE Transactions on Biomedical Circuits and Systems*, pp. 1–10, 2024, ISSN: 1940-9990. DOI: 10.1109/tbcas.2024.3418085.

Chapter 1

Introduction

Futuristic scenarios in science fiction envision high cognitive abilities in compact form factor. One example is an exoskeleton armor, the so-called “Iron Man Suit”, which possesses the capability to monitor the health parameter of its wearer and the environment in real-time. The captured data, such as blood toxicity or flight acceleration, are analyzed instantaneously for accurate interpretation and corresponding reaction. Although this example describes an ideal system interaction in a fictitious story, the core element of continuous data stream analysis in mobile devices is highly desirable in many application fields, especially the domain of healthcare. In this case, the monitoring of vital parameters enable automated detection of asymptomatic heart anomalies, such as atrial fibrillation (AF) [1], and, potentially, save lives with early treatments by healthcare professionals. Recent studies already aim to incorporate this functionality in wearable devices, e.g. smartwatches [2].

When we imagine the idealistic scenario of vital parameter monitoring, one essential component to realize the desired cognitive ability is the data processing component. It receives data from sensors and processes it into an interpretation of the data. The challenges involved with the design of such data processing component for mobile devices usually revolve around two opposing targets: High QoS and low power consumption [JL1]. One the one hand, high quality analysis require high complexity [3] and, therefore, high power consumption. Low power devices, on the other hand, feature long device operation, but are severely limited in terms of task complexity and quality [4]. Especially, biomedical signal processing requires robustness for the classification algorithm, since distinctive features in the data are very subtle and sensitive to noise. As the characteristic electrophysiological signal is measured indirectly through surface potentials of the patients body [5], the variety of noise sources is diverse and can impact data analysis significantly.

ANN provide the capability to solve highly complex tasks. One prominent example is the classification of predefined and labeled datasets, in which it achieves remarkable performances even exceeding humans in several data domains, such as images [6] or ECG data [3]. Nevertheless these state-of-the-art networks come with a significant cost. Best performing ANN models require massive amounts of memory and computational resources in traditional hardware [7]. Based on the trends over the last decade the computational requirements are increasing in an exponential rate [8]. In the context of mobile ECG monitoring, this trend poses significant challenges on multiple abstraction levels of the device design process. In essence, the design of the ANN model as well as the hardware component performing its inference need to be rethought from scratch to provide "the best of both worlds".

To push the limits of state-of-the-art data processing components, the research field of *Neuromorphic Computing* draws inspiration from its biological reference - the human brain [9]. Optimized over countless generations in evolution, the human brain is able to achieve cognitive tasks not yet in reach for conventional computing systems, whilst only consuming little power [10]. Although the field of neuromorphic computing started with the emulation of analog implementations of neural features in silico [11], it evolved into a wider domain focusing on non-von Neumann computers with neuro-inspired functions and structures [9]. In the context of vital parameter monitoring in mobile devices, these neuro-inspired features promise to achieve both high QoS and low power consumption at the same time. One prominent neuro-inspired principle is the communication with sparse activations, or more specifically spikes [12]. SNNs embody the core feature of many state-of-the-art neuromorphic systems on both a large and small scale [13]. While these systems face different challenges and target different objectives, a major desired feature of spiking communication is the compact representation of information. In the end, this representation is expected to provide the same information content for less energy compared to traditional methods. For practical applications like ECG classification, a SNN model can be acquired with different approaches.

To summarize, the continuous monitoring of biomedical signals with high diagnostic quality pose a significant challenge for specialized hardware in mobile devices. Within this domain, neuro-inspired concepts, such as neural networks and even more biologically inspired features, promise high quality classification, which need to be fitted for real-time application case. In this application, the low data-rate of the input signal compared to the clock frequency of modern digital circuits [14] introduces a unique design constraint to be considered in the optimization process.

1.1 Thesis Contributions

The target of this thesis is to answer the following research question:

> What neuro-inspired concepts enable low-power digital processing of real-time, low data rate signals?

First, a suitable application task, i.e. ECG classification is defined for exploration. Especially, the prerequisites need to be clarified for a systematic selection and optimization of available methods. In essence, the state-of-the-art design methodologies and evaluation methods are discussed in the context of the target application (see Section 2). Then, the adapted design methodology and evaluation method is applied in the design of an ANN accelerator. The co-design of the resulting ECG classification systems spans across multiple abstraction levels of a digital design flow. The design steps include algorithm design, e.g. training and optimization of an ANN classifier, over the mapping to hardware components in system level and RTL down to the structured placement of standard cells (see Section 3). The ANN accelerator is used as a baseline to incorporate two features, which are observed in a biologically plausible system. Firstly in Section 4, domain-invariant feature representations are incorporated into the ANN classification system. These are realized with hardware friendly correction layers inserted into a pre-trained ANN, such that the features are aligned across domains with low hardware overhead. Secondly in Section 5, the ANN classification system is converted into an equivalent SNN. Its quality of the classification is undistinguishable to the fixed-point reference, while the sparse activation representation as well as its corresponding temporal computation result in less addition operations than the ANN baseline. In the end, this thesis shows that systematic selection of neuro-inspired concepts can ensure high-quality classification, robust classification and low power real-time processing of low data rate signals.

Chapter 2

Prerequisites for Efficient Real-Time Processing of Low Data Rate Streams

Considering the overarching research question from Section 1.1 the investigation needs to cover following items:

- **Representative Application:** The chosen application benchmark needs to reflect the expected target features. A well fitting use case is ECG monitoring as it is commonly sampled with a low data rate and benefits from the real-time processing in the digital domain. In Section 2.1, the properties of ECG signals, the corresponding processing task and their benchmarking is discussed in the context of recent literature.

- **System Scope:** The state-of-the-art in neuro-inspired concepts, especially neural networks, exhibit different granularities of bio-plausible features as well as digital processing concepts. To enable a structured exploration over existing and novel methods, the scope of the exploration needs to be defined to a promising subset within a global application context. Section 2.2 discusses the suitability of ECG processing on the edge, in particular ANNs, as an effective early warning system and design concepts, which target this setup for pre-processing on the edge.

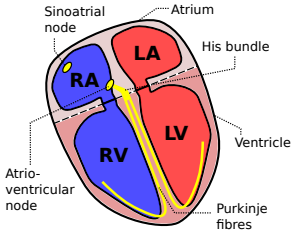
- **Design and Evaluation Strategy:** Despite an existing pool of valid methods, which can be used for a design space exploration, a methodology is required to systematically select methods to converge against high quality designs. Especially, the co-optimization of both algorithm model and hardware requires design steps across multiple abstraction levels, which need to be guided by application specific constraints. Section 2.3 outlines a methodology for design space

exploration specialized for streaming input data and discusses an energy estimation policy for ANN workloads.

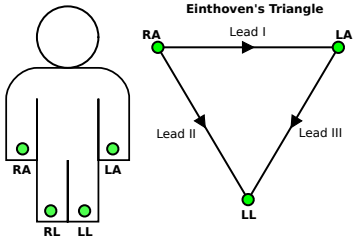
2.1 Application Context of ECG Monitoring

An *electrocardiogram (ECG)* is a time-continuous signal representing the electrical activity from the human heart and is used for the diagnosis of life-threatening heart conditions [15]. Surface potentials on the skin are picked up over electrodes and the difference between two electrodes is measured over time. These measurements are used to derive the electrical stimuli responsible for heart contraction. Potential deviations from standard cardiac rhythms indicate abnormal function of the heart and, thus, a cardiovascular disease (CVD).

2.1.1 Cardiac Electrophysiology



(a) Pacemaker cells automatically generate stimuli for propagation in specialized conduction tissue to trigger the contraction of the heart muscle sections in a specific order.



(b) Position of electrodes for standard limb leads I (LA → RA), II (LL → RA) and III (LL → LA)

Figure 2.1: Overview of cardiac stimuli for the heart and the first three ECG leads and corresponding electrode positions at the extremities adapted from [15].

Figure 2.1 shows a minimal overview of cardiac electrophysiology. The heart muscle is automatically stimulated by electrical signals generated by pacemaker cells and their spread through special conduction tissue (see Fig. 2.1a). The resulting contraction of the heart muscle leads to blood pumping through the cardiovascular system. A non-invasive method to capture these electrical stimuli is to attach electrodes on the surface of the skin and derive directions between the attached electrodes, i.e. leads. A standard ECG consists of 12 leads to capture the electrical propagation front from a wide variety of angles and planes. The first three leads are visualized in Fig. 2.1b. In principle, they capture three different angles within frontal plane of the human body. Since the propagation front of

the electrophysiological signal is three dimensional, 12 leads should provide sufficient information for analysis in all directions.

2.1.2 Mobile ECG Monitoring

Although the 12 lead ECG is preferable in a clinical setting for maximum amount of information, it is not practical to realize in a mobile setting. An effort to develop wearable cardiac monitoring devices started in the early 1950s, i.e. the Holter monitor [16], and still persist today to achieve the "ultimate" monitoring device without restrictions for the wearer. While classical Holter monitors consists of a recording device taped to the patient and attached to a reduced number of leads, modern heart monitors even go down to single lead heart monitors. These feature different form factors and user interfaces such as a heart patch [17], a smartwatch [2] or a dry electrode pair for fingers [18]. The influence of the device form factor is even more pronounced in implantable cardiac monitoring (ICM) devices [19].

As evident simply from the diverse set of device form factors, the setup to capture ECG data is not always coherent. One example is the position of electrodes, which differs from device to device, i.e. finger-to-finger vs. two adjacent patches. The variety of measurement setups is reflected in the variety of ECG datasets in popular databases, such as PhysioBank [20]. Furthermore, external noise is also influencing ECG data quality through e.g. baseline wander, power-line interference or muscle artifacts [21]. Evidently, the influence of noise is more pronounced in a mobile application setting than a stationary setup. As noise sources can be mitigated in a known environment, one challenge in a mobile monitoring setting is the handling of noise in an unknown environment.

Since a variety of CVD is associated with characteristics within the ECG beat, it is essential to distinguish those features from noise. In the continuous monitoring use case, the arrhythmia types are typically asymptomatic, i.e. do not exhibit explicit symptoms. One of the most common cardiac arrhythmia is AF. In 2023, the European Union is anticipated to have an estimated amount of 14-17 million people diagnosed with AF [1]. This trend is increasing every year and these patients are likely to exhibit life-threatening diseases, which is often missed due to the lack of proper monitoring methods deployable in daily life. It has been shown that screening at-risk population with conventional single-lead measurements is already cost-effective [1]. Further improvements in terms of coverage of population can be achieved by automating the screening process using already prevalent wearable devices.

2.1.3 Application Scope of This Work

Within this work, the PhysioBank database, in specific datasets with AF as a classification task is selected as a benchmark for the classification systems. For instance, the *Computing in Cardiology Challenge 2017 (CinC'17)* [18] is chosen, since it features single lead measurements in a large-scale, i.e. roughly 8500 samples, and comprises one of the largest collection of labeled AF data at its time of creation. Further, it mimics a real-world conditions by using the commercially available AliveCor device for classification with a special focus on AF detection. It is ideal as an application baseline as it necessitates long-term ECG monitoring in a low-power mobile device. Not only is the detection of AF complex due to its sparse sensitive features, but it also is a real-world problem highly relevant to the general public. Further, it poses challenges in the algorithm-hardware co-design of the specialized hardware to enable high quality processing capabilities on the edge.

2.2 Pre-processing on the Edge with ANNs

The co-design of digital processing modules requires both fine-tuned classification algorithms and tailored hardware to satisfy objectives for both high-quality classification and low-power processing.

2.2.1 State-of-the-art ECG Classification Using ANNs

Classification with ANNs reaches back more than two decades starting with multi-layer perceptrons (MLPs) [22], [23]. Publicly available datasets, such as the MIT-BIH Arrhythmia Database [24] or CinC'17 [18] within the PhysioBank database [20], accelerated ANN research. In combination with better access to better graphics processing unit (GPU) acceleration for training and supporting machine learning frameworks, like PyTorch [25] and TensorFlow [26], a large variety of classifiers emerged.

State-of-the-art ECG classification is summarized in various surveys, e.g. [27]–[29]. The predecessors of ANN models are traditional machine learning approaches such as support vector machines, decision trees/random forest, etc. Given well-selected expert features they already perform very well on even complex datasets, such as CinC'17 [18]. Within early machine learning classification, simple ANN models such as MLPs are also utilized, often for early ECG datasets, i.e. MIT-BIH arrhythmia, or as a baseline for hardware implementations. One prominent type of an ANN is the convolutional neural network (CNN). In the case of one dimensional temporal data, such as ECG, the network is also often described as a TCN [30]. In general, many ANN topologies have been tested for ECG

classification, such as recurrent neural networks (e.g. long-short term memory [31], gated recurrent units [32] etc.), deep belief networks [33] and exotic archetypes with probabilistic features or fuzzy logic [27]. The combination of different models is also common, further increasing the variety of architectures.

The classification process is mainly split into two consecutive stages: data pre-processing and the classification. The former deals with the extraction of relevant information in terms of features, which include e.g. interesting frequency components in the signal or morphological features etc. [34]. Alternatively, the features are extracted using trained feature extractors along with the classifier [35]. The latter deals with the analysis of extracted features into the prediction of a label. This usually involves a training process using labeled data from the benchmark dataset.

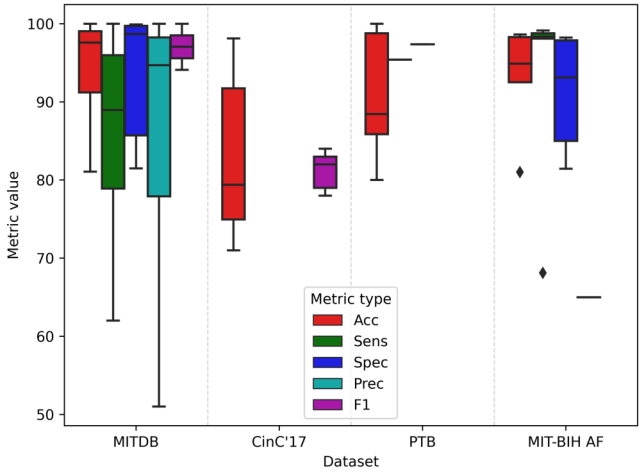


Figure 2.2: Example distribution of quality metrics for ECG classifiers derived from references in [28].

In this process, the target metric is decisive to determine the quality of the classification task, i.e. quality of service (QoS). Figure 2.2 shows an excerpt from the key performance indicator (KPI) of recent ECG classifiers. While the performance of the classifiers is seemingly very high, not only the data specifications vary in terms of above mentioned parameters, but also the classification task is redefined. For instance, the tasks include user authentication, i.e. classifying patients, specific sets of beat types, beat grouping schemes, i.e. ISO [36] or custom, or global labels for signal segments. The consequence is that there are low complexity ANNs, which can solve low complexity tasks

down to no error, and there are high complexity ANNs, which solve high complexity tasks with larger error compared to the same metric. The metric usually revolve around derivations from the confusion matrix, e.g. accuracy (Acc), sensitivity or recall (Sens), specificity (Spec), precision or positive predictive value (Prec), F1 score or harmonic mean of precision and recall (F1). In the exploration in Sections 3 to 5, the AF classification benchmarks are chosen as the target application, as the corresponding datasets, e.g. the CinC'17, require more complicated models and are not yet solvable with near perfect QoS. Thus, the resulting classifiers are expected to scale to realistic test cases.

2.2.2 State-of-the-art Digital Processing of ECG Classifiers

Despite significant advances in the quality of ECG classifiers, the deployment of those classifiers on edge devices, i.e. smartphone or wearable devices [37], is highly desirable. The motivation is clear: security risks are inherent in the transmission of critical personal data, such as vital parameters, for cloud processing [38]. Local data processing eliminates those security risks and also enables direct and personalized feedback on your own device. Nevertheless, these edge devices are limited in terms of battery capacity and their computational resources, since they are designed to be convenient in daily life. Hence, the overall deployed system needs to adapt these advanced ECG classification algorithms, i.a. ANN models, towards energy constrained devices and these devices need to consume low power in a continuous operation mode.

When considering the options to realize the digital back-end of the ECG classification device, the range of digital computing architectures is diverse. Conceptually, the architectures can be categorized in terms of flexibility and include, for instance, programmable processor architectures, field programmable gate arrays (FPGAs) and application specific integrated circuits (ASICs).

State-of-the-art Edge Processing of ECG classifiers using General-Purpose Processors

For instance, Kim et al. uses a single instruction multiple data (SIMD) processor to execute the ECG signal processing workload [39]. The algorithmic operations are translated into available instructions of the processor architecture, e.g. through a compiler. The degree of parallelism achieved is dependent on the instruction set available. Typically the processor comprises a reduced instruction set computer (RISC) architecture, such as an ARM processor [40], for less complex core logic and specialized instruction set. The optimization of program code for the specific processor architecture

is essential to reduce the latency and program memory. This software-hardware co-optimization process already leads to highly efficient designs consuming power down to tenths of a microwatt [39], [41] for low complexity ECG processing algorithms. Another method is to use the programmable processor architecture on the edge to execute a reduced classification model resulting from the *split learning* technique. For instance, Ayad et al. [42] uses the split learning technique to split an ANN model into two separately inferrable models: a large model on the server and a small model on the edge. NVIDIA Jetsons are used to deploy the reduced model on the client side. Figure 2.3 shows

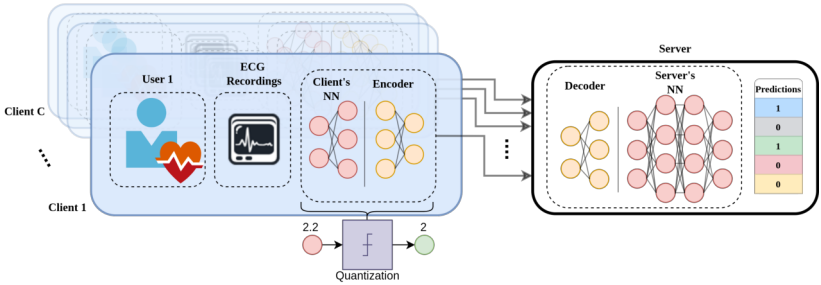


Figure 2.3: System overview of ECG classifier with a modified split-learning mechanism from [42].

the overall concept of the split learning mechanism. This concept allows to adjust the size of the ANN model, which is inferred on the edge. The general quality of the classification remains high, as intermediate features are transferred to the server side for further processing. From the perspective of the client, the computational workload and the memory required to store the model is adjusted on the algorithm level. The size of the resulting client side model after training will influence both the memory consumption and the computational efficiency on the client processor. The deployment of the model on the processor architecture is well supported ML frameworks like TensorFlow [26] and PyTorch [25]. Further reductions is achieved through ANN model quantization from a floating-point to fixed-point. It is shown that the energy consumption of one MAC operation is extremely reduced. Taking a 32-bit floating-point number representation as a reference, the post-training quantization of the ANN model can consume just 0.5 % of the original energy in 4-bit fixed-point executing otherwise the same operation. However, a minor quality reduction needs to be tolerated due to the reduced representable range of numbers.

State-of-the-art Edge Processing of ECG classifiers using FPGAs

In contrast to programmable processor architectures, FPGAs are not limited to several processing cores with a fixed hardware architecture. Instead, it consists of programmable logic blocks and interconnects. During the design time the logic blocks and interconnect is configured with a hardware description language (HDL). The circuit design can be designed to feature arbitrary logic level designs within the available resources of the FPGA. In the case of ECG classification systems, Lu et al. designed a CNN accelerator for ECG classification of MIT-BIH Arrhythmia Database [43], [44]. It features dedicated weight, bias and activation memory and buffers as well as an array of processing units. As logic optimizations are available, larger algorithm designs could be inferred with a larger amount of parallel processing units, therefore achieving remarkable hardware resource efficiency of up to 16.71 [44] GOPS/kLUT. Even though FPGAs are very powerful in the prototyping phase of the hardware design, the reconfigurable hardware requires trade-offs in terms of available resources, such as number of logic blocks, memory blocks, component sizing, power partitioning etc. Thus, the utilization of available resources is key for the efficiency of the deployed system. For instance, it is unlikely that all available resources are actually used for a specific design, as the FPGA is not designed with specific architecture with specific amount of hardware modules in mind. In this case, the idling components are still contributing to power consumption, e.g. through leakage, and, hence, significantly influence the efficiency of the overall system.

State-of-the-art Edge Processing of ECG Classifiers using ASICs

In contrast to FPGA implementations, ASICs are intended to be task-specific instead of a general purpose design. A HDL description is directly mapped to transistor layouts, which is fabricated in a foundry, e.g. in a complementary metal-oxide-semiconductor (CMOS) technology node. Early ASIC implementations for ECG processing focus on sample-and-send type devices, i.e. the edge device's function comprises the sampling of the ECG only and sends the data to another device for further processing [45], [46]. Another task with limited processing on the digital back-end is the delineation of the ECG signal. This task deals with the extraction of biomarkers in characteristic waves, which are diagnostic for heart arrhythmia [34]. Although early low complexity algorithms, such as the Pan-Tompkins algorithm [47] or others [48], already achieve sufficient detection quality, they are further investigated for more robustness across datasets and input data. One example is the ECG delineation using CNNs, in specific a U-Net architecture with around 12 convolution blocks [49]. It

is evident, that computational complexity (CC) is increased significantly to achieve greater capacity for generalization and QoS. Dedicated HW implementations for ECG delineation, however, focus on the efficient implementation of early algorithms, i.e. wavelet-based [50], [51] or Pan-Tompkins [52]. The trend towards low complexity algorithms in the hardware design domain extends into the task of ECG classification using ANNs. For instance, MLPs are used to extract features to compare with similarity functions [53], [54] or directly as the classifier using extracted features as input [55]–[57]. Interestingly, the complexity of the implemented MLP computations spreads over two orders of magnitude for classes like premature ventricular contraction, i.e. 3 neurons [55], in comparison to clinical AF, i.e. 156 neurons [57].

In contrast to MLP architectures, there are more complex ANN topologies for ECG classification on ASICs, such as CNNs [JL1], [JL3], [58], [59], gated recurrent unit (GRU) [35] and long-short term memory (LSTM) [60]. Typically, these architectures classify the raw input either sampled equidistantly over time [58] or using a level-crossing analog-to-digital converter (ADC). Other methods pre-process the signal using digital filters. For instance, discrete wavelet transforms [JL1], [JL3] or trainable filter kernels [35] are used to extract features e.g. in the frequency domain. More biologically inspired ASIC implementations utilize SNNs for ECG monitoring. These include large-scale multi-core [61], [62], mixed-signal single-core [63], [64] and digital single-core solutions [65], [66]. These works target the design of a SNN accelerator using a bottom-up approach. For instance, variants of the integrate-and-fire (IF) neuron are implemented in dedicated circuits to support the model behavior [63]–[66]. Then, the accelerator is designed with the composition of those fixed modules. Other works utilize existing neuromorphic solutions such as Loihi [62] or DYNAP [61] to reconfigure the architecture towards the ECG classification application.

Comparison of Computing Architectures for Ultra-Low Power Processing

Figure 2.4 shows the quantitative evaluation of different digital computing architectures scaled to an example technology node. The programmable general purpose processors feature a low energy and area efficiency. Special purpose circuits, such as full-custom ASICs or standard cell-based design, show superior energy and area efficiency, but feature only limited amount of programmability and/or reconfigurability. In the shown example the difference in efficiency spans across multiple orders of magnitude. It is evident that ultra-low power requirements are most likely to be achieved by ASIC designs.

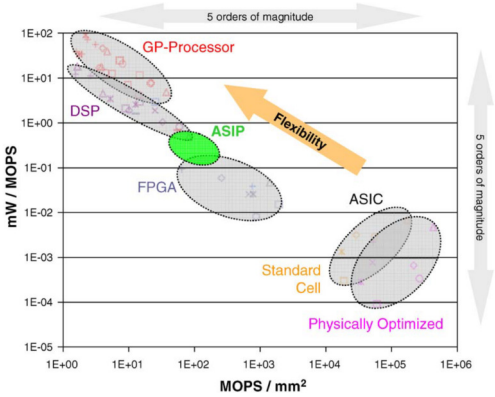


Figure 2.4: Energy and area efficiency for digital computing architectures scaled to 130-nm CMOS technology [67].

2.2.3 Cascaded Classification of ANN classifiers

From a system perspective, a monitored ECG does not contain an equally distributed amount of traces with different labels. In an optimal case, the system only records normal sine signals in a healthy patient. In a realistic setup, noise artifacts and uncritical heart conditions can influence the recording such that some segments need to be identified as such in irregular intervals. Further, the appearance of CVD is usually sporadic [1] and occupies a low percentage of the recorded data stream.

Contemporary systems utilize preliminary wake-up stages to reduce the overall system power consumption. The concept leverages skewed data distributions and complexity necessary to classify certain classes to concentrate the major computational workload on low complexity classifiers [68]. High complexity classifiers or eventually a human expert are “activated” in problematic cases, where the preliminary classification stage is expected to fail.

Preliminary investigations show that the CC, i.e. the number of operations necessary to compute the ANN model, can be adjusted over multiple orders of magnitude, while QoS, i.e. the quality of the classification, remains the close to the high complexity model [JL2]. A detailed analysis of pass-on-criteria is performed for off-the-shelf classifiers to investigate the impact on QoS and CC for configurations of 2- and 3-staged cascades. Figure 2.5 shows the concept of the cascaded classifier. All input samples first pass through an initial classifier of lowest complexity, i.e. lowest number of computation, to “filter” high confidence samples. When samples do not exceed a certain confidence threshold, the sample is passed on to the next classifier with higher complexity. The complexity of

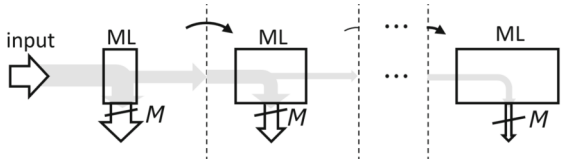


Figure 2.5: Structure of cascaded classifier for balanced datasets and MLs classifiers, here ANNs, with increasing CC [JL2].

the classifiers are chosen in such a way, that a classifier with higher CC also results in a higher QoS. Consequently, the last classifier is the best performing classifier with highest QoS, but also highest CC In the case of image classification, the quality metric is the prediction accuracy. Confidence values are calculated on the output vector of each classifier, which contains prediction probability p_i of each class i. The L1 norm of this vector should be guaranteed to be 1, such that probabilistic metrics can be applied on the vector. This is achieved by linear normalization or the softmax function. The confidence metrics, which are investigated within [JL2], are the absolute value of the prediction, "best versus second best", variance, kurtosis, entropy and Kullback-Leibler divergence. It is observed, that the first four confidence metrics perform similarly well on the MNIST dataset achieving a trade-off between accuracy and CC (normalized MAC operations). The range, in which the CC can be adjusted gracefully is five orders of magnitude for MNIST and two orders of magnitude for CIFAR10. With only small error tolerances of 1 %, a CC reduction of 263.17 $\times$ and 2.55 $\times$ is possible for MNIST and CIFAR10, respectively.

In the end, this investigation shows that a low complexity ANN classifiers are capable to achieve high QoS for a majority of samples. The split of the classifier into multiple stages proves especially useful, if the preliminary stage is used to effectively filter "easy" samples. In the case of an ECG monitoring system, the target is to design this preliminary stage well, such that only relevant or rare indecisive events are forwarded to reliable ANN models or trained experts.

2.2.4 System Scope of This Work

As seen in state-of-the-art solutions, the efficient monitoring of ECG signals can be tackled from a variety of perspectives. In the following, a use case is sketched to provide a context for the implemented classifier system. The use case provides proper requirements for the optimization of the system.

Figure 2.6 shows an application scenario, where the continuous monitoring of ECG data can be utilized. The target classification system is applied in a sensor node, where the ECG data is acquired

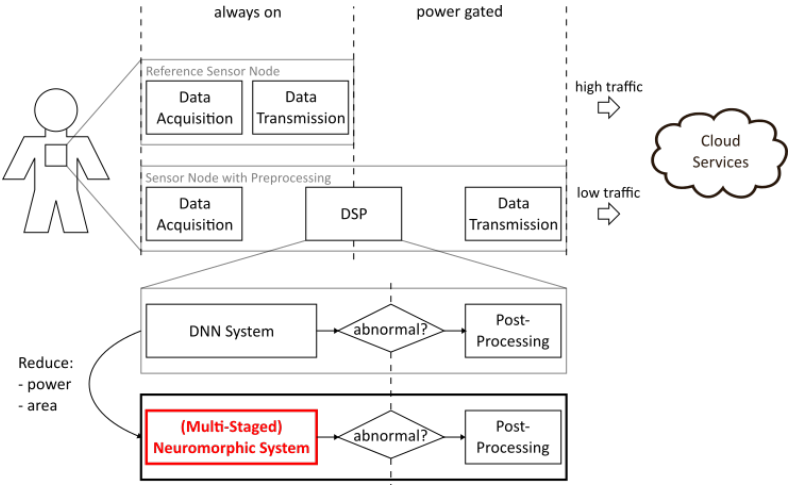


Figure 2.6: Hardware accelerator as gating mechanism for post-processing steps.

and processed for subsequent analysis. The top row of the diagram shows a typical sample-and-send setup. In this setup, the data is sampled on the mobile device and directly transferred to an expert for further processing or manual monitoring. Evidently, there is no digital processing on the device resulting in high data rates. The second scenario processes the raw data inside a digital processing module, such that compressed information about the monitored system, here a patient with a ECG trace, is sent to the expert. The compressed information is, for instance, classified labels or snippets of an ECG trace, which is found to be abnormal. This highly reduces the data transmitted resulting in less time spend by the expert. The challenge is to design an efficient pre-processing system to detect the abnormal classes with high quality and low power. A high quality classification guarantees that only relevant samples are selected for expert diagnosis and no critical samples are neglected. A low power operation is necessary for long monitoring durations as is necessary for critical sparse features as AF. Considering the energy budget of battery cells, which are available today, the target consumption for several weeks of operation is in the region microwatt for the whole system.

The scope of this work aims to design the digital signal processing module for this use case. In this use case, we assume a digital data stream as an input and a classified label as an output. An assumption is that the continuous input data stream is directly processed by the classification system in an always-on manner, such that the coverage of the whole stream is guaranteed. Possible post-processing steps can be activated based on the acquired label. A DNN classifier is designed

such that it can be computed on dedicated hardware components targeting the always-on processing of this data stream. Ultra-low power requirements for limited energy budgets are achievable in an dedicated ASIC design. For a long-term monitoring setup, an operation of multiple weeks with a battery charge is assumed as the target for the ECG classification system. For contemporary battery cells, a realistic target covering this time frame would result in average current consumptions in the region of microampere [69]. Therefore, an ASIC solution is targeted as a proof-of-concept. In the next chapter, the methodology is discussed how to systematically explore the design space and evaluate the performance of the target solution.

2.3 Design and Evaluation Strategy

The target is to design a system, in which the algorithm, i.e. the DNN classifier, is adjusted to its hardware and vice versa, such that multiple optimization objectives are met. In the following, we denote this as *algorithm-hardware co-design*. First, we evaluate how the state-of-the-art is addressing the co-optimization of DNNs. Then, we derive our own co-design strategy tailored to our target application.

2.3.1 State-of-the-Art of Energy Estimation of Batched ANN Workloads

Influential literature has targeted design schemes for the efficient processing of DNNs, as high classification quality in ML tasks requires high computational cost. Especially, the inference is performed on embedded devices and, hence, the main subject of previous works. It has been identified that the efficient processing of popular reference DNN architectures, such as AlexNet [6], relies on an efficient dataflow [70]. In specific, the energy required to access data in memory is up to 1-2 orders of magnitude more than the energy to process one MAC operation, i.e. the most common operation in the DNN. Hence, the orchestration from memory to processing unit plays an important role in the evaluation of the DNN workload.

Considering ANNs trained in a supervised fashion, labeled data is used to provide desired input-output pairs for the model and allow the backpropagation of error values during the training phase. Computer vision benchmarks serve as baselines for the research and development of ANNs and corresponding ASICs accelerators. For instance, MNIST sparked initial research on ANN models, e.g. CNNs [71], and continues to be used for proof-of-concept designs for more advanced neuro-inspired principles, e.g. memristive SNNs [72], even until today. Nevertheless, the increasing complexity of

more realistic larger scale datasets, i.e. ImageNet [73], forces algorithms scale in their complexity and triggered key innovations in ANN research.

Popular ANN models resulting from the ImageNet challenge are e.g. AlexNet [6] or ResNet [74]. The former spawned a variety of digital accelerators specifically targeting DNN, i.e. ANNs with deep layered architectures, which are described by a major body of survey works on ANN accelerators [70], [75]. A major bottleneck in larger ANN models is that the amount of parameters, i.e. weights and biases of the ANN, exceed typical on-chip storage/memory capacities. Therefore, off-chip memory, e.g. DRAM, needs to be accessed to buffer required chunks of model parameters for processing. The dataflow regarding the movement of parameters and activations is a major focus in DNN accelerator design. The access sequence is crucial for the consumed energy per inference, as each memory access to different levels of the memory hierarchies consume different energy. For instance, it is less desirable to access off-chip memory than local register files as the consumed energy is up to two orders of magnitude higher [70].

A systematic design space exploration typically uses a quantitative model for KPI evaluation. Analytical models provide a good initial measure to quantify the consumed energy for a large design space of accelerator architectures with different topologies [76]. Further they provide an indicator for the suitability of dataflow schemes for DNN specific workloads.

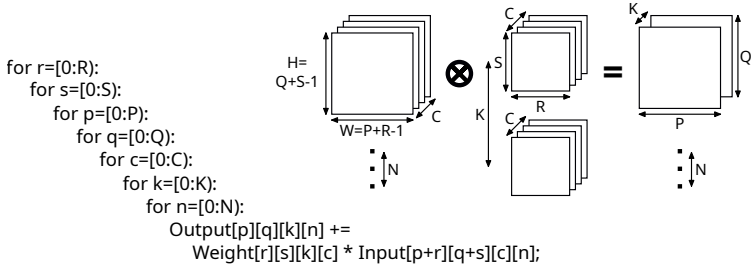


Figure 2.7: Loop nest representation of computational workload of a 2D-convolution operation for a batch of input feature maps [76].

Figure 2.7 shows a convolution operation represented in nested loops over the corresponding dimensions of two dimensional feature maps. In the case of batched image data, there are seven dimensions: three dimensions for the output feature maps, i.e. P, Q (spatial) and K (channel), two dimensions for the convolution kernel, i.e. S and R (spatial), one dimension for the input features maps, i.e. C (channel), and one dimension for the batch, i.e. N. The other dimensions, e.g. spatial dimensions of the input feature map, can be derived, respectively. Based on this representation, loop

nest optimizations are derived from predetermined PE array sizes and memory limitations. The loops can be tiled or unrolled spatially or temporally based on the hardware architecture. In the end the actual numerical dimensions of a specific DNN layer can be inserted, which estimates the number of cycles and memory accesses for the specified workload.

However, conventional DNN energy estimation methods typically reduce the loop nest optimization problem [76], [77]. For instance, the regularity of DNN workloads is used to abstract to calculation of memory accesses. Parashar et al. calculate the first, second and last iteration for the loop tiles for extrapolation [76]. Other works reduce the problem to values indicating the number of tiles within one loop, i.e. how many iterations are required to process one loop dimension with given parallel PEs, e.g. [77]. This is possible, since many DNN accelerator architectures operate in pre-defined patterns, e.g. execute convolution row by row from left to right. One example are two nested loops: The x_1 iterator of the inner loop increments until the end of the range. Then, the x_2 iterator of the outer loop increments, while the inner loop restarts from the start of the range. Above mentioned loop nest optimizations consider the loop ordering, tiling, unrolling etc., however, they do not allow for arbitrary sequences of (x_1, x_2). An example would be to traverse all combinations of the 2D-tuple (x_1, x_2) without a sequential order of x_1 or x_2.

2.3.2 Exploration of Dataflow Sequences for Convolution Operations

Previous works distinguish the dataflow based on which data in the convolution is held stationary inside the PE, which are denoted as input, weight and output stationary dataflow[1] [70], [78]. Typically, the number of data held in PEs is less than the number of elements in the processed data tensor. In the conventional dataflow models, the sequence is predetermined by the loop order in the loop nest representation (see Fig. 2.7). It is straightforward to process these loops in a row-by-row or column-by-column fashion. However, the design space of possible sequences is theoretically much larger. This raises the question whether another sequence could provide a more efficient alternative, since memory accesses contribute largely to the energy consumption [70].

From a different perspective, the problem can be viewed as a sequential traversal of elements in a tensor. Instead of iterating the tensor on a dimension-by-dimension basis, a curve can be defined, which passes through all elements in the tensor. In mathematics, the search for these *space-filling curves* date back more than a century [79], [80]. The idea is that these curves traverse the

[1] For simplicity we omit special cases such as row stationary dataflows in our investigation

multi-dimensional space, such that "locality" is preserved, i.e. points, which are close in the multi-dimensional grid, are also close in the traversal order [81]. However, the sequence in the dataflow models require different optimization constraints, which possibly results in solutions with different properties than already proposed space-filling curves.

Therefore in our exploration, we approach the search for the sequence from a new perspective. The concept is to define costs related to the execution sequence in a dataflow model. Based on this cost model, any sequence has a cost, which can be compared with each other and optimized globally. The required steps to achieve this are summarized as follows:

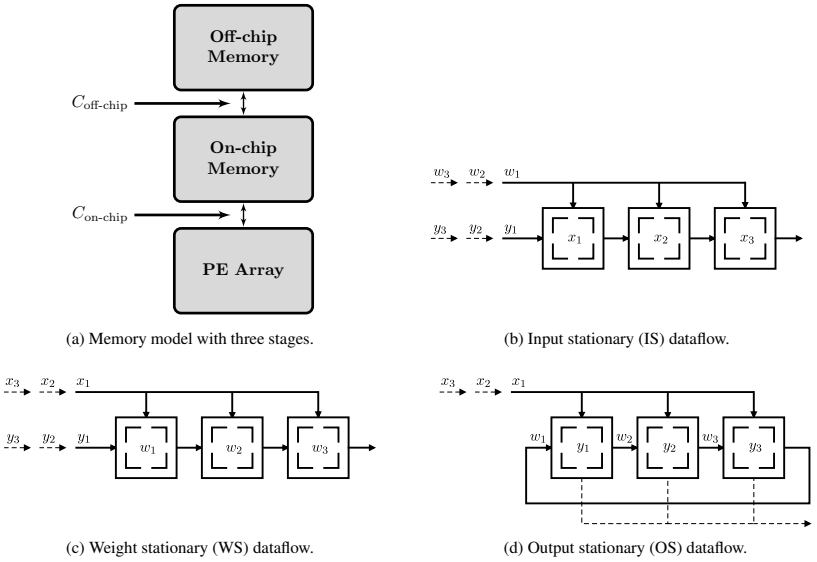
1. Memory model to estimate access patterns of an ANN workload
2. Index mapping of data samples in PE to required data in memory
3. Cost of a sequence of memory accesses during a convolution operation

First, a model needs to be established to define the memory access patterns resulting from different sequences. Then, the relationship between the data in the PE array and the corresponding data, which needs to be stored in the on-chip memory, is defined in a mapping function. This mapping function relates a set of indices from one dataspace to another. This is used in the last step to find overlaps between two consecutive states in a sequence. The cost of this sequence can then be calculated as the sum over all states in a sequence. In the following, these three steps are further presented in detail.

Memory model

As the memory access cost from off- and on-chip buffers deviate by one or two orders of magnitude [70], it is key to model the number of these two hierarchies separately. We investigate a memory hierarchies with three levels: Off-chip, on-chip memory and registers on the PE level (see Fig. 2.8a). In this model the on-chip memory is limited in size, while the off-chip memory is able to contain all data for a typical DNN inference.

The costs to access the on-chip and off-chip memory are denoted as $C_{\text{on-chip}}$ and $C_{\text{off-chip}}$, respectively. The access pattern of the on-chip and off-chip memory and, thereby, the number of accesses $N_{\text{on-chip}}$ and $N_{\text{off-chip}}$, as function of the dataflow scheme used by the DNN accelerator. The considered dataflow schemes are depicted in Fig. 2.8b, Fig. 2.8c and Fig. 2.8d. The dataflow is visualized for the example of the systolic array with three PEs [78]. One cell represents a PE and contains the stationary data. It calculates one MAC operation, i.e. multiplies two values and subsequently adds one value to



(a) Memory model with three stages.

(b) Input stationary (IS) dataflow.

(c) Weight stationary (WS) dataflow.

(d) Output stationary (OS) dataflow.

Figure 2.8: Simplified memory model to estimate data access costs for elementary dataflow schemes (adapted from [78]). A cell indicates the PE and the data held within it until all corresponding partial sums are calculated.

a partial sum resulting in a new partial sum. The required data within the on-chip memory is further exemplified for the weight stationary dataflow in the next step - the index mapping.

All in all, the memory model uses the number of memory accesses $N_{\text{on-chip}}$ and $N_{\text{off-chip}}$ to calculate the memory access cost

$$C_{\text{mem}} = N_{\text{on-chip}} \cdot C_{\text{on-chip}} + N_{\text{off-chip}} \cdot C_{\text{off-chip}} \tag{2.1}$$

per inference. The computational cost C_{comp} is also considered as

$$C_{\text{comp}} = N_{\text{mac}} \cdot C_{\text{mac}} \tag{2.2}$$

with N_{mac} being the number of the MAC operations in the DNN inference and C_{mac} being the cost to execute one operation. In the end, the total cost is the sum of both components resulting in

$$C_{\text{tot}} = C_{\text{comp}} + C_{\text{mem}}, \tag{2.3}$$

Index mapping

In our investigation, we consider a convolution layer and an input feature map with c_{in} input channels to produce output feature maps with c_{out} output channels (see Eq. (2.4)).

$$\mathbf{y}(c_{\text{out}}) = \sum_{c=0}^{c_{\text{in}}-1} \mathbf{W}(c_{\text{out}}, c) * \mathbf{x}(c) \tag{2.4}$$

Here, '$*$' denotes the convolution operation, $\mathbf{x}$ the input feature maps, $\mathbf{W}$ the filter kernel and $\mathbf{y}$ the output feature maps. Similar to Fig. 2.7 we assume three dimensional inputs feature maps and convolve it with four dimensional weights to result in three dimensional output feature maps. It is evident, that the computational cost C_{mac} would stay the same regardless of the order, in which input/output channels are computed. However, this is not the case for the memory accesses, due to opportunities for reuse in the on-chip memory.

Since the convolution is performed in a discrete domain, $\mathbf{x}$, $\mathbf{W}$ and $\mathbf{y}$ can be interpreted as scalar values mapped to integer lattice points in a multi-dimensional space, i.e. dataspace of operand and result tensors [76]. For the design space exploration of dataflow sequences, we want to find the relationship between the dataspace of the tensors, which remains stationary in the PE array, and the other two dataspaces. In the weight stationary dataflow, the objective is to find the location, i.e. spatial indices, of the necessary input and output feature maps given a set of weights. Once this mapping function is known, the overlap between two consecutive states can be calculated. In the following, we showcase this in the example of the weight stationary dataflow, however, the transfer to other dataflow schemes is straightforward.

Figure 2.9 visualizes the mapping of indices in a simplified diagram. The convolution is performed partially for a single weight inside the convolution kernel for one step in the sequence. The bold cross and circle indicate the spatial location of the weight in the kernel, for which all partial sums are generated. The non-bold crosses and circles show the spatial locations of the corresponding input and output feature maps, which need to be accessed to calculate the partial sums. It can be observed, that not all input and output feature maps are required for this specific weight. A different set of feature maps is required for two different weights, e.g. at the spatial location of the bold cross and circle. If the stationary weight inside the PE is exchanged, e.g. the bold circle replaces the bold cross, the overlapping samples in the input and output feature maps can be reused. However, the new non-overlapping samples need to be loaded into the memory and their number define the necessary memory accesses for further cost calculation.

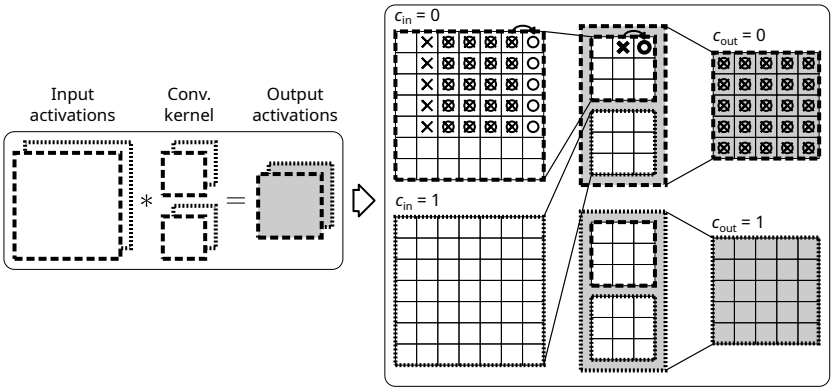


Figure 2.9: Diagram of a memory state for a weight stationary 2D-convolution with a filter kernel 3×3, 2 input/output channels, an output feature map size of 5×5 and a convolution stride 1. The crosses and circles mark data from two separate states. The weight marked in bold is held stationary inside the PE memory until all partial sums has been calculated for its input.

Sequence Cost

As indicated in Eq. (2.3), the total cost of a DNN inference is the sum of computational and memory access costs. Since the number of MAC operations do not change based on the execution order, the number of memory accesses define the cost difference (see Eq. (2.1)). In the dataflow schemes, as shown in Fig. 2.8, the set of data within the PE array is limited and all require multiple iterations to process all data points. In the weight stationary dataflow, this means that the weights are exchanged the PEs in a certain sequence, such that all weights are processed for the convolution operation.

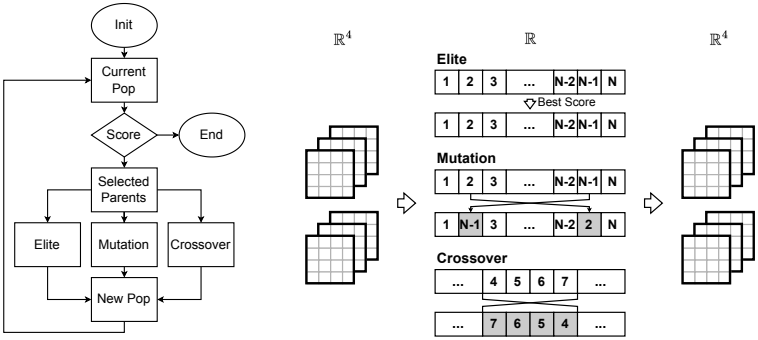
Hence, we define the sequence ξ_{ws} as the ordered set of weight indices. Each sequence maps to $N_{\text{on-chip}}(\xi_{\text{ws}})$ on-chip and $N_{\text{off-chip}}(\xi_{\text{ws}})$ off-chip memory accesses. These are calculated based on the overlapping indices from consecutive memory states as described previously and visualized in Fig. 2.9.

Meta-heuristical design space exploration

Based on above model, the cost of a sequence relative to each other can be determined quantitatively. The unique property of this model compared to state-of-the-art energy estimation models for DNNs [76] is that random sequences, especially those deviating from the loop nest representation, are incorporated and can be estimated in terms of DNN inference cost. To find a cost optimal sequence,

however, it is not feasible to apply a full search. The issue is that the search space of the sequence vector is huge, e.g. a weight kernel of size 3×3 with 32 input and output channels requires $(3 \cdot 3 \cdot 32 \cdot 32)!$ combinations.

Therefore, we employ a metaheuristic optimization method from the class of evolutionary algorithms, in specific the genetic algorithm [82]. The basic principle is to represent the solution, in our case ξ_{ws}, as a genetic representation, on which biologically inspired operations can be performed. Over multiple iterations, denoted as generations, a population of possible candidate solutions will converge towards a potentially globally optimal solution. Every candidate solution is evaluated based on a fitness function or its cost for optimization. The candidate solutions will be modified (subjected to crossover and mutation) based on its costs and selected for the next generation. In our case, the genetic representation of the sequence ξ_{ws} is straightforward, since the spatial index of the weight in the kernel can be mapped unambiguously to linear indices and backwards.



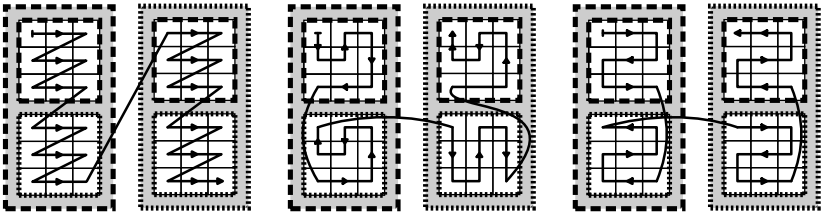
(a) Flow chart of the genetic algorithm.

(b) Detailed description of elite, mutation and crossover children for the next generation.

Figure 2.10: Concept of the genetic algorithm.

Figure 2.10a visualizes the flow chart of the employed genetic algorithm. First, a set of initial sequences is generated for the first generation of individuals. A scoring mechanism determines the fitness value of each individual in the generation. In our case, the fitness value is determined by the cost calculated in Eq. (2.1). Based on the fitness values, parent individuals are selected for the next generation. The individuals of the next generation can either be elite, mutated or crossover individuals. If the change of fitness values across generations does not exceed a threshold, the algorithm terminates and returns a set of potential result sequences. Figure 2.10b shows the details of the individuals of the next generation. Each sequence in the four dimensional space is mapped to a one

dimensional vector with unique scalar indices. The elite individuals comprise a predefined number of sequences in the previous generation with the best fitness value and remain unmodified in the next generation. The mutated individuals are generated from a subset of the previous generation by exchanging two random elements in the sequence vector. In the original genetic algorithm, the crossover step combines two individuals into a new vector. However, for the sequence to be a valid traversal through all elements in the tensor, it needs to be guaranteed that all indices in the vector appear exactly once. Therefore, the exchange of random sections in two individuals is likely to violate this constraint. As a workaround, our crossover step simply flips a random contiguous section, thereby introducing a large change in the individual. An alternative is to search for the same set of elements in two individuals for a crossover, however, this would heavily impact runtime performance of the genetic algorithm.



(a) Conventional execution sequence of weights.

(b) Optimized sequence using genetic algorithm.

(c) Manually optimized sequence with equivalent cost to b.

Figure 2.11: Execution sequence of weights in a weight stationary dataflow assuming 1 PE for processing.

Figure 2.11 depicts the execution sequence of the same example as chosen in Fig. 2.9. In the following we assume that the size of the on-chip memory corresponds to the necessary entries for input and output activations at one step in sequence ξ_{ws}. In the conventional loop order (as indicated in Fig. 2.7 and Fig. 2.11a), the weights are accessed row by row and then channel by channel for input and output, respectively. We use this as a baseline to compare against the following optimized sequence. For the optimization, we ran the genetic algorithm with a population size of 2000 and a maximum of 500 generations. When the cost does not increase over 50 generations, the algorithm is stopped early. The result is shown in Fig. 2.11b, in which the cost is reduced by 23.8 %. Figure 2.12 shows the convergence against the cost optimal solution. The spread in the fitness value is large in the initial population and throughout all generations. However, the elite individuals quickly converge beyond the cost of the conventional loop order, which is used as the reference sequence. The score distribution of individuals resembles a shifted gamma distribution over the course of generations.

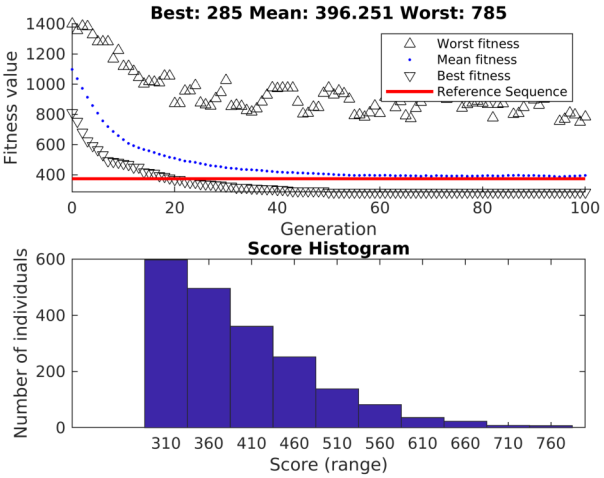


Figure 2.12: Best and mean fitness values during generations of genetic algorithm optimization (above). Fitness value distribution of final generation (below).

The exploration process generated a snake pattern in the 4-dimensional space of the weight kernel. Specifically, the next weight chosen for processing in the PE is always chosen with a Manhattan distance equal to 1, i.e. spatially the next weight is adjacent to the previous weight. In the conventional loop orders, however, there are cases, i.e. from the end to the beginning of a inner loop iteration, in which this locality is not guaranteed. In those transitions, we observe additional costs for memory accesses. Based on this exploration, we could manually construct a sequence with a regular pattern and, even, the same cost as the optimized sequence (see Fig. 2.11c).

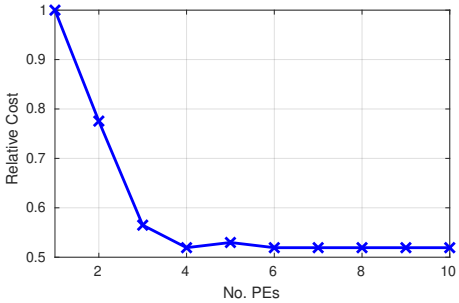


Figure 2.13: Sweep across number of parallel PEs for cost optimal sequences in a weight stationary dataflow using genetic algorithm optimization.

Figure 2.13 shows the relative costs of parallel PEs compared to a single PE. For small number of PEs, the cost is decreasing linearly with increasing PE numbers. However, the cost reduction is saturating at 45 % for the example convolution in Fig. 2.9. In this example, the ideal case minimizes the off-chip accesses, i.e. only one read. This motivates to scope the on-chip memory, such that all activations and parameters can be stored on-chip. Note that the costs are based on the simplified memory model in Fig. 2.8a and does not consider control overhead, multiple memory hierarchies etc.

Implications for the ECG Monitoring Use Case

Based on the sweep performed in Fig. 2.13, the relative cost can be significantly reduced by a few parallel PEs. If the number of PEs correspond to the number of elements in the row of a convolution the cost reduction already comes close to the achievable maximum. This insight aligns with findings of previous literature, where a row stationary dataflow is proposed for the execution of 2D-convolution workloads.

The ECG monitoring use case only requires 1D-convolutions and state-of-the-art ECG classifiers need much less parameters than computer vision models (see Section 2.2.2). In most cases, the whole DNN model is able to fit in the on-chip memory (in the region of kilo-/megabytes), such that no off-chip memory accesses are necessary. This aligns with the result of previous cost exploration, such that a major energy bottleneck common in image processing DNNs can already be eliminated. Further, the dataflow sequence exploration shows that it is beneficial to process spatially adjacent data chunks preferably in rows. It is important to recall that this investigation focuses on batched data processing. As the corresponding DNN models assume that input data is readily available for processing, it needs to be reconfirmed, whether the found principles remain valid for the stream processing use case. This is further investigated in the example design discussed in Section 3.

2.3.3 Design Space Exploration Methodology for Stream Processing Architectures

As seen in the previous section, the application of ECG monitoring implies certain constraints on the designed hardware. The example of batch processing is prevalent in the design of DNN accelerators [70]. However, the real-time monitoring of ECG signals does not provide data in batches. Instead, the data is available as a continuous stream of samples. For ECG data, the number of leads in the monitoring setup determine the number of parallel channels, which are available for analysis. The

key difference is that the samples are only provided one at a time and a sample sequence is used for the analysis.

Considering a batched processing architecture, an example processing flow would be as follows: The input stream is recorded into an internal memory and segmented into batches of ECG traces. Once the data is accumulated, the hardware unit is activated and used to classify the traces. Not only does this setup require a separate memory to record and store the raw data, but it also delays the output of the classifier until the recordings of multiple segments are finished.

However, in a streaming architecture, the input stream is directly processed on-the-fly, e.g. in digital filters. Since past samples do not need to be stored on the device, the input storage can be omitted to reduce the number of components in the design. Further, the processed result is directly available at the output. The latency of the classifier prediction is, therefore, not limited by the recording of the batch. In the following, the stream processing architectures are considered for the chosen ECG monitoring use case.

State-of-the-Art Stream Processing Architectures for ECG Data

In the case of ECG, leads are recorded based on a standardized setup [83]. Each lead is treated separately as a distinct input channel without any spatial relation in the data, for instance in neighboring pixels of an image. The samples in the input stream are ordered through their temporal relationship within the sequence. Figure 2.14 shows a general sketch of specifications to process a continuous data stream and a categorization of state-of-the-art accelerators addressing the stream processing architectures. Basically, the processing schemes can be subdivided into three subcategories.

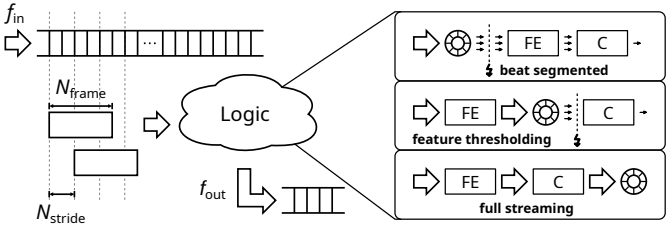


Figure 2.14: Design specifications for the continuous ECG monitoring use case [JL7]. The input is sampled continuously and buffered intermediately for processing, where frames are selected for further analysis. Data is either processed using raw frames or extracted features or in a full streaming fashion.

In the first, the raw data is buffered in frames, e.g. heart beats or sequences, such that these are further processed when the buffer is full. State-of-the-art focuses on the samples aligned around the

R-peak of a beat [53], [54], [56]. In this case, a R-peak detection mechanism, e.g. Pan-Tompkins Algorithms [47], is necessary to detect the center of the sequence to be recorded. Therefore, initial processing stage only comprises few components, such as FIR filters, some alignment/normalization and thresholding units. These trigger the ANN in a subsequent step, which either work as feature extractors [53], [54] or directly classify the buffered data [56].

In the second category, the processing of buffered frames is not triggered by heart beats, but based on lightweight features indicating interesting events. These methods aim to reduce the number of activations of the post-processing step. One example for lightweight features is the instant rate of change [59]. This feature is designed for event-driven ADCs, such as level-crossing ADCs. An alternative approach, which also works on uniformly sampled ADC input, are features like signal line length, area, decay and zero-crossings [58]. When those values exceed a fixed threshold, it triggers the classification of those features or the pre-buffered raw data. In this case, the components, which are executed for every input sample, are more complex than the simple buffering.

In the last category, the accelerators operate in a full streaming fashion [JL3], [35], [57]. Here, the utilized ANN is executed partially for each input sample. It is apparent in the used ANNs that subsampling is a key component to reduce number of operations performed, while keeping a large frame, on which the processing steps are performed on. For instance, DWT is useful to both extract relevant features in the ECG signal and subsampling the signal by a factor of 2^d, in which d denotes the depth of the DWT [JL3], [57]. Alternatively, trained filter banks are also suited to reduce the data rate of the input data stream [35].

Data Stream Coverage and Impact on Power Consumption

As discussed, state-of-the-art ECG stream processing architectures differ from conventional DNN accelerators, which are primarily designed for batch processing [70], [75]. In a continuous data stream, a frame is a subsection of N_{frame} samples within the data stream (see Fig. 2.14). The time interval at which data points are sampled from the stream, is determined by the sampling frequency f_{in} and, consequently, the time to acquire a data frame is fixed to $N_{\text{frame}}/f_{\text{in}}$. Similarly, the time period between two consecutive frames are fixed, as new samples are acquired from the data stream. The delay is determined by the number of temporal samples in between two consecutive frames, i.e. N_{stride}.

State-of-the-art ECG accelerators use different specifications, since they are heavily dependent on the application requirements (see Section 2.2.2). To analyze the impact of these specifications on the KPI of the hardware implementation, a normalized setup can be used to relate the data stream to a batched processing setup. The basic concept is to divide the data stream into non-overlapping frames, such that every input sample is processed exactly once analogous to batched inference. In this case, the stride is equal to the frame size, i.e. $N_{\text{stride,norm}} = N_{\text{frame}}$, resulting in an output frequency $f_{\text{out,norm}} = f_{\text{in}}/N_{\text{frame}}$. Both stride and output frequency are directly determined by the input sampling rate and the frame size used by the post-processing ANN. A factor $\alpha = \frac{N_{\text{stride,norm}}}{N_{\text{stride}}}$ or $\alpha = \frac{f_{\text{out}}}{f_{\text{out,norm}}}$ is introduced to relate the normalized setup to arbitrary stream processing setups. In principle, α determines the overlap between adjacent frames. For $\alpha > 1$, input samples are used multiple times for classification. For $\alpha < 1$, there are gaps in between adjacent frames and input samples are neglected for classification.

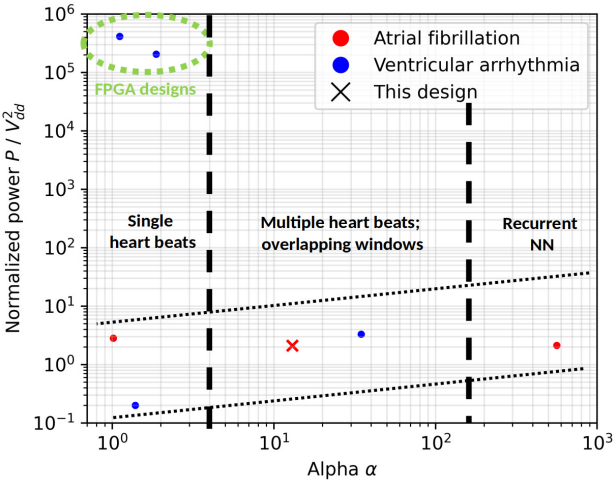


Figure 2.15: Normalized system power in relation to normalization factor α derived from [JL3], [35], [44], [53], [57], [62], [84][2].

Figure 2.15 shows the normalized system power of various ANN accelerators over α, i.e. the coverage of the input stream. While recurrent neural networks, such as GRUs [35], are able to represent the whole temporal sequence in their internal state, other ANN architectures operate on a limited set of input, i.e. single [62], [84] or multiple [JL3], [53] heart beats. The power consumption is lowest

[2]When no data is reported for the output frequency, a normal resting heart frequency of 1.25 Hz is assumed.

for sparse frames and low reuse. In previous works, there is a tendency that classification systems with larger input frames are also designed with a larger percentage of overlap, i.e. greater alpha, to process the data stream.

Note that most designs reuse input data in some form, i.e. $\alpha > 1$, through sliding windows on the input stream or reuse of adjacent RR-peak distances. Hence, features are considered temporally in ECG classification, which require overlapping input frames. This overlaps with the requirement of long analysis windows required for the detection of faint AF features set by expert societies [1]. Hence, the targeted design in Section 3 will adopt overlapping frames for continuous ECG processing.

Top-Down Design Methodology for Algorithm-Hardware Co-Design for Stream Processing

The previous insights are incorporated in a holistic design flow for systematic design space exploration. A top-down design methodology is suitable to address the research question, as requirements are determined by the target application.

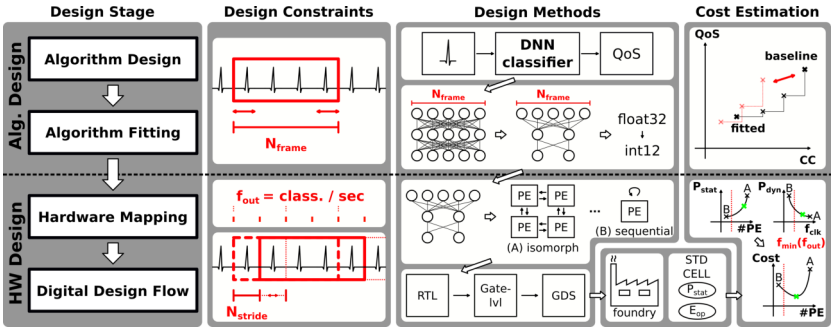


Figure 2.16: Top-down design methodology proposed in [JL7]. The design constraints, specific to streaming ECG data, are used to guide both the algorithm and hardware design. Costs, which map to power and QoS, are continuously assessed to guarantee the convergence towards a cost-optimal design.

Figure 2.16 shows a diagram of the methodology split into the design stages: *Algorithm Design*, *Algorithm Fitting*, *Hardware Mapping* and *Digital Design Flow*. In the *Algorithm Design* stage, a DNN classifier is designed to meet the application requirements, i.e. is able to classify ECG data with a certain quality. Typically, this process involves the filtering of possible solution candidates in terms of DNN architecture. Modern ML frameworks [25], [26] feature a broad selection of layer types, e.g. ranging from typical convolution layers to recurrent or even transformer layers. As the number of layer combinations is theoretically endless, a systematic selection is necessary to narrow down

the solution space. Further the hyperparameters for training also play an important role for model convergence and how well the model fits to the given data.

The *Algorithm Fitting* stage deals with the reduction of a high-performing DNN classifier from the previous design stage, i.e. an optimization of the DNN classifier in terms of complexity. Since the design of a DNN primarily aims to increase the QoS of the target application, the resulting model tends to feature high computational complexity (CC), i.e. high number of MAC operations in deep network architectures. However, a significant reduction of CC can be achieved with minimal impact on QoS through methods like pruning [85] or quantization [86]. For ultra-low power ECG classification a trade-off between CC and QoS needs to be found.

Once the DNN is determined with sufficient QoS, the corresponding accelerator is designed in the *Hardware Mapping* stage. In this stage, the arithmetic operations are conceptually mapped onto processing elements (PEs). This involves the translation of the simple dependence of the operations, e.g. defined in a dependence graph, into a signal flow graph with information about the control and dataflow. In contrast to a typical von-Neumann architecture, the design of application-specific integrated circuits allows the implementation and selection of specialized circuit components [87]. Not only does this reduce unnecessary components within the hardware design, but also more control on the dataflow and degree of parallelism.

In the end, a high-level design description, i.e. a register-transfer level (RTL) description, is used in the *Digital Design Flow* to realize an integrated circuit, which can be simulated in various granularity and fabricated in a semiconductor foundry. In a standard cell-based design flow, the total system performance and KPIs can be modeled as the conjunction of all individual components of the system, e.g. transistor/gate-level timing/power models, wire load etc., which are available in technology specific libraries. In addition to their obvious use in electronic design automation (EDA) tools, these libraries can be used to drive early design decisions in the hardware design phase.

As indicated in Fig. 2.16, each stage evaluates the design quality with cost metrics of different granularity. While in the algorithm design phase the CC is determined by the number of MAC operations in the algorithm, in the hardware design phase the actual power consumption can be determined. The detailed breakdown of component contribution as well as the split into static and dynamic power consumption allows a fine-grained analysis of the design. Even though the diagram shows a sequential order in the design stages, the co-design process also includes the re-iterations of early design stages based-on insights in later ones, e.g. hardware mapping adjustments based on circuit simulations.

2.4 Summary

The processing of low-data rate streams requires a systematic process to enable efficient real-time processing. Before the design and optimization process the requirements from the application need to be considered in a structured design methodology. In a first step, ECG monitoring is chosen as an example use case for exploration. The classification of AF poses a practical example for long-term monitoring in a mobile device. The detection of AF necessitates 24/7 monitoring systems, as indicative features are sparse. Further, the integration into practical devices requires the efficient processing in a small form factor device. In a second step, the context of the design is defined. The focus of this work is to design a digital signal processing module to process an ECG input stream into prediction values, which are used to identify critical segments to alert medical personnel or trigger further analysis. The baseline classifier used for exploration is an ANN, as it shows promising results both in terms of high quality classification and low power devices. Finally, a design space exploration methodology is chosen to co-design algorithm and hardware based on the defined scope. As the target application is clearly defined, a top-down design methodology is used to guide the design process based on the application specifications.

Chapter 3

Co-Optimization of an ANN Accelerator for ECG Monitoring

This section demonstrates the systematic design space exploration to design an efficient ECG classifier for AF, as outlined in the problem definition in Section 2.1 and 2.2. Following the described exploration methodology (see Section 2.3), an ultra-low power and high-quality ANNs classification system is designed. All design considerations through all abstraction levels are detailed in the following subsections. Note that all general principles are generally applicable to low-rate data streams with adjustments in the application constraints.

3.1 Algorithm Design

The chosen benchmark to evaluate the algorithm quality is the CinC'17, which targets the classification of atrial fibrillation among three other classes, i.e. normal sinus rhythm, other heart rhythms and noise. The data consists of single lead ECG recordings of varying length mainly between 30 and 60 seconds collected from an AliveCor device. The recordings are sampled with a frequency of 300 Hz with a 16-bit resolution and a dynamical range of ± 5 mV. The data is split into 8528 recordings in a open training set, on which the model design is performed, and a hidden test set of 3658 recordings. The complete dataset was made publicly available after the challenge on the PhysioBank platform [20].

This benchmark is chosen for the following reasons:

- The dataset contains a large corpus of data covering a large variety of patients. The patient variety allows for inter-patient generalization, as sufficient data is available for training.

- The recordings are captured in a commercially available recording device. The recording does not follow clinical procedures and captures noise and realistic artifacts from patient usage, e.g. lead inversions etc.
- Single lead measurements correspond to various setups in a mobile setting, such as ECG patches, watches etc. [88]

Note that this field of research is rapidly evolving and datasets with better properties, i.e. more labeled data, have emerged over time, e.g. the Incentia11k dataset [17].

Further, the CinC'17 data is accompanied with a range of ML solutions submitted to the challenge. The top performing algorithms include LSTMs [89], [90], Cascaded Binary Classifiers [91], Random Forest Classifiers [92] and Ensemble Classifiers [93]. Although the classification performance of all classifiers are impressive for the given benchmark, the choice of algorithm for the ultra-low power application needs to consider the scalability of the algorithm. Even though classical ML algorithms perform very well on a large or curated set of expert features, the features have their own individual complexity and, correspondingly, a diverse set of numeric operations, which need to be supported by processing elements. For instance, the calculation of entropy or kurtosis as used by Hong et al. [93] requires more computationally complex mathematical operations, i.e. logarithm or exponential functions, than basic addition or multiplication.

For our exploration in [JL1], we chose the CNN baseline by Zihlmann et al. [90] as an inspiration to design a classifier from scratch. In contrast to previous works, the goal is to find an ECG classifier, which is scalable in terms of both QoS and CC. The idea is that the majority of operations in the classifier feature the same mathematical component. Analogous to the processing/algorithm/data-flow proposed in [90], a two-staged processing architecture consisting of a feature extraction component and a classification component is used in this work. Figure 3.1 depicts the targeted processing architecture.

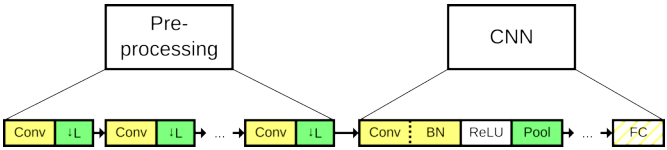


Figure 3.1: ECG classifier structure adapted from [JL3]. The yellow blocks are performing convolution operations and the green blocks perform subsampling operations.

3.1.1 Discrete Wavelet Transform for Feature Extraction

In contrast to the reference architecture, the ECG features are extracted with a discrete wavelet transform (analogous to [94]). This pre-processing component captures both time and frequency information from the signal. Further, the Daubechies wavelets *db2* are chosen based on its similarity to the morphology of the ECG beat.

The computation of DWT coefficients are realized by 4-tap finite impulse response (FIR) filters with fixed filter coefficients and subsampling components in multiple stages. Consequently, the rate of DWT coefficients is halved at each stage after it is convolved with the filter kernel. The mathematical components are limited to MAC operations resulting from the convolution. Figure 3.2 shows the block diagram of this DWT pre-processing stage.

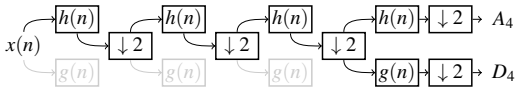


Figure 3.2: Block diagram of wavelet transformation with level 4 decomposition adapted from [JL1]. Lowpass filters $h(n)$ are used in the early levels to generate approximation coefficients A_x from input $x(n)$ for deeper levels. The highpass filters $g(n)$ are only used in the final level to generate the detail coefficients D_x.

When all coefficients are used for further analysis, the discrepancy between the output rates of the DWT coefficients needs to be compensated. Assuming an input rate f_{in} for $x(n)$, D_1 would have an output rate of $f_{\text{in}}/2$ and D_2 would have an output rate of $f_{\text{in}}/4$ etc. The intermediate approximation coefficients $A_1 \dots A_3$ are directly used for the next level. In contrast, we omit $D_1 \dots D_3$ for further analysis in the classifier. Hence, the amount of input for subsequent classification is reduced. Figure 3.3 shows the spectrum of white noise as an example input signal and its reconstruction from the A4 and D4 coefficients after the DWT. The reconstructed frequency response shows that low frequency components, i.e. up to 40 Hz of the signal, are preserved, while high frequency components are dampened. In the time domain, the reconstructed signal from A4 and D4 coefficients has a root mean square error (RMSE) of less than 50 µV, which indicates that they preserve a sufficient quality for further analysis.

3.1.2 Subsampling-based Classification of Temporal Sequences using TCNs

The low-rate features from the DWT are further used in a subsequent CNN for classification. As discussed in Section 2.3.3, the input data is available in form of a temporal sequence. In previous

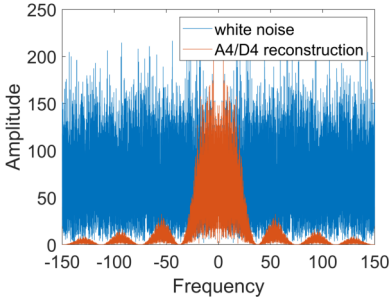


Figure 3.3: Frequency spectrum of white noise and reconstruction using only A4/D4 coefficients after DWT.

works, a neural network operating on this temporal sequence is defined both in general terms as a CNN or as a temporal convolutional network [30], [95]. The former stresses the temporal nature of the input compared to a static data frame, as the convolutions in this architecture remain causal. This guarantees that all output samples are computed from input samples of previous or current time steps [95]. The importance of causality is evident in deployment, as data, which is not captured yet, cannot be used for classification or, even, used to compute intermediate results for the classification. Further, the family of TCN architectures uses a hierarchy of temporal convolutions and pooling or dilations to capture long-range patterns, i.e. increasing the size of the analyzed time window [30]. This proves especially useful as it enables a large effective window necessary to detect AF [1] and utilizes hierarchical subsampling of the data stream for efficient data-driven computation. Both causality of the temporal convolutions and the hierarchical subsampling of the input data stream is exploited in the hardware mapping stage and further described in Section 3.2.1. In the following, we further use temporal convolutional network to denote the used neural network for classification.

Considering the processing blocks of both the DWT pre-processing and the TCN classifier, the processing blocks in the TCN share the same characteristics in each layer: a convolution and a subsampling component. Similar to the DWT block, the TCN intends to convolve the input with internal weights and further reduce the data rate in each layer.

During training, batch normalization is intended to reduce internal covariate shift [96], but it also demonstrates faster convergence and better generalization [97]. However, during inference they introduce additional layers inside the TCN architecture. Therefore, an effective method is to merge adjacent linear transforms, i.e. convolution and batch normalization, to eliminate computations without changing the output.

Consider a matrix multiplication

$$\mathbf{y}_{\text{nn}} = \mathbf{W} \cdot \mathbf{x} + \mathbf{b} \tag{3.1}$$

and a subsequent normalization

$$\mathbf{y}_{\text{bn}} = \frac{\mathbf{y}_{\text{nn}} - \mu}{\sqrt{\sigma}} \cdot \gamma + \beta. \tag{3.2}$$

Inserting Eq. (3.1) in Eq. (3.2) , we derive new weight $\hat{\mathbf{W}}$ and bias $\hat{\mathbf{b}}$ tensors resulting in

$$\hat{\mathbf{y}_{\text{nn}}} = \hat{\mathbf{W}} \cdot \mathbf{x} + \hat{\mathbf{b}} \tag{3.3}$$

with

$$\hat{\mathbf{W}} = \frac{\mathbf{W}}{\sqrt{\sigma}} \cdot \gamma \tag{3.4}$$

and

$$\hat{\mathbf{b}} = \frac{\mathbf{b} - \mu}{\sqrt{\sigma}} \cdot \gamma + \beta. \tag{3.5}$$

The extension of this proof to the convolution operation is straightforward. Note that this assumes that the parameters of the batch normalization layer, especially the mean μ and variance σ is not required to change in inference after TCN training.

The TCN, as proposed in [JL1], [JL3], uses max-pooling operations for subsampling. In addition to a mere rate reduction, the max-pooling operation selects the maximum value of its window as an output. The maximum function can be realized in an iterative manner (see Section 3.2.1). Nevertheless, the rate reduction of the data stream can also be performed by dilated convolutions [95] or by increasing the convolution stride. Their impact on the QoS is not evaluated within the scope of this work.

The final dense layer, or also fully-connected layer, is performed to map the output feature maps from previous layers to a prediction. When performed over a data stream, a prediction can be calculated for every new sample, which is available in the input, similar to the convolution. However, the kernel size of the dense layer determines how many temporal features are used for a prediction and, therefore, the effective time window, on which a prediction is performed on. In [JL1], [JL3], it is selected such that it matches a 1 second ECG sequence.

Despite the TCN structure, the hyperparameters for training are essential for the model convergence. Especially, the chosen TCN is rather shallow compared to state-of-the-art DNNs [3], [90]. In our experiments, we use the PyTorch framework to implement and train the models in a full sweep

across learning rate and batch size. For all training runs, we performed stratified 5-fold cross validation with multiple iterations of a random seed to generate robust results. We observed, that learning rate and batch size show an average local minimum with 0.005 and 128, respectively. However, they also show large variations across folds and iterations, which indicate the necessity for fine-tuning on other datasets. In addition, a stepwise reduction of the learning rate by 10 % after every 30 epoch showed significant performance improvement in achieved QoS.

Another method to increase the classification performance of the TCN without changing its structure is data augmentation. In this use case the following data augmentation techniques are additive noise and offset, beat rate variation, dropout bursts and lead inversion. All techniques imitate common variations in the ECG signal (further discussed in Section 4). Nevertheless, only the additive offset and the lead inversion showed an improvement in the QoS, i.e. $\Delta F1_{\text{offset}} \approx 0.7\%$ and $\Delta F1_{\text{inv}} \approx 2.1\%$, respectively. Both in combination, achieved an improvement of $\Delta F1_{\text{offset+inv}} \approx 2.5\%$. Similar to the hyperparameter exploration, the variations are large across training runs with different samples in the training and validation folds. Therefore, the efficacy of data augmentation techniques need to be evaluated on a case-by-case basis.

Further details on the exploration of training hyperparameters and data augmentation can be found in [98]. In the end, the TCN model show a validation F1 score of up to 79 % [JL3], which constitutes only a reduction of approx. 4 % compared to best-in-class classifiers [18]. The model structure is summarized in Table 3.1.

Table 3.1: TCN structure from [JL3]

Layer[1]	Kernel Size	# Output Channels	Subsampl. Factor
DWT L1	4	1	2
DWT L2	4	1	2
DWT L3	4	1	2
DWT L4	4	2	2
TCN L1	5	10	3
TCN L2	5	13	3
TCN L3	5	20	3
TCN L4	5	63	3
TCN FC	11	4	-

[1] Each layer Lx consists of one convolution (incl. rectified linear unit for TCN layers) and one subsampling block.

3.1.3 Algorithm Fitting

The exploration in the previous stage established a baseline with high QoS for further optimization. The fitting stage uses this baseline and reduces its complexity with minor compromises in QoS.

One method is the number representation and quantization of the TCN model. More importantly, the TCN model is trained based on a floating-point number representation. It is more commonly supported in the ML frameworks, which target CPU/GPU architectures. Even though arithmetic computations with floating-point numbers can represent a huge number range with fine precision, a floating-point operation consumes more energy than a fixed-point operation with equivalent word length [99]. Modern ML frameworks are capable of supporting down to 8 bit fixed point numbers and operations, since int8 instructions are commonly supported in the target processor architectures. However, highly efficient digital signal processing (DSP) blocks, which are established since multiple decades [100], can be designed with arbitrary word lengths in specialized fixed-point arithmetic units. Post-training quantization promises a systematic word length reduction of both weight and activations without the need for re-training with the full preservation of classifier QoS [101].

Analogous to previous explorations a sweep was performed in [98] to determine a possible sweet spot for the proposed TCN model. Surprisingly, the configuration with 12 bit word length, in specific 4 integer bits and 8 fractional bits provided a local optimum with a relative QoS degradation of 0.15 %. In contrast to the image classification tasks, the quantization down to 6 or 8 bit did not yield the expected result, i.e. more than -30 % F1 score reduction or even no model convergence at all. Hence, we conclude that quantization is significantly dependent on the sensitivity of data features towards the target classes. The quantization of weights also has the effect of fully removing the impact of filter kernels of specific channels, since all weights smaller than the least significant bit (LSB) are rounded to zero. In specific cases, whole filter kernels result in zero, e.g. up to 31.76 % [JL1]. In these models, the corresponding channels can be pruned without impact on the final result.

3.2 Hardware Design

In the following, dedicated hardware components are designed for the fitted TCN. Even though the variety of arithmetic operations seems trivial, i.e. only MAC operations and subsampling, the target is to execute the necessary computations in the most efficient way. As the continuous monitoring application requires the classification component to operate in a always-on fashion, the average power

consumption is key to quantify efficiency. Especially, both dynamic and static power needs to be accounted for in the design process.

3.2.1 Hardware Mapping

As mentioned earlier, the hardware mapping stage maps the arithmetic operations onto processing elements with a certain control and dataflow. It includes the storage of data as well as the movement of data in between storage elements and processing elements. The design space is huge regarding the possible mapping variants, e.g. dataflow strategies [70], mixed precision PEs with bit-serial/-parallel multiplication [102] etc.

Therefore, a systematic approach is performed, in which the activation memory and the PE mapping is explored separately based on high-level cost functions. In contrast to Section 2.3.2, the cost functions aim to relate to the power consumption of the target system instead of the energy consumed per inference. The key difference is that the components are quantified by both their active and idle power consumption, which equally contribute to the overall costs of an always-on system. In the following, we assume that the inference of the TCN exhibit a trade-off between active and leakage currents, since it is continuously switching between active processing of input samples and an idle period in between. The cost estimation in this work relies on the detailed characterization of fitted TCN model, i.e. in terms of number of operations, number of intermediate activations in each layer and per channel etc., and the used technology components, i.e. characterized standard cell components.

TCN Memory Components

Early cost estimation for the memory components in a TCN is demonstrated in [JL3], [JL7]. There are two types of data, which require buffering or permanent storage in the TCN: weights and intermediate activations.

The former constitutes the trainable parameters of the TCN model and are usually stored in high density memory components such as static random-access memory (SRAM) [54], [103]. In models with a high number of parameters, these components are the preferred choice due to area limitations. For state-of-the-art DNN models, it is even not possible to store all parameters on-chip. Here, this memory component is used as an intermediate buffer, while an external memory component stores all model parameters (see Section 2.3.2). However, the TCN capable of high-quality AF classification

requires a multiple orders of magnitude less parameters and, therefore, can be stored completely on-chip. Consequently, no off-chip accesses are required and, thus, eliminating expensive data access costs [70]. If no reconfiguration of the model parameters are required, e.g. fixed coefficients in the DWT, fixed coefficient multipliers can be directly synthesized. Applied to a full-flat TCN accelerator a power reduction of 34.8 % has been demonstrated [JL1].

The activation memory stores the intermediate activations for processing in the PE array. In the case of temporal sequences, the activations are streamed sample-by-sample from the input of each layer into the buffer. As convolutions are performed, the kernel slides across the stream of input activations. Hence, the buffer only needs to store the activations within a kernel in a first-in first-out (FIFO) order. For the processing of convolution kernels, the number of items in the buffer remain fixed. Therefore, every new inserted entry results in the oldest entry being removed from the buffer. Further, all data within the buffer needs to be accessible for the convolution operation, which needs to be supported by the memory structure. In this regard, the buffer needs to select the appropriate memory device and logic level implementation for a low cost solution.

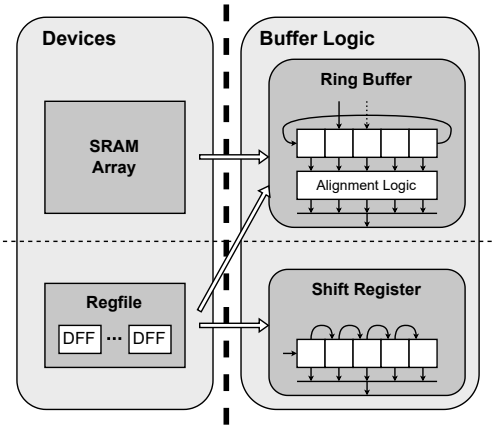


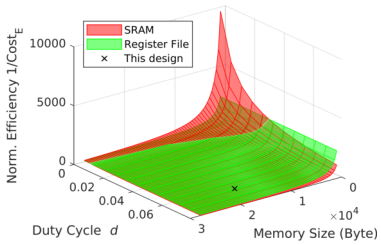
Figure 3.4: Memory devices and logic level implementation of activation buffers [JL3].

Figure 3.4 visualizes that the memory can be designed using SRAM devices or registers. However, conventional SRAM macros do not contain the flexibility to implement internal shift operations and, therefore, need to be implemented as ring buffers. In contrast, D-flip-flop-based registers can realize both ring buffers and shift registers, since individual bit cells can be arranged and connected on RTL. For convolution operations, the logic level implementations needs to implement SIPO buffers

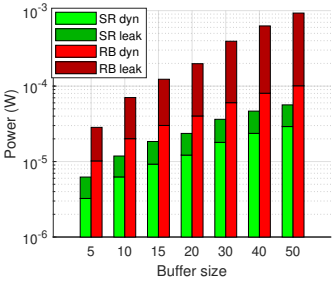
[84]. For shift registers, all register values are concatenated as an output. However, a ring buffer needs to realize an additional alignment logic, e.g. a barrel shifter, for the correct order of the data within the convolution kernel. To find the cost optimal solution for the context of TCN processing, a cost function is defined as follows:

$$\mathrm{Cost_E} = \sum_i \kappa_{\mathrm{op},i} \cdot E_{\mathrm{op},i} + \kappa_{\mathrm{leak},i} \cdot T_{\mathrm{cyc}} \cdot P_{\mathrm{leak},i} \tag{3.6}$$

The concept of Eq. 3.6 is to aggregate all influences of the memory components, which contributes to the overall cost of the activation memory. The total cost $\mathrm{Cost_E}$ is split into contributions from dynamic and static power. The contribution to dynamic power is estimated based on the energy per operation $E_{\mathrm{op},i}$ and the number of operations $\kappa_{\mathrm{op},i}$. The static power is estimated by the number of components $\kappa_{\mathrm{op},i}$ and their corresponding leakage $P_{\mathrm{leak},i}$ normalized by a predefined time period, e.g. T_{cyc}. Note that the figures $E_{\mathrm{op},i}$, $P_{\mathrm{leak},i}$ are extracted from the datasheets of characterized components and they are scaled with scaling factors $\kappa_{\mathrm{op},i}$, $\kappa_{\mathrm{op},i}$, which are obtained from the mapped hardware and an example workload. Figure 3.5a shows the normalized efficiency $1/\mathrm{Cost_E}$ over the duty cycle and the size of the memory component. The duty cycle indicates either no memory transactions, i.e. $d = 0$, or memory transactions in every cycle, i.e. $d = 1$ [JL3]. Since the duty cycle directly relates to the number of operations in the activation memory and the memory size relates to the number of components, the diagram displays the cost tradeoff for different activation configurations. Thus, an early informed design decision can be made based on specific requirements of the targeted TCN accelerator. In the case of the fitted architecture in Section 3.1, there is a cost advantage of the register file compared to a SRAM macro (design point marked in black).



(a) Efficiency comparison of ultra-low leakage SRAM and D-flip-flops. Higher efficiency relates to less power consumption.



(b) Ring buffer (RB) and shift register (SR) simulated with 12 bit words compared for different buffer sizes .

Figure 3.5: Cost comparison for memory devices and simulation results for logic level implementations [JL7].

The classical method to determine the power consumption of logic implementations is to simulate all designs using e.g. post-synthesis or post-layout netlists. Although the design effort is much higher than the high-level cost analysis demonstrated earlier, the power estimation is more precise and leverages industry-standard EDA tools. For instance, it can capture switching activity of each cell in a testbench setup and differentiate between the dynamic energy consumption of rising/falling edge. Figure 3.5b shows the post-synthesis simulation results of ring buffers and shift registers for 12 bit words in a 22 nm CMOS technology. In this case, the shift register requires both less leakage and dynamic power than the ring buffer implementation for random input activity. Since the number of cells required for the alignment logic increases quadratically with the buffer size, leakage is also increased with the increased area. Further, more cells are switching, thus, increasing dynamic power of the ring buffer implementation. Hence, a shift-register outperforms the ring buffer for activation buffering in this configuration.

PE Mapping

As data is stored in SIPO buffers, the activations for the convolution operation are pre-aligned and need to be multiplied with their corresponding weight and accumulated for each input and output channel. In high throughput DNN accelerators, the arithmetic operations are performed in PE arrays [70], which enable parallel MAC operations. While high-throughput architectures prefer a high-utilization of the PE array (e.g. [JL6]), low-power ECG accelerators need to balance the dynamic and static power of the components. This can lead to DNN inference logic with few or even a single PE executing MAC operations sequentially [56]. In the co-designed ANN architecture by Zhao et al., the classification algorithm contained only 592 MAC operations and, thus, the sequential computation does not violate any latency constraints.

In other cases, however, the design choice is not clearly seperable between the two extreme cases. On the one hand, a fully parallel PE design, or full-flat mapping, enables fast computations, but sacrifices area and static power. On the other hand, a fully sequential PE design, does minimize area and static power, but based on the computational complexity of the inferred DNN might violate real-time constraints (as discussed in Section 2.3.3).

It is important to note that power gating might be a viable option to utilize a highly parallel design for DNNs with low MAC operations and, thus, mitigate drawbacks in static power consumption. However, it needs to address how mode transition times and energy impact the architecture in a

stream processing design, which frequently switches between active and inactive mode. Within the scope of this work, power gating is not utilize to simplify the design process.

Given a DNN, the question is how many PEs are requires to execute its computations. The cost model of Eq. 3.6 also provides an informed solution to above mentioned design problem. Again a cost model can be constructed based on the estimated static and dynamic power consumption of the system. In the PE mapping process, however, the concept requires adjustments. Typically, EDA tools use characterized standard cells to perform power estimations. In this case, $E_{\text{op},i}$ and $P_{\text{leak},i}$ are available on gate level, which are summed together using the gate-level netlist after synthesis. However, there are cases, in which it is not desirable to perform synthesis over all design points. For instance, the mapped architecture features too many possible design points, e.g. PE array sizes. Or synthesis and post-synthesis simulations for each design requires too much time for a full exploration. In this case, the cost model should provide an estimate of the qualitative differences between the explored configurations of the design.

The PE configurations for subsampling-based TCNs (see Section 3.1.2) result in specific trade-offs between static power from the implemented PE units and the dynamic power from the sequential processing on those units (c.f. [JL3]). Since both DWT operations and TCN operations in each layer are convolutions, the PE are constructed using MAC units. In alignment with a previous experiment (see Fig. 2.13 in Section 2.3.2) and previous DNN accelerators [70] that the parallel processing of rows, i.e. in the 1D case a whole channel, is chosen for the PE exploration in [JL3]. Therefore, a vector MAC unit is used to perform the convolution for one SIPO buffer of one input channel. In the exploration [JL7], the number of PEs is increased from one unit, i.e. fully sequential execution, to a full flat mapping, i.e. isomorphic architecture [104]. In terms of static power consumption, the number of units scale leakage. Here, the layers of the TCN from Table 3.1 are subsequently inserted starting from the initial DWT layers. The number of parallel PEs is, hence, determined by

$$N_{\text{PE}} = \sum_{i}^{L} k_i \cdot C_i, \tag{3.7}$$

where k_i is the kernel size of the convolution and C_i is the number of output channels of layer i. In terms of dynamic power consumption, the operating frequency can be decreased for more concurrent PEs. This required minimum frequency $f_{\text{sys,min}}$ is calculated based on the input frequency f_{in} and the

subsampling factors d_i in each layer i, hence,

$$f_{\text{sys,min}} = f_{\text{in}} \cdot \prod_{i}^{L_{\text{max}}} d_i. \tag{3.8}$$

For the cost function, we use N_{PE} and $f_{\text{sys,min}}$ as the scaling factors $\kappa_{\text{leak},i}$ and $\kappa_{\text{op},i}$, respectively. For the leakage $P_{\text{leak},i}$ and energy per operation $E_{\text{op},i}$ of a technology node, a characteristic template cell from the standard library is used.

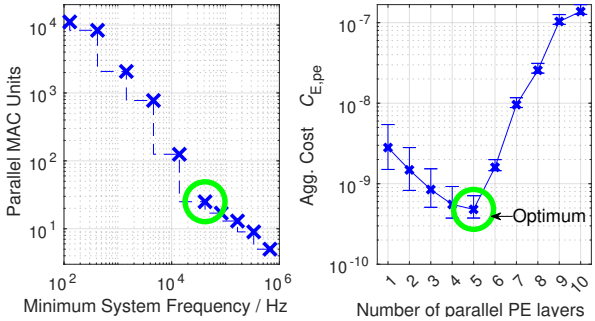


Figure 3.6: Pareto-optimal front (left) and aggregated cost (right) for parallelized layers using a DFF as the template cell. Variations of $P_{\text{leak},i}$ and $E_{\text{op},i}$ for rise/fall transitions are resulting in cost uncertainties [JL3].

Figure 3.6 plots the result of the PE sweep. The left diagram plots the number of parallel MAC units as function of the minimum system frequency of the design points. The aggregated cost $C_{\text{agg,pe}}$ is the sum of above mentioned cost components relating to static and dynamic power. A cost optimal point is visible for five parallel PE layers. This corresponds to fully flat mapped DWT components and one vector MAC unit for the sequential inference of the TCN layers. Since the cost estimation is based on Eq. (3.6), a good correlation of the estimation to the actual system power is dependent on the prior knowledge of the cells used in the final implementation. For instance, with no knowledge of the synthesized standard cells, representative assumptions, e.g. about used cells, need to be made to extrapolate the cost for a component level to system level. A more granular approximation is, hence, available with more knowledge of the expected gate-level netlist after synthesis. One example is to construct the aggregated cost $C_{\text{agg,pe}}$ using not only one characteristic template cell but a selection of cells to construct the components of leakage $P_{\text{leak},i}$ and energy per operation $E_{\text{op},i}$.

3.2.2 Digital Implementation

The previous hardware mapping phase investigated cost-optimal components for the implementation in a complete system. In the following, these components are integrated into a system level architecture to process the DWT pre-processing and TCN in a streaming fashion.

Cascaded Fixed-Coefficient DWT Filters

Given the PE mapping exploration from Section 3.2.1, the initial DWT pre-processing should be realized as dedicated components. As a full streaming architecture is targeted, the cascaded FIR filters are used to realize the DWT low- and high-pass filters.

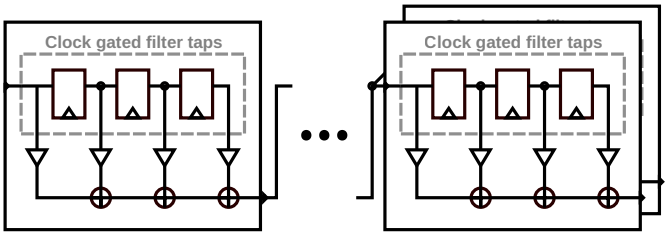


Figure 3.7: Flat mapping of cascaded FIR filters DWT pre-processing.

Figure 3.7 shows the circuit diagram of the DWT pre-processing block as a fully flat mapped component using FIR filters [JL1]. Each block represents one 4-tap FIR filter for the db2 wavelets. The triangles indicate a fixed coefficient multiplication. The result is summed together in an adder tree. The design description in RTL is realized as a behavioral description with parametrizable number of taps, word length and filter coefficients. Therefore, the fixed coefficient multipliers and adder trees can be optimized by synthesis, where registers for the coefficients are omitted and replaced with static wiring for shift and partial product summation. The subsampling is realized by the selective clock-gating of the filter taps, which enables the propagation of the generated output. Conceptually, the control of the enable signal for the clock gates can be generated locally or globally. For instance, a global state machine can orchestrate the enable signals through a common counter logic or there are local counters per layer, i.e. binary counters for subsampling factor 2, which are triggered by preceding enable signals.

The data rate of the ECG data stream is reduced layer by layer, i.e. halved by every stage of the DWT pre-processing component. In Figure 3.8 on the left, the rate reduction is plotted for an input frequency of 300 Hz. The output frequency after each is halved such that the input samples of the

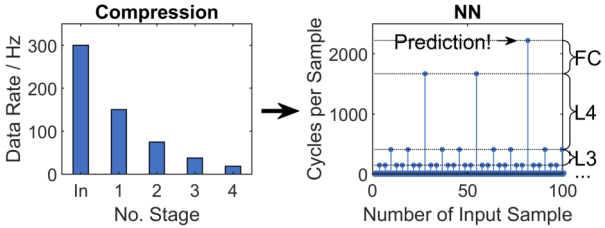


Figure 3.8: Data rate reduction in flat mapped DWT components and number of cycles for partial TCN inference over the number of input samples from the DWT layer [JL3].

TCN are available at 18.75 Hz after four stages. Basically, this stage both compresses the data with fewer samples and increases the time frame for the subsequent sequential TCN inference.

Data-driven TCN Inference on a Vector MAC Unit

The inference of both convolution and dense layers are mapped onto one vector MAC unit using the size of the convolution kernel as the tiling size. Analogous to the FIR filter of the DWT units, the vector MAC units comprises the multiplication with weights and the subsequent accumulation. However, the weight kernels are selected from an external SRAM unit and the input taps are divided into a separate module (see next Section for more information).

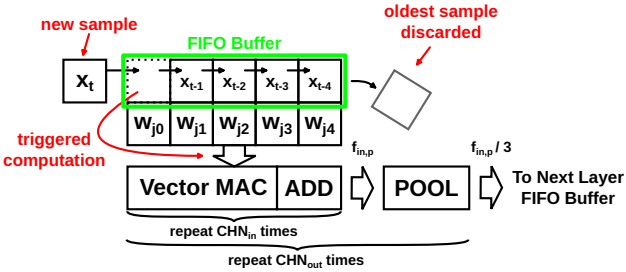


Figure 3.9: Data movement in a SIPO buffer, which provides self-aligned input feature maps for the channel-wise convolution subroutine [JL3].

The concept of the serialized inference is visualized in Fig. 3.9. The buffer receives new temporal sample either from the DWT output or the previous TCN layer. For instance the DWT output generates two new samples, i.e. from A4 and D4 coefficients, which are inserted into two separate shift registers. Since the insertion results in a new alignment of the buffer content to the weight kernel (and the omission of the oldest sample), potential new outputs need to be calculated for the layer. In the computation subroutine, the vector MAC unit is reused to iterate through the input channels,

which are accumulated in the output stationary dataflow. Once all partial sums resulting from the input channels are added for one output channel, the process is repeated for all output channel. The results are each fed into a pooling unit for each output channel. The pooling unit also calculates the pooling operation in an iterative manner. Samples from the pooling unit are emitted for every third input sample, i.e. size of the pooling kernel. The output from the pooling unit are again fed into the buffer units of the next TCN layer or the output prediction buffer. In Figure 3.8 the number of cycles are plotted as a result of the aforementioned process. Each layer requires approx. $C_{\text{in}} \cdot C_{\text{out}} + 1$ cycles, i.e. one cycle for each input and output channel and one pooling operation. The diagram shows that every third input sample results in the computation of the next layer. Hence, the first layer is triggered for every DWT sample. The second layer triggers for every third, the third layer for every ninth etc. The prediction is generated for every 81th sample. For an input rate of 300 Hz, a prediction is generated every $f_{\text{out}} = 4.32\,\text{sec}$ with a stride of $N_{\text{stride}} = 2^4 \cdot 3^4 = 1296$. The effective frame size can be calculated recursively based on the kernel size from the final dense layer using

$$N_{\text{eff},i-1} = N_{\text{eff},i} \cdot k_{\text{pool},i} + (k_{\text{conv},i} - 1) \tag{3.9}$$

with $N_{\text{eff},i}$ being the effective input size at layer i and kernel size $k_{\text{pool},i}$ and $k_{\text{conv},i}$ for pooling and convolution, respectively. This results in approx. 16-17k samples for N_{frame} covering up to nearly one minute of an ECG recording satisfying AF classification requirements [1].

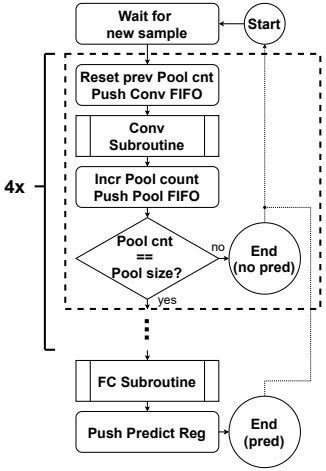


Figure 3.10: Flow chart of the data-driven TCN inference [JL3].

Figure 3.10 exemplifies the TCN inference process in a flow chart. Similar to the DWT components, the trigger mechanisms are controlled through local counters, which enable the insertion of data into the SIPO buffers and the subroutine of the next layer. In the end, the control mechanism is realized in a static state machine, which coordinates the counters and derived enable signals for clock gates according to TCN layer.

WVCNN Accelerator Architecture

The resulting ECG accelerator is denoted as WVCNN for the combination of wavelet transforms and convolutional layers with subsampling. The system level block diagram is depicted in Fig. 3.11. The test environment (blocks marked in blue) interfaces the accelerator (blocks marked in red) for test purposes. Here, a register-file is accessed through a UART-Regfile bridge, which consists of open-source modules for the communication from UART to AXI and AXI to a register file. The register file is used to set registers for accelerator configuration and for the streaming input. On the one hand, the test environment is used to set the weights and biases of the TCN within the SRAM. On the other hand, the register is used to feed the input samples of the ECG signal into the accelerator. For the data-driven TCN inference, the input sample needs to further accommodate a strobe signal to indicate a valid new input to trigger the computation engines. The strobe signal is set to 1 for one clock cycle, which is also used as the enable signal for the first SIPO register. The device under test (DUT) consists of the modules for DWT pre-processing, denoted as compression engine, and the modules for TCN processing, denoted as NN acceleration engine.

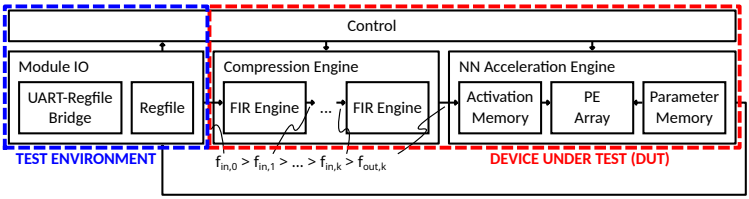


Figure 3.11: System-level architecture of the WVCNN design including the test environment [JL3].

As mentioned above, the compression engine comprises cascaded FIR filters. However, the neural network acceleration engine splits the memory and the PE. Figure 3.12 shows the block diagram of the neural network acceleration engine. The modules are split into three parts: the activation memory, the PE array and the parameter memory. The activation memory is the collection of all SIPO registers

for the input activations of all layers. In contrast to the buffers for the convolution layers, the buffers for the pooling layers do feature additional logic for the subsampling and the maximum function. In the maximum logic the input is directly compared with the stored value and updated with the current maximum. A cyclic counter logic is used to control the reset and the value propagation to the next convolution buffer. In this case, the logic sets enable high, when the counter reaches its maximum value.

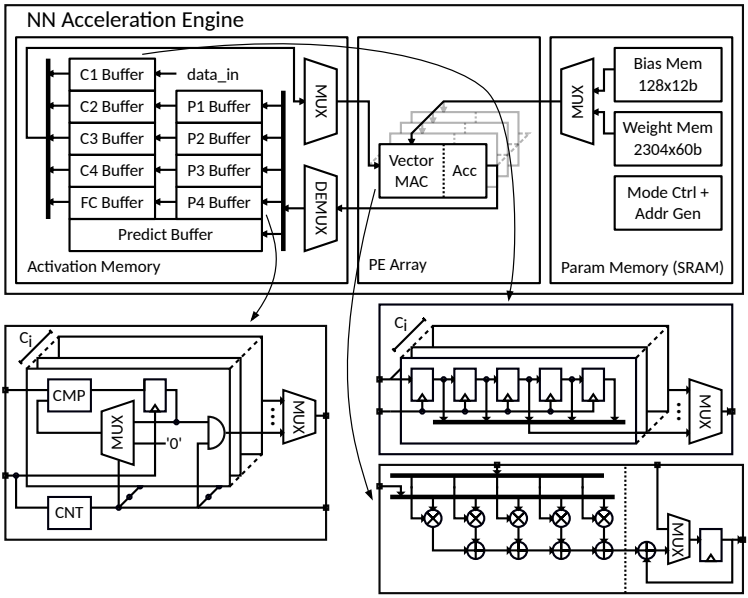


Figure 3.12: Block diagram of the neural network acceleration engine in the WVCNN design [JL3].

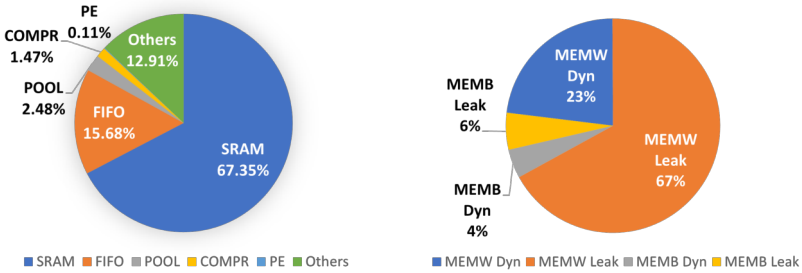
Although the pooling buffers are directly connected to the convolution buffers, the connection from the convolution buffer to the PE array and from the PE array to the pooling buffer needs to be multiplexed. The multiplexer logic is split into two stages, in which the first stage selects the set of buffers corresponding to the inferred layer. In specific, for the inference of layer 3 the C3 buffers are selected as an input to the PE array and the P3 buffer is selected as the buffer, which receives the result of the PE array. The second stage of the multiplexer hierarchy selects the channel of the input and output activations. The same control signals are also used to align weights and biases from the SRAM and to reset the accumulator within the PE array. For the parameter memory two single port SRAM modules are used to store the weights and biases. The weight memory contains 60 bits per

address line, such that 5 weights can be read at once for the whole vector multiplication. Even though the weights and biases can be reconfigured on-the-fly, the size of the memory block is fixed based on the TCN model from Section 3.1.3.

The HDL description is written, such that the counters for number of input/output channel, pooling/convolution kernel size, number of layers are parametrizable. Thus, all TCN model sizes can be accommodated in a dedicated design. Further steps can be made to increase the flexibility and reconfigurability of the design. These could include gated filter taps and PE units or reconfigurable channel sizing. However, this would introduce more logic components, which contribute additionally to the overall power consumption.

3.2.3 Design Evaluation

In the end, the WVCNN design is evaluated in a 22 nm fully-depleted silicon-on-insulator (fdSOI) CMOS technology by GlobalFoundries. Commercial EDA tools were used for the digital design flow. In specific, Cadence Genus is used for synthesis, Cadence Innovus is used for the place-and-route, Mentor QuestaSim is used for the post-synthesis and post-layout simulation and Cadence Voltus is used for the power evaluation on the back-annotated netlists. For synthesis, the ultra-low leakage standard cells are used for the final evaluation as leakage dominates the total power consumption. Simulations are performed at nominal conditions, i.e. TT process corner, 0.8 V supply voltage and room temperature. ECG sequences with a sample frequency of 300 Hz are streamed into the accelerator. The sequence of 60 sec will result in 13 predictions, as long as the operating frequency satisfies Eq. (3.8).



(a) Breakdown of power for components in WVCNN design grouped by function.

(b) Breakdown of power for components in WVCNN design for SRAM components only.

Figure 3.13: Power breakdown of WVCNN design based on back annotated post-synthesis netlists [JL3].

Figure 3.13 shows the power breakdown of the WVCNN design. The breakdown in Fig. 3.13a shows that two third of the system power is dissipated by the parameter memory (SRAM). The activation memory (FIFO) consumes only 15.68 %, while the other components, such as DWT preprocessing (COMPR) and the vector MAC unit (PE), are comparatively low. Due to the big contribution of the SRAM component, it is further inspected in Fig. 3.13a. It is evident, that the weight memory (MEMW) consumes a higher percentage than bias memory (MEMB) due to the bigger memory size. Further, the leakage power from both memories is dominating the power consumption with approx. 73 %. A reason for this is the low utilization of the memory during the data-driven inference of the classifier. When considering the TCN inference (see Fig. 3.8), not every new input sample from the DWT pre-processing results in a vector MAC operation, thus, the PE unit is idling most of the time. Assuming the operation of the accelerator at the minimum clock frequency as defined in Eq. (3.8), the maximum number of cycles is defined by the sample triggering the prediction, i.e. computing all TCN layers. Thus, the PE is utilized to 100 %, if it is computing the maximum number of cycles for every new input sample. Compared to this reference, in the data-driven paradigm the PE only needs to be active about 6.55 % of the time. In general, this utilization is dependent on the number of channels and subsampling factors of the TCN architecture. Nevertheless, we observe that the WVCNN design effectively reduces dynamic power consumption to the degree that the power bottleneck is the leakage of the memory components. The design in Fig. 3.13 were already considering low frequency operation with ultra-low leakage components. Therefore, we expect that further power saving can be achieved through leakage reduction through e.g. power gating and voltage scaling. However, power gating has not been investigated in the scope of this work.

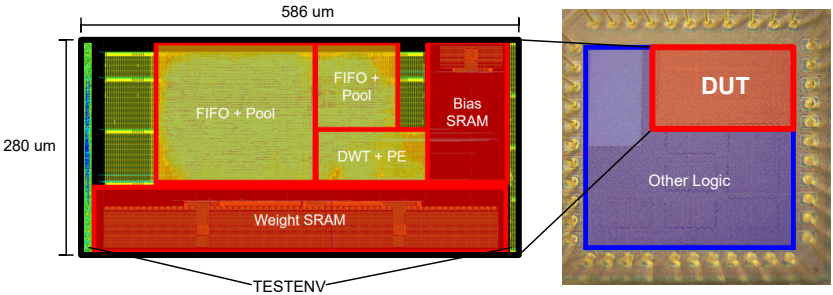


Figure 3.14: Layout level view of the WVCNN design and the chip micrograph [JL3].

In the end, the WVCNN design has been fabricated and the chip micrograph as well as the layout level view can be seen in Fig. 3.14. The design occupies an area of 280 µm × 586 µm. The biggest

area is occupied by the memory components both from the SRAM and the activation FIFOs. The actual PEs for DWT and TCN is comparatively small and located in the center of the memory components. The test environment is surrounding the DUT from the left and right and is not considered in the evaluation.

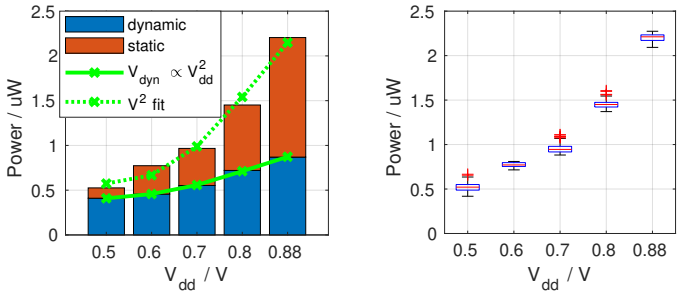


Figure 3.15: Measurement results of the fabricated WVCNN design operating at 500 kHz [JL7]. The left diagram shows the average power consumption split into static and dynamic power and the right diagram shows the distribution of total power over 5 measurements across multiple dies.

Measurements on the fabricated chip are performed for the same input stimuli as the post-synthesis and post-layout netlists. The power domains of test environment and DUT are separated, such that the current can be captured for the DUT only. In the experiment, the current is sampled during the period, when the prediction is calculated. From all 13 predictions in a 60 sec ECG trace the average is taken as one measurement point of an iteration. Five measurement iterations across multiple chips are performed to account for on-chip variations and noise in the measurement setup. Figure 3.15 summarizes the results for voltage sweep from nominal, i.e. V_{dd} = 0.8 V, down to V_{dd} = 0.5 V. On the right, we observe that the statistical distributions across all voltages show robust clusters with small deviations, i.e. up to $\sigma = 6.4\%$. On the left, we can see that the dynamic power is scaling proportional to the squared supply voltage. Further leakage is significantly reduced by voltage scaling, such that the final design only consumes about 525 nW.

Another option available in the fdSOI CMOS technology node used is body biasing [105]. In this method the well is polarized to create a non-zero potential to adjust the gate voltage of the transistor devices. Consequently, gate voltage or leakage of the transistor devices can be further reduced by applying a body bias potential. The latter is interesting for the WVCNN design, since the leakage is a major contributor to overall system power. Figure 3.16 shows the shmoo plot for the fabricated WVCNN design at a reverse body bias V_{RBB} of 0 V, 0.9 V and 1.8 V. The ultra-low leakage devices

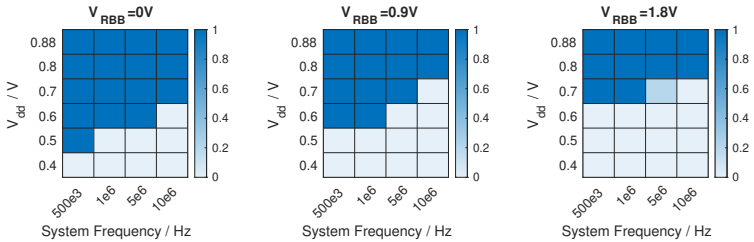


Figure 3.16: Shmoo plot for three different levels of reverse body bias [JL3].

can only be biased in the opposite direction of the gate potential and, thus, reduce the leakage of the devices. It is evident, that without body bias the WVCNN design can be operated at low voltages and low frequencies. If the bias voltage is increased, the design only remains operable at higher supply voltages. In the end, the gain in power consumption, which is achieved through body biasing, is negligible (up to 5 %).

3.3 Summary

Within this chapter, an ECG classification system has been designed from scratch and demonstrates systematic design of an accelerator for smart sensor applications. The key to obtain a system with both low power and high quality classification requires the co-design of algorithm and hardware. In this work, it is shown that a classic DNN model carefully chosen with uniform operations is capable to achieve such requirements. A DWT as pre-processing and a TCN as classifier mainly require convolution operations to achieve high quality AF classification. The subsampling in both components are favorable for the streaming architecture design as it reduces the number of samples consecutively over each processing stage. Based on this algorithm baseline, the hardware components could be designed such that both static and dynamic power is balanced in a cost optimal point. This is achieved through a cost-driven selection of memory devices and a data-driven computation of the TCN. In the end, these classic design techniques applied on the continuous ECG monitoring use case results in a design, which is capable of classifying AF with a F1 score of 79 % and a power of down to 525 nW.

This work intends to show how a systematic selection process in combination with classical design techniques can already achieve state-of-the-art performance. However, as outlined in Section 2.2.1, DNN classifiers applied on the ECG classification task span over all variants of modern machine learning technique. The adoption of modern algorithms to this task shows that the development

of ECG classifiers is still a challenging application to solve. For instance, recent developments in transformer architectures are modified for arrhythmia detection in microcontrollers [106]. In contrast to popular machine learning benchmarks in image classification, the amount and quality of data for biomedical applications is a challenge to provide sufficient generalization. Similar to modern machine learning applications [107], we expect that classifier quality scales with sufficient data and model size. Here, the improvement in classification quality is expected to be applicable across all different DNN architecture variants. The advantage of an architecture with only convolution operations, as demonstrated in this work, is the mapping to less complex computing units and the scalability of the model to support increasing model sizes for the generalization across larger datasets. In complex DNN architectures, such as transformers, the variety of operations, e.g. different activation functions and complex dataflow, need to be supported in the computation platform. The co-design of power optimal designs needs more complex design space exploration than PE and activation memory sweeps (as shown in Section 3.2.1), which need to account for every new operation and component in the design.

Chapter 4

Hardware-Aware Domain Generalization for ANN-Based Feature Alignment

The efficient use of continuous monitoring systems does not only require low power operation of high performing classifiers. It is also dependent on its robustness in a real-life situation, e.g. how well the classifier performs with different input data, noise etc. Especially in the case of the monitoring of cardiac arrhythmia, the classifiers of the algorithm-hardware co-designed systems show incredible performance on their respective dataset (see Section 2.2.1), but rarely discuss the results beyond the scope of this dataset. Therefore, commercial products rarely incorporate such monitoring systems, as their efficacy in real life situations remain to be validated. Instead, traditional thresholding of hand-engineered features are still the preferred methods [2].

One main issue is that the data used for training and the data captured during deployment differ. Hence, the trained DNN does not perform as well, since it is not generalizing well enough beyond the available training data. Especially in biomedical applications with the need for high-quality feedback, it is essential that prediction of these systems are reliable and robust. Hence, it is critical to address this *domain shift* problem for the successful deployment of continuous monitoring systems on mobile devices.

In the following, we first discuss the context of domain generalization with respect to ECG classification as an application. Then, we review the state-of-the-art addressing the domain shift problem and outline a low complexity solution, which generalizes across source and target domain using a single linear layer. In the end, this concept is evaluated and compared against a reference case, where fine-tuning is used for generalization.

4.1 Application Context of Domain Generalization

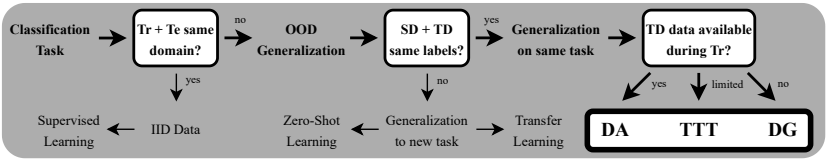


Figure 4.1: Summary of the classification problem of data dependent on the availability of data during training (Tr) and testing (Te). For the robust classification of ECG data in a practical setting the aim is to generalize over OOD data with a domain shift between source domain (SD) and target domain (TD) [JL10].

Figure 4.1 shows the different facets of the classification task. The complexity is defined by the data, which is available during training and deployment of the model used. A typical classification scenario assumes data to be idependent and identically distributed (IID), which is common practice in algorithm-hardware co-designed DNNs. Hence, the task is typically solved using supervised learning, as the distribution of training data matches the test domain. Nevertheless, in many application cases, such as ECG monitoring, it is not possible to capture all features across patients and measurement setups, such that a domain shift between training data and deployment is inevitable. In the scenario, where new labels are introduced in the target domain, the generalization over out-of-distribution (OOD) would extend into complex disciplines, e.g. zero-shot learning [108] or transfer learning [109], which is out of the scope of this work. Instead, we focus on the generalization of different data distributions in source and target domain, but the same label set. Here, the task can be further subdivided into domain adaptation (DA), test-time training (TTT) and DG. The fundamental difference is how much data of the target domain is available for the training phase. While domain adaptation assumes that the target data can be used for training, test-time training only allows the adaptation of the model during test-time with limited data. The ideal scenario would be to generalize to the target domain without incorporating its data for training, i.e. domain generalization.

Figure 4.2 shows the sources of domain shift for ECG data. On the left, the sources are summarized based on the measurement setup. It is evident, that the domain shift is not only defined by noise, but also contains systematic patient specific components such as deviations in the body impedance between electrodes, the contact of the electrode itself and its position. Further, configurations in the analog front end also influence the captured data, such as sampling rate, bandpass filters, resolution etc. On the right, the deviations of openly available ECG datasets, as in PhysioBank [24], are listed. It is evident, that not only the data quality deviates, but also the labeling scheme. For instance, some

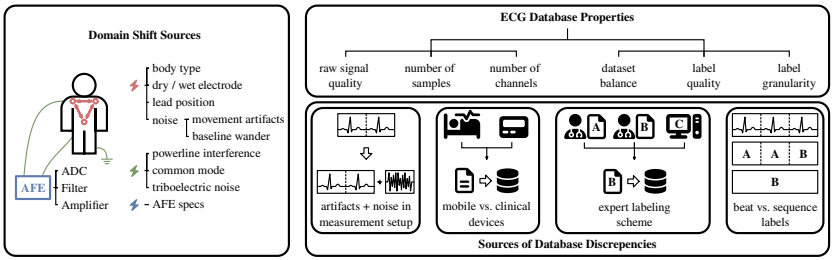


Figure 4.2: Summary of domain shift sources in ECG data acquisition. The discrepancy is databases does not result only from deviations in the measurement setup, but also the labeling scheme [JL10].

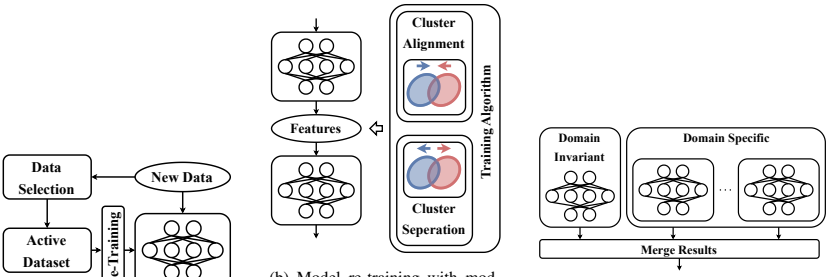
labels are assigned to a whole sequence [18], but a majority is assigning labels to individual beats [24]. Another example is the introduction of grouping schemes to guarantee sufficient samples in training [56]. While it is possible to simply duplicate labels from one sequence to all individual beat in it, it is questionable, whether all beats do contain the features associated with label of the sequence. In the end, the usage of multiple databases requires a method to transform one scheme into another or one is limited to the selection of databases with coherent measurement and labeling setups.

Therefore, the design of DG algorithms and corresponding hardware components need to be viewed under the circumstances of the suitability of available data. The choice of the used datasets and methods need to match a practical scenario suited for the continuous monitoring of ECG for deployment on smart sensors as targeted in this work. In the following, recent literature is reviewed to see how well the state-of-the-art covers this application case.

4.2 State-of-the-art Domain Generalization for ECG Classification

Figure 4.3 shows the structure of most commonly applied DG algorithms in the use case of ECG classification. One method is to continuously extend the set of samples used for training with OOD samples. The collection of samples from the target domain in an active dataset is used to fine-tune the original DNN, such that the network incorporates and adapts to new features [110], [111]. Figure 4.3a depicts a high-level schematic of this method. The samples stored in the active dataset are selected based on a filter criterion to reduce the memory footprint and maximize the diversity of samples. Despite the selection process, the concept of an additional dataset used for model finetuning requires both the memory to store raw ECG data and the model finetuning. In the context of smart sensors, both on-chip memory and computations need to be minimized as much as possible to reduce power

consumption. A DG method with constantly scaling memory requirements, however, is suboptimal from the perspective of power consumption. Therefore, it needs to be further evaluated whether memory limitations impact DG performance.



(a) Model re-training with an extensible active dataset.

(b) Model re-training with modified loss functions for cluster alignment and separation in feature space.

(c) Ensemble models with classifiers for domain invariant features and domain specific features.

Figure 4.3: State-of-the-art DG methods applied for the classification of ECG signals [JL10].

Another method is to align features for the same classes and different domains to each other [111]–[117]. Similarly, features from different classes and different domains are separated in the feature domain (see Fig. 4.3b). This is achieved by the modification of the loss function during the training process. An additional term is rewarding a short/long "distance" of features resulting in the prediction of the same/distinct classes. This concept relies on the assumption, that within the feature space the distributions should be robust against domain shift and the features are similarly clustered. The distance metric is critical for this method, as it quantifies the similarity of features. In this method, the fine-tuning of the whole DNN model is still required.

In a third method, the classifier is split into multiple models, thus, classification is performed by the ensemble of domain in-/variant models (see Fig. 4.3c) [118].Conceptually, each new target domain requires the training of a new DNN model. From a resource perspective, the memory and computational requirements are duplicated for each new classifier added, when assuming similarly sized models for each branch.

4.3 Low Complexity DG Algorithms

It is apparent that data and model up-scaling are focused in recent literature. This results not only in the increased generalization performance, but also similarly scaled resources, which is unfavorable for

resource-constrained environments as smart sensors. The method of model retraining with modified loss functions for feature cluster alignment and separation is promising, since the original model is reused to incorporate the target domain. Therefore, the model complexity remains similar with a different training setup.

4.3.1 Complexity Exploration in State-of-the-Art DG Methods

When looking at the problem of DG for smart sensors from scratch, the key question to answer is: how much complexity is needed to achieve the capability of generalization. In the next experiment, the separation of features is visualized for two feature alignment techniques on homogeneous and heterogeneous domain shift. The idea is to evaluate the generalization performance on the use case of AF classification for different degrees of domain shift and different complexity of the DG algorithm. Therefore, we focused on the binary classification of AF and normal sine signals in the inter-patient scenario as a representative for homogeneous domain shift in the AFDB dataset. The more complex problem is the transition to the CinC'17 dataset with a different ECG device and noise, which represent the heterogeneous domain shift. For the DG algorithm, we chose the instance normalization in each layer to normalize features for each instance, which prevents domain specific means and variances in the intermediate activations. In contrast to this basic normalization, contrast learning is chosen as a representative for the methods depicted in Fig. 4.3b. Although this method still uses the original DNN model for fine-tuning, all weights and biases of the network are adapted instead of the scaling factors for the normalization. As a baseline network a fully convolutional network is trained from scratch on the source domain after which the adaptation to the target domain is performed.

Figure 4.4 visualizes the feature distribution in the last convolution layer before the final dense layer. The features are mapped to the 2D plane using principle component analysis (PCA) and colored by their correspondence to their ground truth label. The classes are clearly clustered and separable in the baseline and source domain. Since no generalization is considered in the baseline the features are not separable for the two target domains within the AFDB dataset and the CinC'17 dataset. Both the instance normalization and the contrastive learning are able to separate the feature clusters for the unknown AFDB patients. However, this is not clearly visible for the CinC'17 samples.

Table 4.1 shows the quantitative results of the experiment. Coherent to the visual representation of the feature distributions in Fig. 4.4, the achieved F1 score is high in the baseline classifier for the source domain. The performance is dropping significantly in the CinC'17 benchmark, i.e. down to

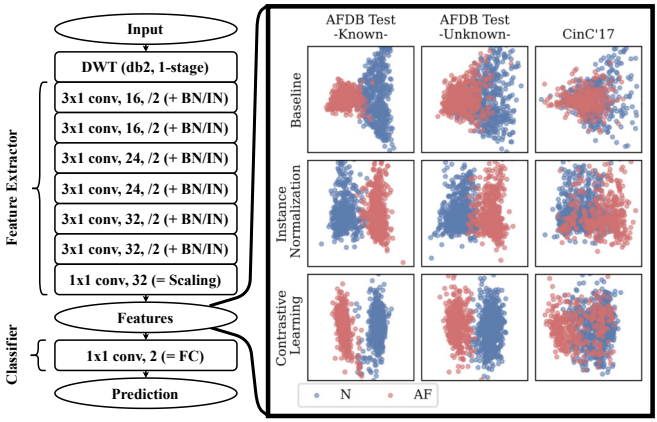


Figure 4.4: Fully convolutional neural network for the evaluation of instance normalization and contrastive learning. Features before the final dense layer are visualized in the 2D plane after principle component analysis [JL10].

	AFDB				CinC'17	
	Test - Known		Test - Unknown		Test - Unknown	
	N	AF	N	AF	N	AF
Baseline (w/o DG)	0.93 ± 0.03	0.90 ± 0.03	0.91 ± 0.07	0.88 ± 0.07	0.67 ± 0.01	0.17 ± 0.21
Instance Normalization	0.96 ± 0.02	0.93 ± 0.03	0.92 ± 0.10	0.91 ± 0.09	0.79 ± 0.02	0.78 ± 0.03
Contrastive Learning	0.97 ± 0.02	0.94 ± 0.02	0.96 ± 0.04	0.94 ± 0.05	0.80 ± 0.01	0.77 ± 0.02

Table 4.1: Validation F1 score of DG techniques on the inter-patient AFDB and CinC'17 dataset using 5-fold cross-validation.

17 %. However, the baseline also performs quite well for homogeneous domain shift with 88-91 %. Both instance normalization and contrastive learning increase the performance on the target domain of AFDB unknown. The F1 score is saturating at 91-92 % and 94-96 % for instance normalization and contrastive learning, respectively. The same trend is also visible for the CinC'17 samples. In this case, the F1 score is increased to 78-79 % and 77-80 % for instance normalization and contrastive learning, respectively. The relative performance gain is more evident for the CinC'17 samples, although it is not visually apparent in the feature distribution. Nevertheless, the difference in classification quality between instance normalization and contrastive learning is minimal.

Based on this experiment, we reevaluate the necessity to fine-tune the baseline DNN to achieve generalization. As seen for instance normalization, the normalization layers simply reduce mean

and variance between source domain and OOD samples and, thus, achieve a similar classification performance as more sophisticated DG algorithms.

4.3.2 Correction Layer (CL)

Hence, the concept of correction layer (CL) is introduced to align the features using a single trainable layer. It borrows the principle of the methods from Fig. 4.3b. However, the training is concentrated on the CL, in which the normalization is performed. Conceptually, the realization of the CL can be chosen as an arbitrary transformation, which is trained within a DNN with frozen weights and biases.

As an initial entry point, linear transformations are investigated. The advantage of linear transformation as a CL is that they can be merged into adjacent layers with linear transforms, e.g. convolution or dense layers. In the following, two types of CL are considered: the channel-wise and the inter-channel transform.

$$\mathbf{f}_{\text{cw}} = (\mathbf{w} + \mathbf{1}) \odot \mathbf{x} \tag{4.1}$$

$$\mathbf{f}_{\text{ic}} = (\mathbf{W} + \mathbb{I}) \cdot \mathbf{x} \tag{4.2}$$

The "$\odot$" and "$\cdot$" denote the element-wise and matrix-vector multiplication, respectively. The channel-wise transform aims to scale each channel with a weight separately. The inter-channel transform comprises the weighted sum of all input channels into separate output channels. It is evident that the former and latter scale linearly and quadratically, respectively. Thus, these two transforms showcase two types of linear transforms with different complexity. In principle, the concept can be extended further to biases and arbitrary transforms, e.g. polynomial or non-linear transforms in general. In the following, the channel-wise and inter-channel transform are used to demonstrate the CL concept.

Figure 4.5 shows the CL training concept. An arbitrary multi-layer DNN architecture is trained on the source domain. In a consecutive step, the parameters are frozen such they do not change during the training of the CL. To train the CL, first, it is inserted into the pre-trained DNN. The input and output dimension should match, such that the original architecture is unmodified. Then the correction layer is trained on a subset of the target domain to align the features at the output of the CL. The validation is performed on the remaining set of the target domain. In this setup, the target domain is chosen such that it contains a balanced number of AF and normal samples to ensure good performance on small samples sizes. In the end, the training procedure consists of two interleaved cross-validation steps with 5 folds to guarantee robust results across the dataset.

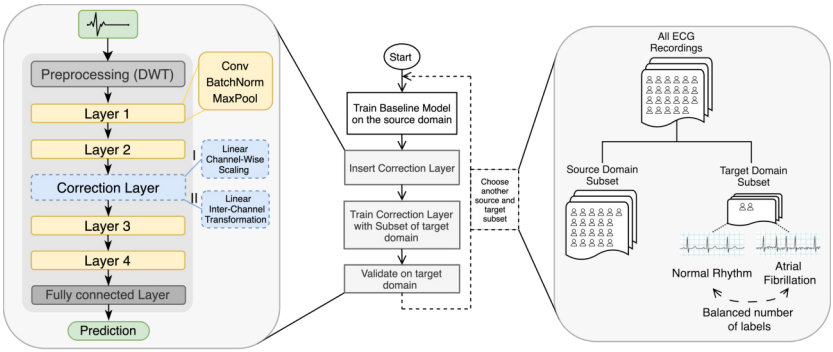


Figure 4.5: Experimental setup for the insertion of CLs in a DNN. The training and validation is performed in two distinct steps. First, the DNN is trained without the CL on the source domain. Second, the CL is trained on a subset of the target domain. Both stages are cross validated for statistically robust results [JL10].

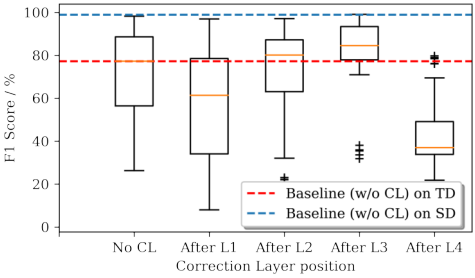
4.4 Evaluation of Correction Layer Performance

Since the objective of the CL is to provide inter-domain generalization, the evaluation of CL performance is divided into classification quality and hardware complexity.

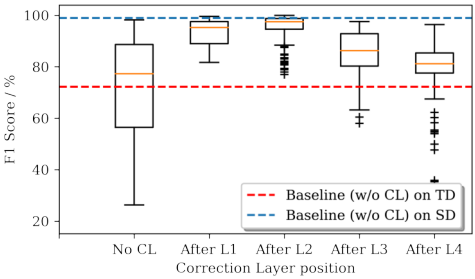
4.4.1 CL Classification Performance

Figure 4.6 shows the quality of service of the baseline and the CL version measured using the F1 score. The CNN from Section 3.1.2 (see also [JL3]) is chosen as a baseline architecture for exploration. We trained the model on the AFDB dataset for binary classification of AF and normal sine signals. The architecture is adjusted to the new task through the number of intermediate channels, i.e. the number of output channels are $(10, 24, 50, 70)$ from the first to last convolution layer. The split of source and target domain is done based on the inter-patient paradigm, where the recordings of both domains contain mutually exclusive subsets of patients. The blue dotted line shows the average F1 score of the baseline trained on the source domain. The average performance is dropping from a near perfect classification score by over 20 % on unseen patient data in the target domain (red dotted line). The insertion of the trained CL is expected to yield performance gains compared to the untrained baseline. The channel-wise CL shows greatly deviating classification performance over different folds and positions. While some positions, e.g. after layer 3, indicate consistently good performance with the exception of a few outliers, other positions, e.g. after layer 4, are consistently underperforming. On the other hand, the inter-channel CL designs show consistently better F1 score in all inserted

positions with few exceptions. Especially, the position after layer 2 shows robust distributions close to the original F1 score of the baseline on the source domain.



(a) Insertion of one channel-wise CL after each layer



(b) Insertion of one inter-channel CL after each layer

Figure 4.6: CL performance based on correction layer position and type. The blue and red dotted lines indicate the average performance on source and target domain without CL training, respectively [JL10].

As a consequence, we observed that CLs with more trainable parameters and larger degree of freedom yield better generalization results. However, this also means that more parameters need to be learned and computations need to be performed.

We also investigated the sample efficiency of CL training. For this experiment the training setup of Fig. 4.5 is kept the same and reduced the number of training samples in the target domain per recording. A 2.99 $\times$ reduction of training samples results in a 1 % F1 score reduction. In an extreme case, the training samples are reduced by 120.48 $\times$ and the F1 score is still preserved within a 6 % range. In essence, the training of the CL can be achieved using few samples and, thus, can be performed during test-time.

4.4.2 CL Integration into an ECG Inference Engine

In the following, the complexity of the CL is evaluated in an ECG inference engine. Basically, the design from Section 3.2 is reused and extended by the CL. Three designs are implemented for comparison.

As a reference design, the WVCNN design remains unmodified. The second design extends the design by including more memory for parameters in the SRAM and the activation memory. The latter requires the insertion of SIPO registers at the corresponding layer and the fitted dimensions to that layer. In specific, the kernel dimension is 1, since the features of one time step are transformed into a new set of features with the same channel dimension. Further, the control logic needs to be modified from a purely convolution-based dataflow to realization of the matrix-vector multiplication of the CL transform. This is achieved by aligning the channel dimension on the vector MAC unit such that one row-by-column multiplication can be executed in an output stationary manner. Hence, the computations of one output element is loop tiled and executed in C/k cycles with C and k being the number of channels and number of elements in the vector MAC unit, respectively. The third design merges the parameters of the CL with an adjacent convolution layer, such that the weights and biases of the reference design are modified in one layer only (see Equations (3.1) to (3.5)).

	Train Acc. (%)	Test Acc. (%)	Area (μm^2)	Seq. Cells	Comb. Cells	Max. Freq. (MHz)	Power (μW)
CL Design	98.98	97.16	19.9k	19.7k	6.9k	9.96	267
CL Design (integr.)			19.3k	19.4k	6.6k	9.97	255
Reference Design	99.01	95.15					

Table 4.2: Post-synthesis results of reference ECG accelerator with and without integrated CL [JL10].

Table 4.2 shows the KPI of the three designs after synthesis. The CL design improves classification on the patients in the test set by approx. 2 %. The circuit area is increased by the additionally synthesized cells for the SIPO registers in the additional layer and the control overhead. This area overhead amounts to approx. 3 % of the reference design. Further, the power consumption increases by only 12 µW, i.e. 4.7 %. All in all, dedicated CL logic amounts to little area and power overhead and, thus, can be integrated into existing an accelerator to provide generalization capability without modification of the reference model. In the CL design, which integrates the layer into existing DNN layers, the same logic from the reference architecture can be reused for inference. Hence, the number

of cells and the area is identical to the reference. In terms of the power of the inference, the difference in switching activity resulting from the changed weight in the merged layer is negligible. In the end, the integrated CL design provides generalization capabilities without any hardware overhead by weight reconfiguration of one layer.

4.4.3 CL Evaluation for Backpropagation On-Chip

The concept of CL further reduces training complexity for the target domain to a single layer. Therefore, we demonstrate the savings in terms of memory consumption and MAC operations compared to full DNN fine-tuning. In the reference case, the whole DNN is fine-tuned as common in state-of-the-art ECG DG algorithms (see Section 4.2). Typically, hardware accelerators for on-device DNN training feature the training of all layers [119]. For the training of a single layer, the backpropagation algorithm requires error signals propagated from the final output layer. This error signal is used to calculate gradients and weight deltas to update the weight. Since only one layer is updated in the CL method, the calculation of intermediate results can be reduced by pruning gradient calculation at other layers. Further, the error signal does not need to be propagated beyond the CL.

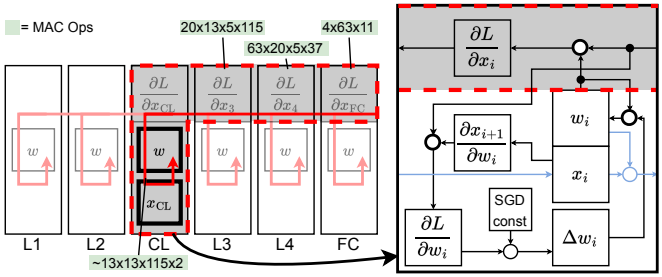
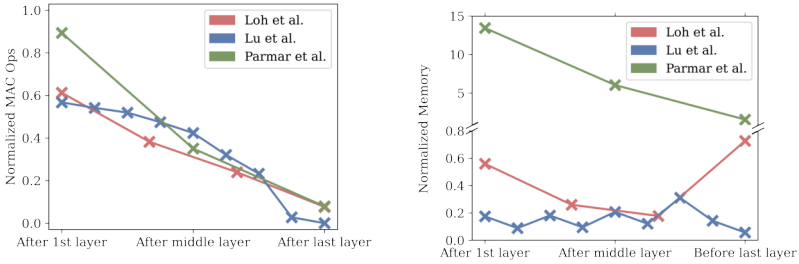


Figure 4.7: Theoretical concept of CL training. In contrast to DNN fine-tuning, the training of CL only requires partial memory resources (bold solid box) and a fraction of computations (grey area in red dashed line) [JL10].

Figure 4.7 visualizes the described mechanism. The chain rule of the backpropagation algorithm results in partial derivatives, which are reused in each layer for gradient calculation, i.e. indicated as $\frac{\partial L}{\partial x_i}$. For convolution operations, the partial derivative with respect to one input is the corresponding weight. The gradient $\frac{\partial L}{\partial w_i}$ is calculated with a layer specific term $\frac{\partial x_{i+1}}{\partial w_i}$ to eventually derive Δw_i for the weight update. In this evaluation, the number of MAC operations are used to estimate the computational complexity of the "reduced" backpropagation algorithm. For simplification, multiplication

without an addition are also counted as one MAC operation, as we assume that the cost of a multiplication is much larger than an addition. The bold circles on the right of Fig. 4.7 show the calculation considered in the backpropagation algorithm. Other calculations involved in the stochastic gradient descent (SGD) algorithm are neglected, since many variants and extensions exist. Nevertheless, the evaluation accounts for a majority of the calculations in the learning algorithm and indicates a lower bound of its complexity. Regarding the memory, the weights and activations need to be stored to calculate the weight updates. For backpropagation, the intermediate activations of the inference pass are required for the partial derivative $\frac{\partial x_{i+1}}{\partial w_i}$. As the update is only performed in one layer, the stored activations in the CL method are reduced significantly. In the evaluation, it is assumed that the reference case does not contain a CL and the investigated designs with CL introduce an additional layer with additional weights and activations corresponding to the inter-channel CL.



(a) Normalized MAC operations for different CL position (left:input, right: output)

(b) Normalized memory for different CL position (left:input, right: output)

Figure 4.8: estimated MAC operations and memory necessary for CL training based on different DNN architectures and CL position. The costs are normalized against the reference case, in which the DNN is fine-tuned [JL10].

Figure 4.8 shows the normalized results in different DNN architectures common for classifiers utilized in state-of-the-art ECG accelerators [43], [57]. The designs are normalized against the reference case, i.e. the whole DNN is fine-tuned. It is evident from all designs that the CL method is both superior in terms of computational complexity and memory consumption except special cases. In Fig. 4.8a all design points show less MAC operations than the reference. The number of operations also depend on the position of the CL. If the CL is positioned close to the output, less partial derivatives need to be computed for the update of the CL's parameters. If it is positioned close to the input, the computation of the chain rule needs to traverse through all layers of the DNN. Nevertheless, MAC operations are still reduced, since the gradient computations of other layers are omitted. Compared to

those, the computations of the additional CL is negligible. Figure 4.8b shows less consumed memory compared to the reference in the CNN architectures by Lu et al. and Loh et al. [JL7], [43]. One exception is the architecture of Parmar et al. [57]. In principle, the omitted activations from the forward pass should outweigh the number of additional parameters required for the CL. However, in the case of Parmar et al. a fully-connected DNN was used with few layers. The total number of activations do not exceed the number of additional weights required to store the CL's trainable parameters. Especially, the inter-channel CL requires the squared number of output channel for the weights and an additional set of activations for the added layer.

In the end, the evaluation assumes an additional dedicated CL. As previously described, the CL is arithmetically integrable into adjacent convolution and dense layers. For the in-field training of the CL, a method could be derived to leverage this integration further, e.g. through single layer training or on-the-fly weight updates.

4.5 Summary

Within this chapter, an ECG classifier is extended to feature domain generalization capabilities through an additional layer, i.e. correction layer (CL). Inspired from low complexity generalization methods, the CL aligns the features of the source and target domain in a separate layer, while preserving the original DNN architecture. Since the correction layer uses linear transformations for alignment the layer can be merged into adjacent convolution or dense layers from the DNN. Cross-validation over source and target domain is performed to validate the performance of the correction layer. A center position of the inter-channel CL shows the best classification performance. An evaluation on an ECG inference engine, that the area and power overhead to integrate an additional CL is low. If the CL parameters are merged with adjacent layers, area and power remain the same as the reference design without domain generalization. The training of CL on-device show significant CC and memory reductions compared to conventional fine-tuning. For the investigated example in this work, the number of MAC operations and memory are reduced by more than 50 %.

In terms of robustness across domain variations, algorithm design feature generalization capabilities, which are supported by dedicated hardware components to a limited degree. For instance, ECG accelerators provide the support for DNNs, in which the weights and biases can be reconfigured [29]. The in-field training of such modules has only been demonstrated for single layers [120], in which the discussed CL method provides additional flexibility in terms of layer position. It is shown that the

re-training of a single layer is less complex than DNN fine-tuning by design. The accelerators incorporating on-chip circuits for re-training [119] would provide additional features into state-of-the-art reconfigurable ECG accelerators with low overhead.

Chapter 5

Temporal Coding for Numerically-Equivalent Conversion of ANNs to SNNs

Spiking neural networks (SNNs) integrate one bio-plausible feature into the computations of conventional DNNs. The spiking communication promises fast inference and low power consumption due to the event-driven processing of spikes within the neurons [121]. The concept relies on the exchange of information with temporal events, i.e. spikes, which are transferred in a network of mainly uniform computing nodes interconnected through synapses.

In the following, we investigate how SNNs provide benefit to the classification of ECG in dedicated hardware components. First, we outline the landscape of SNN algorithm design and existing integrated circuits for SNNs. Then, one conversion method is introduced to map activations to a temporally encoded representation. The main operation of an ANN, i.e. the sum of products, can be calculated with less additions than classical methods. In the use case of ECG monitoring, the model from Section 3.1 is mapped to its spiking counterpart and the activations are analyzed for efficient hardware mapping. In the end, a hardware architecture is designed to infer the converted SNN. The evaluation shows that this mapping fully preserves the classification quality of the reference ANN and achieves complexity reductions at the same time.

5.1 Application Context of SNN Algorithm and Hardware Design

The concept of SNNs is used in different contexts both from the perspective of algorithm development and supporting hardware. Figure 5.1 shows an overview of the corresponding design concepts

for SNNs and how the target application influences the design choices of SNN neuron models and supporting hardware modules. These are further detailed below.

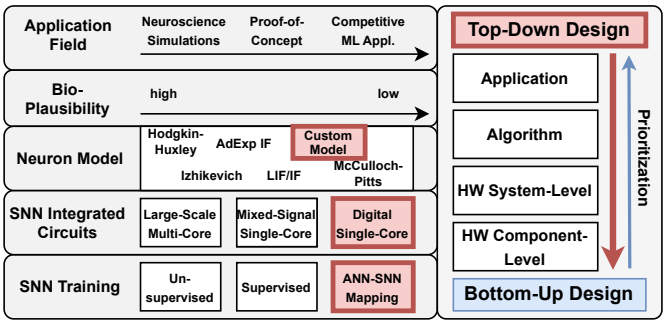


Figure 5.1: Simplified overview of SNN design concepts in context of custom neuron model proposed in [JL4].

The common denominator of SNN is the communication with spikes, which can be abstracted as binary signaling events with spatio-temporal features encoding their information. Spiking communication is a feature, which is not only used in neuroscientific simulations of the brain, e.g. the microcircuit model [122], [123], but also in machine learning applications. In the former, network architectures are built in close relation to the experimental evidence, which map brain structures and electrophysiological properties to a mathematical model with analogous behavior. In the latter, artificial network architectures are designed to solve e.g. classification tasks [JL4], [120]. This typically involves an optimization process to adapt the internal state of the model to produce a desired input/output behavior corresponding to training data or similarity metrics. The used neuron models, i.e. the elementary computing units of the network, mainly deviate based on the granularity of bio-plausible features contained in them. For instance, the Hodgkin-Huxley [124] model can mimic the dynamics of an action potential using conductances of ion channels. While this may prove useful in the replication of neuron-level simulations, the complexity of the computations limits its use in large scale networks. In ML applications the spatio-temporal correlation of spikes is used as input and feature representations in e.g. classification tasks. In this case, the timing of spikes and their integration in an internal state is more important than the dynamics of the action potential[1]. The variants of the integrate-and-fire models represent the spikes in binary signaling events, while the temporal progression of the internal state can incorporate leak terms or adaptive thresholds. In terms

[1]The width of the action potential is suggested to influence synaptic plasticity [125].

of hardware, the circuits supporting those models can range from large-scale multi-core to digital single-core systems. Again, the application requirements determine the complexity of the system, in which power-performance-area (PPA) trade-offs might favor the one or the other design paradigm. For instance, large-scale multi-core systems are suited for neuroscience simulations as they require scalability. On the other hand digital single core architectures are favorable for mobile ML applications as sparse activations in a small footprint circuit can meet demands for low power etc. In the end, the deployed SNN need to be trained before their usage. While some methods follow traditional un-/supervised training methods with adjustments to the non-linearity of some neuron models, there are also conversion methods, which transfer the parameters of a baseline ANN into a spiking counterpart.

In a task-oriented application, such as ECG classification, the design is mainly guided by application requirements. For instance, real-time processing is constrained by input rate etc. (see Section 2.3.3). The processing algorithm and executing hardware is designed such that it adheres to constraints set by the application down to the component level hardware design, i.e. top-down design approach. The bottom-up approach significantly differs from the top-down approach, as the design starts from the component level design and extends from there to the system level. An example, in this case, is the design of a cluster to emulate IF neurons on a large scale. In this example, the design of a neuron core is extended with network structures for communication etc.

In the case of health monitoring, the exact processing mechanism is not pre-defined and remains flexible as long as it achieves its functionality under certain quality metrics. Therefore, a top-down approach is chosen for the remainder of this Section to design a neuro-inspired accelerator. Further, the requirements on bio-plausibility are flexible, as the target is to optimize against the quality metrics. In the case of ECG monitoring, it is desirable to classify artifacts with high quality and low power. In the context of former, state-of-the-art ANN already achieve high quality with low power in a custom digital circuit (see Section 3). Hence, the aim is to integrate biologically-inspired features into existing artificial neural network to further reduce the power consumption. The method of ANN-SNN conversion fits perfectly for this goal, since it aims to mimic the reference ANN in the conversion scheme. Therefore, the activations are closely matching the reference and, in the ideal case, provide the same classifications with sparse communication and computations.

5.2 State-of-the-art ANN-SNN Conversion

The usage of spikes as communication signals is fundamental to SNNs. The expectation is the communication between computing nodes is sparsely represented through the temporal and binary nature of the spikes. The sparsity should result in less circuit activity, which in turn reduces the dynamic power of the system for further power reductions. Neural networks with spiking neurons cannot be trained conventionally with gradient descent algorithms, since they need differentiable activation functions. Hence, a variety of methods are introduced to train SNNs [121]. While some concepts adapt existing training techniques to directly train the SNN, e.g. through spike probabilities [126], others leverage baseline ANNs for conversion into the spiking domain. In an ideal scenario, the conversion process from activations with real numeric values to spike representations does not alter the encoded numeric values, such that both models are equivalent in their input-output response. The ANN-SNN conversion method is used in this section to include the feature of spiking communication into health monitoring systems for further optimization.

The basis of the conversion method is the encoding of real-valued activations in the ANN to spikes. A series of spikes contain the information in binary signaling events with temporal properties corresponding to the reference. Although recent literature discovered many methods of spike encoding schemes(see review [12]), popular methods mainly focus on two subcategories: rate and temporal coding.

The most common method is the rate coding of the activations [127]–[131]. In essence, the numeric value of the ANN neuron outputs are represented in the spike rate. This coding method is widely used, since IF neurons can be used to generate the spikes with a certain rate proportional to the spike rates of the neuron's input channel. This neuron model is highly popular due to its simple phenomenological structure [132]. The idea is that the accumulated input signals are increasing the internal state of the neuron, i.e. membrane voltage, until it exceeds a threshold. By adjusting the neurons parameters, e.g. its threshold, and the synaptic weights, the spike generation can be influenced. For a constant input, the basic IF neuron will generate a spike train with constant spike rate.

However, it is also suggested that SNNs require a very low spiking activity to justify an advantage energetically compared to their ANN counterpart [133]. SNN with spike-rate encoding require many spikes to represent their numeric value, especially, when this value is high. An alternative are temporal encoding methods, which encode information in the timing of their spikes [12]. The advantage of

temporal encoding schemes are the sparsity of used spikes. In specific, the time-to-first-spike (TTFS) encoding schemes use only one spike to represent the numeric value in the spike timing in relation to a global reference [134], [135].

Rueckauer et al. introduced a temporal coding scheme to represent the activations in the spike timing of the first spike [134]. The neuron models generating the spike have long refractory periods, such that all other input after the spike is ignored. This guarantees the sparsity of the output, but also looses information content of the input in the refractory period. The timing of the output spike t_i is designed to follow an inversely proportional relationship to the ANN activation a_i, i.e. $t_i = 1/a_i$. The classification quality of the SNN with their temporal encoding relies on the precision, in which this relationship is actually preserved. Since spikes are lost during the refractory period of the neuron model, accuracy degradations are expected in the converted model. Park et al. use IF neurons, which perform spike encoding and decoding in two separate phases. In the decoding phase, input spikes of a neuron are integrated into a membrane potential. In the encoding phase, the membrane potential is encoded into an output spike. The split into two phases addresses the problem of neglected input spikes. In their work, there is an error between the value before encoding and the value restored after decoding. The error results from the chosen time constant in the kernel for dynamic threshold used for spike encoding.

In the discussed examples, the temporal encoding is, in general, affected by a conversion error. The error can be backtracked to neuron model discrepancies of the reference ANN and the converted SNN. As the intention of the spike encoding is to reduce the complexity of the ANN computation, it seems natural that the encoding scheme is approximate. The expectation is that the reference ANN can tolerate small approximation errors resulting from temporal encoding without deteriorating the classification quality of the network model.

Assuming a sensitive reference ANN, conversion errors could result in many misclassifications. In the following, we investigate a concept, which can perform temporal spike encoding without precision loss and how it impacts the performance of dedicated hardware accelerators.

5.3 Lossless Temporal Coding for Fixed-Point Numbers

In the examples of Rueckauer et al. and Park et al., the models assume real values for the timing of spikes as well as the activations of the ANN reference. In the simulation framework, these values are typically approximated using floating-point precision, in which the exact computation is

performed within machine precision and variations in compilation settings. However, state-of-the-art co-designed ANNs already utilize quantization to reduce memory consumption and computational complexity [136]. Quantization is a well-investigated field, in which low precision can be achieved without quality loss. The fixed-point representation of the quantized ANN is used as the baseline for various novel computing principles such as computing arrays with memristive devices or time-domain computing [137]. Therefore, we investigate whether the fixed-point representation can be exploited to derive an equivalent SNN design. This concept, which we describe in more detail in the following sections is published in [JL4].

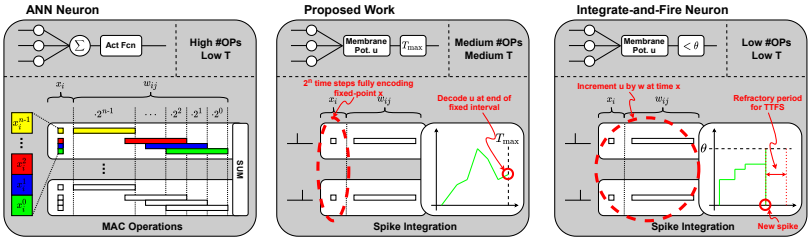


Figure 5.2: Concepts of the conventional ANN neuron (left), the IF neuron (right) and the neuron model with proposed temporal coding (center) [JL4].

Figure 5.2 shows the concepts of the ANN neuron, the IF neuron and the lossless neuron model. The IF neuron receives a binary input, which is multiplied with its weight and integrated in the membrane potential. Various variants of the IF model introduce additional features such as a leakage term [132] or adaptive thresholds [138]. Once the membrane potential exceeds a threshold, a spike is generated. In sparse temporal coding schemes, the number of output spikes typically limited to the timing of the first spike. Other subsequent spikes are prohibited by a long refractory period defined in the neuron model.

In contrast, the ANN neuron has activations and weights with limited precision, which need to be multiplied and summed. The result of the scalar product between activation and weight vector is fed through an activation function to generate the output activation for the next layer. Hence, the main computation is the sum of products resulting from fixed-point multiplications.

The idea of the lossless neuron model is that the membrane potential can encode this sum of products in fixed-point without loss. In this case, the magnitude of the activation is represented in the timing of the corresponding spike. Since the number representation of the operands and the result is discrete and finite, the number of corresponding time steps is also discrete and finite. Hence, the

calculation of multiplications and additions can be performed sequentially, analogous to sequential addition in classical computer arithmetic.

In the following, we derive a temporal coding scheme analytically. This coding scheme enables a temporal representation of fixed-point values and an alternative way to calculate the sum of products with fixed-point values. Since this coding method provides fully equivalent numerical values to its reference, the resulting SNN neuron model is lossless compared to its ANN counterpart.

Let's consider a scalar product of two vectors with N elements

$$y = \sum_{i}^{N} x_i \cdot w_i. \tag{5.1}$$

Assuming fixed-point quantized activations x_i and weights w_i, it can be described as

$$y = 2^{-2f} \cdot \sum_{i} x_{\text{int},i} \cdot w_{\text{int},i} \tag{5.2}$$

with $x_{\text{int},i}, w_{\text{int},i} \in \{0, 2^n - 1\} \subset \mathbb{N}_0$. In this case, f and n describe the number of fractional bits and word length of the fixed-point number, respectively. The sum and multiplication of two integers $x_{\text{int},i}$ and $w_{\text{int},i}$ can be described as the sum of partial products resulting in

$$N_{\text{ADD,ref}} = n \cdot N \tag{5.3}$$

additions. $N_{\text{ADD,ref}}$ denotes the number of additions required for the sum of N fixed-point products without temporal encoding.

On the other hand, the integer value $x_{\text{int},i}$ can be represented as a binary signal $x_{\text{bin},i}(t)$ with $T = 2^n$ timesteps. The binary signal $x_{\text{bin},i}(t)$ is 1 for $x_{\text{int},i}$ timesteps and 0 for the other. It is notable, that the area under this binary signal corresponds to the integer value. Assuming that $x_{\text{bin},i}(t)$ is defined for $0 \leq t < T$ with $t \in \mathbb{N}_0$, then

$$x_{\text{int},i} = \sum_{\tau=0}^{T-1} x_{\text{bin},i}(t). \tag{5.4}$$

When $x_{\text{bin},i}(t)$ is represented as a Heaviside step function, $t_{\text{int},i}$ is the timestep, in which the value of $x_{\text{bin},i}(t)$ turns from 0 to 1. For a defined timeframe $0 \leq t < T$, this timestep is defined as

$$t_{\text{int},i} = (T-1) - x_{\text{int},i}. \tag{5.5}$$

Looking at the binary representation of $x_{\text{int},i}$, the timestep $t_{\text{int},i}$ can be obtained by inverting all bits, i.e. from 0 to 1 and vice versa.

The multiplication of one element $x_{\text{int},i}$ with $w_{\text{int},i}$ could be represented as the double integration of a spike signal at $t_{\text{int},i}$ weighted with its corresponding weight. The weighted spike function is defined as

$$w_{\text{inc},i}(\tau) = \begin{cases} w_{\text{int},i} & \tau = t_{\text{int},i} \\ 0 & \text{otherwise} \end{cases}. \tag{5.6}$$

The result can be visualized as the area of a rectangle with the height of the weight and the length of the activation. The area of the rectangle is accumulated over time in a state variable, i.e. the membrane potential $u_{\text{int}}(t)$, by adding an offset (the numerical value of the weight) to the state. The offset is updated once at the time of the encoded activation. The advantage of this representation is in the superposition of multiple multiplications in an increment over time. The number of additions can be summarized to

$$N_{\text{ADD,encoded}} = 2^n + N. \tag{5.7}$$

The first term describes the number of additions in each time step to increment the membrane potential. The second term describes the number of updates of the increment, which is the number of elements in the vector of the scalar product.

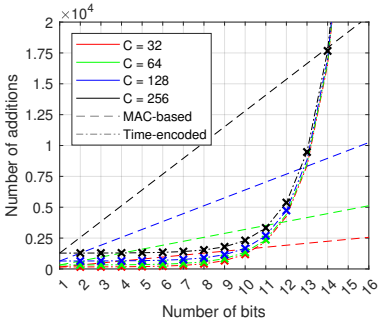


Figure 5.3: Number of additions for a 1D-convolution with kernel size $k = 5$ and C input channels [JL4].

Figure 5.3 shows the number of required additions as function of word length for different kernel sizes. A 1D-convolution with a weight kernel and a bias term is compared for different channel sizes. The computations in the 1D-convolution correspond to a scalar product with $N = k \cdot C$ elements. The

diagram shows a linear and an exponential function for the MAC-based and time-encoded convolution, respectively. For higher number of bits n, e.g. greater than 16, the trend obviously prefers the conventional multiplication and additions of partial products. However, the time-encoded method has less additions in the lower range of word lengths. The break-even point is a function of the number of elements in the scalar product and is determined by solving

$$\frac{2^n}{n} = N - 1 \tag{5.8}$$

with a constant number of elements N to be multiplied. Convolution and dense layers in popular DNNs are quantized to 8 bit and even lower [86]. Hence, the temporal encoding scheme remains feasible for modern DNN architectures.

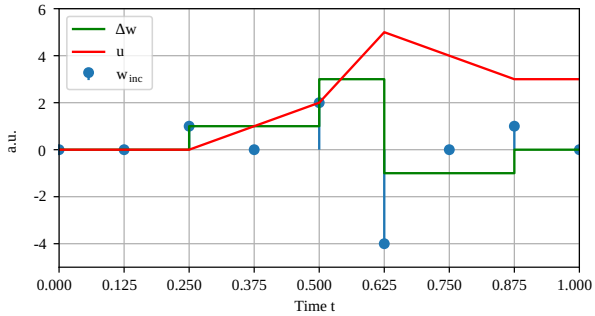


Figure 5.4: Example encoding of a scalar product with four elements and normalized time axes [JL4].

Figure 5.4 shows the membrane potential of a scalar product with four elements changing over time. The blue graph indicates the superpositioned weighted spikes w_{inc}, which are integrated to the weight increment Δw. The step function Δw is the momentary change in the membrane potential u, which results from another integration of Δw. The membrane potential is a piecewise linear function, in which the last value in temporal range represents the result of the scalar product.

$$u_{\text{int}}(t = T - 1) = \sum_i x_{\text{int},i} \cdot w_{\text{int},i} \tag{5.9}$$

Note that the encoding method is shown for the multiplication and addition of integer valued elements. Fractional bits of fixed-point numbers can be accounted for with static shift operations. In the example of Fig. 5.4 the time axis is normalized to the region between 0 and 1, which will be explained in

the next section. For a 3 bit value, the number of discrete time steps is 8, in which the membrane voltage u can change. In the example, the scalar product of activations $\hat{\mathbf{x}} = [1,2,-4,1]^T$ and weights $\hat{\mathbf{w}} = [1,2,-4,1]^T$ is calculated. The weight increment Δw changes four times corresponding to the number of elements in the scalar product. Since small activations now correspond to large latencies after encoding, it is expected that the membrane voltage is changed less in the initial time steps than the last time steps (see Fig. 5.6)

The shown encoding method is conceptually applicable to most quantized DNNs and theoretically promises less additions. Hence, the computational complexity is reduced in the proposed ANN-SNN encoding scheme with identical numeric computations. In the following, this assumption is validated in the HW design of the ECG classifier from Section 3.1.

5.4 Algorithm Design of a Temporally Encoded ECG Classifier

Since the previous encoding method is designed for arbitrary sum of products, the application on ANN layers is straightforward. Especially, CNNs consist of primarily convolution and fully connected layers. Therefore the mapping of the ECG classifier, i.e. the model from Section 3.1, mainly requires the transformation of activations to temporal signals and the computation as described in the previous section. The result before the activation function of the layer is represented in fixed-point. Thus, subsequent components such as activation function and pooling remain unchanged from the WVCNN design in Section 3.2.

Nevertheless, the conversion of the WVCNN model into its spiking counterpart - further denoted as the WVSNN design - requires preparatory steps, such that the converted model can be mapped to dedicated hardware components. In the following, the normalization process for the activations are discussed and its impact on the classifier quality. Then, the activations of the WVSNN design are analyzed to identify properties in the activation distribution, which can be leveraged in the hardware component design of the next section. Further, the impact of the resolution is discussed as a trade-off between latency and classification quality.

5.4.1 Activation Normalization of the Reference Classifier

The normalization of the activations serves the purpose of representing the activations in the same numerical range. After conversion the timesteps representing the activation values are in a predetermined range. The model is normalized with the following three steps:

1. Integration of batch normalization layers
2. Normalization of DNN activations
3. DNN quantization

The first and last step follow the same principles from Section 3.1. In the second step, the activations are normalized into the same numerical range - typically [0,1] [128], [134]. The final step quantizes the activations within this range and determines the resolution of the representable numbers. Since the second step is unique for the conversion, it is discussed further within this subsection.

The normalized range is used to encode the activation vector $\mathbf{x}$ into spike latencies $\mathbf{t} = \mathbf{1} - \mathbf{x}$ in the same range throughout the whole network. The normalization is performed through linear scaling using parameters δ for offset and λ for scaling. Here, the scaling factors are determined by the statistics in the activation distributions, in specific the maximum and minimum of the input. Naturally, the boundaries can be chosen based on percentile criteria to suppress outliers etc. The scaling factors of the input layer are defined, such that $\mathbf{x}_{\text{norm}} = \lambda \cdot \mathbf{x} + \delta$, resulting in

$$\lambda^0 = \frac{1}{x^0_{\max} - x^0_{\min}} \tag{5.10}$$

$$\delta^0 = -x^0_{\min} \cdot \lambda^0 \tag{5.11}$$

The superscript denotes the output of the zeroth layer of the DNN and, hence, the input of the DNN. The scaling factors for the outputs of layer l are defined as:

$$\lambda^l = \frac{1}{x^l_{\max}} \tag{5.12}$$

$$\delta^l = 0 \tag{5.13}$$

This normalization accounts for non-positive input to the network, which are normalized to a positive only range for encoding. The scaling within the network scales only with the positive boundary for the activations to preserve the alignment to the rectified linear unit (ReLU) activation function. In the end, λ^l and δ^l are applied to the weights and biases resulting in new weights and biases.

$$\mathbf{w}^l \rightarrow \mathbf{w}^l \cdot \frac{\lambda^l}{\lambda^{l-1}} \tag{5.14}$$

$$\mathbf{b}^l \rightarrow (\mathbf{b}^l - \mathbf{w}^l \cdot \frac{\delta^{l-1}}{\lambda^{l-1}}) \cdot \lambda^l + \delta^l \tag{5.15}$$

Note that $\delta^l = 0$ for $l > 0$. Therefore, this term can be omitted. Further, the term $\mathbf{w}^l \cdot \frac{\delta^{l-1}}{\lambda^{l-1}}$ is also only non-zero for the first layer. Hence, the normalization for layers $l > 1$ only needs a single factor multiplied with the weights and biases.

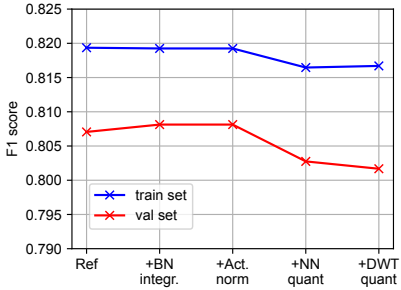


Figure 5.5: Ablation study on the normalization steps as preparation for the temporal encoding scheme [JL4].

Figure 5.5 shows an ablation study for the normalization. In this study the floating-point reference is consecutively normalized for conversion. First, the batch normalization layer are integrated and the activations are normalized. These steps did not result in any QoS differences, except a few samples even classified better in the validation set. This could be explained by numerical imprecisions from the floating-point number representation. Then, the activations for the pre-processing and the classifier are quantized to 12 bit. A F1-score drop of less than a percent is observed due to the introduced quantization error.

5.4.2 Distribution of Spike Timings After Normalization

After normalization, the activations of the reference ANN is mapped linearly to the desired range between 0 and 1. The corresponding timings of the normalized activation is expected to show a non-uniform probability distribution over the whole temporal range. Therefore, we inspect the histogram of spike timings over a random set of ECG samples in the dataset.

Figure 5.6 shows the histograms of spike timings after normalization. The probability is shown on the y-axis and encoded latency is shown on the x-axis through the number of timesteps used. In this example, the activations are encoded using 12 bits resulting in 4096 timesteps. The input activations from the DWT pre-processing are normalized with an additional offset according to Eq. (5.10). The corresponding histogram shows a cluster around the central timesteps. Since most activations before the normalization are clustered around zero, the linear offset now centers them around the offset.

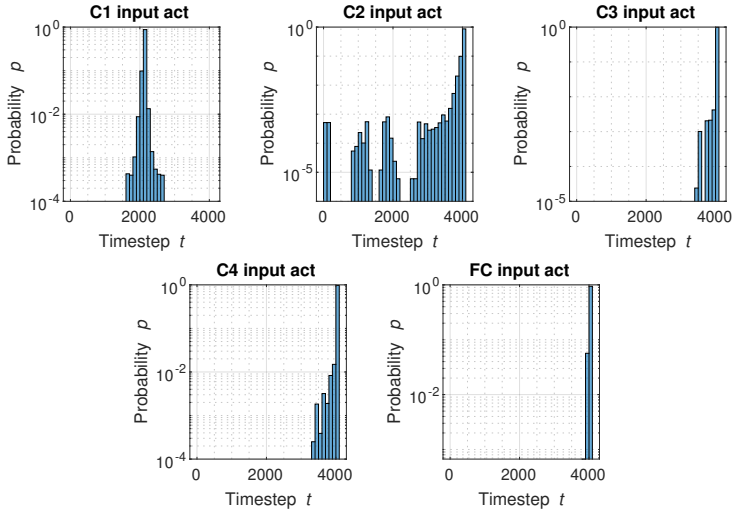


Figure 5.6: Histogram of spike timings after normalization for the inference of 86 ECG samples. Each ECG sample results in multiple computation sequences per layer and neuron, as multiple frames per ECG sample are selected using the sliding window approach.

As the normalization used the maximum values of the activation distribution to calculate the upper and lower bound, a large degree of the temporal range remains unused. In contrast, the second layer receives the output of the first layer and shows more activations spread across the whole time axis. Nevertheless, most activations are found around the last timesteps. The histogram resembles an exponential distribution with a monotonically increasing distribution from the first timestep towards its maximum at the last timestep. A similar trend is seen in the subsequent layers. However, the clustering around the last timesteps is more pronounced the deeper the layer is. In the subset, which is used for analysis, there are actually no samples in the first 75 % of the time period within layers C3 to FC layer.

This highly skewed distribution can be exploited on the circuit level of the hardware design. For instance, zero skipping circuits are often utilized to omit weight kernels or kernel elements, which are known to produce partial sums with value zero [103]. The same principle applies for the accumulation cycles in the temporal encoding scheme. The “late-start” logic described in Section 5.5 is used to skip those initial cycles. The impact of this logic on system power consumption is evaluated below in Section 5.5.4.

5.4.3 Toggle Activity of the Membrane Potential

As described in Section 5.3, the input spikes trigger a change in the increment of the membrane potential, which is accumulated sequentially over the entire time period. Therefore, it is expected that a major part of circuit activity is in the sequential adder used for accumulation.

Since the activity inside the sequential adder depends on both the weight, the input spike and its timing, conventional ML frameworks do not provide the functionality to simulate cycle and bit accurate behavior for this SNN neuron model. Hence, a software emulation of this SNN neuron model is implemented in MATLAB to analyze the activity before the design of a component in a HDL. Analogous to the analysis in the previous section, possible properties of the activity distribution can be exploited in the hardware design phase.

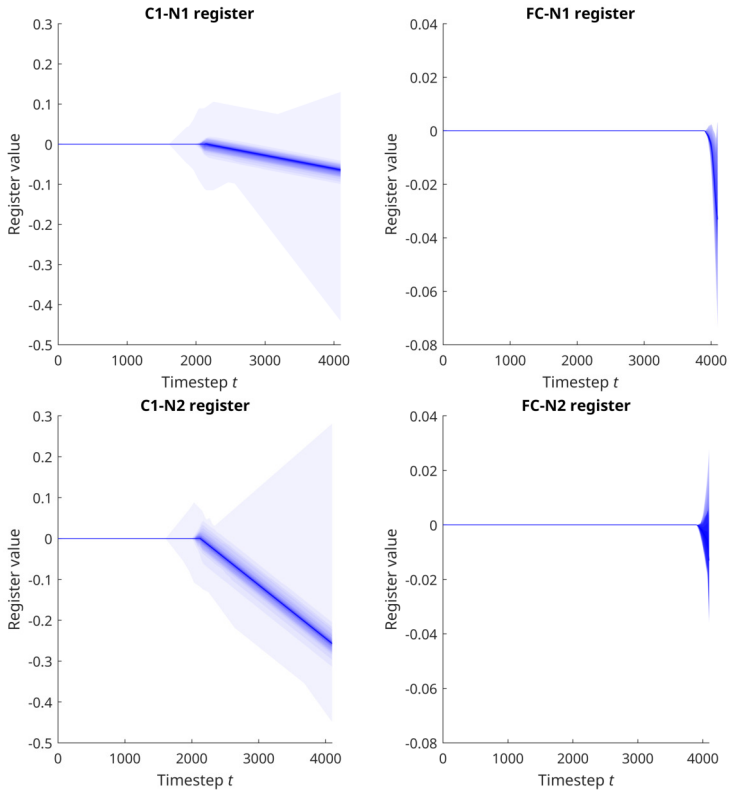


Figure 5.7: Probability distribution over time of the activation value in the accumulation register of the WVSNN design. A strong color depth indicates a high probability.

The challenges in this software implementation include the accurate mapping of rounding, saturation and timing of each computation step. For instance, the rounding scheme is dependent on the used rounding mode in the software implementation, which deviates across implementations especially for the handling of negative number. This needs to be adapted to the RTL implementation, which is performing round-to-nearest with the edge cases being strictly rounded up. Further, the timing of quantization steps and overflows need to be accounted in the simulation implementation. Especially, the execution order is critical in the validation of the stream-based processing design. For an identical behavior of the software simulation and the hardware the timing of the subsampling need to be aligned. For instance, for an equidistant subsampling of factor 2 there are two possible values, which can be selected for processing. In the WVSNN design, there are $2^4 \cdot 3^4 = 1296$ possible permutation of starting conditions, i.e. 2 for each DWT layer and 3 for each pooling layer. In the software simulation, this is manually set in parametrized configurations to match the RTL simulation.

Figure 5.7 shows the simulation of the membrane potential for two exemplary output neurons in the first convolution layer and the final dense layer. The activation values are sampled over all sliding windows over a 60 second ECG trace. The mean of the final value as well as the trend of the distribution over time is neuron specific. There is a tendency for the first convolution layer to feature a wider spread, i.e. approx. [0.3, -0.42], than the final dense layer, i.e. approx. [0.03, -0.07]. The variance of the distribution is also wider spread in the first layers than the final dense layer.

Figure 5.8 shows the activity of each bit in detail. The activations and weights are assumed to be quantized to 12 bits. The size of 24 bit is chosen as the word length of the accumulation register in correspondence to the ANN reference implementation. As mentioned before, the preliminary cycles do not show any activity at all, since no activations are encoded in earlier latencies for the example ECG trace. For the bits with low significance, i.e. fractional bits, the bits are toggling about 50 % of the time. The MSBs of the register show a high number of 1s for the active cycles. This is also expected, since the majority of values is negative. The transitions seem also equally distributed with the exception of few hotspots in the MSB region. For comparison, the L1 norm of the register value is plotted in Fig. 5.9. The activity of the MSBs are lower, since no signs are flipped. One can observe the gradual increasing activity of higher significant bits with increasing timesteps, while the sign bit remains inactive. Hence, the toggling of MSBs is caused by the transition of negative values to positive values. A similar trend is expected for the other subsequent layers.

On the circuit level, the activity caused by flipping MSBs can be reduced by introducing arithmetic transformations to guarantee a calculation with positive numbers only. For instance, the increments

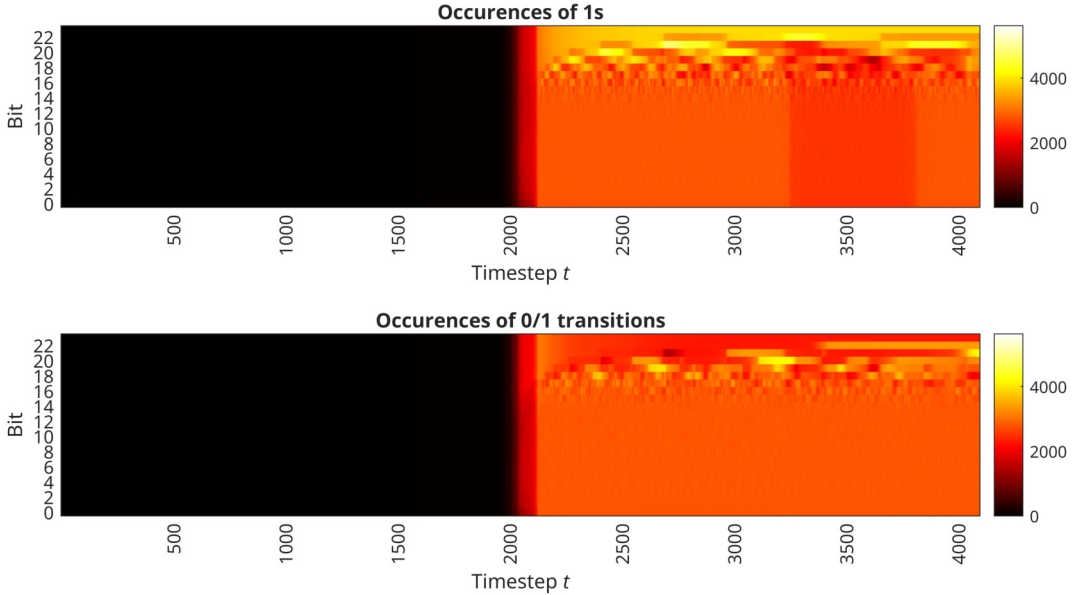


Figure 5.8: Bit-level statistics of the accumulation register of the WVSNN neuron for one 60 seconds ECG trace. The top diagram shows how many times the specific bit is 1 in a particular timestep. The bottom diagram shows the number of transitions from 0 to 1 and vice versa between two consecutive timesteps.

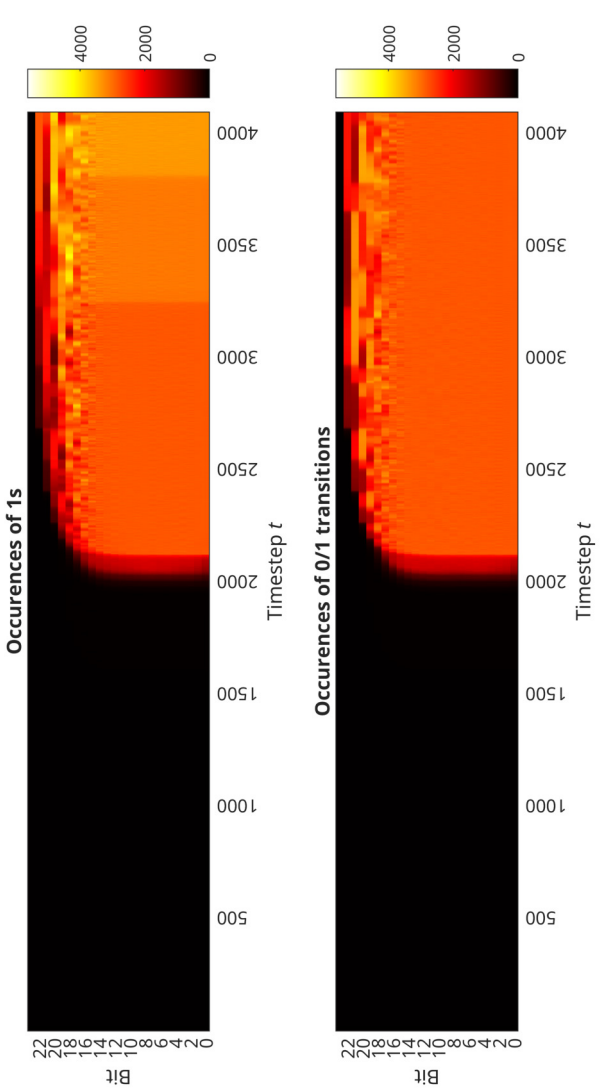


Figure 5.9: Bit-level statistics of the accumulation register of the WVSNN neuron after L1-norm. The top diagram shows how many times the specific bit is 1 in a particular timestep. The bottom diagram shows the number of transitions from 0 to 1 and vice versa between two consecutive timesteps.

can be shifted by a constant offset such that all values are positive. Since it is known how many timesteps are used for the accumulation of the membrane potential, the offset can be compensated with other offset at the end of the accumulation period. This is especially efficient, if the offsets are chosen as a power of two, since constant shifts do not require logic components.

However, the activity in the membrane potential is limited to the time period after the first received input spike. Looking at the temporal distribution (see Section 5.6), the impact of flipping MSB bits is constrained to the few cycles at the end. Especially for the layers apart from the first layer, the percentage of toggling MSBs over the entire time period of 4096 cycles is less than five percent. Therefore, we do not opt for a circuit level optimization for toggling MSBs for the chosen application and network model. However, the activity distribution is expected to deviate for different data and SNN models. Hence, this design choice needs to be reevaluated on a case by case basis.

5.4.4 Impact of Temporal Resolution on Latency and Classification Quality

The precision of the temporal encoding scheme is a function of the temporal resolution for computation. In a digital circuit with a fixed clock frequency, the number of cycles determine the latency of the inference. Since all layers are computed in sequence and the computation of each layer requires the encoding and decoding of the activation, the critical path can be approximated by the number of clock cycles from the input of the first layer to the output of the last layer, hence

$$N_{\text{cyc,crit}} = 2^n \cdot L \tag{5.16}$$

cycles with number of layers L and number of bits n. The resulting critical path $\tau_{\text{crit}} = N_{\text{cyc,crit}}/f_{\text{sys}}$ determines the bottleneck of the design in Section 5.5. In the application context of continuous processing (see Section 2), the SNN engine is subject to real-time constraints set by the input data rate. Therefore, there is a lower bound for the clock frequency f_{sys}, in which the design can be operated.

Figure 5.10 shows the trade-off between the quality of the SNN classification and the latency of one SNN layer. It is evident, that the temporal resolution is closely related to 2^n with additionally few cycles for peripheral computations, such as activation function, pooling, etc. Similarly to conventionally quantized ANNs, the QoS is increasing for higher resolutions and reaches a plateau starting around 12 bits. This is to be expected, as the temporal encoding scheme is intended to provide identical results to the fixed-point ANN reference. Further QoS improvements are possible, when the

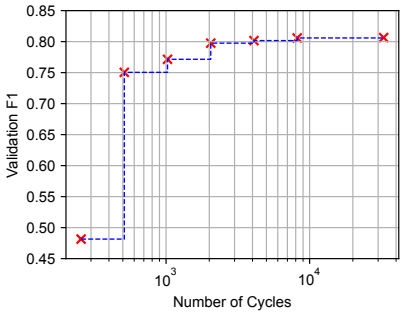


Figure 5.10: Pareto-optimal front for SNN quality and the number of cycles required to calculate one layer [JL4].

ANN reference is improved. This includes fine-tuning in the post-training quantization scheme and possibly quantization-aware training schemes.

5.5 Hardware Design of a Temporally Encoded ECG Classifier

As discussed in the previous section, the converted SNN model uses sparse activation representations for design efficiency. In the following, a hardware implementation is designed to support the encoded activations and corresponding computation in dedicated modules. The expectation is that the hardware KPI of both a reference implementation and the design with sparse temporal coding are directly comparable, since the numerical values stay identical.

5.5.1 System-level Hardware Mapping

An isomorphic hardware mapping for the system architecture is used for the framework of comparison. As a reminder, the temporally encoded design is denoted as “WVSNN”. Figure 5.11 depicts the block diagram for hardware modules from the input of the SNN classifier, i.e. encoded DWT output, to the prediction. Analogous to Section 3.2.1, the intermediate activations are stored in SIPO registers. However, the activations are fed into custom logic to realize the encoding and decoding according to Section 5.3. The logic is split into a compare module to generate the sparse representation of the activation and the processing element (PE) for the accumulation of the membrane potential. The encoding and decoding are executed in parallel and orchestrated by a control module. This control module contains a finite state machine (FSM) to track the executed layers and the time step of the temporal coding. The modules of the isomorphic design are subsequently enabled by the FSM based

on the data-driven dataflow as described in Section 3.2.2 with the exception of non-shared PE units. Hence, the activation registers and pooling units are triggered by modules adjacent in the processing pipeline of the SNN.

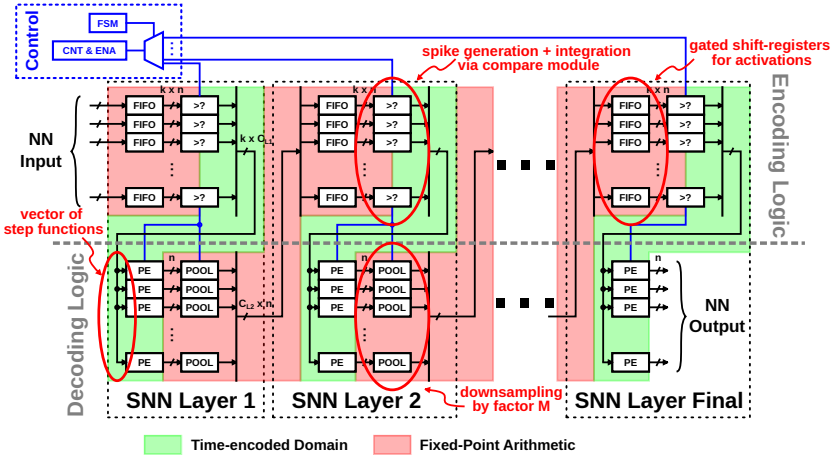


Figure 5.11: System-level architecture of the mapped SNN accelerator [JL4]. The convolution is performed using the time-encoding scheme and can be exchanged with conventional PEs with fixed-point MAC units for comparison.

The coloring in Fig. 5.11 indicate that the computations in the architecture are a combination of time-encoded and fixed-point arithmetic. Although previous works describe winner-take-all methods to perform the max-pooling operation in the temporal domain [139], the classic approach digital approach is selected for comparison. The target is to interchange the modules in the green region to quantify the effect of temporal encoding in the neuron computation.

5.5.2 Temporal Encoding Logic for Activations

Starting from the activation registers the encoding logic is the first major difference to previous design in Section 3.2. The inversion of the activation $\mathbf{x}$ and the first integration of the unweighted spike is combined into one module. Basically, a vector of Heaviside step functions with temporal offset t_i for every input channel i is generated through the comparison of a decrementing counter with the stored activation value. The counter is initialized with the maximum value of the representable number. Since the number range is normalized to $[0,1]$ all bits are used as fractional bits. In case of a 12 bit number, the initial value is $x_{\text{count,init}} = 1 - 2^{-12}$. Therefore, the logical compare function creates a

boolean signal as an output with

$$\int t_i(\tau)\mathrm{d}\tau = \begin{cases} 1 & x_{\text{count,init}} \geq x_i(\tau) \\ 0 & x_{\text{count,init}} < x_i(\tau) \end{cases}. \tag{5.17}$$

5.5.3 Temporal Decoding Logic in PE Unit

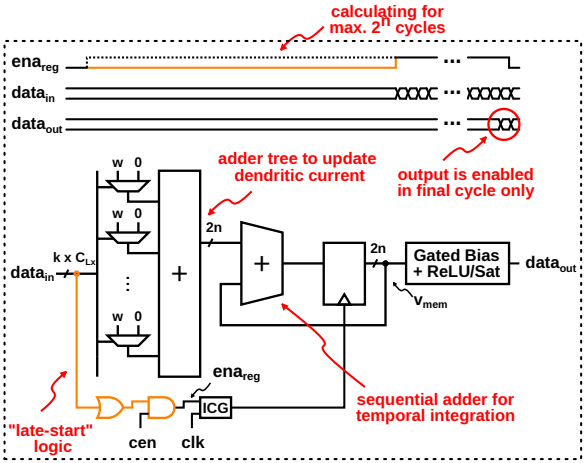


Figure 5.12: Block diagram of PE unit for the time-encoded convolution [JL4]. The encoded signal is inserted for the sequential decoding within the membrane potential.

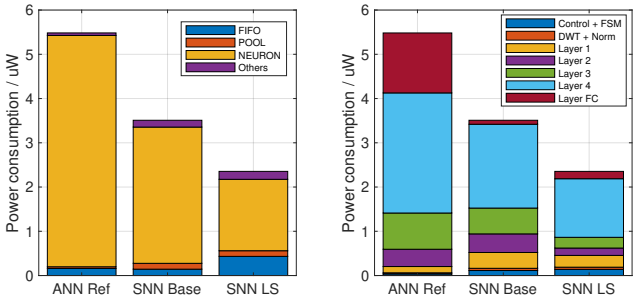
The output of the compare module is used as an enable signal for the PE module as shown in Fig. 5.12. A vector of constant weights is multiplexed into an adder tree to generate the weight increment Δw. The increment is used in a sequential adder for the second integration resulting in the membrane potential u. The result in the final time step is used in the activation function module to generate the output. The accumulation in the register is controlled by the external enable signal and is enabled for the whole duration of the temporal sequence, i.e. $2^{12} = 4096$ cycles. The "late-start" logic disables initial cycles of this time period based on the collective input through the integrated clock gating (ICG) cell. If all input channels are zero, it disables the clock from switching the clock pin at the accumulation register. Therefore, activation distribution with a tendency for small x_i, i.e. large t_i, can benefit from a reduced energy consumption in the inference process.

5.5.4 Design Evaluation

Analogous to Section 3.2.3, the design is evaluated in a 22 nm fdSOI CMOS technology node. The setup is kept under nominal conditions and the design is synthesized for simulation. The post-synthesis netlists are analyzed for comparison during a 60 second period, while the ECG trace is streamed in real-time into the design at 300 Hz. In this case, however, we investigate the active calculation period of the ANN reference design and the temporally encoded SNN designs. Three designs are compared: The reference design "ANN Ref", the WVSNN design without and with "late-start" logic, i.e. "SNN Base" and "SNN LS", respectively. The reference design implements the calculation of the scalar product using a conventional MAC unit. This unit receives fixed-point activations as input and returns the convolved result as an output within one cycle. In all designs, the weight memory is configured during the design phase as parameters in the HDL description and is optimized by synthesis.

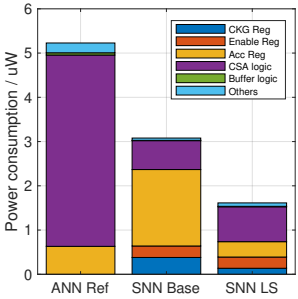
Figure 5.13 shows the power breakdown of the WVSNN design. From Fig. 5.13a, it is evident that the PE unit, i.e. the neuron logic, consumes up to 95.3 % of total power. In the layer breakdown (see Fig. 5.13b), major savings are visible in the WVSNN design mainly in the dense layer and the third/fourth convolution layer. Other layers remained similar or even consume more resources than the ANN reference. Due to the trade-off shown in Fig. 5.3, the temporal encoding is expected to provide less additions for a specific resolution of the activation in relation to the number of elements of the scalar product. For the chosen resolution of 12 bits, the break-even point between the conventional MAC unit and the temporally encoded neuron depends on the kernel size and the number of channels. According to Eq. (5.8), the number of elements N is the product of kernel size $k = 5$ and number of channels C. If the equation is solved for C with $n = 12$, the break-even point is reached with $C_{\text{break-even}} \approx 64$. In the hardware simulation, however, we observe that the break-even point is in between $C = 13$ (Layer 2) and $C = 20$ (Layer 3). Since the theoretical estimation is purely based on the number of addition and does not account for overhead resulting from control etc., a deviation is expected. With the "late-start" logic, this trade-off is shifted even further, since only a power increase is observed for Layer 1 compared to the ANN reference. In the breakdown of the PE unit (see Fig. 5.13c), the power reduction from the adder logic can be seen from the reference design to the base WVSNN design. Further reductions are achieved by reducing the adder power through clock gating of initial inactive cycles.

In the end, it is evident that the WVSNN design reduces the power by up to 2.32 $\times$ for the ECG



(a) Breakdown of power for components in WVSNN design grouped by function.

(b) Breakdown of power for components in WVSNN design grouped by layer.



(c) Breakdown of power for computing unit inside the WVSNN design.

Figure 5.13: Power breakdown of the WVSNN design grouped into functional units, layers and the components inside the computational neuron units [JL4].

monitoring use case. Note that this reduction does not exhibit the losses in the classification quality, since the result of the converted model is identical to the reference. Power reductions are achieved not only through the encoding to sparse activations, but also through circuit-level optimizations. These optimizations exploit probability distributions of the converted spikes (see Section 5.4.2).

5.6 Summary

In this section, we explored ANN-SNN conversion methods for efficient inference of an ECG classifier. In detail we investigated a conversion method, which encodes activations of an ANN reference into spike latencies. The encoding method ensures equivalent representations of activations in fixed-point and corresponding spike latencies. The basic operation of an ANN, i.e. a scalar product of a activation vector and a weight vector, is calculated over time without numerical difference to the

arithmetic operations in fixed-point. Comparing the number of additions, the algorithm in the encoded representation features less additions for a range of configuration, i.e. number of bits in fixed-point number or number of elements in input/weight vector.

An investigation of converted activations and the membrane potential of the derived SNN neuron model shows that circuit activity is not evenly distributed. In the ECG monitoring example, the computations are concentrated in the final cycles of the sequential accumulation within the SNN neuron. In the designed co-optimized hardware modules, this is exploited through circuit-level optimizations, e.g. clock gating of initial cycles.

The integration of the encoding principle in a system level hardware design confirms the complexity reductions predicted theoretically in the algorithm analysis. The isomorphic mapping of both a ANN reference design and the WVSNN design shows significant power reductions resulting from the addition logic and circuit activity in the PE unit. All in all, the classification of ECG signals exhibit a $2.32\times$ power reduction without loss in QoS compared to the reference design.

Nevertheless, there are still many promising directions to improve the design further. A promising direction for future design optimization is to investigate dataflow mechanisms for the reuse of PE units. Analogous to the in-depth exploration in Section 3, the WVSNN design can leverage the layered inference structure of feed-forward networks architectures and sequentially execute the computations on shared PE units.

Chapter 6

Conclusion

In this thesis, we systematically explored methods to co-design algorithm and hardware for real-life tasks. The targeted research question is as follows:

Research Question:

> What neuro-inspired concepts enable low-power digital processing of real-time, low data rate signals?

The approach started with the definition of a use case requiring real-time and low-power processing of low data rate signals. We identified one task in the biomedical signal processing domain as a suitable application. The identification of cardiac arrhythmia with high-quality and long-term monitoring capabilities in a mobile ECG sensing device is ideal to demonstrate the desired features of neuro-inspired concepts, i.e. high classification quality and low power processing. To guide the exploration, specifications are defined to guarantee real-time capability, while optimizing for both classifier quality and complexity. A top-down design methodology is chosen to adopt design methods in multiple abstraction layers of the system design, i.e. algorithm and hardware design. Thus, the co-design of a neural network classifier is guaranteed to adhere to the application specifications.

As a first design, the WVCNN design is implemented from scratch as a baseline for further neuro-inspired features. The final design shows that a simple ANN architecture with wavelet transforms as pre-processing and a CNN as a classifier is capable of achieving low power processing at high classification quality. The uniform arithmetic operations, i.e. MAC operations and the layered subsampling modules allow adjusting the degree of parallel PE units to balance static and dynamic power consumption. The temporal execution of convolution allows data-driven computation of convolution kernels triggered by each input sample. Early cost estimations guide the the design of individual memory components and number of PEs. An implementation in a 22 nm FDSOI technology shows that the WVCNN design consumes only 525 nW, while achieving 79 % F1 score in the CINC'17 benchmark.

Result 1:

A systematic top-down approach is capable to design a baseline classifier using conventional CNNs capable of achieving ultra-low power classification with high quality of service.

As an extension of the baseline design, the domain generalization algorithms with low complexity are explored for robust classification. Similar to the human brain, the design needs to feature adaptive capabilities to generalize on unseen data. For the example of ECG classification, it is shown that low complexity algorithms like normalizations across instances are competitive compared to high complexity methods requiring fine-tuning of the whole ANN architecture. Therefore, correction layers are introduced to concentrate the generalization on a single linear layer. The correction layer with more trainable parameters, e.g. inter-channel transforms, provide superior capacity to align the feature of both source and target domain. The correction layer favors a central position in the network for best classification results. Evaluated in an inter-patient validation scheme for AF classification, the correction layer improves the F1 score on average in the target domain by more than 20 % and only less than 2 % difference compared to the F1 score on the source domain. The integration of the correction layer into the WVCNN design shows a power overhead of 4.7 %. An integration of the correction layer into adjacent convolution layers does not differ from the reference design without generalization. For CL training, the WVCNN design requires $2\times$ less MAC operations and memory compared to conventional fine-tuning.

Result 2:

The incorporation of correction layers allows the generalization across domains with low overhead in terms of computational complexity and memory both for inference and training.

Another neuro-inspired feature is the spiking communication in neurons. Using the temporal encoding of activations into spike timings, an ANN is converted into its SNN counterpart. Since fixed-point numbers are discrete in their value, they can be represented fully using discrete timesteps. The linearly encoded of fixed-point numbers in the temporal domain can be used to calculate multiplication of numbers as well as the sum of multiple products. In the case of a convolution operation, the calculation in the temporal domain requires less additions than the conventional computer arithmetic for low word lengths and high number of products to be summed. The integration of temporal

encoding into an ECG classifier shows skewed activity distributions, which can be exploited in the logic design. A direct comparison of an isomorphically mapped WVCNN design and an equivalent WVSNN design reveals power reductions of 2.32 $\times$. Since the input/output behavior is numerically identical, power reductions are achieved without loss in classification quality.

Result 3:

> The ANN-SNN conversion into equivalent spike timings enable sparse communication and low power inference without loss in classifier quality.

In the end, a top-down design approach for an ANN classifier achieves an efficient design for continuous high-quality processing. The incorporation of bio-inspired elements such as spiking communication and generalization on unseen data enables low-power digital processing of biomedical signals in real-time.

Nevertheless, the achieved design elements closely match conventional neural network acceleration, while additional features do not change the general processing concept. The main advantage of this approach is that ANNs are proven to succeed for real-life tasks and those gain additional desired features without major performance drawbacks. On the contrary, there are more disruptive methods pushing the borders of bio-inspired computing to find more efficient designs for future applications. For instance, three factor rules promises bio-plausible training methods to leverage spiking neural networks with bio-plausible neuron models. In combination with bio-plausible network architectures, e.g. distributions of excitatory and inhibitory synapses or general connectome in cortical areas, this might lead to more compact and efficient systems for real-life tasks. However, a direct comparison is necessary to reveal the benefits and drawbacks of these new concepts against more conservative approaches as proposed in this work.

Bibliography

[1] P. Kirchhof, S. Benussi, D. Kotecha, *et al.*, "2016 ESC guidelines for the management of atrial fibrillation developed in collaboration with EACTS," *European Heart Journal*, vol. 37, no. 38, pp. 2893–2962, Aug. 2016. DOI: 10.1093/eurheartj/ehw210.

[2] M. V. Perez, K. W. Mahaffey, H. Hedlin, *et al.*, "Large-scale assessment of a smartwatch to identify atrial fibrillation," *New England Journal of Medicine*, vol. 381, no. 20, pp. 1909–1917, Nov. 2019. DOI: 10.1056/nejmoa1901183.

[3] A. Y. Hannun, P. Rajpurkar, M. Haghpanahi, *et al.*, "Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network," *Nature Medicine*, vol. 25, no. 1, pp. 65–69, Jan. 2019. DOI: 10.1038/s41591-018-0268-3.

[4] Y.-P. Chen, D. Jeon, Y. Lee, *et al.*, "An injectable 64 nw ecg mixed-signal soc in 65 nm for arrhythmia monitoring," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 1, pp. 375–390, Jan. 2015, ISSN: 1558-173X. DOI: 10.1109/jssc.2014.2364036.

[5] Y. M. Chi, T.-P. Jung, and G. Cauwenberghs, "Dry-contact and noncontact biopotential electrodes: Methodological review," *IEEE Reviews in Biomedical Engineering*, vol. 3, pp. 106–119, 2010, ISSN: 1941-1189. DOI: 10.1109/rbme.2010.2084078.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[7] Y. Wang, Q. Wang, S. Shi, *et al.*, "Benchmarking the performance and energy efficiency of ai accelerators for ai training," in *IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, May 2020. DOI: 10.1109/ccgrid49817.2020.00-15.

[8] J. Sevilla, L. Heim, A. Ho, T. Besiroglu, M. Hobbhahn, and P. Villalobos, "Compute trends across three eras of machine learning," in *International Joint Conference on Neural Networks (IJCNN)*, IEEE, Jul. 2022. DOI: 10.1109/ijcnn55064.2022.9891914.

[9] C. D. Schuman, S. R. Kulkarni, M. Parsa, J. P. Mitchell, P. Date, and B. Kay, "Opportunities for neuromorphic computing algorithms and applications," *Nature Computational Science*, vol. 2, no. 1, pp. 10–19, Jan. 2022. DOI: 10.1038/s43588-021-00184-y.

[10] E. R. Kandel, J. H. Schwartz, T. M. Jessell, S. A. Siegelbaum, A. Hudspeth, and A. J. Hudspeth, *Principles of Neural Science, Fifth Edition*. 2012.

[11] C. Mead and M. Ismail, Eds., *Analog VLSI Implementation of Neural Systems*. Springer US, 1989. DOI: 10.1007/978-1-4613-1639-8.

[12] D. Auge, J. Hille, E. Mueller, and A. Knoll, "A survey of encoding techniques for signal processing in spiking neural networks," *Neural Processing Letters*, vol. 53, no. 6, pp. 4693–4710, Jul. 2021, ISSN: 1573-773X. DOI: 10.1007/s11063-021-10562-2.

[13] A. Basu, L. Deng, C. Frenkel, and X. Zhang, "Spiking neural network integrated circuits: A review of trends and future directions," in *IEEE Custom Integrated Circuits Conference (CICC)*, Apr. 2022. DOI: 10.1109/cicc53496.2022.9772783.

[14] K. Gavaskar, R. Dhivya, and R. Dimple Dayana, "Low power cmos design of phase locked loop for fastest frequency acquisition at various nanometer technologies," *Wireless Personal Communications*, vol. 125, no. 3, pp. 2239–2251, Mar. 2022, ISSN: 1572-834X. DOI: 10.1007/s11277-022-09654-6.

[15] A. L. Goldberger, Z. D. Goldberger, and A. Shvilkin, *Goldberger's Clinical Electrocardiography*. Elsevier, 2018. DOI: 10.1016/c2014-0-03319-9.

[16] N. J. Holter, "New method for heart studies," *Science*, vol. 134, no. 3486, pp. 1214–1220, Oct. 1961. DOI: 10.1126/science.134.3486.1214.

[17] S. Tan, G. Androz, A. Chamseddine, *et al.*, "Icentia11k: An unsupervised representation learning dataset for arrhythmia subtype discovery," *arXiv preprint arXiv:1910.09570*, Oct. 21, 2019. [Online]. Available: https://arxiv.org/abs/1910.09570.

[18] G. Clifford, C. Liu, B. Moody, *et al.*, "AF classification from a short single lead ECG recording: The physionet computing in cardiology challenge 2017," in *Computing in Cardiology Conference (CinC)*, Sep. 2017. DOI: 10.22489/cinc.2017.065-469.

[19] Y. Yin, S. M. Abubakar, S. Tan, *et al.*, "A 2.63 uw ECG processor with adaptive arrhythmia detection and data compression for implantable cardiac monitoring device," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 15, no. 4, pp. 777–790, Aug. 2021. DOI: 10.1109/tbcas.2021.3100434.

[20] A. L. Goldberger, L. A. N. Amaral, L. Glass, *et al.*, "PhysioBank, PhysioToolkit, and PhysioNet," *Circulation*, vol. 101, no. 23, Jun. 2000. DOI: 10.1161/01.cir.101.23.e215.

[21] S. Chatterjee, R. S. Thakur, R. N. Yadav, L. Gupta, and D. K. Raghuvanshi, "Review of noise removal techniques in ECG signals," *IET Signal Processing*, vol. 14, no. 9, pp. 569–590, Dec. 2020. DOI: 10.1049/iet-spr.2020.0104.

[22] D. Cubanski, D. Cyganski, E. M. Antman, and C. L. Feldman, "A neural network system for detection of atrial fibrillation in ambulatory electrocardiograms," *Journal of Cardiovascular Electrophysiology*, vol. 5, no. 7, pp. 602–608, Jul. 1994. DOI: 10.1111/j.1540-8167.1994.tb01301.x.

[23] H. Holst, M. Ohlsson, C. Peterson, and L. Edenbrandt, "A confident decision support system for interpreting electrocardiograms," *Clinical Physiology*, vol. 19, no. 5, pp. 410–418, Oct. 1999. DOI: 10.1046/j.1365-2281.1999.00195.x.

[24] G. Moody and R. Mark, "The impact of the MIT-BIH arrhythmia database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, 2001. DOI: 10.1109/51.932724.

[25] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

[26] M. Abadi, P. Barham, J. Chen, *et al.*, "TensorFlow: A system for Large-Scale machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.

[27] S. H. Jambukia, V. K. Dabhi, and H. B. Prajapati, "Classification of ecg signals using machine learning techniques: A survey," in *International Conference on Advances in Computer Engineering and Applications*, IEEE, Mar. 2015. DOI: 10.1109/icacea.2015.7164783.

[28] Z. Ebrahimi, M. Loni, M. Daneshtalab, and A. Gharehbaghi, "A review on deep learning methods for ecg arrhythmia classification," *Expert Systems with Applications: X*, vol. 7, p. 100 033, Sep. 2020, ISSN: 2590-1885. DOI: 10.1016/j.eswax.2020.100033.

[29] X. Liu, H. Wang, Z. Li, and L. Qin, "Deep learning in ecg diagnosis: A review," *Knowledge-Based Systems*, vol. 227, p. 107 187, Sep. 2021, ISSN: 0950-7051. DOI: 10.1016/j.knosys.2021.107187.

[30] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017.

[31] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5–6, pp. 602–610, Jul. 2005, ISSN: 0893-6080. DOI: 10.1016/j.neunet.2005.06.042.

[32] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, Sep. 3, 2014. [Online]. Available: https://arxiv.org/abs/1409.1259.

[33] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006, ISSN: 1530-888X. DOI: 10.1162/neco.2006.18.7.1527.

[34] F. Rincón, J. Recas, N. Khaled, and D. Atienza, "Development and evaluation of multi-lead wavelet-based ecg delineation algorithms for embedded wireless sensor nodes," *IEEE Transactions on Information Technology in Biomedicine*, vol. 15, no. 6, pp. 854–863, Nov. 2011, ISSN: 1558-0032. DOI: 10.1109/titb.2011.2163943.

[35] M. Jobst, J. Partzsch, C. Liu, *et al.*, "Zen: A flexible energy-efficient hardware classifier exploiting temporal sparsity in ecg data," in *IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, IEEE, Jun. 2022. DOI: 10.1109/aicas54282.2022.9869958.

[36] "Testing and Reporting Performance Results of Cardiac Rhythm and ST Segment Measurement Algorithms," International Organization for Standardization, Geneva, CH, Standard, 2012.

[37] M. Verhelst and B. Moons, "Embedded deep neural network processing: Algorithmic and processor techniques bring deep learning to iot and edge devices," *IEEE Solid-State Circuits Magazine*, vol. 9, no. 4, pp. 55–65, 2017, ISSN: 1943-0582. DOI: 10.1109/mssc.2017.2745818.

[38] R. Kumar and R. Goyal, "On cloud security requirements, threats, vulnerabilities and countermeasures: A survey," *Computer Science Review*, vol. 33, pp. 1–48, Aug. 2019, ISSN: 1574-0137. DOI: 10.1016/j.cosrev.2019.05.002.

[39] H. Kim, S. Kim, N. Van Helleputte, *et al.*, "A configurable and low-power mixed signal soc for portable ecg monitoring applications," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, no. 2, pp. 257–267, Apr. 2014, ISSN: 1940-9990. DOI: 10.1109/tbcas.2013.2260159.

[40] G. Sivapalan, K. K. Nundy, S. Dev, B. Cardiff, and D. John, "Annet: A lightweight neural network for ecg anomaly detection in iot edge sensors," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 16, no. 1, pp. 24–35, Feb. 2022, ISSN: 1940-9990. DOI: 10.1109/tbcas.2021.3137646.

[41] N. Van Helleputte, M. Konijnenburg, J. Pettine, *et al.*, "A 345 µw multi-sensor biomedical soc with bio-impedance, 3-channel ecg, motion artifact reduction, and integrated dsp," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 1, pp. 230–244, Jan. 2015, ISSN: 1558-173X. DOI: 10.1109/jssc.2014.2359962.

[42] A. Ayad, M. Barhoush, M. Frei, B. Völker, and A. Schmeink, "An efficient and private ecg classification system using split and semi-supervised learning," *IEEE Journal of Biomedical and Health Informatics*, vol. 27, no. 9, pp. 4261–4272, Sep. 2023, ISSN: 2168-2208. DOI: 10.1109/jbhi.2023.3281977.

[43] J. Lu, D. Liu, Z. Liu, *et al.*, "Efficient hardware architecture of convolutional neural network for ecg classification in wearable healthcare device," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 7, pp. 2976–2985, Jul. 2021, ISSN: 1558-0806. DOI: 10.1109/tcsi.2021.3072622.

[44] J. Lu, D. Liu, X. Cheng, L. Wei, A. Hu, and X. Zou, "An efficient unstructured sparse convolutional neural network accelerator for wearable ecg classification device," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 11, pp. 4572–4582, Nov. 2022, ISSN: 1558-0806. DOI: 10.1109/tcsi.2022.3194636.

[45] T. H. Teo, X. Qian, P. Kumar Gopalakrishnan, *et al.*, "A 700-uw wireless sensor node soc for continuous real-time health monitoring," *IEEE Journal of Solid-State Circuits*, Nov. 2010, ISSN: 1558-173X. DOI: 10.1109/jssc.2010.2064030.

[46] X. Zhang, Z. Zhang, Y. Li, C. Liu, Y. X. Guo, and Y. Lian, "A 2.89 uw dry-electrode enabled clockless wireless ecg soc for wearable applications," *IEEE Journal of Solid-State Circuits*, pp. 1–12, 2016, ISSN: 1558-173X. DOI: 10.1109/jssc.2016.2582863.

[47] J. Pan and W. J. Tompkins, "A real-time qrs detection algorithm," *IEEE Transactions on Biomedical Engineering*, vol. BME-32, no. 3, pp. 230–236, Mar. 1985, ISSN: 0018-9294. DOI: 10.1109/tbme.1985.325532.

[48] B.-U. Kohler, C. Hennig, and R. Orglmeister, "The principles of software qrs detection," *IEEE Engineering in Medicine and Biology Magazine*, vol. 21, no. 1, pp. 42–57, 2002, ISSN: 0739-5175. DOI: 10.1109/51.993193.

[49] G. Jimenez-Perez, A. Alcaine, and O. Camara, "Delineation of the electrocardiogram with a mixed-quality-annotations dataset using convolutional neural networks," *Scientific Reports*, vol. 11, no. 1, Jan. 2021, ISSN: 2045-2322. DOI: 10.1038/s41598-020-79512-7.

[50] R. F. Yazicioglu, S. Kim, T. Torfs, H. Kim, and C. Van Hoof, "A 30 uw analog signal processor asic for portable biopotential signal monitoring," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 209–223, Jan. 2011, ISSN: 1558-173X. DOI: 10.1109/jssc.2010.2085930.

[51] X. Liu, J. Zhou, Y. Yang, *et al.*, "A 457 nw near-threshold cognitive multi-functional ecg processor for long-term cardiac monitoring," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 11, pp. 2422–2434, Nov. 2014, ISSN: 1558-173X. DOI: 10.1109/jssc.2014.2338870.

[52] R. A. Abdallah and N. R. Shanbhag, "An energy-efficient ecg processor in 45-nm cmos using statistical error compensation," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 11, pp. 2882–2893, Nov. 2013, ISSN: 1558-173X. DOI: 10.1109/jssc.2013.2280055.

[53] S. Yin, M. Kim, D. Kadetotad, *et al.*, "A 1.06-uw smart ecg processor in 65-nm cmos for real-time biometric authentication and personal cardiac monitoring," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 8, pp. 2316–2326, Aug. 2019, ISSN: 1558-173X. DOI: 10.1109/jssc.2019.2912304.

[54] S. K. Cherupally, S. Yin, D. Kadetotad, *et al.*, "Ecg authentication hardware design with low-power signal processing and neural network optimization with low precision and structured compression," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 14, no. 2, pp. 198–208, Apr. 2020, ISSN: 1940-9990. DOI: 10.1109/tbcas.2020.2974387.

[55] Z. Chen, H. Xu, J. Luo, T. Zhu, and J. Meng, "Low-power perceptron model based ecg processor for premature ventricular contraction detection," *Microprocessors and Microsystems*, vol. 59, pp. 29–36, Jun. 2018, ISSN: 0141-9331. DOI: 10.1016/j.micpro.2018.03.006.

[56] Y. Zhao, Z. Shang, and Y. Lian, "A 13.34 uw event-driven patient-specific ann cardiac arrhythmia classifier for wearable ecg sensors," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 14, no. 2, pp. 186–197, Apr. 2020, ISSN: 1940-9990. DOI: 10.1109/tbcas.2019.2954479.

[57] R. Parmar, M. Janveja, J. Pidanic, and G. Trivedi, "Design of dnn-based low-power vlsi architecture to classify atrial fibrillation for wearable devices," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 3, pp. 320–330, Mar. 2023, ISSN: 1557-9999. DOI: 10.1109/tvlsi.2023.3236530.

[58] J. Liu, Z. Zhu, Y. Zhou, *et al.*, "4.5 bioaip: A reconfigurable biomedical ai processor with adaptive learning for versatile intelligent health monitoring," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb. 2021. DOI: 10.1109/isscc42613.2021.9365996.

[59] Z. Wang, Y. Liu, P. Zhou, *et al.*, "A 148-nw reconfigurable event-driven intelligent wake-up system for aiot nodes using an asynchronous pulse-based feature extractor and a convolutional neural network," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 11, pp. 3274–3288, Nov. 2021, ISSN: 1558-173X. DOI: 10.1109/jssc.2021.3113257.

[60] S. Sadasivuni, R. Chowdhury, V. E. G. Karnam, I. Banerjee, and A. Sanyal, "Recurrent neural network circuit for automated detection of atrial fibrillation from raw ecg," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2021. DOI: 10.1109/iscas51556.2021.9401666.

[61] F. C. Bauer, D. R. Muir, and G. Indiveri, "Real-time ultra-low power ecg anomaly detection using an event-driven neuromorphic processor," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 6, pp. 1575–1582, Dec. 2019, ISSN: 1940-9990. DOI: 10.1109/tbcas.2019.2953001.

[62] K. Buettner and A. D. George, "Heartbeat classification with spiking neural networks on the loihi neuromorphic processor," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Jul. 2021. DOI: 10.1109/isvlsi51109.2021.00035.

[63] A. Amirshahi and M. Hashemi, "Ecg classification algorithm based on stdp and r-stdp neural networks for real-time monitoring on ultra low-power personal wearable devices," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 6, pp. 1483–1493, Dec. 2019, ISSN: 1940-9990. DOI: 10.1109/tbcas.2019.2948920.

[64] Y. Liu, Z. Wang, W. He, *et al.*, "An 82nw 0.53pj/sop clock-free spiking neural network with 40μs latency for alot wake-up functions using ultimate-event-driven bionic architecture and computing-in-memory technique," in *IEEE International Solid- State Circuits Conference (ISSCC)*, Feb. 2022. DOI: 10.1109/isscc42614.2022.9731795.

[65] H. Chu, Y. Yan, L. Gan, *et al.*, "A neuromorphic processing system with spike-driven snn processor for wearable ecg classification," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 16, no. 4, pp. 511–523, Aug. 2022, ISSN: 1940-9990. DOI: 10.1109/tbcas.2022.3189364.

[66] Y. Xing, L. Zhang, Z. Hou, *et al.*, "Accurate ecg classification based on spiking neural network and attentional mechanism for real-time implementation on personal portable devices," *Electronics*, vol. 11, no. 12, p. 1889, Jun. 2022, ISSN: 2079-9292. DOI: 10.3390/electronics11121889.

[67] T. G. Noll, T. von Sydow, B. Neumann, J. Schleifer, T. Coenen, and G. Kappen, "Reconfigurable components for application-specific processor architectures," in *Dynamically Reconfigurable Systems*, Springer Netherlands, 2010, pp. 25–49. DOI: 10.1007/978-90-481-3485-4_2.

[68] K. Goetschalckx, B. Moons, S. Lauwereins, M. Andraud, and M. Verhelst, "Optimized hierarchical cascaded processing," *IEEE Journal on Emerging and Selected Topics in Circuits*

and Systems, vol. 8, no. 4, pp. 884–894, Dec. 2018, ISSN: 2156-3365. DOI: 10.1109/jetcas.2018.2839347.

[69] D. Griffith, "Toward zero: Power consumption trends in low data rate wireless connectivity," *IEEE Solid-State Circuits Magazine*, vol. 14, no. 4, pp. 51–60, 2022, ISSN: 1943-0590. DOI: 10.1109/mssc.2022.3195122.

[70] 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, Dec. 2017, ISSN: 1558-2256. DOI: 10.1109/jproc.2017.2761740.

[71] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998, ISSN: 0018-9219. DOI: 10.1109/5.726791.

[72] P. Zhou, D.-U. Choi, S.-M. Kang, and J. K. Eshraghian, "Backpropagating errors through memristive spiking neural networks," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2023. DOI: 10.1109/iscas46773.2023.10182006.

[73] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2009. DOI: 10.1109/cvpr.2009.5206848.

[74] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[75] C. Latotzke and T. Gemmeke, "Efficiency versus accuracy: A review of design techniques for dnn hardware accelerators," *IEEE Access*, vol. 9, pp. 9785–9799, 2021, ISSN: 2169-3536. DOI: 10.1109/access.2021.3050670.

[76] A. Parashar, P. Raina, Y. S. Shao, *et al.*, "Timeloop: A systematic approach to dnn accelerator evaluation," in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Mar. 2019. DOI: 10.1109/ispass.2019.00042.

[77] L. Ke, X. He, and X. Zhang, "Nnest: Early-stage design space exploration tool for neural network inference accelerators," in *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*, ACM, Jul. 2018. DOI: 10.1145/3218603.3218647.

[78] Kung, "Why systolic architectures?" *Computer*, vol. 15, no. 1, pp. 37–46, Jan. 1982, ISSN: 0018-9162. DOI: 10.1109/mc.1982.1653825.

[79] G. Peano, "Sur une courbe, qui remplit toute une aire plane," *Mathematische Annalen*, vol. 36, no. 1, pp. 157–160, Mar. 1890, ISSN: 1432-1807. DOI: 10.1007/bf01199438.

[80] D. Hilbert, "Ueber die stetige abbildung einer line auf ein flächenstück," *Mathematische Annalen*, vol. 38, no. 3, pp. 459–460, Sep. 1891, ISSN: 1432-1807. DOI: 10.1007/bf01199431.

[81] C. Gotsman and M. Lindenbaum, "On the metric properties of discrete space-filling curves," in *Proceedings of the 12th IAPR International Conference on Pattern Recognition*, IEEE, 1994. DOI: 10.1109/icpr.1994.577130.

[82] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Berlin Heidelberg, 1996, ISBN: 9783662033159. DOI: 10.1007/978-3-662-03315-9.

[83] F. N. Wilson, C. E. Kossmann, G. E. Burch, *et al.*, "Recommendations for standardization of electrocardiographic and vectorcardiographic leads," *Circulation*, vol. 10, no. 4, pp. 564–573, Oct. 1954, ISSN: 1524-4539. DOI: 10.1161/01.cir.10.4.564.

[84] M. Janveja, M. Tantuway, K. Chaudhari, and G. Trivedi, "Design of low power vlsi architecture for classification of arrhythmic beats using dnn for wearable device applications," in *IEEE Nordic Circuits and Systems Conference (NorCAS)*, Oct. 2021. DOI: 10.1109/norcas53631.2021.9599864.

[85] Y. LeCun, J. Denker, and S. Solla, "Optimal brain damage," *Advances in Neural Information Processing Systems*, vol. 2, 1989. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf.

[86] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," *Advances in Neural Information Processing Systems*, vol. 29, 2016. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2016/file/d8330f857a17c53d217014ee776bfd50-Paper.pdf.

[87] D. V. Christensen, R. Dittmann, B. Linares-Barranco, *et al.*, "2022 roadmap on neuromorphic computing and engineering," *Neuromorphic Computing and Engineering*, vol. 2, no. 2, p. 022 501, May 2022, ISSN: 2634-4386. DOI: 10.1088/2634-4386/ac4a83.

[88] Q. Lin, S. Song, I. D. Castro, *et al.*, "Wearable multiple modality bio-signal recording and processing on chip: A review," *IEEE Sensors Journal*, vol. 21, no. 2, pp. 1108–1123, Jan. 2021, ISSN: 2379-9153. DOI: 10.1109/jsen.2020.3016115.

[89] T. Teijeiro, C. A. Garcia, D. Castro, and P. Felix, "Arrhythmia classification from the abductive interpretation of short single-lead ecg records," in *Computing in Cardiology Conference (CinC)*, Sep. 2017. DOI: 10.22489/cinc.2017.166-054.

[90] M. Zihlmann, D. Perekrestenko, and M. Tschannen, "Convolutional recurrent neural networks for electrocardiogram classification," in *Computing in Cardiology Conference (CinC)*, Sep. 2017. DOI: 10.22489/cinc.2017.070-060.

[91] S. Datta, C. Puri, A. Mukherjee, *et al.*, "Identifying normal, af and other abnormal ecg rhythms using a cascaded binary classifier," in *Computing in Cardiology Conference (CinC)*, Sep. 2017. DOI: 10.22489/cinc.2017.173-154.

[92] M. Zabihi, A. Bahrami Rad, A. K. Katsaggelos, S. Kiranyaz, S. Narkilahti, and M. Gabbouj, "Detection of atrial fibrillation in ecg hand-held devices using a random forest classifier," in *Computing in Cardiology Conference (CinC)*, Sep. 2017. DOI: 10.22489/cinc.2017.069-336.

[93] S. Hong, M. Wu, Y. Zhou, *et al.*, "Encase: An ensemble classifier for ecg classification using expert features and deep neural networks," in *Computing in Cardiology Conference (CinC)*, Sep. 2017. DOI: 10.22489/cinc.2017.178-245.

[94] S. Saadatnejad, M. Oveisi, and M. Hashemi, "Lstm-based ecg classification for continuous monitoring on personal wearable devices," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 2, pp. 515–523, Feb. 2020, ISSN: 2168-2208. DOI: 10.1109/jbhi.2019.2911367.

[95] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, Mar. 4, 2018. [Online]. Available: https://arxiv.org/abs/1803.01271.

[96] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML)*, Lille, France, 2015, pp. 448–456.

[97] N. Bjorck, C. P. Gomes, B. Selman, and K. Q. Weinberger, "Understanding batch normalization," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[98] J. Wen, "Logic Design of an energy-efficient CNN for ECG Classification based on Wavelet Transformation," *M.S. thesis, Chair of Integrated Digital Systems and Circuit Design, RWTH Aachen University*, 2020.

[99] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, Feb. 2014. DOI: 10.1109/isscc.2014.6757323.

[100] P. Pirsch, *Architectures for Digital Signal Processing*. Wiley Diagnostic and Therapeutic Radiology, 1998.

[101] C. Latotzke, B. Balim, and T. Gemmeke, "Post-training quantization for energy efficient realization of deep neural networks," in *21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, Dec. 2022. DOI: 10.1109/icmla55696.2022.00243.

[102] V. Camus, C. Enz, and M. Verhelst, "Survey of precision-scalable multiply-accumulate units for neural-network processing," in *IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, Mar. 2019. DOI: 10.1109/aicas.2019.8771610.

[103] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017, ISSN: 1558-173X. DOI: 10.1109/jssc.2016.2616357.

[104] E. Lee and D. Messerschmitt, "Synchronous data flow," *Proceedings of the IEEE*, vol. 75, no. 9, pp. 1235–1245, 1987, ISSN: 0018-9219. DOI: 10.1109/proc.1987.13876.

[105] F. Arnaud, S. Clerc, S. Haendler, *et al.*, "Enhanced design performance thanks to adaptative body biasing technique in fdsoi technolologies," in *IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, Oct. 2017. DOI: 10.1109/s3s.2017.8308754.

[106] P. Busia, M. A. Scrugli, V. J.-B. Jung, L. Benini, and P. Meloni, "A tiny transformer for low-power arrhythmia classification on microcontrollers," *IEEE Transactions on Biomedical Circuits and Systems*, pp. 1–11, 2024, ISSN: 1940-9990. DOI: 10.1109/tbcas.2024.3401858.

[107] T. B. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, May 28, 2020. [Online]. Available: https://arxiv.org/abs/2005.14165.

[108] F. Pourpanah, M. Abdar, Y. Luo, *et al.*, "A review of generalized zero-shot learning methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–20, 2022, ISSN: 1939-3539. DOI: 10.1109/tpami.2022.3191696.

[109] F. Zhuang, Z. Qi, K. Duan, *et al.*, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, Jan. 2021, ISSN: 1558-2256. DOI: 10.1109/jproc.2020.3004555.

[110] M. A. Rahhal, Y. Bazi, H. AlHichri, N. Alajlan, F. Melgani, and R. Yager, "Deep learning approach for active classification of electrocardiogram signals," *Information Sciences*, vol. 345, pp. 340–354, Jun. 2016, ISSN: 0020-0255. DOI: 10.1016/j.ins.2016.01.082.

[111] G. Wang, C. Zhang, Y. Liu, *et al.*, "A global and updatable ecg beat classification system based on recurrent neural networks and active learning," *Information Sciences*, vol. 501, pp. 523–542, Oct. 2019, ISSN: 0020-0255. DOI: 10.1016/j.ins.2018.06.062.

[112] Y. Bazi, N. Alajlan, H. AlHichri, and S. Malek, "Domain adaptation methods for ECG classification," in *International Conference on Computer Medical Applications (ICCMA)*, IEEE, Jan. 2013. DOI: 10.1109/iccma.2013.6506156.

[113] Y. Ganin, E. Ustinova, H. Ajakan, *et al.*, "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[114] W. Yin, X. Yang, L. Li, *et al.*, "Self-adjustable domain adaptation in personalized ECG monitoring integrated with IR-UWB radar," *Biomedical Signal Processing and Control*, vol. 47, pp. 75–87, Jan. 2019. DOI: 10.1016/j.bspc.2018.08.002.

[115] M. Chen, G. Wang, Z. Ding, J. Li, and H. Yang, "Unsupervised domain adaptation for ECG arrhythmia classification," in *42nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Jul. 2020. DOI: 10.1109/embc44109.2020.9175928.

[116] H. Hasani, A. Bitarafan, and M. Soleymani, "Classification of 12-lead ECG signals with adversarial multi-source domain generalization," in *Computing in Cardiology Conference (CinC)*, Dec. 2020. DOI: 10.22489/cinc.2020.445.

[117] Z. Shang, Z. Zhao, H. Fang, *et al.*, "Deep discriminative domain generalization with adversarial feature learning for classifying ECG signals," in *Computing in Cardiology (CinC)*, Sep. 2021. DOI: 10.23919/cinc53138.2021.9662844.

[118] F. Deng, S. Tu, and L. Xu, "Multi-source unsupervised domain adaptation for ECG classification," in *IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, Dec. 2021. DOI: 10.1109/bibm52615.2021.9669755.

[119] J. Lee and H.-J. Yoo, "An overview of energy-efficient hardware accelerators for on-device deep-neural-network training," *IEEE Open Journal of the Solid-State Circuits Society*, vol. 1, pp. 115–128, 2021, ISSN: 2644-1349. DOI: 10.1109/ojsscs.2021.3119554.

[120] R. Mao, S. Li, Z. Zhang, *et al.*, "An ultra-energy-efficient and high accuracy ecg classification processor with snn inference assisted by on-chip ann learning," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 16, no. 5, pp. 832–841, Oct. 2022, ISSN: 1940-9990. DOI: 10.1109/tbcas.2022.3185720.

[121] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: Opportunities and challenges," *Frontiers in Neuroscience*, vol. 12, Oct. 2018, ISSN: 1662-453X. DOI: 10.3389/fnins.2018.00774.

[122] T. C. Potjans and M. Diesmann, "The cell-type specific cortical microcircuit: Relating structure and activity in a full-scale spiking network model," *Cerebral Cortex*, vol. 24, no. 3, pp. 785–806, Dec. 2012, ISSN: 1047-3211. DOI: 10.1093/cercor/bhs358.

[123] K. Kauth, T. Stadtmann, V. Sobhani, and T. Gemmeke, "Neuroaix-framework: Design of future neuroscience simulation systems exhibiting execution of the cortical microcircuit model 20× faster than biological real-time," *Frontiers in Computational Neuroscience*, vol. 17, Apr. 2023, ISSN: 1662-5188. DOI: 10.3389/fncom.2023.1144143.

[124] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, Aug. 1952, ISSN: 1469-7793. DOI: 10.1113/jphysiol.1952.sp004764.

[125] M. B. Jackson, A. Konnerth, and G. J. Augustine, "Action potential broadening and frequency-dependent facilitation of calcium signals in pituitary nerve terminals.," *Proceedings of the National Academy of Sciences*, vol. 88, no. 2, pp. 380–384, Jan. 1991, ISSN: 1091-6490. DOI: 10.1073/pnas.88.2.380.

[126] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha, "Backpropagation for energy-efficient neuromorphic computing," *Advances in Neural Information Processing Systems*, vol. 28, 2015. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2015/file/10a5ab2db37feedfdeaab192ead4ac0e-Paper.pdf.

[127] Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *International Journal of Computer Vision*, vol. 113, no. 1, pp. 54–66, Nov. 2014, ISSN: 1573-1405. DOI: 10.1007/s11263-014-0788-3.

[128] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *2015 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Jul. 2015. DOI: 10.1109/ijcnn.2015.7280696.

[129] D. Neil, M. Pfeiffer, and S.-C. Liu, "Learning to be efficient: Algorithms for training low-latency, low-compute deep spiking neural networks," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing (SAC)*, Apr. 2016. DOI: 10.1145/2851613.2851724.

[130] E. Hunsberger and C. Eliasmith, "Training spiking deep networks for neuromorphic hardware," *arXiv preprint arXiv:1611.05141*, Nov. 16, 2016. [Online]. Available: https://arxiv.org/abs/1611.05141.

[131] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in Neuroscience*, vol. 11, Dec. 2017, ISSN: 1662-453X. DOI: 10.3389/fnins.2017.00682.

[132] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, Jul. 2014, ISBN: 9781107447615. DOI: 10.1017/cbo9781107447615.

[133] S. Davidson and S. B. Furber, "Comparison of artificial and spiking neural networks on digital hardware," *Frontiers in Neuroscience*, vol. 15, Apr. 2021, ISSN: 1662-453X. DOI: 10.3389/fnins.2021.651141.

[134] B. Rueckauer and S.-C. Liu, "Conversion of analog to spiking neural networks using sparse temporal coding," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2018. DOI: 10.1109/iscas.2018.8351295.

[135] S. Park, S. Kim, B. Na, and S. Yoon, "T2fsnn: Deep spiking neural networks with time-to-first-spike coding," in *Proceedings of the 57th ACM/EDAC/IEEE Design Automation Conference (DAC)*, Virtual Event, USA: IEEE Press, 2020, ISBN: 9781450367257.

[136] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. van Baalen, and T. Blankevoort, "A white paper on neural network quantization," *arXiv preprint arXiv:2106.08295*, Jun. 15, 2021. [Online]. Available: https://arxiv.org/abs/2106.08295.

[137] J. Lou, F. Freye, C. Lanius, and T. Gemmeke, "An energy efficient all-digital time-domain compute-in-memory macro optimized for binary neural networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 71, no. 1, pp. 287–298, Jan. 2024, ISSN: 1558-0806. DOI: 10.1109/tcsi.2023.3323205.

[138] N. Qiao, H. Mostafa, F. Corradi, *et al.*, "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses," *Frontiers in Neuroscience*, vol. 9, Apr. 2015, ISSN: 1662-453X. DOI: 10.3389/fnins.2015.00141.

[139] G. Orchard, C. Meyer, R. Etienne-Cummings, C. Posch, N. Thakor, and R. Benosman, "Hfirst: A temporal approach to object recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 10, pp. 2028–2040, Oct. 2015, ISSN: 1939-3539. DOI: 10.1109/tpami.2015.2392947.

Curriculum Vitae

20.03.2025	Doctoral Examination for Dr.-Ing.
2018-2025	Research Assistant and Doctoral Candidate
	Chair of Integrated Digital Systems and Circuit Design
	RWTH Aachen University, Germany
2012-2018	Studies of Electrical Engineering, Information Technology and
	Computer Engineering
	RWTH Aachen University, Germany
May 2018	Master of Science
2017	Exchange Semester at Queensland University of Technology,
	Brisbane, Australia
Septemper 2015	Bachelor of Science
1999-2012	General Higher Education Entrance Qualification (Abitur)
	Städtisches Gymnasium Eschweiler, Germany
1993	born in Aachen, Germany