Power-Accuracy Trade-offs for Heartbeat Classification on Neural Networks Hardware

Adarsha Balaji[*,1], Federico Corradi[2], Anup Das[1],
Sandeep Pande[2], Siebren Schaafsma[2], Francky Catthoor[3]

[1]Dept. of ECE, Drexel University, Philadlephia, PA, 19104, USA

[2]Stichting IMEC Nederland, High Tech Campus 31, Eindhoven 5656 AE, The Netherlands

[3]IMEC Leuven, Kapeldreef 75, 3001 Heverlee, Belgium
KU Leuven, ESAT Department, B-3001 Leuven, Belgium

*corresponding author: Adarsha Balaji

Address:

Drexel University

Department of ECE

Chestnut Street, 3101

Philadelphia, PA, 19104, USA

Office : 267-469-8244

Email : adarsha.balaji@drexel.edu

Power-Accuracy Trade-offs for Heartbeat Classification on Neural Networks Hardware

Adarsha Balaji, Federico Corradi, Anup Das, Sandeep Pande,
Siebren Schaafsma, Francky Catthoor

Abstract– Heartbeat classification using electrocardiogram (ECG) data is an essential feature of modern day wearable devices. State-of-the-art machine learning-based heartbeat classifiers are designed using convolutional neural networks (CNN). Despite their high classification accuracy, CNNs require significant computational resources and power. This makes the mapping of CNNs on resource- and power-constrained wearable devices challenging. In this paper, we propose heartbeat classification using spiking neural networks (SNN), an alternative approach based on a biologically inspired, event-driven neural networks. SNNs compute and transfer information using discrete spikes that require fewer operations and less complex hardware resources, making them energy-efficient compared to CNNs. However, due to complex error-backpropagation involving spikes, supervised learning of deep SNNs remains challenging. We propose an alternative approach to SNN-based heartbeat classification. We start with an optimized CNN implementation of the heartbeat classification task and then convert the CNN operations, such as multiply-accumulate, pooling and softmax, into spiking equivalent with a minimal loss of accuracy. We evaluate the SNN-based heartbeat classification using publicly available ECG database of the Massachusetts Institute of Technology and Beth Israel Hospital (MIT/BIH), and demonstrate a minimal loss in accuracy when compared to 85.92% accuracy of a CNN-based hearbeat classification. We demonstrate that, for every operation, the activation of SNN neurons in each layer is sparse when compared to CNN neurons, in the same layer. We also show that this sparsity increases with an increase in the number of layers of the neural network. In addition, we detail the power-accuracy trade-off of the SNN and show a 87.76% and 96.82% reduction in SNN neuron and synapse activity,respectively, for accuracy loss ranging between 0.6% and 1.00%, when compared to a CNN-only implementation.

Keywords– Heartbeat classification, spiking neural network(SNN), convolution neural network(CNN)

1 INTRODUCTION

With the advance in electrocardiogram (ECG) sensor technology, modern day wearable devices are able to measure, monitor and classify several key biological metrics associated with the human heart [1–3]. In recent years, heartbeat classification techniques using ECG signals is successful at detecting cardiac abnormalities such as (1) cardiac arrhythmia, related to abnormal heart rhythms; (2) ischemia, related to poor blood flow to the heart muscles, and (3) past heart attacks, to name a few [4].

Over the past few decades, significant research is conducted to autonomously and accurately classify heartbeats. Deterministic algorithms classify heartbeat by manually identified features such as ECG morphology and heartbeat intervals [5–12]. Although accurate, these techniques requires relevant features to be identified by medical experts. Many of these approaches fail to classify heartbeats for patients with irregular heart conditions such as for patients following Transcatheter Aortic Valve Implantation (TAVI) procedure [13]. Additionally, the resource and power requirements for extracting the most relevant information from ECG signals limits the portability of these algorithms on wearables. Contrarily, the machine learning approaches allow to find hidden features in the ECG signal that may have been missed in these deterministic approaches. Machine learning algorithms such as support vector machines [14] (SVM) and deep learning [15] such as convolutional neural networks (CNN) have achieved excellent heartbeat classification accuracies.

A typical CNN uses four operators: (1) Multiply-accumulate (convolution), (2) zero thresholding (rectified linear unit, ReLU) , (3) downsampling (pooling), and (4) classification. These operations are computation, memory, and power intensive, and require larger systems like multi-core CPUs and GPUs to compute quickly and accurately. This makes the mapping of CNNs on resource- and power-constrained wearable devices challenging.

Recently, spiking neural networks [16] (SNN) are used to solve classification problems as alternatives to CNN. SNNs are bio-inspired, event driven, analog neural networks. When mapped on dedicated neuromorphic hardware such as CxQuad [17], Loihi [18], NeuCube [19], NeuroGrid [20], and HICANN [21], SNNs are highly energy and resource efficient compared to CNNs. This makes SNNs the perfect candidate for computing on wearable devices. However, the classification accuracy of SNNs are not at-par with the CNNs. This is due to the non-differentiable discrete activation function in SNNs making it challenging to use the gradient-based error back-propagation for optimization during the supervised training.

In this paper, we propose heartbeat classification using SNNs compatible with neuromorphic hardware. In our approach, we first train a CNN to perform heartbeat classification. To extract the time-domain and the frequency-domain features from ECG signals effectively, we propose time-frequency joint distribution of ECG signals over a time window containing the critical QRS peak. We then use this joint distribution as features and the hand-labeled QRS classes as labels for the CNN. We optimize the CNN topology to obtain the highest classification accuracy. Next, we map the CNN operations into their SNN equivalents ensuring a near lossless conversion. Our approach achieves promising classification accuracy while demonstrating significantly lower neuron and synaptic activation when simulated.

Contributions: Following are our key contributions.

- We propose a CNN-based heartbeat classification using time-frequency joint distribution of ECG signals
- We propose a technique to map CNN operations into SNN equivalent with near lossless conversion;
- We demonstrate the sparsity of the Spiking Neural Network activations, and show that this sparsity increases with the number of layers in the network;
- We perform power-accuracy trade-off analysis of the trained SNN by measuring the reduction in operation-count for SNNs;

The remainder of this paper is organized as follows. Related works are discussed in Section 2. Background on CNNs and SNNs are provided in Section 3. The overall design flow is introduced in Section 4. Results are presented in Section 5 and conclusions in Section 6.

2 RELATED WORKS

Heartbeat classification is an important research problem for medical diagnosis. Heartbeat classification can be performed using two methods: (1) classification using heuristic techniques based on deterministic features extracted from ECG, and (2) classification using machine learning.

2.1 Heartbeat Classification with Clinically-Relevant Features Extracted by Experts-in-the Field

Clinically-relevant heartbeat features of ECG signals is extracted from the cardiac rhythm (also known as the RR interval). The features in the RR interval is used with great effect to distinguish the types of heartbeats. De Chazal et al. [5] propose a method for heartbeat classification into five groups: normal beats, VEBs, SVEBs, fusion of normal and VEBs, and unknown beat types using ECG morphology, heartbeat intervals and RR intervals. This method then uses two linear discriminant classifiers (LDC), in tandem, to classify the heartbeats. The authors have demonstrated a sensitivity of 75.9%-77.7% and a predictivity of 38.5% -81.9% across the five classes. Christov et al. [6] propose the classification of QRS complex in five classes using ECG morphological and time-frequency features extracted with matching pursuits. This method uses the K nearest neighbor rule to classify the heartbeats. The authors demonstrate an accuracy between 90.7%-99% across the five heartbeat classes. Llamedo et al. [7] propose a heartbeat classification technique using R-R interval and morphology based features from the ECG signal, the two dimensional vectocardiogram (VCG) loop and the discrete wavelet transform of the ECG signal. This method then uses two linear discriminant classifiers to classify the heartbeats. The authors demonstrate a global accuracy of 93% across several heartbeat classes. Rakowski et al. [12] propose a ECG heartbeat classification technique using ECG morphology and R-R intervals based features. This method uses a SOM and Learning Vector Quantization algorithm to classify the ECG signals. The author compares the classifier using features from original ECG signals and features from a mathematically morphed ECG signals. The author reports accuracies ranging between 49.56%-97.14% for classification using original ECG signals and 41.86%-97.14% for classification using preprocessed signals. Lin et al. [22] propose heartbeat classification using normalized RR intervals and morphological features, extracted using wavelets. This method then uses a linear discriminant classifier (LDC) to classify heartbeats. Authors have demonstrated an accuracy of over 93% for supraventricular ectopic beats.

2.2 Heartbeat Classification with Automatic Feature Extraction using Machine Learning Techniques

In recent years, alternative models for heartbeat classification using machine learning algorithms are becoming popular. This is due to their ability to extract hidden features from ECG signals and use them to achieve high classification accuracy. Dutta et al. [23] propose a cross-correlation based approach cross-spectral density information in the frequency domain of the ECG signal is used as features. These features are then used in a least-square support vector machine classifier. The ECG heartbeats are classified into three classes – normal beats, PVC beats, and other beats. Authors have demonstrated an accuracy of 95% for these classes. Faziludeen et al. [24] propose to use daubechies wavelet, a type of time-frequency joint distribution, to extract heartbeat features from ECG signals. These extracted features are classified using a one-against-one (OAO) SVM. Authors have demonstrated an accuracy of 99.92% accuracy for premature ventricular contraction beats. Melgani et al. [25] propose a two-step approach for the classification of five heartbeat classes. First, a SVM is used to classify the QRS heartbeat by autonomously extracting the hidden features. Next, it uses particle swarm optimization, an evolutionary algorithm, to improve the generalization performance of the SVM classifier. The combined technique achieves an average accuracy of 89.72% for all the critical heartbeats. Ye at al. [8] propose an approach for heartbeat classification using morphological features extracted using wavelet transform, independent component analysis, and RR intervals. The method uses a Support Vector Machine to classify the heartbeat into sixteen separate classes. The authors demonstrate a classification accuracy of 99.3% across the sixteen classes.

Apart from SVM, neural networks are also proposed for heartbeat classification. Hu et al. [26] propose a

method for QRS detection and heartbeat classification using an adaptive multi-layer perceptron (MLP). The QRS detection technique employs a MLP that enhances the QRS complex while avoiding the non-linear background noise. The classifier works with 12 classes. Osowski et al. [27] propose a method for the heartbeat classification using a fuzzy hybrid neural network. The features used are the cumulants of the second, third, and fourth order. The classifier is a combination of a fuzzy self-organizing subnetwork (using c-means and Gustafson-Kessel algorithms) followed by a multilayer perceptron. They achieve good accuracy. The classification accuracy presented by the author ranges between 90.28% to 96.94% across seven different wavelet types. Yu et al. [28] present a method for ECG beat classification using a probabilistic neural network (PNN) classifier. Sub-bands of the discrete wavelet transformed (DTW) ECG signal, along with AC power and the RR interval are used as features for classification. The author demonstrates a classification accuracy of 99.65% using the PNN classifier with a 11 dimension features set. Al Rahhal et al. [29] propose an approach for classification of ECG signals using a deep neural network (DNN). The classification is achieved in two steps: (1) features are extracted from a raw ECG signal using a unsupervised stacked denoising autoencoder (DEA) with sparsity constraints, and (2) most reliable and label ECG signals are used to train the DNN classifier. The author shows accuracies ranging from 86.39% to 99.77% while using the publicly available ECG database. Zubair et al. [30] propose a ECG classification method using convolution neural networks. The model extracts hidden features from raw ECG signals. The ECG signals are classified into five different classes. Authors reports superior accuracy than most of the state-of-the-art methods for ECG signal classification. Rajpurkar et al. [31] present an algorithm for heart arrhythmia detection using ECG signals. The method uses a 34-layer CNN as the classifier. The author also attributes the performance of the classifier to a larger annotated dataset, 500 times larger than the number used in other studies. The accuracy of the CNN is presented as a F1 score ranging between 0.8 and 0.809.

Recently, SNNs are used for building medical applications on wearables. Das et al. [32], propose to encode the ECG signals into a spike train (using temporal coding). The discrimination of ECG signals is done using an unsupervised classifier built using a network of recurrently connected spiking neurons followed by a probabilistic inference unit at the output layer. The approach demonstrates high classification accuracy and low power consumption for 23 distinct heartbeat classes.

3 BACKGROUND

3.1 Background on Convolution Neural Networks

Convolutional neural networks (CNNs) are a class of deep neural networks [15] used extensively for image classification, feature clustering, and object recognition. CNNs are arrangement of neurons into layers, with synaptic weights connecting the neurons. Figure 1 shows a typical organization of CNN, which performs four key operations: (1) convolution, (2) non-linear activation, (3) pooling, and (4) classification. We briefly describe these four operations involved in a CNN.

3.1.1 Convolution

The convolution operation is used to extract features from the input image. Typically, a small matrix (3×3 or 5×5), also called the kernel, is used to extract these features. The kernel is slide over the input image to compute the dot product of the matrix coefficient and the image pixels. The resultant matrix is called the activation map or the feature map, and contains the encoded features of the input image.

3.1.2 Non-linear Activation

The results of the convolution operations are subjected to a non-linear function, Rectified Linear Unit (ReLU) to replaces all negative pixel values in the feature map by zero. Mathematically, ReLU computes the function $f(x) = max(0, x)$, i.e., the activation is thresholded to zero. Other non-linear activation functions are also used for neural networks, such as sigmoid and tanh.

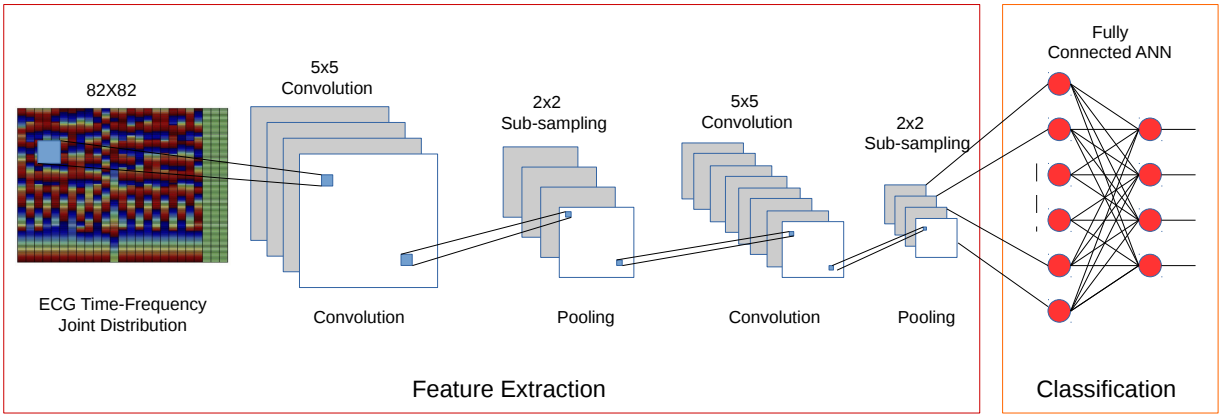

Figure 1: Example of a Convolution Neural Network (CNN)

3.1.3 Pooling

Spatial pooling is a downsampling technique to reduce the dimensionality of each feature map while retaining the most important features from the map. Typically, a spatial neighborhood is defined for this process. Downsampling is performed selecting one value for the neighborhood. If the largest element of the neighborhood is selected, the downsampling process is called the max pooling. If the average or the sum of the neighborhood elements is selected, the downsampling process is called the average pooling.

3.1.4 Classification

The output from the convolutional and pooling layers represent high-level features of the input image. The purpose of the fully-connected layer is to use these features for classifying the input image into various classes based on the training dataset. For this purpose, a softmax activation function, which takes a vector of arbitrary real-valued scores and squashes it to a vector of values between zero and one that sum to one.

3.2 Background on Spiking Neural Networks

Spiking neural networks (SNNs) [16] are computational models inspired by the dynamics of human brain. From an implementation perspective, SNNs are collection of neurons that communicate by sending short pulses (called spikes) across connections (synapses) to other neurons. In contrast to convolutional neural networks, the output of the the neurons in SNNs are asynchronous binary events. Event-based asynchronous computations make SNNs energy and resource-efficient, the key requirements in modern wearable devices.

Two widely used encoding schemes for SNNs are rate coding and temporal coding [33–35]. Rate coding encodes information in terms of the number of spikes within a timing window, while temporal coding encodes information in terms of the inter-spike intervals. Rate coding is used for spatial classification such as handwritten digit recognition [36], while temporal coding is used for time-series processing, such as speech recognition [37] and EEG-based brain-machine interface [38].

Similar to its analog counterpart, SNNs can also be organized into feedforward and recurrent typologies. In feedforward SNNs, the synaptic connections allow spikes to be propagated in the forward direction only. Figure 2(a) shows a feedforward SNN. The neurons of one layer connect to the neutrons in the next layer. Connections between layers can be one-to-one, one-to-all, or one-to-many, depending on the application. In recurrent SNNs, the synaptic connections are organized to allow spike propagation in the forward and backward directions. Figure 2(b) shows a recurrent SNN. Several computational models have been proposed for recurrent SNNs. Examples include liquid state machine [39], and Hopfield networks [40].

As SNNs use spikes rather than analog values to encode and communicate information between neurons, supervised learning using error backpropagation in SNNs remain challenging. This is because, the backprop-

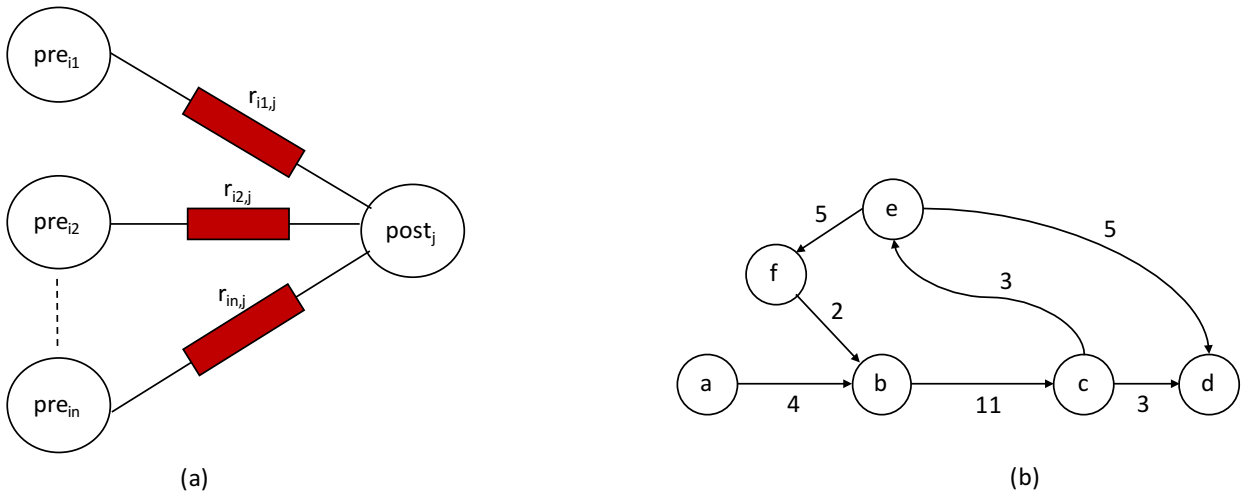

Figure 2: Example of feedforward and recurrent neural networks.

agation algorithm requires to compute the gradient of the error function, which is infeasible due to its discrete nature. To overcome this limitation, gradient-following approach is used. This usually requires simplifications of some sort, such as calculating the gradient in a time range where the post-synaptic potential can be estimated as a linear function of time up to a certain degree of accuracy [41], and using stochastic gradient descent where neuron firings are modeled with Poisson firing rate [42]. An alternative to backpropagation is using spike timing dependent plasticity [43], a form of Hebbian learning for spiking neurons. Based on STDP, remote supervised learning method (ReSuMe) is proposed [44], which uses local STDP to adapt the synaptic potentials across the networks of SNNs.

Recently, alternative approaches to supervised learning in SNNs are proposed, which involve conversion of a CNN to an equivalent SNN by transforming the CNN operations directly into SNN equivalents. Diehl et al. [45] propose a near lossless conversion of an ANN into a SNN. The SNN built using this technique out-performed all previous SNNs working on the MNIST database, with an accuracy of 98.64%. Cao et al. [46] propose to train a CNN to perform object recognition using the CIFAR-10 database. The CNN is then tailored to reduce the complexity of conversion to a SNN. The tailoring process includes the elimination of negative valued convolution features, restricted use of activation functions (ReLU only) and the removal of biases from all the layers of the CNN. Weights of the trained CNN are then applied to the SNN architecture, that is derived from the CNN. Rueckauer et al. [47] propose an extension to [46]. Here, the implementation of SNN-compatible versions for operations such as max pooling, softmax, and batch normalization have been achieved, hence allowing for a larger class of CNNs to be converted to an SNN.

4 Design Flow

4.1 High-level Overview

Figure 3 shows the high-level flow of our proposed approach. There are three major steps in our approach. First, we convert ECG data recorded from ECG sensors (see description in Section 4.2.1) to ECG joint distribution signatures using time-frequency joint distribution of ECG signals (see Section 4.2.2). Next we train convolutional neural networks (CNN) to classify these time-frequency joint distribution signatures into heartbeat classes. The CNN is optimized by varying its topology (layer width, layer depth, etc), design parameters (epochs, dropout rate, etc.), and operations (max ReLU, mean ReLU etc.) (see Section 4.2.3). Finally, we convert the CNN operations to SNN equivalents with minimal loss in accuracy (see Section 4.2.4). The resultant SNN is analyzed in terms of its operation counts and the energy-performance trade-offs is demonstrated by varying layers of the CNNs in the step marked as 'Optimization' in Figure 3.

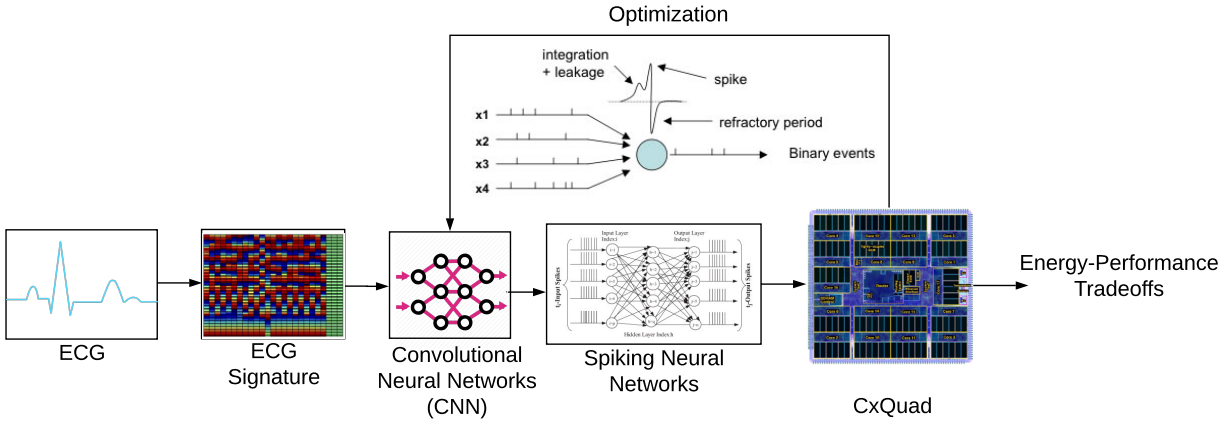

Figure 3: High-level design-flow.

4.2 Detailed Design

We now describe the steps of our approach in detail.

4.2.1 ECG Heartbeat Classification

Figure 4 shows a typical QRS complex in ECG signal. The ECG signal is characterized by five valleys and peaks, labeled P, Q, R, S and T. In general, ECG signals are analyzed using the QRS complex along with the T and P waves. The database of the Massachusetts Institute of Technology and Beth Israel Hospital (MIT/BIH) [48] is commonly used for QRS detection and classification as this is one of the databases indicated for performance evaluation of ventricular arrhythmia detection systems by the Association for the Advancement of Medical Instrumentation, AAMI/EC57 [49]. The heartbeats in the MIT-BIH database can be classified into 23 classes. As a proof of concept, we train our CNN with five classes – (1) normal beats, (2) superventricular beats, (3) ventricular beats, (4) fusion beats and (5) unclassified beats. Our approach can be trivially extended to consider all 23 classes.

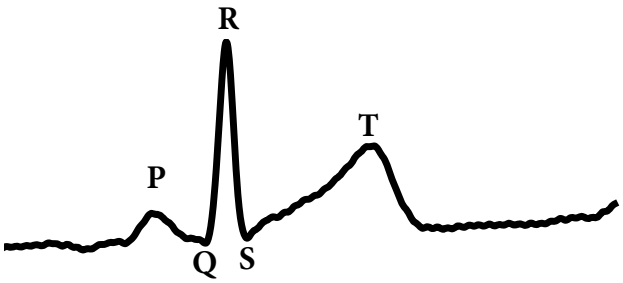

Figure 4: QRS complex in ECG.

4.2.2 Generating ECG Sparse Distributed Signatures

The classical Fourier analysis assumes that signals are of infinite duration or periodic in nature. However, non-stationary signals such as the ECG and speech are of short duration (transient), and change substantially over this duration. To accurately capture time varying frequency components of these transient signals, time-frequency joint distributions (such as short-time Fourier transform [50] and wavelet transform [51]) are frequently used. We use the short-time Fourier transform of ECG signals due to its real-time properties, low resource requirements, and energy efficiency.

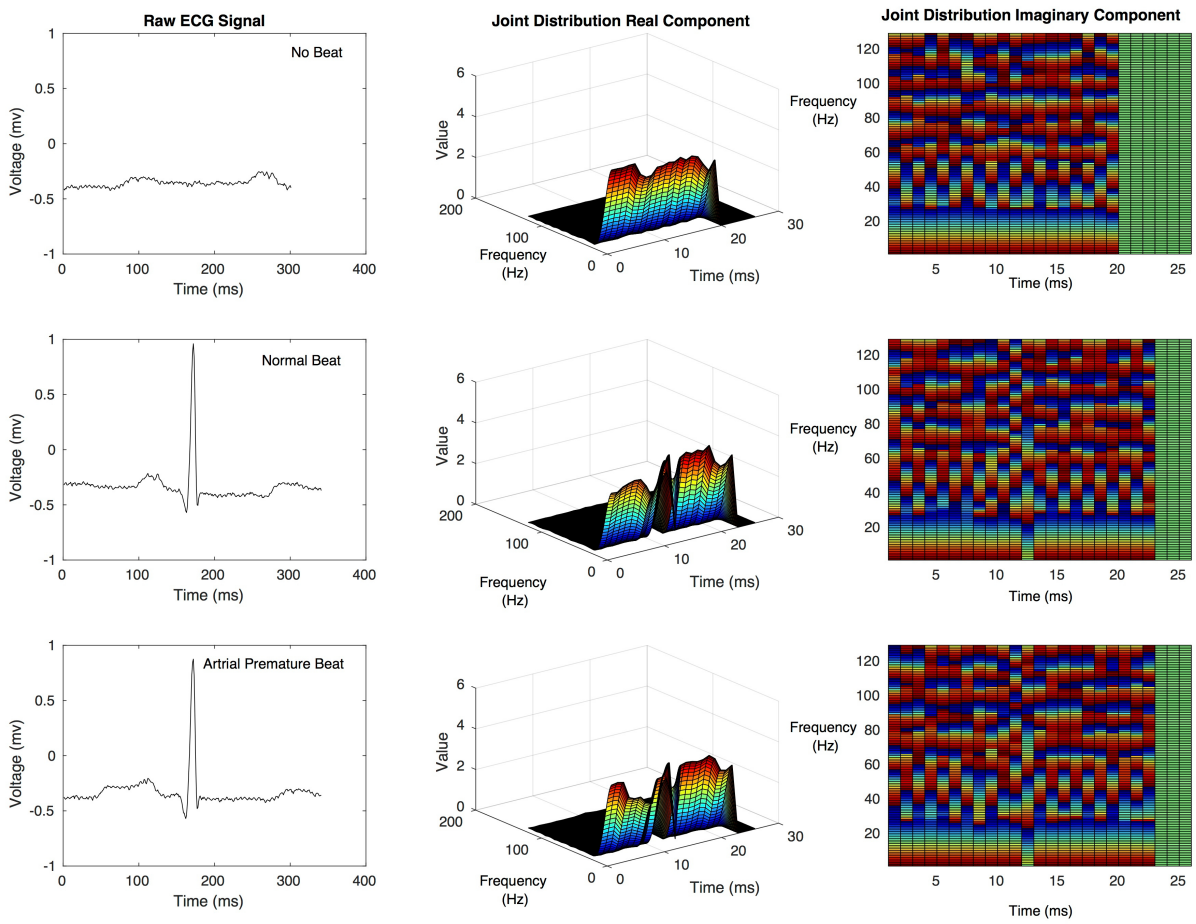

Figure 5: ECG signal segments and plot for real and imaginary components of time-frequency joint distribution. Also shown are the sparse distributed signatures [4].

The short-time Fourier transform of a discrete-time signal $x(t)$ is given by the following equation [52]

$$X(t,\omega) = \sum_{\tau=-\infty}^{\infty} h(t-\tau) \cdot x(\tau) \cdot e^{-j\omega\tau} \tag{1}$$

where, $h(t)$ is the analysis window, which is narrow in time and frequency, and is normalized such that $h(0) = 1$. This equation is similar to the classical Fourier transform, except that $X(t,\omega)$ is now a function of both time and frequency, and represents only a local behavior of $x(\tau)$ as viewed through the sliding window $h(t-\tau)$. Using the real and the imaginary components, we create a signature as follows.

$$\mathcal{S} = [f\{\mathbb{R}(X)\} \quad f\{\mathbb{I}(X)\} \quad \mathbb{P}] \tag{2}$$

where $f\{x\}$ is a transformation of x; $\mathbb{R}(X)$ and $\mathbb{I}(X)$ are respectively, the real and the imaginary components of X (Equation 1); and $\mathbb{P}$ is the padded zero component, which is introduced for future extension of the approach to incorporate additional parameters. The signature generated using the short-time Fourier transform components are shown in the third subplot of Figure 5. Each signature is a 82×82 matrix.

Each QRS peak is first transformed into its time-frequency representation. The ECG signal together with its real and imaginary components of time-frequency joint distribution are plotted in Figure 5 for three scenarios – (1) No Beat, (2) Normal Beat and (3) Atrial Premature Beat. As seen from Figure 5, the real and imaginary plots differ from each other for all three scenarios indicating the significance of both these components. We propose to combine these components to form unique signatures. These ECG signatures together with their original labels are then used for training the CNN. This is described next.

4.2.3 *Classifying ECG Heartbeats using CNN*

In this section, we describe a CNN model that we use to classify ECG signals. The input to the CNN is a 82x82 matrix that represents the sparse distributed signature of the ECG QRS complex. The network is trained using back propagation algorithm to classify the ECG signals into five heartbeat classes. All the remaining heartbeat classes are considered as a separate class. In total, the CNN is trained with six classes.

The CNN is designed using a series of generic convolution layers. A convolution layer consists of a convolution feature extraction layer, followed by a ReLU activation function, a pooling layer and a final ReLU activation function. Our network uses a maximum of two such convolution layers. The first convolution layer filters the 82x82x1 input sparse signature of an ECG signal with 64 kernels of size 5x5x1 and a pooling filter of size 2x2. The second convolution layer operates on the output of the first convolution layer. This layer consists of 64 kernels of size 5x5x1 and a pooling filter of size 2x2. The convolution layer is followed by a fully connected layer of neurons that perform the classification of the input ECG signals into 6 classes. As the fully connected layers are computationally expensive we set a dropout ranging between 0.3 and 0.5 across each of the fully connected layers. Weights of the neurons are initialized using the Xavier initialization function. The final decision layer uses a softmax function to produce the probability distribution across the six classes.

During the training phase, our network employs the conventional back-propagation algorithm [53] to compute the stochastic gradient descent with a batch size of 100, momentum of 0.9, and a learning rate of 0.03. We trained our network for 300 epochs with a training set of 6500 ECG signals selected randomly from 120K QRS complexes in the database.

4.2.4 *Converting CNN Operations to Equivalent SNN Operations*

In this section, we perform a one-to-one mapping of CNN neurons and their connection weights to SNN. In the following we highlight the design considerations to achieve this one-to-one mapping.

- *ReLU Activation Functions:* The ReLU activation function acts as a firing rate approximator for an Integrate and Fire (IF) neuron wherein, the output of the activation function is proportional to the spike rate of the IF neuron.

- *Bias:* A bias in CNN allows the classifier to translate its decision boundary by a fixed value (positive or negative). In SNNs, bias is represented as a constant input current to its neurons, the value of which is proportional to the CNN biases. The constant current contributes to the membrane potential of the SNN neuron. However, by including a zero bias in the training phase of the CNN we ensure that the SNN neurons are only defined by their threshold function and the synaptic weights. This reduces the complexity of the design space.

- *Weight Normalization:* Weight normalization is a method used to control the firing rate of SNN neurons. This is done to ensure that the firing rate of SNN neuron are not saturated. Unfortunately, this technique can also induce a low firing rate for neurons, thereby increasing the latency with which data reaches the higher layers of the network. The normalization is performed layer-wise and the weight normalization factor (λ) is set to all the neurons in a layer. This is done to ensure that the firing rate of all the neurons in the layer do not exceed the maximum firing rate of the network. This unfortunately induces very high activations for a small group of neurons in a layer, while the remaining neurons are normalized to a low firing rate. In order to minimize the temporal delay of the neuron and simultaneously ensure that the neuron firing threshold is not too low, we weight-normalize the first layer depending on the maximum spike-based input received by the first layer.

  After the threshold of the first layer is set, we are provided with a representative spike train at the output of the first layer which enables us to generate the input spike-stream for the next layer. The process is continued sequentially for all the layers of the network. Our approach ensure that the proposed weight-normalization scheme accounts for the actual SNN operation during the conversion process. By applying weight normalization per operation of the SNN we can ensure a balance between temporal delay and excessive firing rates of the SNN.

- *Spiking Softmax:* Softmax activation functions are used in the output layer of the CNN. The softmax activation function generates the probability distribution or the likelihood of the output belonging to a particular class. To replicate this behavior in a spiking neuron, an external spike generator, like a Poisson generator, is used to generate spikes based on the weighted sum accumulated by each spiking neuron.
- *Spiking Pooling Layers* Pooling layers in CNNs are often used to reduce the size of a convolution output. Max-pooling and average-pooling are the commonly used pooling functions in CNNs. Max-pooling in the CNNs detect the maximum output with-in the neighborhood of its filter. In SNNs, the neuron that fires first is said to have the maximum value as its input. We can hence disregard the output from the other neighboring neurons. Average pooling however is simpler to implement on SNNs and achieve similar results as max-pooling. Therefore, we use average-pooling in our CNN design.

4.2.5 Comparing CNN operations to SNN operations:

During the classification stage of the CNN, the number of operations required to complete one forward pass remains a constant. The operation count is calculated by finding the total number of incoming connections to every neuron in the network. For instance, in the convolution and fully-connected layers, neurons perform a multiple and accumulate operation (macc). Therefore, depending on the number of incoming connections to each neuron, we can calculate the number of MACC operations performed by neuron in the layer. Similarly, in during the classification stage of the SNN, we only require the addition operator [47] to update the state of each neuron. The number of operations are calculated using the number of synaptic connections of each neuron.

It is known that addeultiply operation. Raueckauer et al. [47] show that, in certain cases, the addition operation is upto 14x faster. Therefore, it is important to note that even if SNNs use a similar number of operations as the CNN to perform classification, the SNN operations require less time and far less complex computational resources to perform classification.

5 RESULTS AND DISCUSSION

The performance of the SNN depends on two factors: (1) a well-trained CNN, (2) a near lossless conversion of CNN operations into SNNs. The proposed method for the above approaches are discussed in section 4. In this section, we discuss the performance of the CNN for heartbeat classification, followed by the performance of the CNN-SNN conversion method.

5.1 Performance of CNN:

We train, validate and test the CNN using the MIT-BIH database. The QRS signals are preprocessed using python: short-time Fourier Transform using *specgram* from the *python-matplotlib* and the CNN is trained using the *lasagne* library along with *python-theano*. The heartbeat database consists of 23 classes [1]. But as the dataset is unbalanced (does not have same number of samples for every class), we only choose top 5 performing classes. The remaining 18 classes are grouped into the sixth class. The training and validation set contains 6500 labeled QRS signals selected from 120K QRS signals in the database. The test set consists of all the 120K QRS signals (excluding the ones chosen for the training dataset). The training, validation and test QRS signals represent all 48 patients and 6 classes. We explore multiple configurations of the the CNN and choose the best performing configuration to be converted into the SNN.

5.1.1 Exploring CNN configurations:

Table 1 shows the CNN configurations considered for heartbeat classification. The output layer of the CNN consists of 6 neurons. The performance of the CNN configurations are reported in Table 2. As can be seen

Config #	Learning Rate	# of Convolution Layers	# of Convolution Filters	Filter Size	Pooling Filter Size	# of Hidden layers	# of Neurons
1	0.03	2	32,32	5,5	2,2	1	256
2	0.03	2	64,64	5,5	2,2	4	512,256,256,256
3	0.03	2	64,64	7,7	2,2	2	512,256
4	0.03	2	64,64	3,3	2,2	2	512,256
5	0.05	2	128,128	5,5	2,2	1	512

Table 1: CNN configurations

from Table 2, the classification accuracy of CNN-1 is consistent across all the six chosen classes. The accuracy achieved ranges between 76.31% - 96.71% (average of 85.92%) across the chosen classes.

In CNN1, the number of neurons in the input layer is 6724 (82x82 Sparse Distributed Signature of ECG signal). CNN-1 consists of two convolution layers (CONV1 and CONV2 - with 32 filters of size 5x5, each), ReLU activation (ReLU1 and ReLU2), two max pooling layers (POOL1 and POOL2 - with filter size of 2x2) after every convolution layer, followed by two fully connected layers (FC1 - with 256 neurons and 6 neurons respectively). The output layer uses the softmax function.

The performance of CNN 1, verified using patient specific data is seen in Figure 6.

Configuration #	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
1	82.88%	76.31%	89.32%	96.71%	84.4%	45.29%
2	83.78%	73.26%	90.88%	96.43%	72.6%	56.84%
3	82.08%	62.98%	83.62%	96.29%	96.40%	35.62%
4	80.22%	81.91%	92.29	88.29%	90.60%	45.29%
5	82.83%	59.49%	76.61	97.57%	82.80%	38.87%

Table 2: Performance of CNNs across six classes

5.2 Performance of CNN-SNN Conversion Technique:

The CNN is converted to an SNN using the method described in 4.2.4. The neurons in the trained CNN are replaced by Integrate and Fire (IF) neurons with positive threshold and no incoming synaptic delay. The input joint distributed signatures are temporally-coded with a minimum period of $10\,\mathrm{ns}$ and a maximum period of $100\,\mu\mathrm{s}$. Additionally, we limit the number of spikes required for the computation of each stimulus by adding a decision delta threshold at the output layer. Empirically the decision threshold value is set to 4, this value minimizes conversion loss. We used *n2d2* software [54] for exploring conversion parameters for converting the CNN into an efficient SNN.

The conversion of the chosen CNN into an SNN leads to a conversion loss of 0.8% (+-0.2% on average) in accuracy for each of the 6 classes.

5.2.1 Analyzing the SNN Network

Table 3 shows the average number of neuron activations per layer of the network per input image. From the table we can see that, on average, the SNN performs 96.82% fewer neuron activations when compared to CNN neurons, for a single input image.

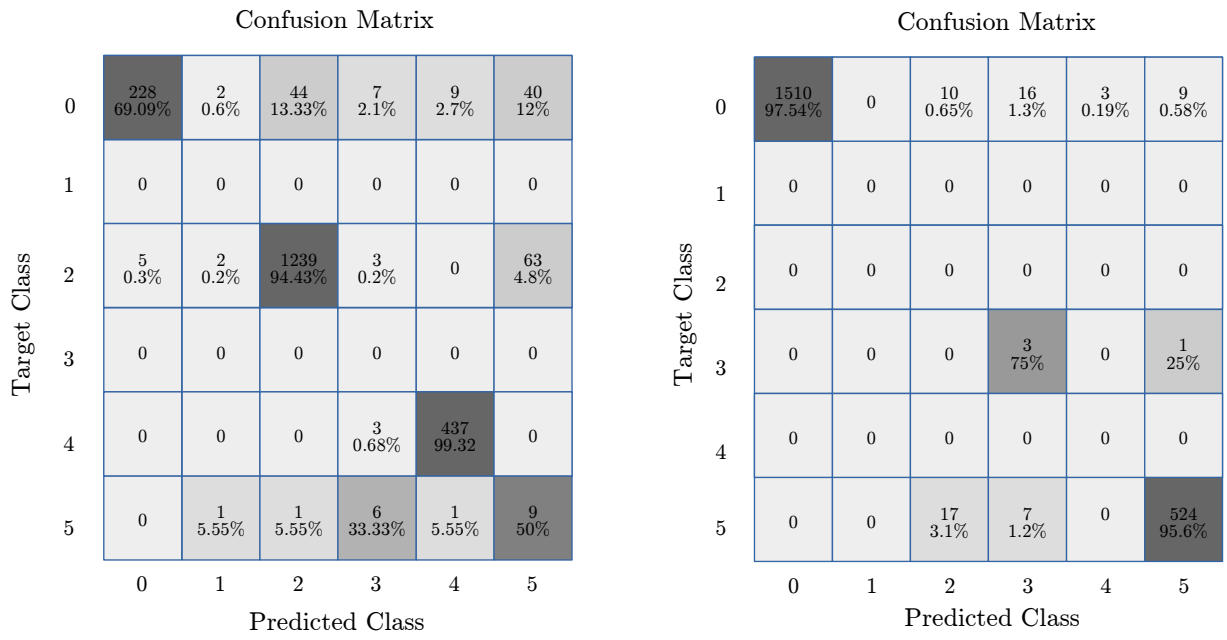

Figure 6: The Confusion Matrices of the proposed CNN using patient specific data. We have chosen two patients at random from the 48 available patients.

As discussed in Section 4, every CNN layer can be represented by basic operations such as multiply and accumulate (macc), compare (comp) and accumulate (acc), while SNN neurons only use the accumulate (acc) operator. On average, for every input image, each layer of the network performs: CONV1 - 3.04 M macc, ReLU1 - 121.68k comp, POOL1 - 121.68k acc, CONV2 - 30.63M macc, ReLU2 - 61.25k comp, POOL2 - 64.8k acc and FC1 - 4.15M macc operations respectively. On analyzing the CNN we see that 18.79 events (macc, comp or acc) are generated per connection per input image. However, the SNN only generates 2.35 events (acc only) per synapse per input image. On comparison, for every input, the average number of events per synapse of the SNN is 87.76% lower when compared to a CNN connection. This in-turn signifies that, for every forward pass of network, the SNN neurons and synapses are far less active when compared to their CNN counterparts.

Layer	CNN-Neuron Activation	SNN-Neuron Activation
Conv - 1	121,680	5,293
Pool - 1	30,420	1,170
Conv - 2	61,250	613
Pool - 2	16,200	157
FC - 1	256	75

Table 3: Average number of neurons activated per layer of the network per input image.

It is also important to note that the SNN operation (accumulate) is faster to execute and requires less complex hardware when compared to CNN operations (MACC). Therefore, we are able to reduce the total number of operations of the network and ensure that the operation are less complex and computationally demanding when compared to the CNN approach.

Figure 7 shows the firing rate of SNN neurons at every layer of the SNN. From the figure we see that deeper layers in the network have sparser firing rates. We know that, in general, deeper CNNs have better

classification accuracies. As we are able to convert these CNNs into SNNs in a near loss-less manner (0.8% loss in accuracy), and see that SNN activations and synapse events become more sparse the deeper we traverse through the network, we conclude that our proposed method for the conversion of CNN into SNN is successful at reducing the energy consumption of the network with a minimal loss in accuracy.

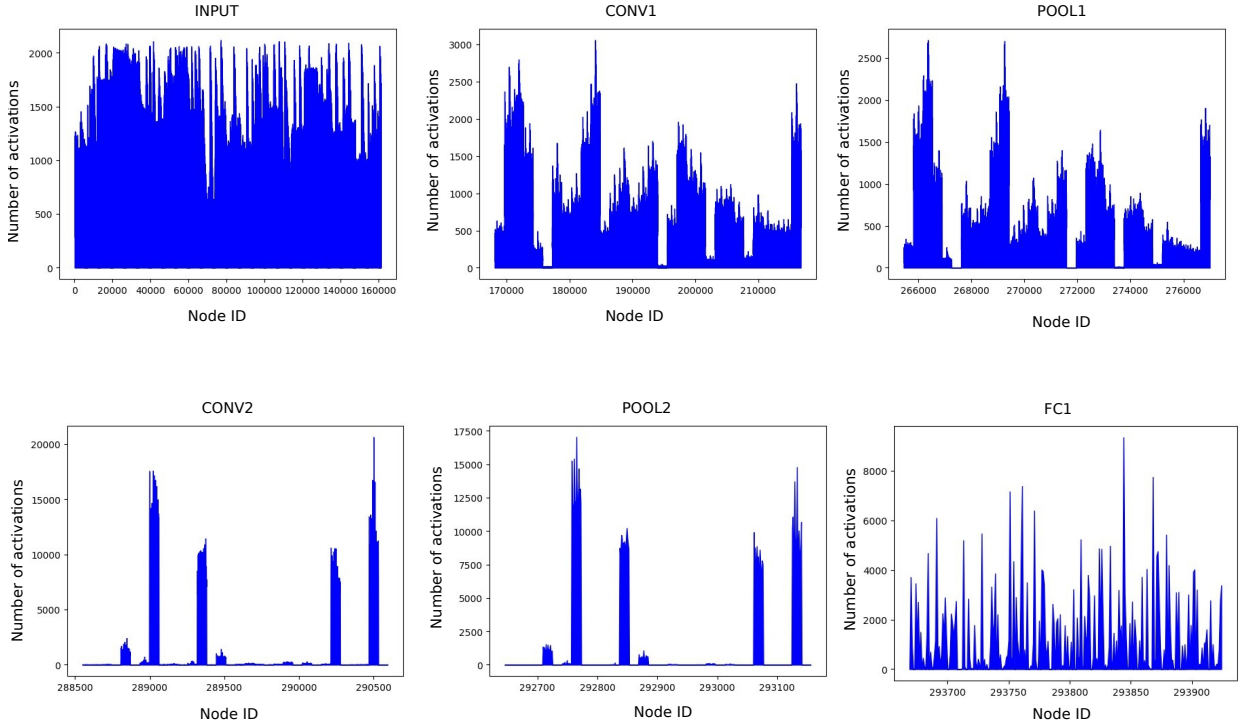

Figure 7: Layer-wise firing rates of SNN neurons

6 Conclusions

In this paper we propose heartbeat classification from Electrocardiogram (ECG) signals using event-driven spiking neural networks (SNN). We first convert the window of ECG signals containing the QRS peak into joint distribution signatures using short-time Fourier transform, a time-frequency joint distribution of ECG signals. Next, we build a convolution neural network (CNN)-based heartbeat classifier using the ECG joint distribution signatures. We then convert the CNN operations into spiking equivalents with minimal loss in accuracy. We then simulate the SNN and demonstrate that for every layer of the network, on average, the SNN generates 87.76% fewer events per synapse and 96.82% fewer neuron activations for a single input when compared to the CNN, and does so with an accuracy loss of only 0.6% -1.0% across the 6 classes.

We conclude that our approach of converting a heartbeat classification CNN to SNN is more energy efficient and has shorter runtime latency, with minimal loss of classification accuracy, when compared to previously-proposed neural networks-based heartbeat classification.

Acknowledgements

This work is supported in parts by EU-H2020 grant NeuRAM3 Cube (NEUral computing aRchitectures in Advanced Monolithic 3D-VLSI nano-technologies) and ITEA3 proposal PARTNER (Patient-care Advancement with Responsive Technologies aNd Engagement togetheR).

REFERENCES

[1] D. Phan, L. Y. Siong, P. N. Pathirana, and A. Seneviratne, "Smartwatch: Performance evaluation for long-term heart rate monitoring," in *International Symposium on Bioelectronics and Bioinformatics (ISBB)*, pp. 144–147, IEEE, 2015.

[2] R. M. Aarts and M. Ouwerkerk, "Apparatus for monitoring a person's heart rate and/or heart rate variation; wrist-watch comprising the same," Nov. 14 2006. US Patent App. 12/097,548.

[3] J. Penders, J. van de Molengraft, M. Altini, F. Yazicioglu, and C. Van Hoof, "A low-power wireless ecg necklace for reliable cardiac activity monitoring on-the-move," IEEE, 2011.

[4] A. Das, F. Catthoor, and S. Schaafsma, "Heartbeat classification in wearables using multi-layer perceptron and time-frequency joint distribution of ecg," *Workshop on Deep Learning and Edge Computing in IOT-centered Health Applications*, 2018.

[5] P. De Chazal, M. O'Dwyer, and R. B. Reilly, "Automatic classification of heartbeats using ecg morphology and heartbeat interval features," *IEEE transactions on biomedical engineering*, vol. 51, no. 7, pp. 1196–1206, 2004.

[6] I. Christov, G. Gómez-Herrero, V. Krasteva, I. Jekova, A. Gotchev, and K. Egiazarian, "Comparative study of morphological and time-frequency ecg descriptors for heartbeat classification," *Medical engineering & physics*, vol. 28, no. 9, pp. 876–887, 2006.

[7] M. Llamedo and J. P. Martínez, "Heartbeat classification using feature selection driven by database generalization criteria," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 3, pp. 616–625, 2011.

[8] C. Ye, B. V. Kumar, and M. T. Coimbra, "Heartbeat classification using morphological and dynamic features of ecg signals," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 10, pp. 2930–2941, 2012.

[9] Z. Zhang, J. Dong, X. Luo, K.-S. Choi, and X. Wu, "Heartbeat classification using disease-specific feature selection," *Computers in biology and medicine*, vol. 46, pp. 79–89, 2014.

[10] I. Jekova, G. Bortolan, and I. Christov, "Assessment and comparison of different methods for heartbeat classification," *Medical Engineering & Physics*, vol. 30, no. 2, pp. 248–257, 2008.

[11] E. J. d. S. Luz, W. R. Schwartz, G. Cámara-Chávez, and D. Menotti, "Ecg-based heartbeat classification for arrhythmia detection: A survey," *Computer methods and programs in biomedicine*, vol. 127, pp. 144–164, 2016.

[12] W. Rakowski and P. Tadejko, "Mathematical morphology based ecg feature extraction for the purpose of heartbeat classification," in *2007 6th International Conference on Computer Information Systems and Industrial Management Applications(CISIM)*, vol. 00, pp. 322–327, 06 2007.

[13] C. Tamburino, D. Capodanno, A. Ramondo, A. S. Petronio, F. Ettori, G. Santoro, S. Klugmann, F. Bedogni, F. Maisano, A. Marzocchi, *et al.*, "Incidence and predictors of early and late mortality after transcatheter aortic valve implantation in 663 patients with severe aortic stenosis," *Circulation*, vol. 123, no. 3, pp. 299–308, 2011.

[14] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.

[15] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[16] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[17] G. Indiveri, F. Corradi, and N. Qiao, "Neuromorphic architectures for spiking deep neural networks," pp. 4–2, IEEE, 2015.

[18] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[19] N. K. Kasabov, "Neucube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data," *Neural Networks*, vol. 52, pp. 62–76, 2014.

[20] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 699–716, 2014.

[21] J. Schemmel, D. Briiderle, A. Griibl, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," pp. 1947–1950, IEEE, 2010.

[22] C.-C. Lin and C.-M. Yang, "Heartbeat classification using normalized rr intervals and morphological features," *Mathematical Problems in Engineering*, vol. 2014, 2014.

[23] S. Dutta, A. Chatterjee, and S. Munshi, "Correlation technique and least square support vector machine combine for frequency domain based ecg beat classification," *Medical engineering & physics*, vol. 32, no. 10, pp. 1161–1169, 2010.

[24] S. Faziludeen and P. Sabiq, "Ecg beat classification using wavelets and svm," in *Information & Communication Technologies (ICT), 2013 IEEE Conference on*, pp. 815–818, IEEE, 2013.

[25] F. Melgani and Y. Bazi, "Classification of electrocardiogram signals with support vector machines and particle swarm optimization," *IEEE transactions on information technology in biomedicine*, vol. 12, no. 5, pp. 667–677, 2008.

[26] Y. H. Hu, W. J. Tompkins, J. L. Urrusti, and V. X. Afonso, "Applications of artificial neural networks for ecg signal detection and classification.," *Journal of electrocardiology*, vol. 26, pp. 66–73, 1993.

[27] S. Osowski and T. H. Linh, "Ecg beat recognition using fuzzy hybrid neural network," *IEEE Transactions on Biomedical Engineering*, vol. 48, no. 11, pp. 1265–1271, 2001.

[28] S.-N. Yu and Y.-H. Chen, "Electrocardiogram beat classification based on wavelet transformation and probabilistic neural network," *Pattern Recognition Letters*, vol. 28, no. 10, pp. 1142–1150, 2007.

[29] M. M. Al Rahhal, Y. Bazi, H. AlHichri, N. Alajlan, F. Melgani, and R. R. Yager, "Deep learning approach for active classification of electrocardiogram signals," *Information Sciences*, vol. 345, pp. 340–354, 2016.

[30] M. Zubair, J. Kim, and C. Yoon, "An automated ecg beat classification system using convolutional neural networks," in *IT Convergence and Security (ICITCS), 2016 6th International Conference on*, pp. 1–5, IEEE, 2016.

[31] P. Rajpurkar, A. Y. Hannun, M. Haghpanahi, C. Bourn, and A. Y. Ng, "Cardiologist-level arrhythmia detection with convolutional neural networks," *arXiv preprint arXiv:1707.01836*, 2017.

[32] A. Das, P. Pradhapan, W. Groenendaal, P. Adiraju, R. T. Rajan, F. Catthoor, S. Schaafsma, J. L. Krichmar, N. Dutt, and C. Van Hoof, "Unsupervised heart-rate estimation in wearables with liquid states and a probabilistic readout," *Neural Networks*, vol. 99, pp. 134–147, 2018.

[33] S. Schliebs and N. Kasabov, "Evolving spiking neural networka survey," *Evolving Systems*, vol. 4, no. 2, pp. 87–98, 2013.

[34] R. Van Rullen and S. J. Thorpe, "Rate coding versus temporal order coding: what the retinal ganglion cells tell the visual cortex," *Neural computation*, vol. 13, no. 6, pp. 1255–1283, 2001.

[35] M. C. Van Rossum, "A novel spike distance," *Neural computation*, vol. 13, no. 4, pp. 751–763, 2001.

[36] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in computational neuroscience*, vol. 9, no. 0, pp. 0–0, 2015.

[37] A. Tavanaei and A. S. Maida, "A spiking network that learns to extract spike signatures from speech signals," *Neurocomputing*, vol. 240, pp. 191 – 199, 2017.

[38] F. Corradi and G. Indiveri, "A neuromorphic event-based neural recording system for smart brain-machine-interfaces," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 9, pp. 699–709, Oct 2015.

[39] W. Maass, "Liquid state machines: motivation, theory, and applications," in *Computability in context: computation and logic in the real world*, pp. 275–296, World Scientific, 2011.

[40] W. Maass and T. Natschläger, "Networks of spiking neurons can emulate arbitrary hopfield nets in temporal coding," *Network: Computation in Neural Systems*, vol. 8, no. 4, pp. 355–371, 1997.

[41] S. M. Bohte, J. N. Kok, and J. A. La Poutré, "Spikeprop: backpropagation for networks of spiking neurons.," in *ESANN*, pp. 419–424, 2000.

[42] H. S. Seung, "Learning in spiking neural networks by reinforcement of stochastic synaptic transmission," *Neuron*, vol. 40, no. 6, pp. 1063–1073, 2003.

[43] S. Song, K. D. Miller, and L. F. Abbott, "Competitive hebbian learning through spike-timing-dependent synaptic plasticity," *Nature neuroscience*, vol. 3, no. 9, p. 919, 2000.

[44] F. Ponulak, "Resume-new supervised learning method for spiking neural networks," *Institute of Control and Information Engineering, Poznan University of Technology*, vol. 42, 2005.

[45] 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 *Neural Networks (IJCNN), 2015 International Joint Conference on*, pp. 1–8, IEEE, 2015.

[46] 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, 2015.

[47] 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, p. 682, 2017.

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

[49] A. for the Advancement of Medical Instrumentation *et al.*, "Testing and reporting performance results of cardiac rhythm and st segment measurement algorithms," *ANSI/AAMI EC38*, vol. 1998, 1998.

[50] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236–243, 1984.

[51] I. Daubechies, "The wavelet transform, time-frequency localization and signal analysis," *IEEE transactions on information theory*, vol. 36, no. 5, pp. 961–1005, 1990.

[52] L. B. Almeida, “The fractional fourier transform and time-frequency representations,” *IEEE Transactions on signal processing*, 1994.

[53] M. T. Hagan and M. B. Menhaj, “Training feedforward networks with the marquardt algorithm,” *IEEE transactions on Neural Networks*, vol. 5, no. 6, pp. 989–993, 1994.

[54] CEA-LIST, “N2D2,” in *https://github.com/cea-list/n2d2*.

7 BIOGRAPHIES

Adarsha Balaji received a Bachelors degree from Visvesvaraya Technological University, India, in 2012 and a Master's degree from Drexel University, Philadelphia, PA, in 2017. He is currently pursuing a Ph.D. degree from the Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA. His current research interests include design of neuromorphic computing systems, particularly data-flow and power optimization of spiking neural networks (SNN) hardware.

Federico Corradi is a research and development scientist at Imec-nl, Eindhoven, The Netherlands. He received a B.Sc. degree in Physics (2007) from Universit degli studi di Parma, Italy, a M.Sc. degree (cum laude) in Physics (2010) from La Sapienza University, Rome, Italy, a Ph.D. degrees in Natural Sciences (2015) from the University of Zrich, Switzerland, and a Ph.D. in Neuroscience (2015) from the ETH Neuroscience Center Zrich, Switzerland. His research activities are at the interface between neuroscience and neuromorphic engineering. His research is focused towards a new generation of computing architectures based on bio-inspired neural networks, and his contribution in the field includes novel online learning circuits for neuromorphic VLSI systems.

Anup Das is an Assistant Professor at Drexel University. He received a Ph.D. in Embedded Systems from National University of Singapore in 2014. Prior to his Ph.D., he was a research engineer for more than 7 years at ST Microelectronics (India and Grenoble) and LSI Corporation (India). Following his Ph.D., he was a post-doctoral fellow at the University of Southampton from 2014 to 2015 and a researcher at IMEC from 2015 to 2017. His research focuses on neuromorphic computing, from algorithm development to architectural exploration. His other research interests include System-level design techniques for lifetime and energy optimization, Soft-error tolerance of FPGA configuration bitstream, Synchronous data flow graph based task mapping and scheduling, Probabilistic energy and performance optimization of multiprocessor systems, architectural adaptations for lifetime improvement of multiprocessor, and Design-for-testability (DFT) of multi-power domain SoC.

Sandeep Pande graduated with a Ph.D. (in Electronics Eng.) from the National University of Ireland, Galway, Ireland in 2014 and Bachelor and Master in Electronic Eng. studies from Nagpur University (India) in 1996 and 2000 respectively. His Ph.D. research addressed key challenges for the design of neuromorphic systems and applications. Since 2001, He was with leading universities and semiconductor research companies, where he developed and deployed system-level design methodologies and SoC architectures. His research interests include next-generation cognitive systems such as nature-inspired computing techniques, and neuromorphic systems; and their application to the development of autonomous systems.

Siebren Schaafsma is an R&D manager in the IoT unit of Imec The Netherlands (Imec-nl). This unit is part of the Holst Center in Eindhoven. He is responsible for two teams of Analog and Digital IC designers building new state of the art Radio ICs and sub GHz Radar (BT-LE, Wifi 11.ah, subGHz, etc). He is also responsible for a team of embedded hardware and software engineers working in the domain of IoT and Artificial Intelligence. He received two masters (Nuclear physics in 1988 and computer science in 1989) at the Rijks Universiteit Groningen (RUG). His dissertation in the latter one addresses a neural networks implementation on a transputer cluster. He received his Ph.D. (Dr.) from the University of Nijmegen (KUN) in the Biophysics Department. His dissertation addresses the coding of optic flow in the visual cortex. He holds two patents on his research inventions from his period in research at Ericsson Telecommunications.

Francky Catthoor received a Ph.D. in EE from the Katholieke Univ. Leuven, Belgium in 1987. Between 1987 and 2000, he has headed several research domains in the area of synthesis techniques and architectural methodologies. Since 2000 he is strongly involved in other activities at IMEC including deep submicron

technology aspects, IoT and biomedical platforms, and smart photovoltaic modules, all at IMEC Leuven, Belgium. Currently he is an IMEC fellow. He is also part-time full professor at the EE department of the KULeuven. He has been associate editor for several IEEE and ACM journals. He was elected IEEE fellow in 2005.