SNN with Gradient-based Backpropagation algorithm for ECG arrhythmia classification with LIF neuron and AdEx neuron

1st Bana Shanmuga Sai Badrinatha Reddy*, 1st Priya K*, 2nd Binsu J Kailath
Electronics and Communication Enggineering, IIITDM Kancheepuram
Chennai, India
badrinathreddy.bana@gmail.com, ec21d0008@iiitdm.ac.in, bkailath@iiitdm.ac.in

***Abstract*—Signal processing is used by many to diagnose cases of Arrhythmia in a person's heart by analyzing the Electrocardiogram (ECG) of the patient. Recently, ML algorithms and neural networks have been adopted to process the time-varying ECG signal and diagnose medical conditions. As part of this, Spiking neural networks (SNN) have gotten significant attention for their potential in analyzing time-varying signals like ECG. This project explores the SNN's ability to process data with a few neuron models that can mimic the behavior of the biological neuron. This project focuses on finding the efficacy and complexity of the SNN's that are modeled for the classification of the ECG into cases of Arrhythmia with the parameters that are extracted from the ECG signal. In this work, two 3-layered fully connected SNN models (3 x n x 5 architecture), where n is the number of neurons in the hidden layer, are created for 5 classes of arrhythmia cases (N, V, S, A, R) according to AAMI standard with 2 different neuron models namely LIF and AdEx, and input features extracted from the ECG data from the MIT-BIH database. The model was trained using the backpropagation algorithm to compute the loss gradient for updating the synapse weights. The model is designed to maximize accuracy and specificity with less complexity. An accuracy of 85.94% is achieved with AdEx neurons with 10 neurons in the hidden layer (n = 10) and 81.65% is achieved with the LIF neurons with 15 neurons in the hidden layer (n = 15).**

***Index Terms*—AdEx, Arrhythmia, BPTT, ECG, LIF, SNN.**

I. Introduction

Cardiovascular diseases continue to be the leading cause of many health problems and fatalities worldwide, thus timely and accurate diagnosis is critical [1]. Electrocardiogram (ECG) data captures the heart's electrical activity, which is critical for diagnosing arrhythmia and other cardiac disorders. Arrhythmia occurs when a person's heart beats too rapidly, too slowly, or irregularly as a result of unusual electrical signals. Modern developments in computational neuroscience have contributed to an increasing interest in working with Spiking Neural Networks (SNNs) to handle temporal data, as ECG signals change over time [2]. SNNs closely mimic real neurons as they can characterize neural activity through discrete spikes, unlike ANNs, which operate on continuous activation values.

Many SNN models have been created in the past with various architectures and training algorithms for either high accuracy or low power and resource requirements [3] [4]. A simple less complex architecture using a gradient-based backpropagation algorithm to train SNN instead of STDP or TSTDP is also reported [5].

SNN with its inherent biological plausibility and low power consumption can bridge the gap between current artificial intelligence systems and the brain as reported by J. K. Eshraghian et. *al* [6]. This work also explains how to implement algorithms like backpropagation through time, and surrogate gradient descent, and explains how to convert the ANN models to SNN. The authors explore encoding data as spikes, challenges in applying gradient-based learning to SNNs, and the relationship between temporal backpropagation and STDP.

A novel supervised learning method for training deep SNNs using backpropagation is given in [7]. This technique treats membrane potentials of spiking neurons as differentiable signals for allowing the error backpropagation similar to conventional deep networks. This method also addresses the challenge of non-differentiable spike events and aims to capture spike statistics more precisely. It demonstrates improved accuracy and computational efficiency compared to previous SNN methods and conventional CNN's. This paper also introduces regularization techniques for stable and balanced learning in SNNs.

The main objective of this work is to look at the complexity and effectiveness of different SNN models used to classify ECG arrhythmias. A variety of neuron models, including Adaptive Exponential integrate-and-fire model (AdEx) neurons and Leaky Lntegrate-and-Lire model (LIF) neurons are used to show how choosing a neuron model influences both computing requirements and classification results.

SNN models and neuron models discussed in this paper are designed in the Python environment using *snnTorch* library that increases PyTorch's functionality by applying its GPU-accelerated tensor computation to SNNs. This implementation

This work has been carried out as part of the project funded by the C2S project of MeitY, Govt. of India.

*Both Bana Shanmuga Sai Badrinatha Reddy and Priya K contributed equally to this work.

is carried out as per the procedure given in [6]. The proposed model is trained with data available in MIT-BIH Arrhythmia database [8].

II. Neuron models

The SNN model created with the LIF is referred to as SNN1 and the SNN model created with the AdEx neuron is referred to as SNN2 and hierarchy of this model in *snnTorch* is highlighted in Fig. 1.

A. Leaky Integrate and Fire neuron (LIF)

The integrate and fire neuron model is a simple yet effective neuron model that is commonly utilized in SNNs and other areas of neuroscience [9]. It integrates the membrane potential over time, taking input current as a parameter, and generates a spike when the membrane potential exceeds a predetermined threshold voltage. It is a basic RC circuit that can simulate the spiking behavior of a neuron.

$$\tau\frac{dU(t)}{dt} = -U(t) + RI(t) \tag{1}$$

τ is the time constant, U is the membrane potential of the neuron, R is resistance and I is the input current for the neuron.

The simplification and approximation of (1) are explained in the paper [6]. Equation (2) below is the final equation that is used in this study referred to as Leaky as given in *snnTorch*.

$$U[t+1] = \beta U[t] + WX[t+1] - S[t]U_t \tag{2}$$

The $\beta = (1 - \frac{1}{\tau})$, R is assumed as 1, $I = W.X[t]$, and U_t is the reset threshold of the LIF neuron.

B. Adaptive Exponential Neuron model (AdEx)

The Adaptive Exponential Integrate-and-fire model (AdEx) uses two variables to describe spiking neurons. In (3), which includes an activation component with an exponential voltage dependency, represents the kinetics of the membrane potential, whereas (4) determining how adaptability and voltage are connected [10].

$$\frac{C\delta U}{\delta T} = -g_L(U - E_L) + g_L\Delta T exp(\frac{U - V_T}{\Delta T}) + I - \omega \tag{3}$$

$$\frac{\delta\omega}{\delta T}\tau_\omega = a(U - E_L) - \omega \tag{4}$$

$$\begin{cases} U = E_L & \text{if } U > 0 \\ \omega = \omega + b \end{cases} \tag{5}$$

Equation (5) explains how the U and ω update when the spike occurs.

These equations have the hyperparameters as ω, C, g_l, E_L, V_T, Δ_t and 'a' that stand for adaptation variable, membrane capacitance, leak conductance, leak reversal potential, threshold, and slope factor, respectively. Increasing 'a' and 'b' can increase the strength of adaptation and hence decrease the firing rate and vice versa.

C. Neuron model implementation

The equation (2) is implemented in *snnTorch* as *nn.Leaky* with the input hyperparameter β. This model is used in the SNN1.

The equations (3), (4), (5) are used in the SNN2. The *spikingneuron* class in *snnTorch* serves as a parent class for creating new neuron models. AdEx neuron is constructed on top of it so that it can be incorporated in the SNN2.

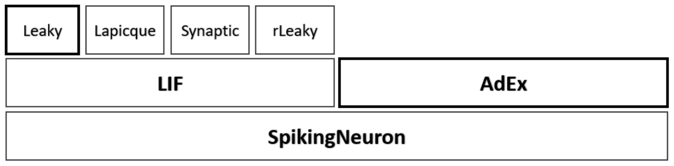


Fig. 1. Hierarchy of the classes in snn.torch. Neuron models used in SNN1 and SNN2 are highlighted.

III. SNN Model

A. Synapse

Both of the neuron models described above use a similar logic to determine when the spike is created, which is when the membrane potential crosses a threshold value. Each synaptic connection to the post-neuron can be represented as shown in Fig. 2.

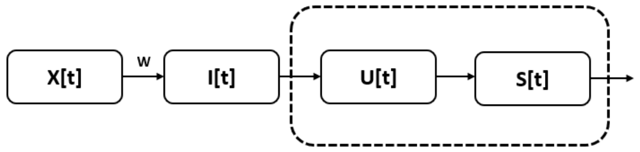


Fig. 2. X represents the spike train from the pre-neuron, I is the input current to the post-neuron, U is the membrane potential of the post-neuron and S is the spike function of the neuron.

The two components in the dashed box are neuron-dependent and the remaining two are Linear connection layer between two neurons (pre and post neurons). Spikes are observed for each input data value over certain time intervals known as steps, and the number of spikes generated in the particular window of time varies with that input current.

B. Back Propogation Through Time (BPTT) Algorithm

The training procedure involves two fundamental paths: *forward* and *backward* pass. Forward and backward passes can be represented as seen in Fig. 3.

$$\frac{\delta L}{\delta W} = \frac{\delta L}{\delta S}\frac{\delta S}{\delta U}\frac{\delta U}{\delta I}\frac{\delta I}{\delta W} \tag{6}$$

The function L(t) described in equation (6) is a Loss function, which will be covered in the following sections. The backpropagation algorithm's objective is to minimize Loss. To do this, the chain rule is applied from the last layer back to each weight, calculating the gradient of the loss with respect to the learnable parameter W. The weights are then changed using the gradient calculated using equation (6) to reduce error. If this gradient is ”0,” no weight update occurs. [6].

The relation of spikes generated by neuron with the Membrane potential of the neuron can be represented as

$$S[t] = \Theta(U[t] - U_t) \tag{7}$$

Where Θ is the Heaviside Step function.

$$\Theta(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \tag{8}$$

In the backpropagation path, differentiation of the Heaviside function of S[t] with respect to U[t] is needed, which results in an impulse function at the threshold value. This function creates an issue termed the Dead neuron problem, which makes the Loss gradient with respect to weight uncomputable.

1) Dead Neuron Problem: The term $\frac{\delta S}{\delta U}$ in the loss gradient function is an impulse function with values $0, \infty$. One of the ways to solve this problem is to replace the Heaviside function with some other function to make it differentiable. The term $\frac{\delta \mathring{S}}{\delta U}$ is replaced with another function.

$$\frac{\delta \ddot{S}}{\delta U} \leftarrow \frac{1}{\pi} \frac{1}{1 + [U\pi]^2} \tag{9}$$

This is commonly known as the *surrogate gradient approach*. The Surrogate gradient approach replaces the Heaviside function with a *shifted atan* function, as shown in Fig. 4, with the derivative being differentiable. This substitution does not affect the SNN’s learning process because it is robust to all of these modifications.

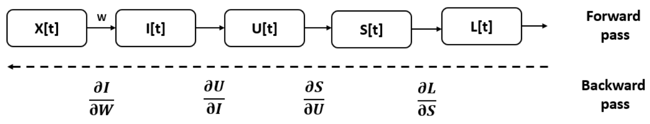
$$atan(x) = tan^{-1}(x) \tag{10}$$

This replacement of the Heaviside function with the shifted atan function is done while designing the neuron model.

2) Back Propagation Through Time (BPTT): The term $\frac{\delta S}{\delta U}$ equation in (6) is replaced with $\frac{\delta \mathring{S}}{\delta U}$ giving the following equation.

$$\frac{\delta L}{\delta W} = \frac{\delta L}{\delta S} \frac{\delta \mathring{S}}{\delta U} \frac{\delta U}{\delta I} \frac{\delta I}{\delta W} \tag{11}$$

Equation (11) calculates the gradient for a single step, known as instantaneous influence. BPTT algorithm calculates and totals the gradient for the loss to all previous stages, also known as prior influence.



X[t]
w
I[t]
U[t]
S[t]
L[t]
Forward pass
$\frac{\partial I}{\partial W}$
$\frac{\partial U}{\partial I}$
$\frac{\partial S}{\partial U}$
$\frac{\partial L}{\partial S}$
Backward pass

Fig. 3. The forward path calculates the Loss of the model and the backward pass calculate the gradient of the loss with respect to synapse weight using chain rule.

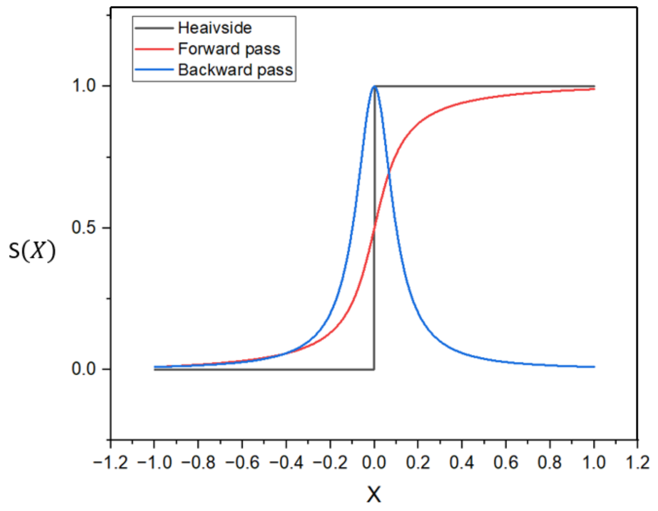


Fig. 4. Heaviside function is replaced with the atan in forward pass, the differentiation of this atan replaces the impulse function in the backward pass.

Each time step, the weight W is applied, and the loss is determined. The global gradient will be calculated by adding the weight’s effect on both current and prior losses as given in (12).

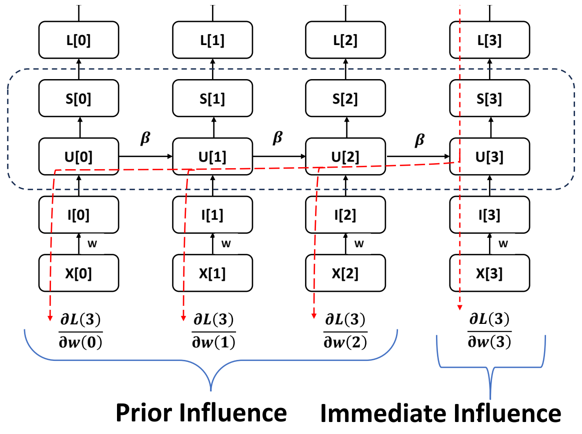
$$\frac{\delta L}{\delta W} = \sum_t \frac{\delta L[t]}{\delta W} = \sum_t \sum_{s \leq t} \frac{\delta L[t]}{\delta W[s]} \frac{\delta W[s]}{\delta W} \tag{12}$$

By restricting s¡t, consider the impact of current and prior effects on the loss. In a recurrent system, the weights have shared access to all time steps inside the time window, so the W[0]=W[1]=...=W implying $\delta W[s]/\delta W$=1

$$\frac{\delta L}{\delta W} = \sum_t \sum_{s \leq t} \frac{\delta L[t]}{\delta W[s]} \tag{13}$$

Fig. 5 shows the BPTT algorithm for a single synapse.

3) Loss Function: The network’s desired outcome ideally is that the right output neuron generates the highest amount



L[0]
L[1]
L[2]
L[3]
S[0]
S[1]
S[2]
S[3]
U[0]
U[1]
U[2]
U[3]
β
β
β
I[0]
I[1]
I[2]
I[3]
w
w
w
w
X[0]
X[1]
X[2]
X[3]
$\frac{\partial L(3)}{\partial w(0)}$
$\frac{\partial L(3)}{\partial w(1)}$
$\frac{\partial L(3)}{\partial w(2)}$
$\frac{\partial L(3)}{\partial w(3)}$
Prior Influence
Immediate Influence

Fig. 5. Total loss calculation of a synapse by backpropagation in 4 time intervals.

of spikes, while the remaining neurons do not generate spikes. But, in reality, all neurons generate spikes, therefore the weights of the synapses that connect the neurons should be adjusted so that the correct neuron generates more spikes than the others.

The *Softmax* for the membrane potential of output neurons is taken at every step to create a probability distribution of the outcomes.

$$p_i[t] = \frac{e^{U_i[t]}}{\sum_{i=0}^{4} e^{U_i[t]}} \tag{14}$$

The Cross entropy loss of the softmax is calculated and the output target $y_i \in \{0,1\}^5$ is taken to find the loss at a particular step.

$$L[t] = -\sum_{i=0}^{4} y_i \log(p_i[t]) \tag{15}$$

The Loss at each step is summed to find the total loss.

$$L = \sum_{t} L[t] \tag{16}$$

This derived loss value is used in the gradient calculation and the weights are modified accordingly to minimize this loss value.

C. Layers

The 3-layer model as shown in Fig. 6, described as 3 x n x 5 in this work has one input layer with one neuron for each input value (RR, QRS, P-wave), and one output layer with one neuron for each output class (N, V, S, A, R). The number of neurons in the hidden layer 'n' is a hyperparameter that can be tuned according to the requirements.

These three layers are connected with fully connected linear connections that take spikes of pre-neuron in the form of vectors as input and generate an output vector that is described by (17) and is given to the post-neuron as current.

$$I = W.X - B \tag{17}$$

I is the current vector given to the post-neurons connected to the end of the linear layer, W is the vector containing the weights of the synapses, X is the vector of spikes generated by the pre-neurons connected to the linear layer at a particular step and B is a bias vector for the synapses.

IV. Proposed Architecture

The input data for the SNN is extracted by encoding the ECG signal from the MIT-BIH database into 2 channels of spikes with the AdEx neuron, one channel for the positive values and the other channel for the negative values of the ECG data. These 2 channels of spikes are analyzed and parameters like RR, QRS-complex, and P-wave are found. ECG is encoded directly, without any pre-processing and from the modulated spike train the specified parameters are determined. The values are classified into various classes according to the AAMI standards. The proposed architecture is shown in Fig. 7.

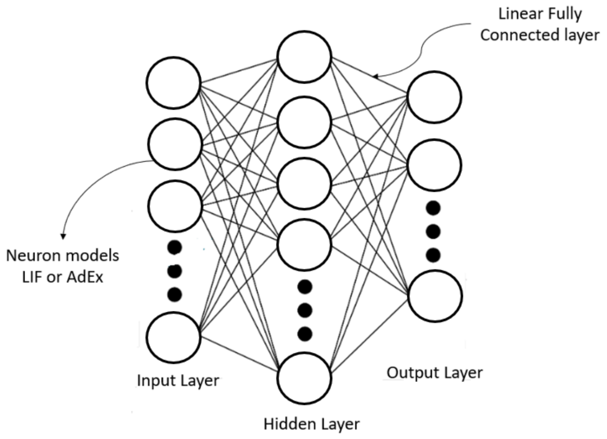


Fig. 6. 3 layer SNN model with 3 x n x 5 architecture.

TABLE I
Micro Average scores, Precision, Recall and F1_Score compared for SNN1 and SNN2

Neuron	MicroAvgPre	MicroAvgRec	MicroAvg F1
SNN1 (LIF)	0.7800	0.7800	0.7800
SNN2 (AdEx)	0.8600	0.8600	0.8600

From the extracted parameters, in this model the values RR (ms), QRS-complex (ms), and presence and absence of P-wave (in form of 1 or 0) are given as inputs after normalizing as the neurons are sensitive to the scale of input. The normalized values are given as the input vector to the SNN model. This normalized dataset is used in training both SNN1 and SNN2 models with the BPTT algorithm by finding the gradient of the loss vector with respect to synapse weights.

The entire dataset containing samples of 11,000 is divided into 2, 70% of the data is used for training and 30% of the data is used for testing the trained model. Adam optimizer is used in minimizing the loss function i.e. CrossEntropyLoss as discussed in the previous section. The model gives the beat-vise classification of the ECG signal by classifying the data into 5 classes N, V, S, A, R.

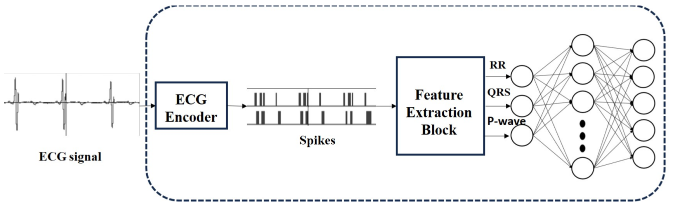


Fig. 7. ECG signal is given to 2 AdEx neurons in ECG Encoder to Encode the ECG into 2 spike trains one for positive values (above) and the other for negative values (below) in ECG. The 2 spike trains are given to the Feature extraction block to extract the input features, normalize them, and give them to the SNN model.

V. Results and Comparison

The SNN1 (SNN model with the LIF neuron) is designed with 15 neurons in the hidden layer with a β value of 0.8 and a learning rate of $5X10^{-3}$. The SNN2 (SNN model with AdEx neurons) is designed with the values of the parameters taken from the paper [12]. The values for the tonic spiking are taken and the value of Δt is modified to 10 to get optimal spikes after observing the spikes generated by the model.

The performance of SNN1 and SNN2 are analyzed and compared by the following parameters Accuracy (acc) Specificity (spe), Precision (pre), Sensitivity (sen) , F1_score (F1) and Balanced accuracy as stated in (18) to (23), where TP, TN, FP, FN represents True Positive, True Negative, False Positive and False Negative values respectively. The calculated parameters are compared and is presented in Fig. 8, in which overall accuracy of the SNN model with LIF neurons is 81.62% and the accuracy for SNN model with AdEx neurons is 85.94% . It clearly shows AdEx is comparitively better than LIF for SNN implementation.

$$acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{18}$$

$$spe = \frac{TN}{TN + FP} \tag{19}$$

$$pre = \frac{TP}{TP + FP} \tag{20}$$

$$sen = \frac{TP}{TP + FN} \tag{21}$$

$$F1 = \frac{2 \times pre \times sen}{pre + sen} \tag{22}$$

$$Balanced\ Accuracy = \frac{\frac{TP}{TP+FN} + \frac{TP}{TP+FP}}{2} \tag{23}$$

Since the dataset for training the SNN is an imbalanced dataset, Metrics such as Balanced accuracy, Micro Average (and MacroAverage scores are also calculated, shown in Table I and II, shows AdEx has significantly higher Micro and Macro average precision and hence overall and across each class, its performance is good. The confusion matrix of both SNN models with SNN1 (LIF) and SNN2 (AdEx) models are shown in Fig. 9.

The results of the proposed models are compared with the other state of art models and is presented in Table III.

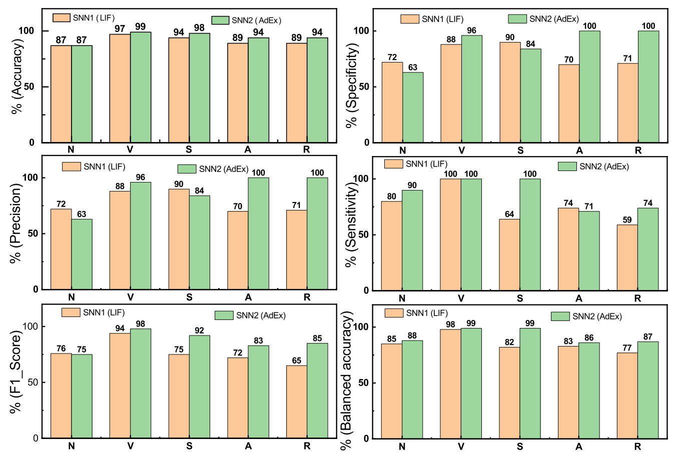


Fig. 8. Comparison of Accuracy, Specificity, Precision, Sensitivity, F1_Score, Balanced Accuracy of the classes N, V, S, A, R in percentage for both the neuron models SNN1 (LIF) (orange) SNN2 (AdEX) (green).

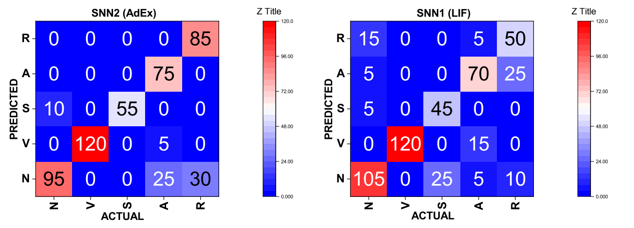


Fig. 9. Confusion matrix of SNN models with AdEx (Left) and LIF (right) neurons.

TABLE II
Macro Average scores, Precision, Recall and F1_Score compared for SNN1 and SNN2

Neuron	MacroAvgPre	MacroAvgRec	MacroAvg F1
SNN1 (LIF)	0.7855	0.7551	0.7635
SNN2 (ADEX)	0.8879	0.8716	0.8649

TABLE III
Proposed SNN models compared with state-of-the-art models

Network	Architecture	Training	Accuracy
RSNN-LIF pool [2]	256Ex-64In-153pool	SVM	ECG full resolution 94.2%, ECG ENC Delta mod 84.5%
MLP (Nyquist) [11]	800-fc640-fc15	BP	96.69%
Spiking MLP [11]	160-fc128-fc15	Firing rate oriented STBP	95.29% (fixed8)
SNN Inference Assisted by On-Chip ANN Learning [3]	96-fc32-fc16-fc5	On-chip learning	Independent testing w/o on-chip testing 97.35%, Independent testing w/ on-chip testing 93.67%
SNN1 (LIF)	3-fc15-fc5	Gradient based BPTT	81.65%
SNN2 (AdEx)	3-fc10-fc5	Gradient based BPTT	85.94%

The number of output classes in [2], [11] is 15, and in [3] is 5. Ex - Excitatory neuron, MLP - Multi-Layer Perceptron, In - Inhibitory neuron, fc - Fully Connected.

VI. Conclusion

This paper presents a simple SNN model focused on resource management with less number of neurons for the beat-wise classification of the ECG signal. The SNN models designed with AdEx and LIF neurons have 18 and 23 neurons respectively and have a total accuracy of 81.62% and 85.94% when tested on the testing dataset. The accuracy of the models discussed is low compared to the other architectures discussed in [2], [11], and [3] but the complexity of this model is less compared to the other models. ECG Features are extracted by a dedicated module in the proposed system, whereas in previous work, feature extraction and classification are carried out by neural network making such system more complex than the proposed one. Additional features of ECG such as PR-interval, PT-segment, QT-segment etc can be used to train the proposed SNN models, which will result in decrease in FP and FN, thereby improving both performance and accuracy. Hardware resource requirement is an important factor to be considered while developing standalone arrhythmia detection system. It can be observed from second column in Table III, that the architecture of the proposed system is much simpler than the previous one, making it more hardware friendly.

The SNN with AdEx neurons has shown a higher accuracy with less number of neurons compared to the SNN with LIF neurons. But the AdEx neuron being a more complex model than LIF, the training and simulation process is longer compared to training the SNN model with LIF neuron. This project also explored creating new neuron models in the *snn.torch* and utilizing them instead of the default neuron models available in the library.

References

[1] ”Cardiovascular diseases (CVDs)”, 2019, [online] Available: https://www.who.int/en/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds).

[2] F. Corradi et al., ”ECG-based Heartbeat Classification in Neuromorphic Hardware,” 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 2019, pp. 1-8, doi: 10.1109/IJCNN.2019.8852279.

[3] R. Mao et al., ”An Ultra-Energy-Efficient and High Accuracy ECG Classification Processor With SNN Inference Assisted by On-Chip ANN Learning,” in IEEE Transactions on Biomedical Circuits and Systems, vol. 16, no. 5, pp. 832-841, Oct. 2022, doi: 10.1109/TBCAS.2022.3185720.

[4] H. Chu et al., ”An Energy-Efficient and Robust SNN Classifier for LC-ADC Sampled ECG Signals,” 2023 8th International Conference on Integrated Circuits and Microsystems (ICICM), Nanjing, China, 2023, pp. 502-506, doi: 10.1109/ICICM59499.2023.10365892.

[5] Masquelier T, Guyonneau R, Thorpe SJ. Competitive STDP-based spike pattern learning. Neural Comput. 2009 May;21(5):1259-76. doi: 10.1162/neco.2008.06-08-804. PMID: 19718815.

[6] J. K. Eshraghian et al., ”Training Spiking Neural Networks Using Lessons From Deep Learning,” in Proceedings of the IEEE, vol. 111, no. 9, pp. 1016-1054, Sept. 2023, doi: 10.1109/JPROC.2023.3308088..

[7] Lee JH, Delbruck T, Pfeiffer M. Training Deep Spiking Neural Networks Using Backpropagation. Front Neurosci. 2016 Nov 8;10:508. doi: 10.3389/fnins.2016.00508. PMID: 27877107; PMCID: PMC5099523.

[8] G. B. Moody and R. G. Mark, ”The impact of the MIT-BIH Arrhythmia Database,” in IEEE Engineering in Medicine and Biology Magazine, vol. 20, no. 3, pp. 45-50, May-June 2001, doi: 10.1109/51.932724.

[9] A. S. Chouhan, “An analytical study of leaky integrate–and-fire neuron model using matlab simulation,” International Journal of Engineering Research Technology, vol. 2, no. 4, 2013.

[10] V. Pradeep Kumar, Dr. Binsu J Kailath., “ECG Encoding using AdEx Neuron,” in IBM IEEE CAS EDS AI Compute Symposium, 2022, held virtually from IBM USA on 13th October 2022.

[11] H. Chu et al., ”A Neuromorphic Processing System for Low-Power Wearable ECG Classification,” 2021 IEEE Biomedical Circuits and Systems Conference (BioCAS), Berlin, Germany, 2021, pp. 1-5, doi: 10.1109/BioCAS49922.2021.9644939.

[12] O. T. Yule Wang, “An optimization on the neuronal networks based on the adex biological model in terms of lut-state behaviors: Digital design and realization on fpga platforms,” Biology 2022, no. 11, p. 1125, 2022.